Internet of Things 9 (2020) 100157

Contents lists available at ScienceDirect

Internet of Things

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

Research article

An IoT based Congestion Control Algorithm

Lal Pratap Verma [a], Mahesh Kumar [b,*]

[a] Department of Computer Science and Engineering, Moradabad Institute of Technology, Moradabad, Uttar Pradesh, India
[b] Department of Computer Science and Engineering, Jaypee University of Engineering and Technology, Guna, Madhya Pradesh, India

ARTICLE INFO

Article history:
Received 17 June 2019
Revised 12 November 2019
Accepted 24 December 2019
Available online 28 December 2019

Keywords:
IoT
TCP
Fairness
Congestion control
RTT

ABSTRACT

Internet of Things provides a common platform to connect heterogeneous devices over the internet. As a result, the number of connected devices increases over the Internet exponentially. Therefore, congestion over the Internet also increases. TCP is a reliable transport layer protocol for wired and wireless networks that control the transmission rate according to network congestion. When network conditions change abruptly, the network delay and bandwidth also change accordingly. Therefore, it is very important to adjust the transmission rate according to the network status. This paper introduces a new congestion control policy to adapt the transmission rate quickly whenever the available bandwidth and delay changes. The proposed approach maintains a steady-state to reduce packet drop and yield improved throughput. This paper also proposes adaptive techniques to maintain fairness with widely deployed TCP Cubic. The experimental results show that the proposed TCP achieves better performance in terms of throughput and inter-protocol fairness.

1. Introduction

Internet of Things (IoT) is the global network platform which offers interconnection between dissimilar devices having the ability to send data over the network. These interconnected devices can be sensor node use for any type of monitoring to automate the home, industry, healthcare, environmental data collection and prediction, etc. [1,2]. Ericsson conducted a survey and suggested that the number of connected devices over the Internet was 16 billion in 2016 and this number will be reached up to 29 billion in 2022 [3]. Another survey forecasts that the number of connected devices over the Internet will increase to 20.4billion by 2020 [4]. Therefore, as a number of connected devices increases over the Internet, it requires more research attention to deal with upcoming challenges in the IoT domain.

Major applications of IoT are smart homes [5], smart transportation system [6,7], industrial monitoring systems [8], healthcare monitoring systems [9], smart grids [10], smart cities [11], and environment monitoring systems, etc. To implement the IoT, it requires a set of protocols to control communication over the Internet. There are many types of application protocols available for IoT environment to provide data transmission between different devices. The XMPP (Extensible Messaging and Presence Protocol) [12], MQTT (Message Queuing Telemetry Transport) [13], AMQP (Advanced Message Queuing Protocol) [14] are IoT application protocols which use Transmission Control Protocol (TCP) [15] as a transport layer protocol to offer data transmission. However, CoAP (Constrained Application Protocol) [16] is another IoT application protocol designed to run over the User Datagram Protocol (UDP) to provide a fast connection between devices. Fig. 1 shows the IoT

* Corresponding author.
E-mail addresses: er.lpverma1986@gmail.com (L.P. Verma), mahesh.chahar@gmail.com (M. Kumar).

https://doi.org/10.1016/j.iot.2019.100157
2542-6605/

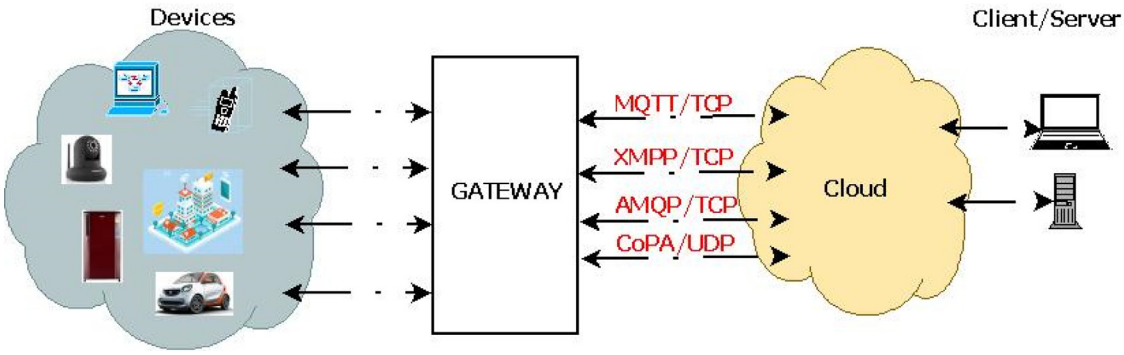


Fig. 1. IoT Application Protocols.

application protocols use to offer data transmission. It shows that TCP is an integral part of IoT application protocol. TCP provides reliable, host-to-host, reliable, connection-oriented service.

The preliminary objective of TCP is to utilize available bandwidth and maintaining fairness with competing flows. Therefore, all TCP variants (loss-based and delay-based TCP) have different designs and their own policy to adjust the transmission rate. As number of connected devices increases over the Internet, network congestion also increases. Therefore, TCP needs some modification according to IoT device requirements to start connection with marginally good capacity, adjust the transmission rate as congestion increases and make transmission rate stable when network resources are stable.

1.1. Motivation

In IoT, different devices have their own requirements like communication speed, delay, and reliability. However, most of the IoT devices are having battery-powered, with limited computation capabilities [2]. Therefore, the data transmission rate is an important factor need to take care during communication. On the other hand, due to the exponential growth of smart devices over the Internet, network congestion is a critical issue [17] need to handle to ensure the proper and timely delivery of data. In traditional networks, application data is generally transported by the

TCP, which provide congestion control and transmission rate adaptation policy. After analysis of the various TCP variants [15,18–36], we conclude that loss-based TCP utilizes the available bandwidth appropriately, but not suitable for applications which work in low bandwidth environments. However, delay-based TCP attempts to reduce the packet losses and retransmissions, but not able to utilize the available bandwidth. Both delay and loss based TCP variants are not able to provide appropriate adaptation of variable requirements of IoT devices. Therefore, we need a new congestion control policy which works in IoT environment to adjust the transmission rate according to the requirement of devices.

Initialization of the *cwnd* and *ssthresh* is another important issue for TCP. This factor is also dependent on network resources like bandwidth and path delay. Many authors suggested the initial *cwnd* value start with some constant value [37–42], but such policies do not include the network characteristics like bandwidth and delay. Therefore, we need a proper *cwnd* initialization method which includes network available bandwidth as a factor to decide the initial congestion window value.

