Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Improving TCP performance over a common IoT scenario using the Early Window Tailoring method

Marcos Talau [*], Thiago A. Herek, Mauro Fonseca, Emilio C.G. Wille

Federal University of Technology - Paraná (UTFPR), CPGEI, Curitiba, Paraná, Brazil

ARTICLE INFO

Keywords:
IoT
AQM
TCP
Congestion control
Receiver's advertised window

ABSTRACT

The Internet of Things (IoT) is already a reality in homes, transforming the home network into a complex composition of different equipment where each one has its own attributes in terms of TCP protocol variants, transmission speed, reliability and delay. In home network scenarios where IoT devices are used, it is common to have a 802.11 environment with several devices sharing the same Access Point (AP). When TCP is used in this environment its performance is affected by many things, such as wireless link losses, queue saturation on access points, and fairness. To improve the TCP performance in these scenarios, this paper presents a new version of the Early Window Tailoring (EWT) approach. The EWT-IoT do not require any modification in the TCP, and it does not require that all routers in the path use the same Active Queue Management (AQM), the use in the AP is sufficient. To evaluate its performance, comparison tests against drop-tail, RED, ARED and EWA were performed. The EWT-IoT proved to be effective in congestion control, achieving greater performance than the other approaches, reducing the transfer time, having a superior goodput, maintaining a satisfactory fairness, and presenting no losses. Also, the EWT-IoT allowed the existence of a number of flows on average 65.2% better than its best competitor and 71.3% better when no AQM scheme was used.

1. Introduction

The Internet of Things (IoT) involves connecting billions of new devices to the internet of the future, made possible by the accelerated convergence of many technologies including wireless communication, big data, real-time analytics and artificial intelligence [1].

In homes, this new reality is already present, transforming an old homogeneous environment, mainly composed of computers, into a diverse network formed by new elements such as: personal assistants, smartphones, televisions, smartwatches, other gadgets and sensors. This wide range of equipment that will certainly expand in the future and is not limited only to domestic use, also creates possibilities for new applications for smart cities, smart mobility, health, agriculture and many other domains, moving us to a horizon where: "everyone, anytime, anywhere will be connected to everything" [2,3].

In this environment, the home network becomes a complex and integrated composition of technologies, this is the scenario explored in this work. It is an 802.11 network, shown in Fig. 1, where a variety of elements share the same access point and may present their own attributes in terms of TCP protocol variants, transmission speed, reliability and delay [4].

In a wireless network with these characteristics the main limitations of the TCP protocol are: (a) The inability of TCP congestion algorithms to differentiate the cause of lost segments between link failure and actual network congestion, leading to the application of its contention mechanism, causing severe degradation of throughput when there is often no congestion. (b) The reducing of the transmission rate performed by the sender for "congestion control" affects the fairness of sharing and maintains a low level of network utilization. (c) In the loss recovery phase, the low growth of congestion window requires a greater number of ACK segment exchanges to achieve a better level of network utilization, which results in an increase in the time needed to complete the data transfer between the source and destiny [5].

TCP protocols that use proactive congestion control mechanisms, that is, those that rely on delay to measure congestion, such as: Vegas [6] and Westwood [7], are not able to use the full network transmission capacity. Likewise, reactive protocols such as Reno [8], New Reno [9,10], BIC [11] and CUBIC [12] that use segment losses to identify congestion, are also not suitable for the characteristics of these

[*] Corresponding author.
E-mail addresses: mtalau@utfpr.edu.br (M. Talau), therek@utfpr.edu.br (T.A. Herek), maurofonseca@utfpr.edu.br (M. Fonseca), ewille@utfpr.edu.br (E.C.G. Wille).

https://doi.org/10.1016/j.comnet.2023.109875
Received 20 February 2023; Received in revised form 3 May 2023; Accepted 11 June 2023
Available online 22 June 2023

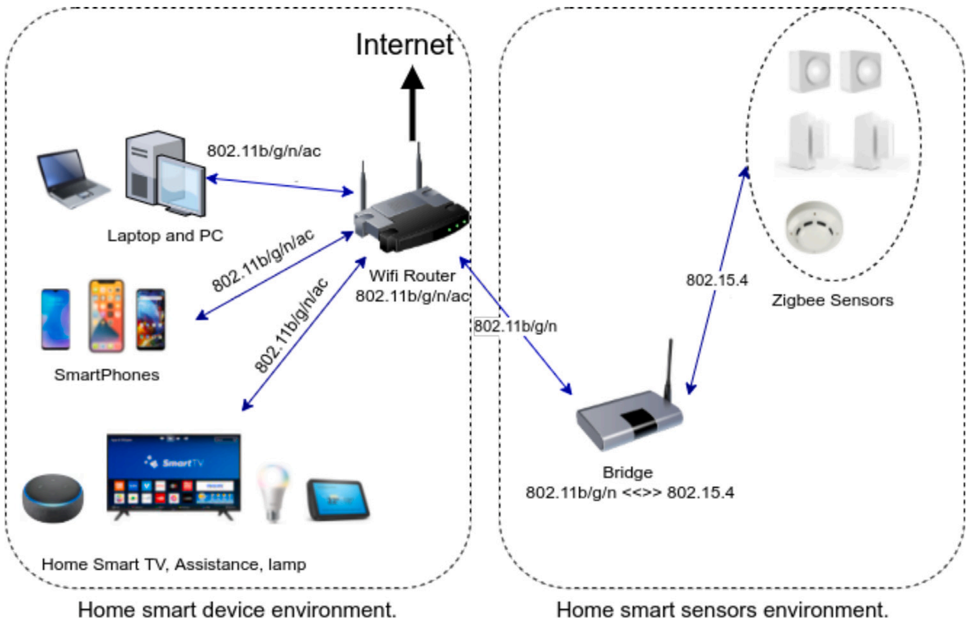


Fig. 1. IoT scenario with several devices over 802.11 network.

environments, as they are not adapted for devices that operate with low bandwidth [13].

Moreover, the possibility of the co-existence of multiple TCP variants, where through more aggressive algorithms, some can occupy network resources faster than others, makes this characteristic of the environment a factor that accentuates the problem of fairness.

In such a scenario, a better use of the available bandwidth, fairness and the prevention of discarding segments due to queue overflow are essential requirements for the operation of the various applications and, consequently, for the user experience.

In [14], the Early Window Tailoring (EWT) was proposed as new method to improve the TCP congestion control. The EWT acts by changing the value contained in the receiver's advertised window of the TCP ACK segments. The change is made in proportion to the memory space available in the gateway. Like other network feedback techniques, the EWT does not require any modification in the TCP implementations at the edges, and for its operation, only the gateway installation is sufficient. This method proved to be efficient in a network where the round-trip time (RTT) is low, as in a client–server communication in a corporate LAN. However, it was not originally conceived for a scenario where the RTT is high, common in communications between distant clients and servers, as in home wireless local area networks composed of several IoT devices accessing services on the internet.

When using the EWT, a low RTT allows keeping the sources updated of the current conditions of the network, however when the RTT is high the sources may not know the real conditions of the network, as they receive this information with delay.

In an IoT scenario, all the challenges presented above are accentuated and the eventual retransmissions due to buffer saturation also occupy precious resources that could not be wasted. To mitigate most of the problems discussed above, the main purpose of this paper is to present a proposal to modify the EWT method to work in an IoT environment, using a new mechanism to measure buffer occupancy and inform sources of network conditions, with the following objectives:

- Improve fairness, throughput and efficiency;
- Reduce segment losses due to buffer overflow;
- Maintain the compatibility with TCP variants and the simplicity of the original implementation of the EWT method.

The main contributions of this paper are:

- Proposing the EWT-IoT, a new version of EWT for IoT scenarios;
- Modeling the EWT-IoT analytically;
- Evaluating the EWT-IoT together with other techniques in the ns-3 network simulator.

The remainder of this paper is organized as follows: Section 2 presents the literature on congestion control in TCP networks, while Section 3 reveals a proposal for a new version of EWT for IoT environment. In Section 4 the simulation scenario is exposed, followed by Section 5 where the results are discussed. Conclusions and suggestions for future work are presented in Section 6.

