EC-Elastic an Explicit Congestion Control Mechanism for Named Data Networking

Asmaa EL-BAKKOUCHI[1], Mohammed EL GHAZI[2]
Anas BOUAYAD[3], Mohammed FATTAH[4], Moulhime EL BEKKALI[5]
Artificial Intelligence, Data Sciences and Emerging Systems Laboratory
Sidi Mohamed Ben Abdellah University, Fez, Morocco[1, 2, 3, 5]
IMAGE Laboratory, Moulay Ismail University, Meknes, Morocco[4]

***Abstract*—In recent years, Named Data Networking (NDN) has attracted researchers' attention as a new internet architecture. NDN changes the internet communication paradigm from a host-to-host IP model to a name-based model. Thus, NDN permits the retrieval of requested content by name, from different sources and via multiple paths, and the use of caching in intermediate routers. These features transform the transport control model from sender to receiver and make traditional end-to-end congestion control mechanisms incompatible with NDN architecture. To deal with this problem, a reliable congestion control mechanism becomes necessary for a successful deployment of NDN. This paper presents a new hybrid congestion control mechanism for NDN, EC-Elastic (Explicit Congestion Elastic), which adopts the basic concept of Elastic-TCP to control the sending rates of the interest packets at the consumer nodes. In the intermediate nodes, a queue has been associated with the Controlled Delay-Active Queue Management CoDel-AQM to measure the packet sojourn time and notify the consumer to decrease its interest packet sending rate when it receives an explicit congestion signal. EC-Elastic was implemented in ndnSIM and evaluated with Agile-SD, CUBIC, and STCP in different scenarios. Simulation results show that EC-Elastic provides a significant improvement in bandwidth utilization while maintaining lower delay and packet loss rates.**

Keywords*—NDN; named data networking; congestion control; explicit congestion control; TCP-elastic

I. INTRODUCTION

The use of the internet has grown exponentially from point-to-point communications to the distribution of information everywhere. This growth has increased the number of internet users where these users are more interested in getting data in a short period of time than the location of that data. To facilitate connectivity between these users, high-speed and long-distance networks have been widely employed in many countries [1] [2]. However, this evolution poses some problems, namely, the current Transmission Control Protocol/Internet Protocol TCP/IP internet architecture and its variants have seen poor performance [3], and cannot cope with this growth, as they are designed for end-to-end communications. The use of high speed and long distance networks requires consideration of two major problems that are often encountered in this type of environment and that affect network performance negatively. The first problem concerns the use of large buffer regimes and long distances which leads to very long RTTs while the second problem concerns the need to increase the congestion window (cwnd) as much as possible to maximize the use of available bandwidth.

The first problem concerning the current TCP/IP internet architecture has motivated the researchers to explore new architectures for the future internet [4]. Information-Centric Networking (ICN) [5] has been proposed as a new content-centric internet architecture to replace the current host-centric internet architecture. ICN has proposed several architectures that are all based on the content name rather than the IP address. Among these architectures, Named Data Networking (NDN) [6], an important research topic that has quickly encountered considerable interest from researchers. NDN uses hierarchical names to exchange two types of packets (interest packets and data packets [6]) between consumers and content producers. A consumer requests content via an interest packet, and then any node that has the requested data sends it through a data packet. These data packets follow the reverse path of interest packets. Each NDN node has three components, namely Content Store (CS), Pending Interest Table (PIT) and Forwarding Information Base (FIB) as shown in Fig. 1. Content Store (CS) : works as a content cache [7]. When CS receives data packets, it can store them temporarily in a cache and use them again in case of a request for the same data [8]. Pending Interest Table (PIT): The PIT contains interests that have been transmitted upstream but have not yet been satisfied [7]. It also contains the incoming interface list from which the interest packet for that name was received and the outgoing interface list from which the interest packet was sent [8]. Forwarding Information Base (FIB) : This is a database that contains prefix names for identifying the location of content producers, and an interfaces list for determining which interface is needed to forward the interest packet [8].

The NDN architecture has new features such as connectionless, one-interest-one-data, caching, multipath and multi-source. However, these features complicate network congestion control, because the existing TCP/IP solutions cannot be applied directly in NDN, which made congestion control an active research topic to be studied. The different characteristics between the two architectures (NDN and TCP/IP) mainly lie in:

- In NDN, communication is receiver-based and connectionless, whereas in TCP/IP it is connection-oriented between two end points.