The inter-protocol fairness is very important when more than one TCP variant running on the same network. TCP Cubic is widely implemented TCP because most of Linux and Android operating system uses the TCP Cubic as transport layer protocol. Thus, the fairness with TCP Cubic is another challenge for new TCP variants. Both types of TCP (delay and loss based) variants also suffer from the inter-fairness problem.

1.2. Highlight

To alleviate the above discussed problem, this paper makes the following contribution:

- This paper propose a new congestion control approach which works in different modes according to network congestion. This approach adjusts the data transfer rate according to the requirement of IoT devices.
- This paper also propose a new congestion window initialization approach. This approach decides the initial congestion window value according to the available bandwidth of the path.
- The proposed approach also included the adaptive property of TCP to utilizes the available bandwidth according to the demand of the application and maintains the inter-protocol fairness with TCP Cubic flow.

The rest of the paper is organized as follows: Section 2 presents the literature of IOT and TCP congestion control policies while section 3 propose a new congestion control algorithm based on combination of loss-based and delay-based TCP. In section 4, we present the performance of proposed method, and section 5 concludes the overall performance of the proposed approach.

2. Related work

IoT is a growing technology which provides a common platform to connect heterogeneous devices to offers smart services like smart monitoring and control, information collaboration, sharing, and analysis, etc. [43]. Therefore, a different type of IoT protocol architecture has been designed by various authors to fulfill the need of IoT devices [44]. There are many types of application protocols available for IoT environment to provide data transmission between different devices. The XMPP (Extensible Messaging and Presence Protocol) [12] is a communications protocols for message-oriented middleware based on Extensible Markup Language (XML).

Initially, XMPP was developed for real-time instant messaging by an open-source community. Further, The XMPP extension optimizes the older version and support signaling for VoIP, file transfer, gaming, and IoT applications. The XMPP uses TCP as a transport layer protocol to offer process-to-process communication between devices over the Internet. MQTT (Message Queuing Telemetry Transport) [13] is another application protocol for IoT environment which uses TCP to offer process-to-process, reliable, congestion-controlled, and connection-oriented communication services. MQTT has been invented by Dr. Andy Stanford-Clark (IBM), and Arlen Nipper (Eurotech), in 1999. Now, MQTT is a lightweight, open-source protocol suitable for Machine-to-Machine communication. AMQP (Advanced Message Queuing Protocol) [14] is also an application protocol for IoT environment which uses TCP [15] as a transport layer protocol to offer data transmission between different devices. Therefore, TCP has an important role in the growth of IoT [45]. In recent years, multipath communication also gaining more research attention due to its load distribution and transmission rate adaptation policy [46–49]. Therefore, this technology will also play an important role in the growth of IoT [53–56] when the number of connected devices will become more.TCP is a transport layer protocol that provides process-to-process communication over the Internet. Rapidly growing IoT devices generate more traffic and cause congestion on the Internet. TCP has a congestion control policy to adapt the transmission rate of the path according to available bandwidth. TCP's congestion control mechanism uses additive increase multiplicative decrease (AIMD) policy [18]. It uses *cwnd* and *ssthresh* state variables to adjust the transmission rates, according to network congestion. But such type of congestion control policy is not suitable to deal with fast occurring network congestion, because of its higher reaction time. Therefore, Kelly [25] proposed a Scalable TCP (STCP) as a replacement of AIMD policy and introduces a multiplicative increase multiplicative decrease (MIMD) policy. This policy is more aggressive in nature as compared to AIMD. Therefore, this policy is also not suitable for heterogeneous devices communication. On behalf of AIMD and MIMD policies, TCP variants can be classified into two parts:

- Loss-based Congestion Control Algorithm
- Delay-based Congestion Control Algorithm

2.1. Loss-based Congestion Control Algorithm

This type of congestion control algorithms uses packet loss as an indicator of congestion. The loss-based congestion control algorithm for the TCP protocol was initially proposed in [15,18]; based on this algorithm, the fast retransmission and fast recovery technique has been included in TCP Reno [19]. Floyd et al. [20,21] recognized various problems in TCP Reno and proposed a new technique called TCP New Reno. There are other TCP variants available that modify TCP Reno and New Reno, i.e., TCP SACK [22] and TCP FACK [23].

Floyd [24] recognized the efficiency problem of high-speed networks and proposed High Speed TCP (HS-TCP) for the same. HS-TCP achieves high bandwidth utilization and has an intra-fairness quality, but it suffers from the fairness of round trip time (RTT) with different flows. A variant of HS-TCP was proposed by Kelly [25] named Scalable TCP (STCP) to solve the effectiveness problem in high-speed networks. STCP grows *cwnd* very quickly, but it experiences a critical problem of inter-protocol fairness. Other high-speed TCP variants are HTCP [26], BIC-TCP [27], TCP-CUBIC [28] and CUBIC-FIT [29].

2.2. Delay-based Congestion Control Algorithm

Such congestion control algorithms use network delay as a congestion detection factor. When delay increases, congestion increases, and when delay decreases, network congestion decreases. Brakmo and Peterson [30] proposed a delay-based congestion control algorithm called TCP Vegas. It adjusts the *cwnd* according to the difference between the expected and the actual rate. The authors claim packet loss reduction, but it suffers from low bandwidth utilization in high-speed networks. Hasegawa et al. [31] recognized this serious unfairness problem of TCP Vegas and proposed a new TCP variant called Vegas+.

Mascolo et al. [32] proposed TCP Westwood, which estimates the available bandwidth based on the acknowledgement arrival rate. Another revised version of Westwood has been introduced [33] due to overestimation of the bandwidth. Both versions suffer from the inter-protocol fairness problem. Wei et al. [34] introduced a queuing delay-based congestion control algorithm known as FAST-TCP. Apart from the advantages, it suffers from the fairness problem. TCP-FIT [35] is another congestion control algorithm that uses both loss-based and delay-based techniques to control *cwnd*. It improves the throughput and fairness with respect to TCP-Reno, CUBIC and FAST-TCP. Wang et al. [36] took advantage of FAST-TCP and TCP-FIT and proposed a new congestion control algorithm called FAST-FIT. It uses the FAST-TCP growth function to maintain the data flow and a TCP-FIT technique to adjust the *cwnd*. It shows better results in terms of inter-protocol fairness (with TCP-Reno) and high bandwidth utilization in wired and wireless environments.

3. Proposed work

IoT is a platform where different types of devices exchanges different types of data with each other. As the number of IoT devices increases over the Internet, network congestion also increases. Ericsson [3] forecast that the number of connected devices over the Internet was 16 billion in 2016 and this number will be reached up to 29 billion in 2022. Therefore, network congestion will become a big challenge in the upcoming years.