2. Related work

In various wireless network architectures, including IoT, TCP remains a relevant transport protocol due to its reliable delivery property and its wide internet presence [15,16]. Among all the features of the TCP protocol, congestion control is one of the properties that has received the most attention over time and continues to be the object of several studies.

Studies involving this theme generally focus on the following techniques: active queue management, network feedback techniques, changes in the TCP protocol algorithm and their combinations [5,13,17–22]. This section presents a succinct survey of works that have addressed problems related to TCP and IoT.

2.1. TCP and IoT

In [23], an extensive comparative study of TCP congestion control algorithms in an IoT scenario is presented. The work evaluates 14 variants and identifies the advantages and disadvantages of each one. It appears that the greatest difficulties among those evaluated are mainly in throughput and fairness.

As the number of interconnected devices grows, congestion also increases. In the IoT, this growth is even more expressive, given the increasing emergence of new devices capable of connecting to existing networks. In this environment, congestion control must be applied to meet the requirements of a variety of devices that do not always use the same variant of the transport protocol, adjusting transmission rates as network overflow increases, and maintaining network fairness and throughput [23].

2.2. Active queue management

A router, when storing new segments in memory, can be saturated quickly when the rate of new segments is greater than its service capacity. These limitations are usually caused by bottlenecks in the network links. Active Queue Management (AQM) techniques seek to prevent this memory overflow.

Random Early Detection (RED) [19] is one of the first methods of active queue management, performing the early drop of segments using a stochastic technique when detecting an initial congestion. After its introduction, many extensions appeared: GRED [24], ARED [25], DRED [26], BLUE [27], MRED [28], ERED [29], YELLOW [30], EnRED and WRED [20] among others.

In [31] the authors compare six methods: DropTail, ARED, CoDel, PIE, GREEN and PINK in an IoT scenario. The simulations include an environment with multiple TCP variants, multiple congestion levels and multi-RTT. The results highlight equity and goodput issues in these challenging scenarios.

Among recent works there are Enhanced Random Early Detection (EnRED) and Time-window Augmented RED (WRED) presented in [20]. The first proposal consists of a change in the original RED algorithm to consider the current state of the queue, while the original algorithm uses the average queue length. The modification still depends on specifying some parameters for its operation. The second proposal, uses a window operating with average values between the queue length and the total buffer size. The modifications showed better results for the delay and discard of segments compared to other active queue managers like RED, ERED and BLUE in bottlenecked network environments.

2.3. Network feedback techniques

Network feedback techniques are defined as methods that inform, generally explicitly, the senders about the network conditions using Explicit Congestion Notification (ECN) [32]. To simplify the implementation and avoid the need for changes in the transport protocol, some works propose an implicit form of feedback, making changes in the header of the acknowledgment (ACK) segments sent from the receiver to the sender. In these cases, the ACK segment is basically intercepted and the value of the reception window is modified by the network router or gateway according to the congestion, making it possible to indirectly inform the sender of the network conditions and to promote a reduction in the transmission rate [21,22,33–35].

The studies [21,34] present the Smart AP with Low Advertised Window (SAP-LAW), while an improvement of this, named Vegas Over Access Point (VoAP) is proposed in [22]. These works consider the number of TCP flows and UDP traffic on the existing router or gateway to dynamically determine the new value of the receive window of confirmation segments. The approach has shown promise in several metrics such as fairness, throughput and delay in multi-traffic wireless network environments, with the advance of not requiring changes in TCP protocol.

In [33] a combination of RED and SAP-LAW methods is proposed and defined as Smart-RED. The system works with predefined minimum (min_{th}) and maximum (max_{th}) thresholds for the queue size (ql), operating with the SAP-LAW method when $min_{th} \leq ql < max_{th}$ and with the RED method while $ql < min_{th}$ or $max_{th} \leq ql$. The simulation scenario involves a wireless network environment and the method is applied in a shared access point. The results were promising for queue length and throughput in scenarios where there are multiple TCP flows and UDP bursts.

2.4. TCP variants

Modification proposals are usually implementations compatible with the standard TCP [36] or specific enhancements to the control algorithm of other known variants. In these modifications, protocols that use RTT to measure network congestion are classified as proactive, and protocols that use segment losses to identify congestion are reactive ones. According to the authors in [13], although reactive protocols are able to adequately use all available bandwidth, they are not adapted for devices that work with low bandwidth.

On the other hand, proactive protocols seek to reduce segment losses and retransmissions, but are unable to use the entire capacity of the network. Considering these conditions, the work in [13] presents a modification of the TCP protocol, capable of operating in proactive or reactive mode, depending on the network conditions. The protocol starts operating in reactive mode until the identification of an increase in congestion, when it switches the operation mode to proactive. The hybrid variant performed well for transfer rates and fairness compared to the CUBIC, FAST-FIT, HTCP and New Reno variants.

Another recent work focused on transport protocol modifications is presented in [18]. CERL+ is a variant inspired by TCP Reno, its main objective is to maximize throughput while minimizing the negative impact caused by lost segments. The proposal employs a mean and minimum RTT estimate to identify link congestion conditions and distinguish when random segment loss occurs due to a link failure, common in 802.11 networks. Compared to other TCP variants including mVeno, New Jersey+, Westwood+, CUBIC, YeAh, and NewReno the proposal showed better throughput, lower segment loss, alleviating bottleneck congestion in different wireless network configurations.

2.5. Combination approaches

Using a combination of the exposed approaches, the authors of [5] present FAIR+, a composite of AQM techniques, network feedback and TCP protocol changes. The proposal showed positive results for fairness, goodput and latency metrics in multi-hop wireless network environments compared to other variants such as Westwood, OQS and NRT.

Proposals for changes in the protocol and/or explicit feedback depend on interventions in all network devices, so their implementations are difficult even in small environments. On the other hand, the active queue management is a technique that presents greater simplicity in its implementation, as they are applied in routers or gateways. However, a large part of the proposals still work on early discard mechanisms, which causes an aggressive reduction in the transmission flow of the TCP protocol, mainly harming fairness and throughput.

Likewise as some of the reported work, the Early Window Tailoring (EWT) method [14] do not require changes in TCP for its operation, while being compatible with any variant based on the standard TCP, an essential feature in a dynamic IoT wireless network environment. By using the implicit feedback mechanism, the EWT is designed to be applied to a network gateway or router and indirectly inform the sender of congestion conditions.

3. The Early Window Tailoring (EWT)

The EWT brings to TCP an additional control by passing on network information that will be used in its transmission mechanism. This control is done by updating the value contained in the receiver window field (W_r) of ACK segments based on the number of available bytes of memory.

The receiver window value will be updated by EWT if its new value is greater than the Maximum Segment Size (MSS), otherwise the value will be set to one MSS. During ACK processing, it is not considered which flow they belong to, in this way all flows are treated in the same way and receive the same update (proportional to their window

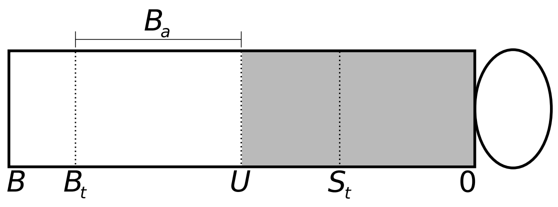


Fig. 2. The variables of the EWT method.

value). By only changing the value of the receiver window, EWT is compatible with standard TCP and does not require protocol changes, as the receiver window is naturally used by the server to limit its transmission rate. Fig. 2 illustrates the memory (buffer) in EWT and its variables.

The limit value provided by EWT is based on the amount of *bytes* available (B_a) in the memory. If the flows respect this limit, losses will hardly occur, however, if the memory is at a high level of utilization and new traffic burst appears, the memory will be full and segments will be discarded. To accommodate traffic bursts, the EWT may use not all the memory in the calculation of B_a, using:

$$B_a = B_t - U, \tag{1}$$

where B_t is the maximum amount of memory to be used and U indicates the current memory utilization. B_t is always less than the amount of total memory (B) and $B - B_t$ is the amount of memory that will be reserved, serving to accommodate bursts of traffic, if it is necessary.