RTT is an important factor in TCP which plays a major role while adjusting the transmission rate in proactive protocols. When the congestion level increases, RTT also increases correspondingly, as a result, the link utilization decreases. The proactive TCP uses a delay-based approach to adjust the transmission rate and reduce the number of packet drops, but it is not able to fully utilize the available bandwidth. Reactive TCP adjusts the transmission rate when packet loss is detected and may fully utilize the available link bandwidth, but it has no control over packet drops. Therefore, both the type of protocol is not suitable for IoT environment. Thus, this paper proposes a combination of proactive and reactive TCP policies to adjust the transmission rate according to the device requirement. The proposed approach will maintain a steady-state and improve the available bandwidth utilization. Thus, this paper introduces a new congestion control algorithm that works in two modes:

- Reactive mode
- Proactive mode

The above-mentioned modes are used on the basis of the congestion detection factor β. The value of β is dependent on RTT_{min}, RTT_{max} and RTT have been discussed in the Congestion Control sub-section. In our proposed schema, TCP uses the following methods to adjust transmission rate of path:

- Congestion window and Slow-start threshold initialization
- Modified Slow-start and Congestion avoidance
- Congestion Control

3.1. Congestion window and slow start threshold initialization

The initial value of the congestion window and slow start threshold is an important factor which specifies that at which rate TCP will start communication. The value of these factors depends on available network resources like bandwidth. Many authors suggested the initial *cwnd* value start with some constant value [37–42], but such policies do not include the network characteristic like bandwidth. Therefore, this paper proposes a new *cwnd* and *ssthresh* initialization algorithm to decide these values according to the available bandwidth of the path.

In a traditional network, the received packet can be handled by a packet-processing system known as de-multiplexing [49]. As a packet arrives, protocol access the contents of Type Field available in the received packet header and decide how to process it. In this approach, a packet passes through one layer at a time. Therefore, de-multiplexing is required for each layer. However, the modern network system has a completely different opinion on the packet processing system. The modern network system classification [50,51] technique (also known as cross-layering) is used as a replacement of de-multiplexing. Such type of technique being used by companies such as Cisco, Juniper, Intel and Netronome [52]. Therefore, this paper utilizes the cross-layering approach to estimate the available bandwidth of the path.

Initially, TCP sends a connection request to the destination by using an SYN packet. This packet reaches the destination through intermediate routers, and the response to the SYN packet also comes back to the source using the same path. In our bandwidth estimation process, each router updates the available bandwidth information in the SYN-ACK packet during the processing of it by using classification [50,51] approach. However, each router has to estimate the available bandwidth of path after reception of the SYN-ACK packet. Therefore, each router calculates the available bandwidth of a specific path by using Algorithm 1. In this algorithm, the router track each link information in terms of the number of packet and size of packet received from the link when it receives an SYN packet. Based on the above available information, the router calculates the lode of that particular link. The router also has the capacity of each link. Therefore, the router subtracts the link capacity from link lode to get available bandwidth. This approach uses the TCP header optional field (32 bit) to store the available bandwidth information.

The TCP source calculates the minimum available bandwidth of the path using SYN-ACK. The algorithm for initial bandwidth estimation is shown in Algorithm 1.

After receiving the SYN-ACK packet, the TCP source learns about the minimum available bandwidth on the path, which plays an important role in further transmission. Thus, the source initializes the *cwnd* and *ssthresh* using the following expressions

$$cwnd_i = \frac{BW}{\alpha} \tag{1}$$

$$ssthresh = \frac{BW}{\alpha} \tag{2}$$

Where BW is the available bandwidth of the path and α is a sensitivity factor used for deciding the initial *cwnd* and *ssthresh* size. The range of α may vary from 1 to the maximum available bandwidth. To find the optimal value of α, we have

Algorithm 1 Initial bandwidth estimation and initialization of cwnd and ssthresh

```
// The bandwidth estimation process at each router
Begin
  If SYN + ACK received
Then
time_interval = Difference between current time & packet arrival time
Load = (Received_packet × Packet_size × 8)/time_interval
Link_available_bandwidth = Link_capacity − Load
        If available_bandwidth < Link_available_bandwidth
        Then
                available_bandwidth = Link_available_bandwidth
                forward the packet
        End If
End If
End
//Initialization of cwnd and ssthresh at source
Begin
cwnd = BW/α                    //BW is available_bandwidth
ssthresh = BW/α
End
```

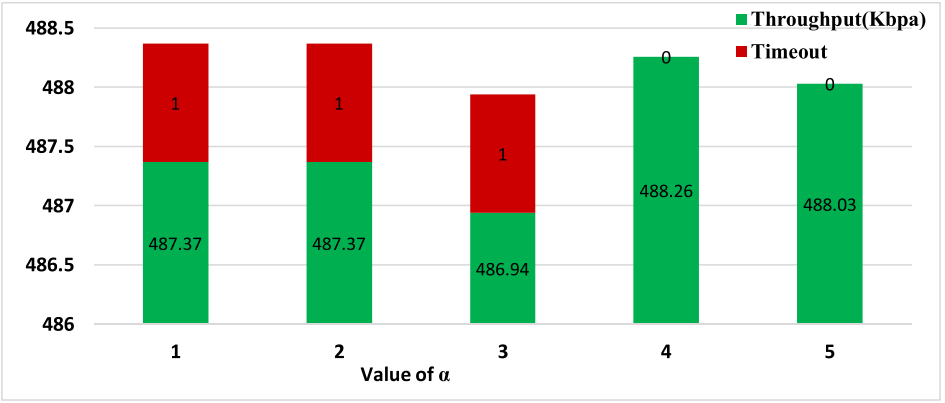


Fig. 2. Assessment of optimal value of α.

conducted 10 number of simulations in network simulator-2. The dumbbell topology has been used to verify the value of α. This topology has 10 source nodes, 10 destination nodes, and 2 routers. All nodes are connected with links of different bandwidth and the simulation time of this setup is 200 second. The result of this simulation is shown in Fig. 2. It shows that as value of α increase the throughput and timeout of TCP also changes. However, it shows the best result on α=4.