New flows tend to have their congestion window low, so the feedback provided by the EWT would not have an immediate effect, to mitigate this and save resources when memory is low, the parameter S_t is used to set the starting point to the operation of the method. With this, the EWT will start working only when the memory usage exceeds the S_t threshold. When this occurs, the EWT changes the value of the segment window proportionally to B_a. Considering W_{ewt} as the return value of the EWT, we have:

$$\begin{aligned} W_{ewt} &= \frac{B_a}{B} \cdot W_r, \\ W_r' &= max(W_{ewt}, MSS) \end{aligned} \tag{2}$$

where MSS is the Maximum Segment Size and W_r' is the new value for W_r calculated by the EWT.

Using these equations, the EWT will reduce the value of W_r proportionally to the available router memory, that is, the window values will be reduced as memory usage increases. The EWT was designed to provide a transmission rate control of all TCP flows based on the router memory used, thus minimizing packet drop while maintaining fairness among flows.

The most appropriate place to use EWT is at the gateway (but it can also be used in other routers), where the contest for resources is greater. To use the additional control provided by EWT, no changes to the TCP protocol are necessary, so it can be used with any TCP congestion control algorithm. In TCP, the transmission window satisfies $min(W_c, W_r)$, thus the EWT becomes effective when the congestion window (W_c) is greater than W_r. Related to this, three main cases can occur: (1) *no congestion*: The value of W_c is high and the memory occupancy reaches a threshold. The EWT will reduce W_r, and with it, the protocol will reduce its transmission rate, this is because W_c will probably be higher than W_r; (2) *congestion*: The TCP transmits data using W_c, but if the congestion is heavy, the W_r calculated by EWT may be smaller and the data burst will be reduced; (3) *congestion recovery*: The W_c is used in the transmission, but W_r can also be used if the memory has a considerable level of occupancy. The use of EWT seeks to avoid the occurrence of congestion, for this reason, cases (2) and (3) tend to occur only due to unexpected bursts of traffic or losses in the physical layer.

EWT proved to be effective in low RTT environments, but it was not originally conceived considering the characteristics IoT scenario, where client–server communications present a high RTT, causing delays in the feedbacks (W_r) of the ACK segments that are used to inform the conditions of the memory and consequently the overflow of the network. The situation is aggravated when retransmissions occur due to buffer overflow and, given the properties of the shared transmission medium of wireless networks, generate an even greater impact in reducing the throughput of all hosts that share this same transmission channel. In view of this, this work proposes the EWT-IoT, which uses a more conservative method to define the buffer occupancy level, avoiding as much as possible its saturation in communications that present a high RTT.

3.1. EWT-IoT

With the use of EWT, TCP sources adjust their transmission rate using the information provided by the EWT. The longer it takes for information to reach the sources, the greater the chances that the information will not reflect the current memory state. Therefore, EWT should be used close to TCP sources, such as in a data-center on a local corporate network, where RTT to sources is low.

But there may be cases where it is required to install EWT on the client side, for example, on a Wi-Fi AP in an IoT scenario, which usually presents a higher RTT to TCP sources. In these scenarios, the existence of multiple flows with high RTT could easily exhaust memory, since all sources would be receiving W_r late and adjusting their flows late, given that EWT adjusts W_r with based on the instant state of memory, as exemplified in Fig. 3.

The objective of this modification is to maximize this protection while maintaining fairness and efficiency in the use of the network. These aspects are important in any network environment, but even more critical in an IoT scenario, where the transmission medium is usually shared. Buffer overflow retransmissions unnecessarily occupy the wireless channel, degrading network performance for all devices sharing the same channel, harming the user experience.

In EWT-IoT, a weighted moving average is considered for the amount of *bytes* available (B_a) in the router's memory, being defined by:

$$\overline{B_a} = \begin{cases} (1-g) \cdot \overline{B_a} + g \cdot B_a, & if\ B_a \geq \overline{B_a} \\ B_a, & otherwise \end{cases} \tag{3}$$

where g is a constant in the range [0, 1], and corresponds to an average smoothing factor. In this version, $\overline{B_a}$ is used instead of B_a, seeking to provide the sources with a more conservative estimate of the amount of bytes available. Finally, to avoid losses as much as possible when $\overline{B_a}$ indicates a strong tendency of buffer saturation, at the moment when $\overline{B_a}$ is greater than B_a its value is reduced to B_a.

The modification proposal was conceived with the objective of correcting the problem of the delay in sending the conditions of the network caused by a high RTT, thus allowing the flows of the sources to be adjusted more securely. Using the moving average to estimate B_a helps to provide TCP sources with a more balanced return on the capacity of the device where the method is located, since the instantaneous state of B_a is not used in the calculation for window adjustment of the receiver. The instantaneous value is not adequate when the RTT to the TCP sources is high, because the adjustment made at the instant t will be used by the TCP source in a time $t + \frac{RTT}{2}$, and the traffic from the source will pass through the device in the time $t + RTT$, with this time difference the value of B_a will probably be different when the traffic returns from the sources, and thus the change will not match the capacity of the device at the time of the passage of the TCP segments. For this reason, the moving average becomes essential when the RTT from the device to the TCP sources is high.

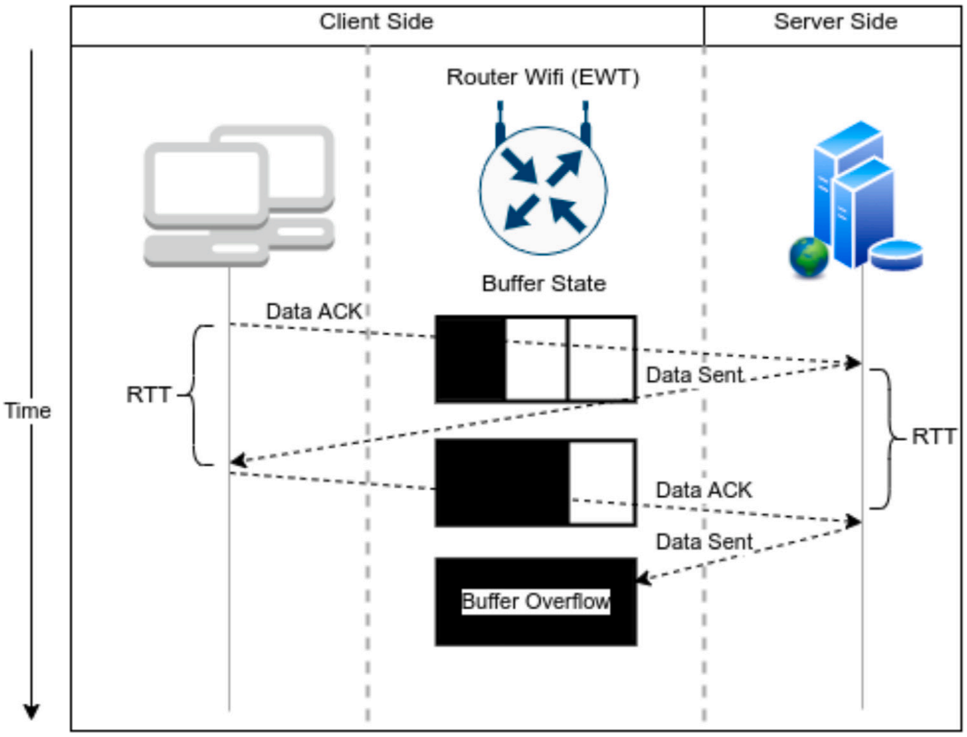


Fig. 3. Illustration of client–server communication in an environment using the EWT method, demonstrating how the source (Server) can receive delayed buffer state returns. Consider that there are several simultaneous flows, which are not represented in the illustration, but which are also stored in the buffer.

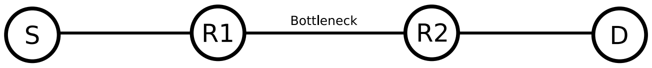


Fig. 4. Network used in the Zhang's model [37].

3.2. Proof of concept

To demonstrate the correct functioning of the EWT-IoT, we considered a discrete-time model to capture the interactions of TCP congestion control algorithm with an AQM mechanism. Our model is based on the model described in Zhang et al. [37], which considers *slow-start* and *congestion-avoidance* algorithms.

The Zhang's model considers the network shown in Fig. 4. It consists of a source (S), two routers (R1 and R2), and a receiver (D). It is assumed that the links S-R1 and R2-D have sufficient capacity, whereas the link R1-R2 has the lowest capacity, being the only *bottleneck* in the network, so AQM is used in R1. Data segments are transmitted from S to D, while the ACKs travel through the reverse path. A single persistent TCP flow is considered. The RTT between S and D and the segment size are constant. ACK segments are never lost. Losses occur only when R1 queue reaches its limit.

The Zhang's model employs window size and average queue size as state variables that are sampled at the end of every RTT period (being k the sample index). In the model, only one variable (the window size, W_k) is used to refer to congestion and transmission window. The window growth is governed by the following equation:

$$W_{k+1} = \begin{cases} min(2W_k,\ ssthresh,\ W_r), & if\ W_k < ssthresh \\ min(W_k + 1,\ W_r), & if\ W_k \geq ssthresh \end{cases} \tag{4}$$

where $ssthresh$ is the threshold value for executing the *slow-start* algorithm, and W_r is the receiver's advertised window size. When the window is smaller than $ssthresh$ the *slow-start* algorithm is executed, where the window is increased exponentially, otherwise the *congestion-avoidance* is executed, generating a linear increase. In both cases, the value of W_r (which is usually 65 segments) is respected as the size limit. Window value reduction occurs only when there are losses, where window value is reduced by half.

The model also deals with the current queue size. This queue size depends on the queue size in the previous period, the current window size, and the number of packets that have left the queue during the previous sampling period, according to the following calculation:

$$\begin{aligned} U_{k+1} &= U_k + W_{k+1} - \frac{C \cdot RTT_{k+1}}{M} \\ U_{k+1} &= U_k + W_{k+1} - \frac{C}{M}(d + \frac{U_k \cdot M}{C}) \\ U_{k+1} &= W_{k+1} - \frac{C \cdot d}{M} \end{aligned} \tag{5}$$

where U_k is the queue utilization, W_k is the window size, C is the link capacity, M is the segment size, and d is the propagation delay. U_k and W_k are expressed in segments, C and M in *bytes*, and RTT and d in seconds.

The pseudocode describing Zhang's model is shown in Fig. 5. Through this model, the behavior of the window W_k and the use of the queue (U_k) can be verified over time. Losses occur when W_k, applied in Eq. (5), results in a usage greater than or equal to the total amount of memory (B). The use of this loss detection mechanism represents the behavior of the *drop-tail*.

To model the EWT-IoT, the Zhang's model was modified as follows. Before the model makes use of the receiver window (W_r) a call to *function* $EWTIoT$ is made, this returns the value of W_r processed by the EWT-IoT. After this process, the value of W_r has already been updated by the EWT-IoT and the model does not need to be changed, as W_r is already used in the calculations that define the window W_k. Note that in the TCP/drop-tail case the value of W_r is fixed; whereas in the TCP/EWT-IoT case, W_r is dynamically adjusted as a function of network conditions. The pseudocode containing the EWT-IoT is shown in Fig. 7. Functions used by EWT-IoT are shown in Fig. 6.

To demonstrate the behavior of the EWT-IoT, the model (Fig. 7) was evaluated considering 100 samples. The same process was also performed for the *drop-tail* model (Fig. 5) and for EWT. The parameters

```
$W_0 \leftarrow 1$
$U_0 \leftarrow 0$
For each $k$:
  if $W_k - \frac{C\cdot d}{M} < B$ then
    if $W_k < ssthresh$ then
      $W_{k+1} \leftarrow min(2W_k,\ ssthresh,\ W_r)$
    else
      $W_{k+1} \leftarrow min(W_k + 1,\ W_r)$
    endif
  else
    $W_{k+1} \leftarrow \frac{1}{2}W_k$
  endif
  $U_{k+1} \leftarrow W_{k+1} - \frac{C\cdot d}{M}$
```

Fig. 5. Pseudocode of the Zhang's model [37].

```
$W_0 \leftarrow 1$
$U_0 \leftarrow 0$
For each $k$:
  if $W_k - \frac{C\cdot d}{M} < B$ then
    $W_r \leftarrow EWTIoT(W_r, W_k)$
    if $W_k < ssthresh$ then
      $W_{k+1} \leftarrow min(2W_k,\ ssthresh,\ W_r)$
    else
      $W_{k+1} \leftarrow min(W_k + 1,\ W_r)$
    endif
  else
    $W_{k+1} \leftarrow \frac{1}{2}W_k$
  endif
```

Fig. 7. Algorithm to evaluate EWT-IoT (adapted from Zhang).

```
function $get\overline{B_a}(B_a)$
  if $B_a > \overline{B_a}$ then
    $\overline{B_a} \leftarrow (1-g)\cdot\overline{B_a} + g\cdot B_a$
    $B_a \leftarrow \overline{B_a}$
  endif
  return $B_a$
end
function $EWTIoT(W_r, W_k)$
  $U \leftarrow W_k - \frac{C\cdot d}{M}$
  $B_a \leftarrow B - U$
  $B_a \leftarrow get\overline{B_a}(B_a)$
  if $U < S_t$ then
    return $W_r$
  endif
  $W_{ewt} \leftarrow \frac{B_a}{B}\cdot W_r$
  return $max(W_{ewt}, 1)$
end
```

Fig. 6. Functions used by EWT-IoT model.

Table 1
Parameter settings.

Parameter	Value
M	500 bytes
C	192 500 bytes
d	0.0228 s
B	30 segments
W_r	65 segments
$ssthresh$	20 segments
S_t	3 segments

used are in Table 1 (based on [37]). The S_t is used to define a threshold on the buffer to EWT-IoT start to work.

In Fig. 8 it can be seen that the window W_k had an exponential growth up to the point $ssthresh$, then there was a linear growth. With the use of EWT-IoT the receiver window was reduced as the queue grew, as can be seen in Fig. 9. In this figure it can also be seen that without segments in the queue, the receiver window (W_r) has not changed, after a few rounds, the queue started to be used, and the threshold (S_t) of the EWT-IoT was exceeded, thus the method began to reduce the value of W_r. The reduction of W_r started to hold the growth of the window W_k when the value of W_r was less than W_k, at this point the system entered into equilibrium, the window W_k started to use W_r and there were no losses in the queue (Fig. 10).

These results show that the convergence of the system is monotonic, with no unwanted oscillations, for both EWT and EWT-IoT (under various values of the constant g), which substantially reduces packet losses and brings a positive impact on other performance metrics (as demonstrated ahead).

4. IoT scenario and simulation issues

In residential scenarios, where IoT devices are used, it is common to have an environment formed by an 802.11 network with several devices sharing the same access point (Fig. 11), this scenario is explored in this section. This topology has already been used in several works, such as [38–42].

The environment is composed of N wireless clients connected to an access point (R1). The AP is connected to router R2 via a 100 Mbps link with 10 ms delay. R2 connects to N servers through 100 Mbps links with random delays in the range $[1, 20]$ ms.

In order to simulate the system, at the beginning of each simulation round, the client i ($i \in \{1, N\}$) establishes a connection with a server i and it starts transferring 5 MB of data. Note that there is only one TCP flow between each client–server pair. Each server starts transmitting after a random wait time of up to eight seconds.

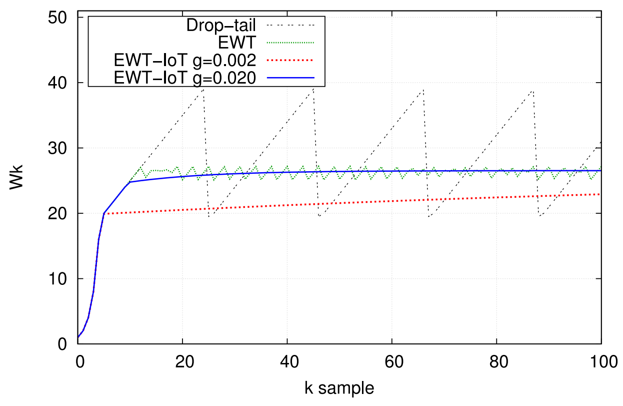


Fig. 8. Register of W_k for k samples of the *drop-tail* and EWT-IoT models.