3.2. Modified slow-start and Congestion avoidance

The the proposed congestion control method is based on TCP New Reno [8], which is widely used over the Internet. TCP New Reno has slow-start, congestion avoidance, fast retransmission, and fast recovery mechanisms to achieve the desired throughput. In this section, we describe the congestion window growth policy to adapt the abrupt data rate of the path. First, the proposed approach modifies the congestion window growth policy in terms of Modified Slow-Start which is shown by Eq. (3) and (4). As we discussed earlier that IoT environment has the heterogeneous types of devices having a different requirements. Initially, when TCP starts data transmission, it is in the slow-start phase. In this phase, TCP grows the *cwnd* in exponential fission. However, when we talked about high-speed network, this growth policy takes more time to reach up to *ssthresh* value. Because we are initializing the *cwnd* and *ssthresh* according to the available bandwidth of the path. Therefore, the proposed approach uses the *ssthresh* and delay of the path to decide the growth of *cwnd*. The proposed approach uses x as a *cwnd* increasing factor in the slow-start phase. The value of x has been estimated by using the product of current *ssthresh* and RTT. The product of *ssthresh* and RTT is treated as a constant. If the value of x is greater than 1, *cwnd* increases with the x-factor. Else it increases the *cwnd* according to TCP New Reno policy. The algorithm for the modified slow start and congestion avoidance is shown in Algorithm 2.

$$cwnd_{i+1} = \begin{cases} cwnd_i + x \ if \ x > 1 \\ cwnd_i + 1 \ if \ x \leq 1 \end{cases} \quad \text{Modified Slow} - \text{Start} \tag{3}$$

$$cwnd_{i+1} = cwnd_i + \frac{1}{cwnd_i} \quad \text{Congestion avoidance} \tag{4}$$

Algorithm 2 Modified Slow-Start and Congestion Avoidance

```
1. //At source node
2. Begin
3.     If cwnd_i ≤ ssthresh_i
4.     Then
5.         x = (ssthresh_i × RTT_i)
6.         Ifx > 1
7.         Then
8.             cwnd_{i+1} = cwnd_i + x
9.         Else
10.            cwnd_{i+1} = cwnd_i + 1
11.        End If
12.    Else
13.        cwnd_{i+1} = cwnd_i + 1/cwnd_i
14.    End If
15. End
```

3.3. Congestion Control

When a router receives data at a higher rate than the forwarding capacity, it starts buffering data in fixed-size queues. When the queue is full, incoming packets get dropped. Most TCP variants reduce the *cwnd* and *ssthresh* whenever packet loss is detected. This paper introduces a new congestion control method which is a combination of proactive and reactive approaches. It adjusts *cwnd* according to the congestion level of the path.

The path delay is an important factor that provides traffic information on the path. As a delay of the path increases, it means that traffic on the path also increases. Therefore, our proposed approach uses path delay(RTT) as an indicator of congestion. We estimate RTT_{min}, RTT_{max} and normal RTT for each received ACK. New, we calculate β as shown in (5).

$$\beta = \frac{RTT - RTT_{min}}{RTT_{max}} \tag{5}$$

Where RTT_{min} is the minimum round trip time, RTT_{max} is the maximum round trip time and RTT is the current round trip time. The β is a congestion indicator which provides information about the traffic intensity of path. The value of β lies between 0 and 1. As the value of β is around 0, it means that the path has normal traffic intensity. However, as the value of β approaches to 1, it means that the path has more traffic intensity. Therefore, the proposed method adjusts the *cwnd* according to β. The value of β used to detect congestion before it happens. Thus, we need a few threshold variables which specify the traffic intensity on the path. So, we use ρ, σ and τ threshold variables that decide path traffic intensity. The threshold variable ρ, σ, and τ triggered when the network is less, partially and highly congested respectively. The values of these variables lie between 0-1 and decided experimentally. The proposed approach uses these variables to change the congestion control modes.

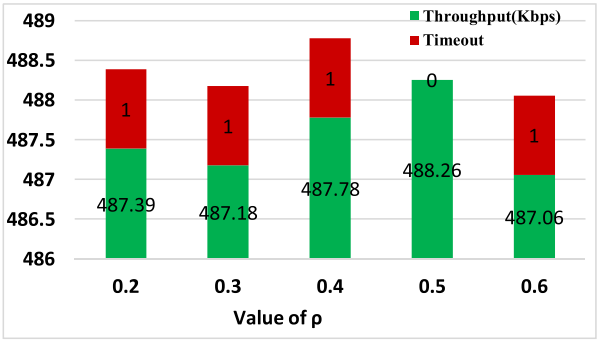
To find the optimal value of these variables, we have performed 10 numbers of simulations. In this simulation, we use dumbbell topology for which has 10 source nodes, 10 destination nodes, and 2 routers. All connected links have1Mbps bandwidth while bottleneck link has 1.5Mbps bandwidth 20ms propagation delay. Fig. 3(a)-(c) shows the simulation results of ρ, σ, and τ. Fig. 3(a) confirms that as the value of ρ increases, throughput and timeout of TCP also changes. It is evident from Fig. 3(a) that TCP achieves the best result at ρ=0.50 where throughput is maximum and timeout is zero. Additionally, a similar pattern also has been observed during the simulation for σ, and τ. Fig. 3(b) and 3(c) shows that at σ =0.80, and τ =0.90, we observed the best performance of the proposed TCP. Therefore, we use the value of ρ=0.50, σ =0.80, and τ =0.90 at the time of the simulation.

Reactive mode: If the value of β is less than ρ, TCP is in reactive mode; otherwise, it is in proactive mode. Initially, TCP starts in reactive mode. In this mode, TCP increases *cwnd* exponentially when *cwnd* $\leq$ ssthresh;otherwise, it increases linearly.

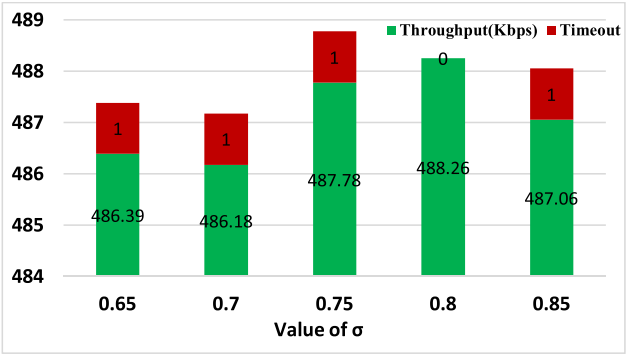
Proactive mode: If β is greater than ρ, then TCP works in a proactive mode. As per the congestion indicator, we put TCP in the following states

- **Less congested:** If $\beta \geq \rho$ and $\beta \leq \sigma$, then TCP is in the less-congested state and *cwnd* increases linearly by adding the factor (1- β).
- **Partially congested:** If $\beta \geq \sigma$ and $\beta \leq \tau$, then TCP is in the partially congested state, which means TCP is in steady-state. Thus, *cwnd* remains constant.
- **Highly congested:** If $\beta \geq \tau$, then the network is experiencing heavy congestion. It means that TCP is running on more than the network capacity. Here, we reduce *cwnd* by 1/*cwnd*.

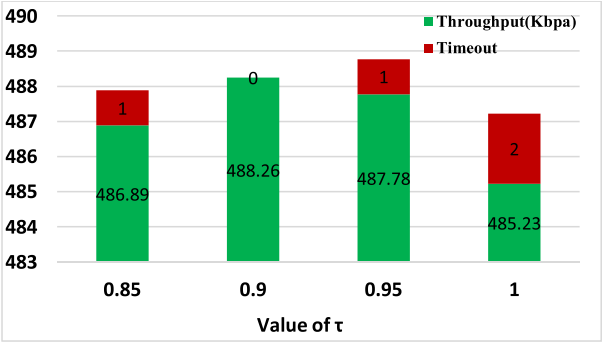
The algorithm for congestion control is shown in Algorithm 3. This algorithm runs on sender side for each received ACK.



(a)



(b)



(c)

Fig. 3. Assessment for optimal value of ρ, σ, and τ.

4. Performance evaluation

This section shows the simulation setup for all the simulated approaches such as New Reno [21], HTCP [26], CUBIC [28], and Fast-Fit [36] for the simulation. The entire simulation is conducted in Network simulator 2 (NS-2). The dumbbell topology is a recommended topology for the congestion control test and the IoT environment also suffers from the same congestion problem. Thus, we use dumbbell topology for validation of the proposed approach which is shown in Fig. 4. This topology has n source nodes, n destination nodes, and 2 routers. All nodes are connected with links of different bandwidth. The value of n may vary according to the simulation requirement. This simulation topology includes various types of traffic generators like CBR, FTP, and VBR. All nodes are configured with the Drop-tail queuing policy. All links have different propagation delays. During the simulation, the values of α, ρ, σ and τ are 4, 0.50, 0.80 and 0.90, respectively. The IoT environment connects the maximum number of low bandwidth and high delay devices. Thus, simulation topology has been configured according to IoT requirements.

Analysis of the throughput of the proposed TCP, Fast-Fit, HTCP, CUBIC, and New Reno have been shown in Figs. 5 and 6. These results have been observed for different simulation times, ranging from 25sec. to 150sec. During the simulation, the

Algorithm 3 Congestion Control Algorithm.

1. **//At source node for each received ACK**
2. **Begin**
3. **If** ACK received
4. **Then**
5. $\beta = \frac{RTT_i - RTT_{min}}{RTT_{max}}$
6. **If** $\beta > \tau$
7. $cwnd_{i+1} = cwnd_i - \frac{1}{cwnd_i}$
8. **Else If** $\beta > \sigma$
9. $cwnd_{i+1} = cwnd_i$
10. **Else If** $\beta > \rho$
11. $cwnd_{i+1} = cwnd_i + \frac{1}{cwnd_i} + (1 - \beta)$
12. **Else If** $cwnd_i \leq ssthresh$
13. $ModifiedSlow\ start$
14. **Else**
15. $Congestion\ avoidence$
16. **End If**
17. **End If**
18. **End**

requirements.

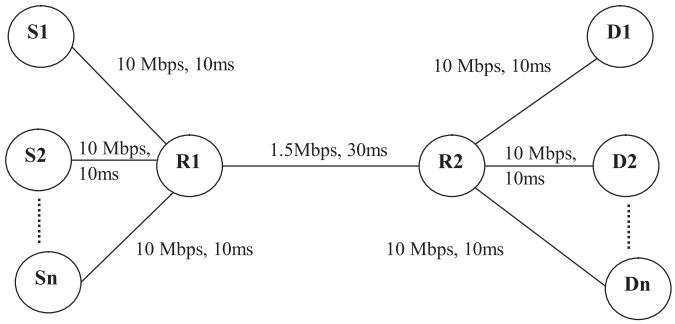


Fig. 4. Simulation topology.

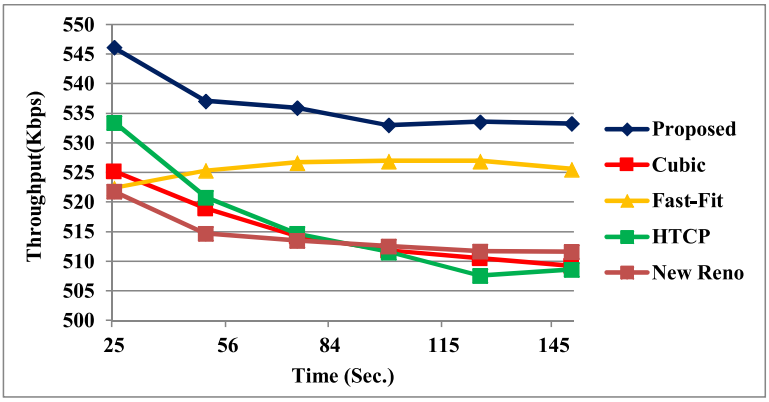


Fig. 5. Throughput of the TCP variants with respect to simulation time.

bottleneck queue size is 50 packets, propagation delay of the path is 50ms and background traffic is 1000Kbps in both directions.

Fig. 5 shows the throughput of TCP variants with different simulation times. It shows that as time increase the throughput of TCP variants decreases. However, Fast-fit shows least utilization at 25sec. and as time increases its utilization get stable while HTCP, Cubic and New Reno demonstrate the linear decreasing pattern. The proposed TCP initially start with good capacity due *cwnd* initialization method and maintain the utilization by using new congestion detection and avoidance approach. Thus, the proposed method achieves higher throughput as compared to other TCP variants.

Fig. 6 shows the average throughput of different TCP variants. It show that the New Reno have least utilization while proposed method have highest utilization amongst the Cubic, HTCP, New Reno and Fast-Fit TCP variants. The proposed

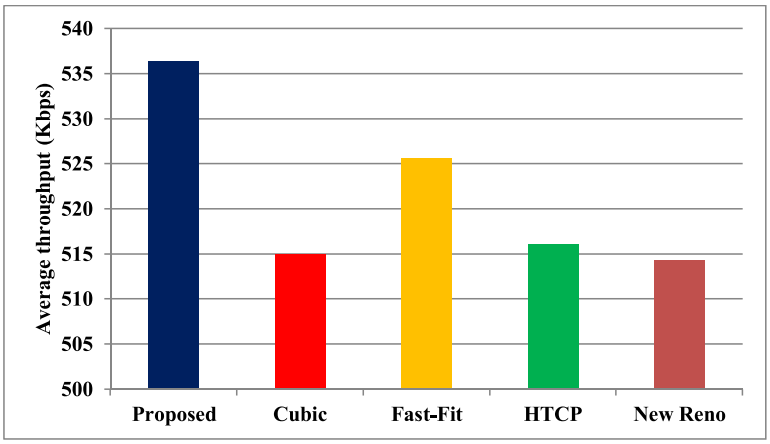


Fig. 6. Average throughputs of different TCP variants.