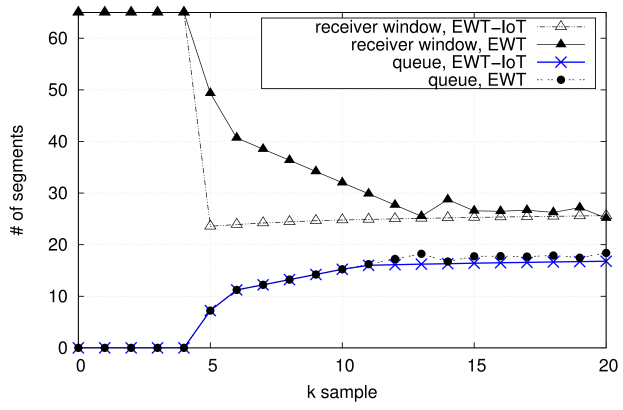


Fig. 9. Joint record of receiver window and queue usage in the EWT-IoT model.

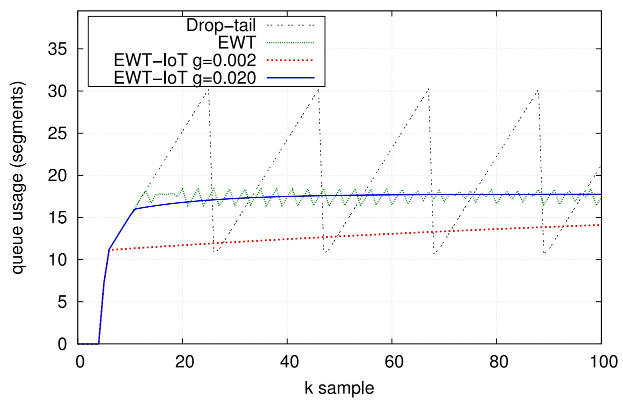


Fig. 10. Queue usage record for k samples of the *drop-tail* and EWT-IoT models.

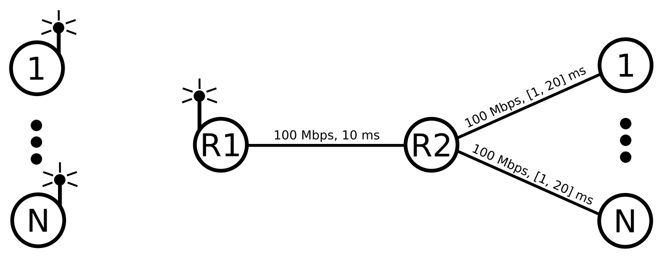


Fig. 11. IoT scenario evaluated.

Table 2
Simulation parameters.

Parameter	Value
Memory size	97 kB
min_{th} (RED)	Memory size/12
max_{th} (RED)	Memory size/4
W_q (RED)	0.002
max_p (RED)	0.02
S_t (EWT-IoT/EWT)	Memory size/3
g (EWT-IoT/EWT)	0.0007
TCP segment size	1458 bytes
TCP	CUBIC
Standard WiFi	802.11 g
Propagation loss model	Friis

The RFC 7928 [43] was considered to establish the congestion level (packet loss rate) in the simulated network. Given the congestion level, it is possible to define the number of TCP flows to be used in the simulations. The RFC defines 0.1% packet loss (in AQM) for mild, 0.5% for medium, and 1% for heavy congestion levels; but it could not be found a number of flows that would produce a percentage of losses as indicated by the RFC. This was due to the delay between the AP and the servers, together with characteristics of the wireless access environment. Thus, the closest ones were considered. Mild level was set to 0.05%, medium to 0.62% and heavy to 1.28%. The number of flows to define the mild level was three, five for medium and seven flows for heavy level.

In the performance evaluation, *drop-tail*, RED, adaptive RED (ARED) and Explicit Window Adaptation (EWA) [44] approaches were used. The methods have been installed on the AP's wireless network interface. The end of each round occurs at the end of all 5MB transfers. The total time of these transfers allows the system to enter a steady-state phase, which guarantees a correct estimate of the analyzed metrics. To calculate 95% confidence intervals, 30 round with different random number seeds were performed [45]. The other simulation parameters are presented in Table 2 (based on [19]).

5. Simulation results

In this section, results obtained using the previously described scenario are presented. The following metrics were considered: TCP efficiency, file transfer time, goodput, goodput fairness, mean loss ratio.

5.1. TCP efficiency

This metric represents the percentage of data (in *bytes*) that was not retransmitted according to [46]. The results are shown in Fig. 12. The *drop-tail*, RED, ARED and EWA approaches had a drop in efficiency as the level of congestion increased. EWT-IoT maintained the TCP efficiency at all three congestion levels.

5.2. File transfer latency

The file transfer latency indicates how many seconds it takes for a file to be successfully transmitted. Here, the average values are shown, calculated from the times recorded by flow. Fig. 13 displays the results. Naturally, with the increase in traffic (characteristic of each level) the transfer time was increasing for all methods. Considering the confidence interval, *drop-tail* and EWA methods presented the same values for mild and medium congestion. EWT-IoT recorded the shortest values at all three levels.

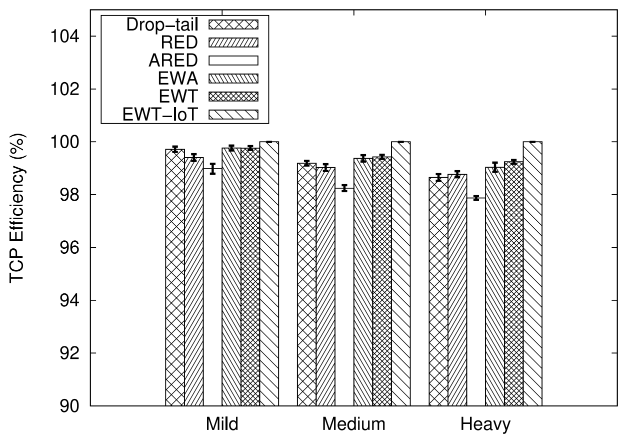


Fig. 12. TCP efficiency vs. Congention level.

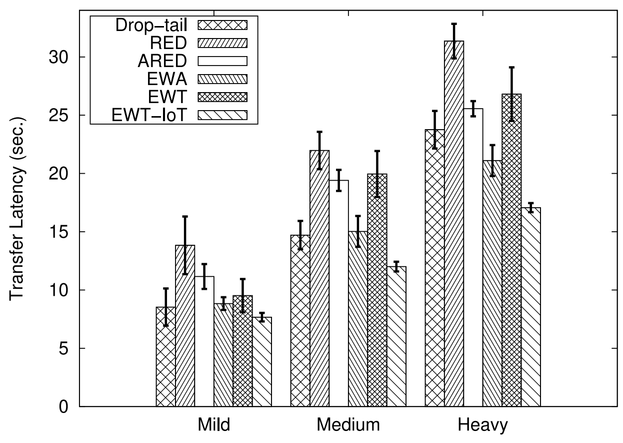


Fig. 13. File transfer latency vs. Congention level.

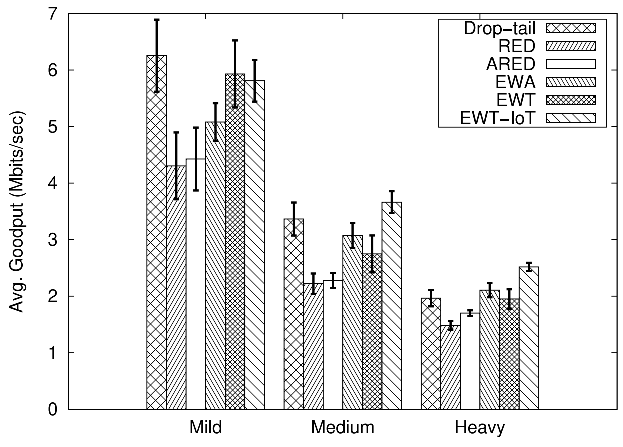


Fig. 14. Goodput vs. Congention level.