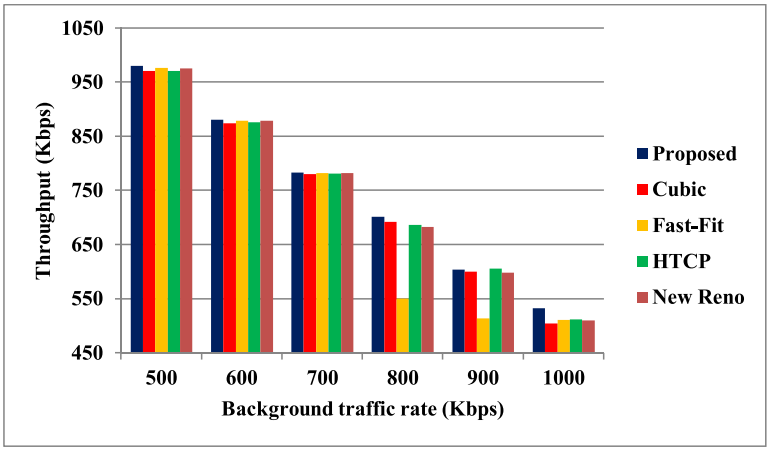


Fig. 7. Throughput of the TCP variants with different background traffic.

method achieves 4% better throughput as compared to Cubic, 1.9% better throughput as compared to Fast-Fit, 3.8% better throughput as compared to HTCP and 4.3% better throughput as compared to New Reno.

IoT has variable traffic intensity due to the huge number of connected devices. Thus, to validate proposed method in such environment, the simulation uses the variable background traffic rate, ranging from 500Kbps to 1000Kbps. During this simulation, the bottleneck queue size is 50, the propagation delay of the path is 50ms and simulation time is 100sec. All the link has been configured according to Fig. 4.

Fig. 7 shows the throughput of different TCP variants with variable background traffic. When the background traffic is low (500Kbps), all of the TCP variants yield similar throughput, but as the background traffic increases, packet drops increase too, which reduces the throughput. Fast-Fit experiences high throughput drops, while New Reno, Cubic, HTCP and the proposed TCP exhibit a linear drop in throughput. The proposed method uses different threshold variable to specify the path traffic intensity and adjust the traffic rate accordingly. Thus, proposed method maintains the utilization under all background traffic conditions and achieves higher throughput than the other TCP variants.

Fig. 8 shows the average throughput of all of the TCP variants under different background traffic conditions. It shows that Fast-Fit shows the least utilization while proposed method shows the highest throughput amongst Fast-Fit, Cubic, HTCP and New Reno. The proposed method improves 6.6% throughput as compared to Fast-Fit, 1.5% throughput as compared to Cubic, 1.3% throughput as compared to HTCP and 1.6% throughput as compared to New Reno.

In IoT environment, most of the devices work on android operating system which uses the TCP Cubic as the transport layer protocol. Thus, the fairness with TCP Cubic is an important factor for STCP. For fairness analysis, we use three TCP Cubic flows with one other TCP variant. In this simulation, TCP Cubic and the other TCP variant's flow share a 1.5-Mbps bottleneck link. The propagation delay of the path is set to 50ms, and the queue size of bottleneck is 50. These results are obtained when background traffic is present in the network. The simulation time of the setup is 100sec.

Fig. 9 shows the fairness analysis of the TCP variants with TCP Cubic. In this figure, other represents the competing traffic with 3 cubic flows. It shows that the proposed TCP shares almost the same amount of bandwidth with the TCP Cubic, while the other TCP variants are not able to maintain the same amount of bandwidth sharing.

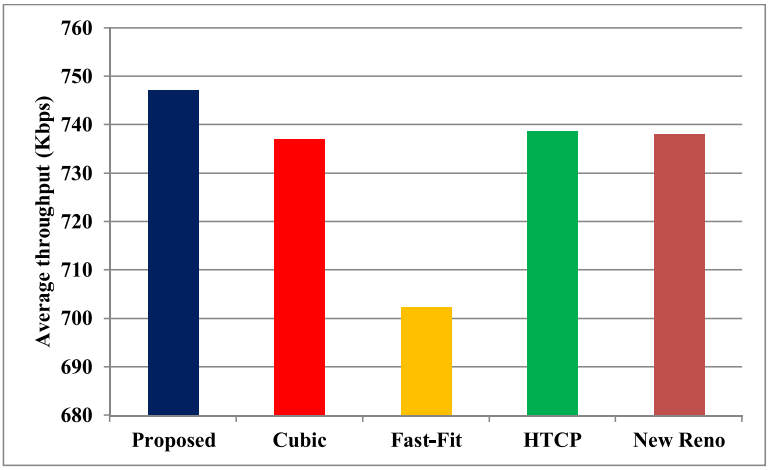


Fig. 8. Average throughputs of different TCP variants.

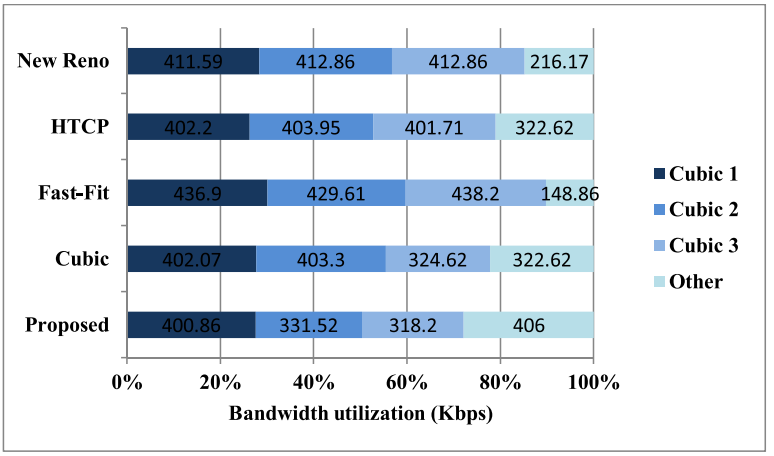


Fig. 9. Bandwidth utilization among the different TCP variants with TCP Cubic.

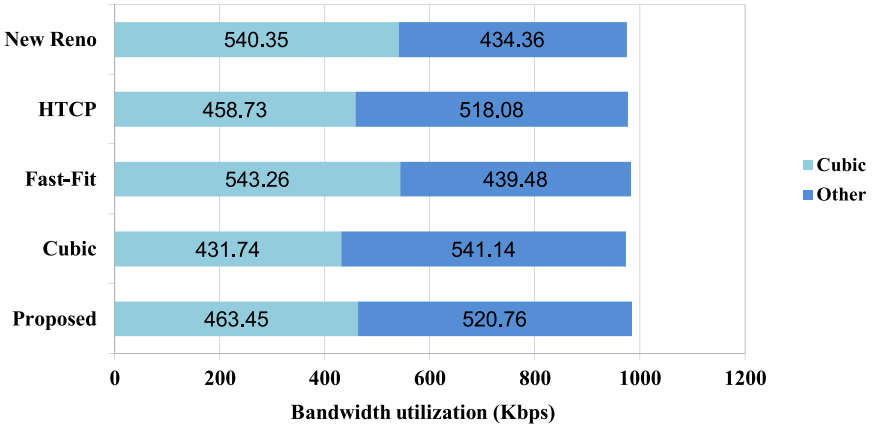


Fig. 10. Bandwidth utilization of the TCP variants with TCP Cubic.

In another simulation, we observed the fairness of TCP variants with one TCP Cubic flow. In this simulation setup, the propagation delay of the path is 50ms, bottleneck queue size is 50 and simulation time is 100sec. The results are observed in the presence of background traffic of 1000Kbps. Fig. 10 shows the bandwidth sharing of one TCP variant with one TCP Cubic flow. It shows that the proposed TCP shares almost equal bandwidth with TCP Cubic flow.

5. Conclusion

This paper proposed a new congestion control algorithm for IoT application protocols to quickly adapt the transmission rate according to network conditions. The proposed approach is a combination of loss-based and delay-based TCP variants, which work in two modes: reactive mode and proactive mode. Initially, it starts in reactive mode, and when network congestion increases, it switched into proactive mode. The proposed approach maintains fairness with TCP Cubic, which is widely implemented over the Internet. Simulation results show that the proposed approach provides better results in terms of throughput and inter-protocol fairness with TCP Cubic. The proposed method is well suited for IoT applications protocol MQTT which work for lightweight smart home appliances, smart city monitoring systems, healthcare providers and sensors communicating to the server via satellite link, etc.

Declaration of Competing Interest

None.

References

[1] K. Ashton, That 'Internet-of-Things' Thing, RFiD Journal 22 (7) (2011) 1.
[2] M. Aissa, A. Belghith, Overview of machine-type communications traffic patterns, in: Web Applications and Networking (WSWAN), in: 2nd World Symposium on, IEEE, 2015, pp. 1–7.
[3] Ericsson mobility report June 2017, https://www.ericsson.com/assets/local/mobility-report/documents/2017/ericsson-mobility-report-june-2017.pdf.
[4] Gartner says 8.4 billion connected 'things' will be in use in 2017, up 31percent from 2016, http://www.gartner.com/newsroom/id/3598917.
[5] D.J. Cook, A.S. Crandall, B.L. Thomas, N.C. Krishnan, CASAS: Asmart home in a box, Computer 46 (7) (2013) 62–69.
[6] Y. Leng, L. Zhao, Novel design of intelligent internet-of-vehicles management system based on cloud-computing and internet-of-things, in: Electronic and Mechanical Engineering and Information Technology(EMEIT), in: 2011 International Conference on, 6, IEEE, 2011, pp. 3190–3193.
[7] M. Gerla, E.-K. Lee, G. Pau, U. Lee, Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds, in: Internet of Things (WF-IoT), 2014 IEEE World Forum on, IEEE, 2014, pp. 241–246.
[8] C. Wang, Z. Bi, L. Da Xu, Iot and cloud computing in automation of assembly modeling systems, IEEE Transactions on Industrial Informatics 10 (2) (2014) 1426–1434.
[9] K. Michaelsen, J.L. Sanders, S.M. Zimmer, G.M. Bump, Overcoming patient barriers to discussing physician hand hygiene: do patients preferelectronic reminders to other methods? Infection Control &Hospital Epidemiology 34 (9) (2013) 929–934.
[10] Y. Yan, Y. Qian, H. Sharif, D. Tipper, A survey on smart grid communication infrastructures: Motivations, requirements and challenges, IEEE communications surveys & tutorials 15 (1) (2013) 5–20.
[11] J. Jin, J. Gubbi, S. Marusic, M. Palaniswami, An information framework for creating a smart city through internet of things, IEEE Internet of Things Journal 1 (2) (2014) 112–121.
[12] P. Saint-Andre, (2011) Extensible Messaging and Presence Protocol (XMPP): Core. https://tools.ietf.org/html/rfc6120. Accessed 10 December 2018.
[13] MQTT Version 3.1.1 (2015) http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.doc. Accessed 12- December - 2018.
[14] P. Saint-Andre, (2011) "Extensible messaging and presence protocol (XMPP): Core," IETF RFC 6120, 2011, Online Available: https://tools.ietf.org/html/rfc6120 Accessed 12-June-2019.
[15] J. Postel, (1981) Transmission Control Protocol. https://tools.ietf.org/html/rfc793. Accessed 12-June-2019.
[16] Z. Shelby, Sensinode, K. Hartke, C. Bormann, and U. B. TZI, (2014) "Internet-draft - constrained application protocol (CoAP)," Online Available: https://tools.ietf.org/html/rfc7252. 12-June-2019.
[17] D. Papadimitriou, M. Welzl, M. Scharf, B. Briscoe, (2011) Open Research Issues in Internet Congestion Control. https://tools.ietf.org/html/rfc6077. Accessed 12 December 2018
[18] V. Jacobson, Congestion avoidance and control, ACM SIGCOMM Computer Communication Review 18 (1988) 314–329.
[19] M. Allman, V. Paxson, W. Stevens, (1999) TCP congestion control. https://tools.ietf.org/html/rfc2581. Accessed 12-June-2019.
[20] S. Floyd, T. Henderson, (1999) The NewReno modification to TCP's fast recovery algorithm. https://tools.ietf.org/html/rfc2582. Accessed 12-June-2019.
[21] S. Floyd, T. Henderson, A. Gurtov, (2004) The NewReno modification to TCP's fast recovery algorithm. https://tools.ietf.org/html/rfc3782. Accessed 12-June-2019.
[22] M. Mathis, J. Mahdavi, S. Floyd, A. Romanov, (1996) TCP selective acknowledgment options. https://tools.ietf.org/html/rfc2018. Accessed 12-June-2019.
[23] M. Mathis, J. Mahdavi, Forward acknowledgement: refining TCP congestion control, ACM SIGCOMM Computer Communication Review 26 (1996) 281–291.
[24] S. Floyd, (2003) HighSpeed TCP for large congestion windows. https://tools.ietf.org/html/rfc3649. Accessed 12-June-2019.
[25] T. Kelly, Scalable TCP: improving performance in high-speed wide area networks, ACM SIGCOMM Computer Communication Review 33 (2003) 83–91.
[26] D. Leith, R. Shorten, (2007) H-TCP: TCP congestion control for high bandwidth-delay product paths. https://tools.ietf.org/html/draft-leith-tcp-htcp-03. Accessed 12-June-2019.
[27] L. Xu, K. Harfoush, I. Rhee, Binary increase congestion control for fast, long distance networks, in: Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, 2004, pp. 2514–2524.
[28] S. Ha, I. Rhee, L. Xu, CUBIC: a new TCP-friendly high-speed TCP variant, ACM SIGOPS Operating Systems Review 42 (2008) 64–74.
[29] J. Wang, J. Wen, Y. Han, J. Zhang, C. Li, Z. Xiong, CUBIC-FIT: A High Performance and TCP CUBIC Friendly Congestion Control Algorithm, IEEE Communication Latter 17 (2013) 1664–1667.
[30] L. Brakmo, L. Peterson, TCP Vegas: end to end congestion avoidance on a global Internet, IEEE Journal on Selected Areas in Communications 13 (1995) 1465–1480.
[31] G. Hasegawa, K. Kurata, M. Murata, Analysis and improvement of fairness between TCP Reno and Vegas for deployment of TCP Vegas to the Internet, International Conference on Network Protocols (2000) 177–186.
[32] S. Mascolo, C. Casetti, M. Gerla, M.Y. Sanadidi, R. Wang, TCP Westwood: Bandwidth estimation for enhanced transport over wireless links, in: International conference on Mobile computing and networking, 2001, pp. 287–297.