5.3. Goodput

The amount of data received by the application, in a period of time, defines the *goodput*. Here, the average value of the *goodput* (Mbits/s) of the flows is presented. The results are shown in Fig. 14. Taking into account the confidence interval, at mild and medium levels the *drop-tail* and EWT-IoT had similar results, with the *drop-tail* having a better performance at the mild level and the EWT-IoT in the medium.

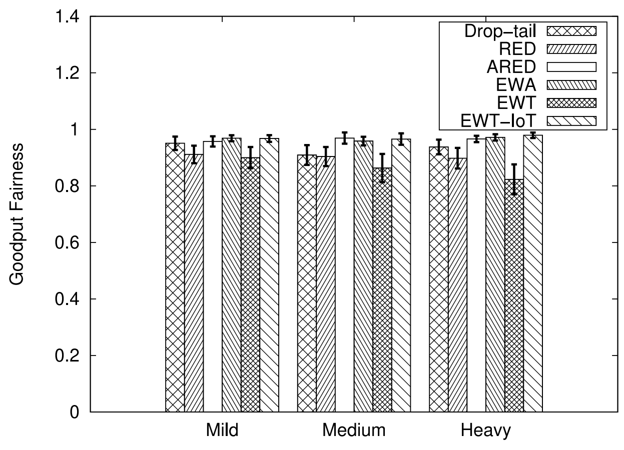


Fig. 15. Goodput fairness vs. Congention level.

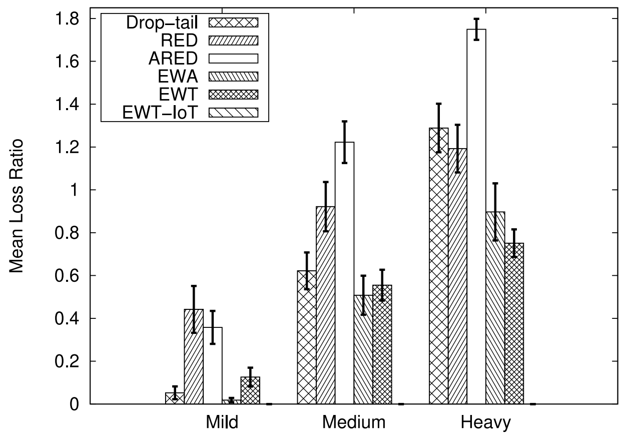


Fig. 16. Mean loss ratio vs. Congention level.

RED/ARED/EWA approaches underperformed at all three levels. At the heavy level, the EWT-IoT reached the highest *goodput*.

5.4. Goodput fairness

The Jain's index [47] was applied to the *goodput* obtained by the N flows, in order to verify the fairness between the flows. The Fig. 15 shows the results. The *drop-tail* and RED approaches registered less fairness. The other methods showed similar results.

5.5. Mean loss ratio

This value represents the average percentage of lost segments during the simulation. The losses considered are those recorded by the network and transport layers. The results are shown in Fig. 16. The *drop-tail*, RED, ARED and EWA methods had an increase in the loss percentage at each level. EWT-IoT did not record losses in the network and transport layers, so it does not appear in the graph.

5.6. Analysis of results

Checking the results presented, it can be observed that the EWT-IoT was effective in congestion control, achieving greater performance than the other approaches, reducing the file transfer time, having a superior *goodput*, maintaining a satisfactory fairness. Likewise, it was able to guarantee optimal efficiency without accounting for losses, regardless of the level of network congestion (see Fig. 12).

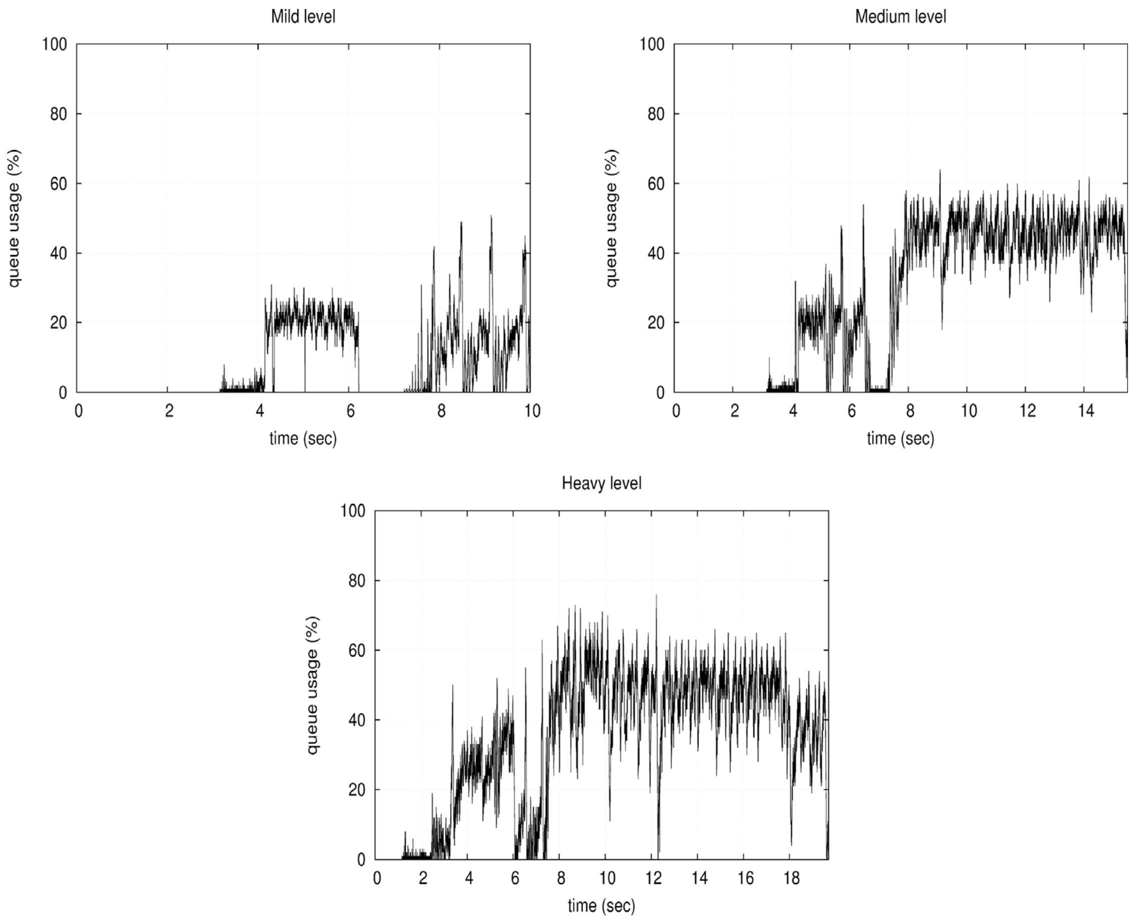


Fig. 17. Queue usage by EWT-IoT for all three levels of congestion.

This was mainly due to the control of the AP's wireless interface queue; Fig. 17 shows the use of this queue to the three levels of congestion. Unlike the original EWT, the EWT-IoT does not directly use the amount of bytes available (B_a), but rather a moving average of B_a ($\overline{B_a}$) and uses B_a when $\overline{B_a} > B_a$, avoiding as much as possible the occurrence of losses in the queue.

The use of moving average was essential to keep track of the AP's wireless interface queue. If the EWT-IoT made use of the instantaneous value of B_a, the W_r to be used by the servers would not reflect on the state of the network at the time of its use, as the time it takes for W_r to reach the server no longer would match the memory state at time $t + delay(AP,\ server)$. In addition, there is a delay from the server to the AP. Therefore, it can be stated that the time of the adjustment of W_r until the segments coming from the server arrive is $t+RTT(AP,\ server)$. Due to the existence of this time, the use of the moving average was essential to carry out congestion control.

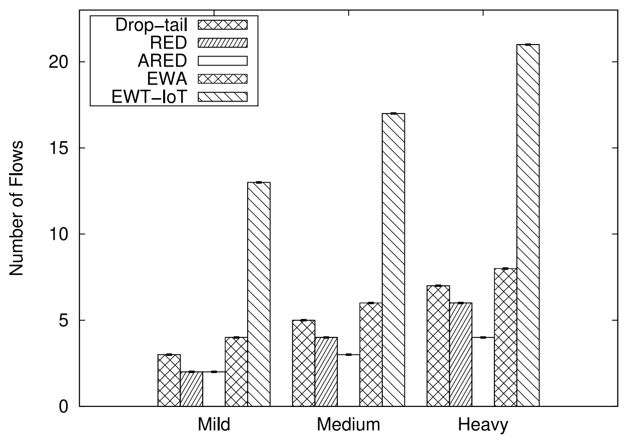


Fig. 18. Number of TCP flows to Meet the RFC 7928.

5.7. Number of TCP flows to meet RFC 7928

In the simulations presented, the congestion levels defined by RFC 7928 were found by increasing the number of TCP flows until a loss rate compatible with each level was reached. As the RFC indicates, no AQM scheme was used in this step. In this section, RED, ARED, EWA, and EWT-IoT approaches were used to find the three congestion levels suggested by the RFC. This was done in order to find the number of flows that each method can serve at each congestion level.

The results are shown in Fig. 18. The method that reached the highest number of flows for each congestion level was EWT-IoT. Compared to its best competitor, it allowed for 65.2% (on average) more flows at each level, and when no AQM scheme was used the percentage was 71.3%. EWT-IoT enabled a greater number of flows by the control generated in the AP's wireless interface queue.

6. Conclusions

This paper proposed a modification in the EWT method that was designed to operate in an environment formed by an 802.11 network with several IoT devices sharing the same access point. The correct functioning of the proposal was proven with the use of a time discrete model. Furthermore, the results obtained via simulation (with the ns-3 simulator) showed the efficiency of the new method. Verifying the results, it was observed that the EWT-IoT proved to be effective in congestion control, achieving greater performance than the other approaches, reducing the file transfer time, having a superior goodput, maintaining a satisfactory fairness, and not accounting losses. When verifying the number of flows to reach each level of RFC 7928, the EWT-IoT allowed the existence of 65.2% more flows at each level, and when no AQM scheme was used the percentage was 71.3%. As future work, the efficiency of the EWT-IoT can be examined in other scenarios, and it can also be implemented in the kernel of an operating system for testing in a production environment.

Declaration of competing interest

All authors declare that they have no conflicts of interest.

Data availability

No data was used for the research described in the article

Acknowledgments

The authors thank the valuable comments of the reviewers.

References

[1] M. Kim, M. Jaseemuddin, Deep reinforcement learning based active queue management for IoT networks, J. Netw. Syst. Manage. 29 (2021) http://dx.doi.org/10.1007/s10922-021-09603-x.

[2] P. Datta, B. Sharma, A survey on IoT architectures, protocols, security and smart city based applications, in: 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2017, pp. 1–5, http://dx.doi.org/10.1109/ICCCNT.2017.8203943.

[3] S. Chaudhary, R. Johari, R. Bhatia, K. Gupta, A. Bhatnagar, CRAIoT: Concept, review and application(s) of IoT, in: 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019, pp. 1–4, http://dx.doi.org/10.1109/{IoT}-SIU.2019.8777467.

[4] S.R. Pokhrel, C. Williamson, Modeling compound TCP over WiFi for IoT, IEEE/ACM Trans. Netw. 26 (2) (2018) 864–878, http://dx.doi.org/10.1109/TNET.2018.2806352.

[5] M.J.A. Jude, v. Diniesh, M. Shivaranjani, Throughput stability and flow fairness enhancement of TCP traffic in multi-hop wireless networks, Wirel. Netw. 26 (2020) http://dx.doi.org/10.1007/s11276-020-02357-5.

[6] L. Brakmo, L. Peterson, TCP Vegas: end to end congestion avoidance on a global Internet, Sel. Areas Commun. 13 (8) (1995) 1465–1480, http://dx.doi.org/10.1109/49.464716.

[7] M. Gerla, M. Sanadidi, R. Wang, A. Zanella, C. Casetti, S. Mascolo, TCP Westwood: congestion window control using bandwidth estimation, in: Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE, Vol. 3, IEEE Press, San Antonio, USA, 2001, pp. 1698–1702, http://dx.doi.org/10.1109/GLOCOM.2001.965869.

[8] E. Blanton, D.V. Paxson, M. Allman, TCP Congestion Control, RFC 5681, 2009, http://dx.doi.org/10.17487/RFC5681, URL https://rfc-editor.org/rfc/rfc5681.txt.

[9] T. Henderson, S. Floyd, The NewReno Modification to TCP's Fast Recovery Algorithm, RFC 2582, 1999, http://dx.doi.org/10.17487/RFC2582, URL https://rfc-editor.org/rfc/rfc2582.txt.

[10] A. Gurtov, T. Henderson, S. Floyd, The NewReno Modification to TCP's Fast Recovery Algorithm, RFC 3782, 2004, http://dx.doi.org/10.17487/RFC3782, URL https://rfc-editor.org/rfc/rfc3782.txt.

[11] L. Xu, K. Harfoush, I. Rhee, Binary increase congestion control (BIC) for fast long-distance networks, in: IEEE INFOCOM 2004, Vol. 4, 2004, pp. 2514–2524, http://dx.doi.org/10.1109/INFCOM.2004.1354672.

[12] S. Ha, I. Rhee, L. Xu, CUBIC: A new tcp-friendly high-speed TCP variant, SIGOPS Oper. Syst. Rev. 42 (5) (2008) 64–74, http://dx.doi.org/10.1145/1400097.1400105, URL http://doi.acm.org/10.1145/1400097.1400105.

[13] L.P. Verma, M. Kumar, An IoT based congestion control algorithm, Internet Things 9 (2020) 100157, http://dx.doi.org/10.1016/j.iot.2019.100157.

[14] M. Talau, M. Fonseca, E.C.G. Wille, Early window tailoring: A new approach to increase the number of TCP connections served, J. Comput. Netw. Commun. 2019 (2019) http://dx.doi.org/10.1155/2019/2187543.

[15] C. Gomez, A. Arcia-Moret, J. Crowcroft, TCP in the internet of things: From ostracism to prominence, IEEE Internet Comput. 22 (1) (2018) 29–41, http://dx.doi.org/10.1109/MIC.2018.112102200.

[16] E. Ancillotti, S. Bolettieri, R. Bruno, RTT-based congestion control for the Internet of Things, in: International Conference on Wired/Wireless Internet Communication, Springer, 2018, pp. 3–15.

[17] M. Talau, M. Fonseca, A. Munaretto, E.C.G. Wille, Early congestion control: A new approach to improve the performance of TCP in ad hoc networks, in: 2016 7th International Conference on the Network of the Future (NOF), 2016, pp. 1–6, http://dx.doi.org/10.1109/NOF.2016.7810143.

[18] T. Saedi, H. El-Ocla, TCP CERL+: revisiting TCP congestion control in wireless networks with random loss, Wirel. Netw. 27 (2021) http://dx.doi.org/10.1007/s11276-020-02459-0.

[19] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Trans. Netw. (TON) 1 (4) (1993) 397–413, http://dx.doi.org/10.1109/90.251892.

[20] A.A. Abu-Shareha, Enhanced random early detection using responsive congestion indicators, Int. J. Adv. Comput. Sci. Appl. 10 (3) (2019) http://dx.doi.org/10.14569/IJACSA.2019.0100347.

[21] C.E. Palazzi, S. Ferretti, M. Roccetti, G. Pau, M. Gerla, What's in that magic box? The home entertainment center's special protocol potion, revealed, IEEE Trans. Consum. Electron. 52 (4) (2006) 1280–1288, http://dx.doi.org/10.1109/TCE.2006.273146.

[22] A. Bujari, M. Massaro, C.E. Palazzi, Vegas over access point: Making room for thin client game systems in a wireless home, IEEE Trans. Circuits Syst. Video Technol. 25 (12) (2015) 2002–2012, http://dx.doi.org/10.1109/TCSVT.2015.2450332.