[33] L.A. Grieco, S. Mascolo, Performance evaluation and comparison of Westwood+, New Reno and Vegas TCP congestion control, in: ACM SIGCOMM Computer Communication Review, 34, 2004, pp. 25–38.
[34] D.X. Wei, C. Jin, S.H. Low, S. Hegde, FAST TCP: motivation, architecture, algorithms, performance, IEEE/ACM Transactions on Networking 14 (2006) 1246–1259.
[35] J. Wang, J. Wen, J. Zhang, Y. Han, TCP-FIT: an improved TCP congestion control algorithm and its performance, in: IEEE INFOCOM Conference, 2011, pp. 2894–2902.
[36] J. Wang, J. Wen, Y. Han, J. Zhang, C. Li, Z. Xiong, Achieving high throughput and TCP Reno fairness in delay-based TCP over large network, Frontier of Computer Science 8 (2014) 426–439.
[37] V.N. Padmanabhan, R.H. Katz, TCP Fast Start: A Technique for Speeding Up Web Transfers, in: IEEE Globecom 98 Internet Mini-Conference, 1998.
[38] S. Floyd, M. Allman, A. Jain, (2007) Quick-Start for TCP and IP. https://tools.ietf.org/html/rfc4782. Accessed 12 January, 2016.
[39] M. Allman, S. Floyd, C. Partridge, (2002) Increasing TCP's initial window. https://tools.ietf.org/html/rfc3390. Accessed 12-June-2019.
[40] N. Dukkipati, T. Refice, Y. Cheng, An argument for increasing TCP's initial congestion window, ACM SIGCOMM Computer Communication Review 40 (2010) 27–33.
[41] R. Sallantin, C. Baudoin, E. Chaput, F. Arnal, E. Dubois, A. Beylot, Initial Spreading: a Fast Start–Up TCP Mechanism, in: 38th annual conference on Local Computer Network, 2013.
[42] J. Chu, N. Dukkipati, Y. Cheng, M. Mathis, (2013) Increasing TCP's Initial Window. https://tools.ietf.org/html/rfc6928. Accessed 12-June-2019.
[43] I. Lee, K. Lee, The internet of things (IoT): Applications, investments, and challenges for enterprises, Business Horizons 58 (4) (2015) 431–440.
[44] E. Soltanmohammadi, K. Ghavami, M. Naraghi-Pour, A survey of traffic issues in machine-to-machine communications over LTE, IEEE Internet of Things Journal 3 (6) (2016) 865–884.
[45] N. Mishra, L.P. Verma, P.K. Srivastava, A. Gupta, An Analysis of IoT Congestion Control Policies, Procedia Computer Science 132 (2018) 444–450.
[46] L.P. Verma, M. Kumar, An adaptive data chunk scheduling for concurrent multipath transfer, Computer Standards & Interfaces 52 (2017) 97–104.
[47] L.P. Verma, V.K. Sharma, M. Kumar, New Delay-Based Fast Retransmission Policy for CMT-SCTP, International Journal of Intelligent Systems and Applications 10 (3) (2018) 59–66.
[48] V.K. Sharma, L.P. Verma, M. Kumar, A Fuzzy-based Adaptive Energy Efficient Load Distribution Scheme in Ad-hoc Networks, International Journal of Intelligent Systems and Applications 10 (2) (2018) 72–84.
[49] D.E. Comer, D.L. Stevens, Internetworking With TCP/IP Volume 2: Design, Implementation, and Internals, Third edition, Prentice-Hall, Upper Saddle River, NJ, 1999.
[50] yuba.stanford.edu/~nickm/papers/classification_tutorial_01.pdf.
[51] D.E. Comer, Packet Classification: A Faster, More Generic Alternative to Demultiplexing", The Internet Protocol Journal 15 (4) (2012) 12–22.
[52] D.E. Comer, Network Systems Design Using Network Processors, Prentice-Hall, Upper Saddle River, NJ, 2006 Intel IXP 2xxx version.
[53] G. Kumar, P. Tomar, "A Survey of IPv6 Addressing Schemes for Internet of Things", International Journal of Hyperconnectivity and the Internet of Things (IJHIoT) 2 (2) (2018) 15.
[54] V.K. Sharma, L.P. Verma, M. Kumar, CL-ADSP: Cross-Layer Adaptive Data Scheduling Policy in Mobile Ad-hoc Networks, Future Generation Computer Systems 97 (2019) 530–563.
[55] V.K. Sharma, M. Kumar, "Adaptive congestion control scheme in mobile ad-hoc networks", Peer Peer Netw. Appl. 10 (3) (2017) 633–657.
[56] L.P. Verma, I. Verma, M. Kumar, An adaptive congestion control algorithm, Modelling, Measurement and Control A 92 (1) (2019) 30–36.