[23] S.Z. Hussain, S. Parween, Comparative study of TCP congestion control algorithm in IoT, in: 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), 2021, pp. 1428–1431, http://dx.doi.org/10.1109/ICAC3N53548.2021.9725642.

[24] S. Floyd, Recommendations on using the gentle variant of RED, 2000, URL http://www.aciri.org/floyd/red/gentle.html.

[25] S. Floyd, R. Gummadi, S. Shenker, Adaptive RED: An algorithm for increasing the robustness of RED's active queue management, 2001.

[26] J. Aweya, M. Ouellette, D. Montuno, A control theoretic approach to active queue management, Comput. Netw. 36 (2001) 203–235, http://dx.doi.org/10.1016/S1389-1286(00)00206-1.

[27] W.-C. Feng, D. Kandlurz, D. Sahaz, K. Shin, BLUE: A new class of active queue management algorithms, 2000.

[28] J. Koo, B. Song, K. Chung, H. Lee, H. Kahng, MRED: a new approach to random early detection, in: Proceedings 15th International Conference on Information Networking, 2001, pp. 347–352, http://dx.doi.org/10.1109/ICOIN.2001.905450.

[29] B. Abbasov, S. Korukoğlu, Effective RED: An algorithm to improve RED's performance by reducing packet loss rate, J. Netw. Comput. Appl. 32 (2009) 703–709, http://dx.doi.org/10.1016/j.jnca.2008.07.001.

[30] C. Long, B. Zhao, J. Yang, The Yellow active queue management algorithm, Comput. Netw. 47 (2005) 525–550, http://dx.doi.org/10.1016/j.comnet.2004.09.006.

[31] C.A. Grazia, N. Patriciello, M. Klapez, M. Casoni, Which AQM fits IoT better? in: IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI), 2017, pp. 1–6, http://dx.doi.org/10.1109/RTSI.2017.8065903.

[32] K. Ramakrishnan, S. Floyd, D. Black, RFC 3168: The addition of explicit congestion notification (ECN) to IP, 2001, URL https://rfc-editor.org/rfc/rfc3168.txt.

[33] A. Bujari, A. Marin, C.E. Palazzi, S. Rossi, Smart-RED: A novel congestion control mechanism for high throughput and low queuing delay., Wirel. Commun. Mob. Comput. (2019) 1–10.

[34] C.E. Palazzi, N. Stievano, M. Roccetti, A smart access point solution for heterogeneous flows, in: 2009 International Conference on Ultra Modern Telecommunications Workshops, 2009, pp. 1–7, http://dx.doi.org/10.1109/ICUMT.2009.5345522.

[35] M. Casoni, C.A. Grazia, M. Klapez, N. Patriciello, How to avoid TCP congestion without dropping packets: An effective AQM called PINK, Comput. Commun. 103 (2017) 49–60, http://dx.doi.org/10.1016/j.comcom.2017.02.010, URL http://www.sciencedirect.com/science/article/pii/S0140366417301913.

[36] DARPA, Transmission Control Protocol, RFC 793, 1981, http://dx.doi.org/10.17487/RFC0793, URL https://rfc-editor.org/rfc/rfc793.txt.

[37] H. Zhang, M. Liu, V. Vukadinović, L. Trajković, Modeling TCP/RED: a dynamical approach, in: Complex Dynamics in Communication Networks, Springer, 2006, pp. 251–278.

[38] J. Ha, C. Choi, WLC29-5: TCP fairness for uplink and downlink flows in WLANs, in: IEEE Globecom 2006, 2006, pp. 1–5.

[39] A. Bujari, M. Massaro, C.E. Palazzi, Vegas over access point: Making room for thin client game systems in a wireless home, IEEE Trans. Circuits Syst. Video Technol. 25 (12) (2015) 2002–2012.

[40] C.E. Palazzi, N. Stievano, M. Roccetti, G. Marfia, Ensuring fair coexistence of multimedia applications in a wireless home, in: 2009 2nd IFIP Wireless Days (WD), 2009, pp. 1–5.

[41] J. Huang, J. Wang, J. Ye, Buffer allocation management for improving TCP fairness in IEEE 802.11 WLANs, in: 2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM), 2010, pp. 1–4.

[42] M. Seyedzadegan, M. Othman, S. Subramaniam, Z. Zukarnain, The TCP fairness in WLAN: A review, in: 2007 IEEE International Conference on Telecommunications and Malaysia International Conference on Communications, 2007, pp. 644–648.

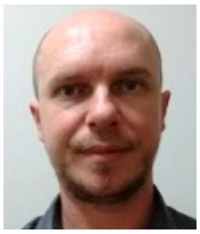
[43] N. Kuhn, P. Natarajan, N. Khademi, D. Ros, Characterization guidelines for active queue management (aqm), 2016, http://dx.doi.org/10.17487/RFC7928.

[44] L. Kalampoukas, A. Varma, K.K. Ramakrishnan, Explicit window adaptation: A method to enhance TCP performance, IEEE/ACM Trans. Netw. 10 (3) (2002) 338–350.

[45] G. Corder, D. Foreman, Nonparametric Statistics: A Step-By-Step Approach, Wiley, 2014.

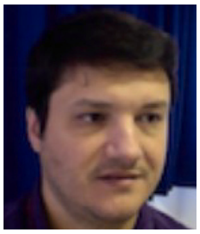
[46] R. Schrage, G. Forget, R. Geib, B. Constantine, Framework for TCP Throughput Testing, RFC 6349, 2011, http://dx.doi.org/10.17487/RFC6349, URL https://rfc-editor.org/rfc/rfc6349.txt.

[47] R. Jain, D. Chiu, W. Hawe, A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems, DEC Research Report TR-301, Digital Equipment Corporation, Maynard, MA, USA, 1984, p. 38, URL ftp://ftp.netlab.ohio-state.edu/pub/jain/papers/fairness.htm.



Marcos Talau received his Ph.D. in Computer Networks from Federal University of Technology - Paraná (UTFPR), Brazil, in 2020. Also, since 2012 he has been an adjunct professor at this university. He obtained his M.Sc. in Computer Networks at UTFPR in 2012. Mr. Talau has been working with research in TCP since 2009. He is a Debian Developer and also works with network simulators, responsible for including RED in the ns-3 in 2011.

Thiago Alexandre Herek was born in Londrina (PR), Brazil. He currently works as an IT Analyst and is also a Ph.D. student conducting research on TCP. He received his M.Sc. in Computer Networks from the Federal Technological University of Paraná - UTFPR (Curitiba, Brazil) in 2018, where he conducted research on Ad Hoc Networks and Vehicular Opportunistic Networks (VANETs).



Mauro Fonseca is currently Associate Professor at UTFPR. He was Full Professor at PUC-PR for 18 years. He received his Ph.D. in Networks from Université Pierre et Marie Curie (Paris 6). Graduated from the PUC Paraná Brazil in 1994 with a Bachelor's Degree in Computers Engineering (specializing in Control), and gained a M.Sc. in Networks and distributed system analysis from the CEFET Paraná Brazil in 1997. He has already been leader of several projects and participated on national and international projects. His research interests focus on service management frameworks, architectures for networks and wireless networks.

Emilio Carlos Gomes Wille was born in Lapa (PR) - Brazil. He received his degree in Electrical Engineering in February 1989, and a M.Sc. in Electrical Engineering in July 1991, both from Federal University of Technology of Paraná - UTFPR (Curitiba - Brazil). He received his Ph.D. degree in Electronic and Telecommunications Engineering from Politecnico di Torino (Italy) in February 2004. During his stay in Italy he was supported by a CAPES Foundation scholarship from the Ministry of Education of Brazil. He is a Full Professor at the UTFPR, and since October 1991 he is with the Electronics Department. His teaching duties at UTFPR comprise graduate and undergraduate-level courses on electronic and telecommunication theory. He has co-authored several papers presented in national and international conferences, all of them in the area of telecommunication systems and networks. His research interests include communication networks, wireless systems, stochastic processes, computer simulation and optimization algorithms.