- NDN uses a One-Interest-One-Data transport mode, the consumer is responsible for retransmitting the interest packet if the desired data is not received. There are no duplicate data acknowledgement (ACKs) as in TCP/IP.
- NDN uses caching in intermediate nodes to satisfy requests from all consumers rather than a single content source used by TCP/IP.
- The use of caching in NDN nodes allows desired data to be fetched from several sources and over several paths, which complicates the use of RTO (Retransmission Time Out) in NDN congestion control as it is intended for single-source TCP/IP communication [9].

These challenges have motivated the research community to design and develop new mechanisms for NDN networks that are able to avoid congestion, increase the use of available bandwidth while maintaining fast delivery time. However, the majority of existing mechanisms in NDN are based on the AIMD mechanism which can prevent full utilization of the available bandwidth due to the huge bandwidth-delay product (BDP: Bandwidth-Delay Product refers to the maximum quantity of data that can be sent over a link or network) in high-speed and long-distance networks, making it a waste of network resources [1] because AIMD takes a long time to reach the maximum capacity of the network links, which leads to underutilization of the bandwidth. Moreover, in case of congestion, AIMD divides the congestion window by 2, which requires more time to reach the maximum throughput again and consequently, the link performance is degraded.

To address the second problem concerning large buffer regimes and very long RTTs, this paper proposes a new hybrid congestion control mechanism for NDN named Explicit Congestion Elastic (EC-Elastic), which adapts the basic idea of Elastic-TCP [1] to control the sending rate of interest packets at the consumer nodes. EC-Elastic uses the Window-correlated Weighting Function WWF that aims to improve the bandwidth utilization of the network. In intermediate routers, EC-Elastic uses a CoDel-AQM queue for each prefix on each interface to measure packet sojourn time. This algorithm allows routers, which have a large buffer, to absorb traffic bursts and to reduce its queues through detecting congestion before the buffer is full [10] then explicitly signals congestion to inform consumers to reduce their traffic rate.

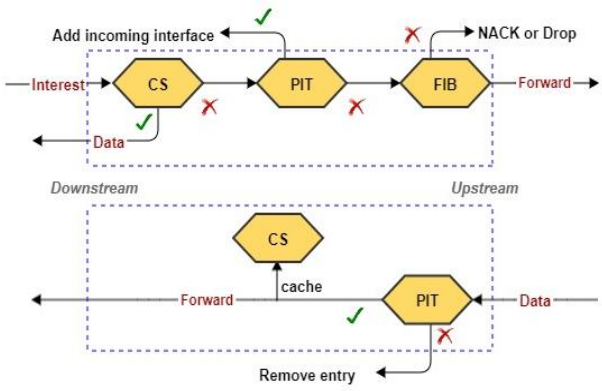

Fig. 1. Forwarding Process at NDN Node.

The rest of the paper is organized as follows: Section II presents the related work while Section III presents the principle of congestion control. Section IV details the proposed "EC-Elastic" mechanism and Section V evaluates the performance of this mechanism, it presents the topologies and measurements used as well as the results and discussion. Finally, Section VI concludes the paper.

II. Related Work

In the literature, several congestion control mechanisms have been proposed for NDN networks. According to [11], these mechanisms can be classified into three categories: Receiver-based method: which is characterized by detecting congestion and controlling the sending rate of interest packets only at the consumer nodes [7]. Hop-by-hop method: which is characterized by detecting congestion and controlling the sending rate of interest packets at each intermediate node [12]. Hybrid method: which is characterized by detecting and controlling congestion at both receiver nodes and intermediate nodes [7] [12].

In NDN, the majority of congestion control mechanisms are inherited from TCP's window-based mechanisms, and most of them adjust the size of their congestion window based on the Additive Increase Multiplicative Decrease (AIMD) mechanism that increases the congestion window by Additive Increase (AI) and decreases the congestion window by Multiplicative Decrease (MD). Specifically, the authors of [13] propose ICP (Interest Control Protocol) a receiver-based congestion control mechanism that detects congestion by measuring the delay and timer expirations and adjusts the congestion window size by AIMD mechanism. The authors of [14] propose ICTP (Information Centric Transport Protocol) which also detects congestion by RTO Timeout and adjusts the congestion window size by AIMD mechanism. These two mechanisms did not consider the case of multiple source scenarios. To solve this problem, the authors of [15] propose a mechanism named ConTug that detects congestion using an RTO value for each content source and then adjusts the size of congestion window by AIMD mechanism. However, the authors of [16] propose CCTCP (Content Centric TCP) which instead of using a single RTO value for each content source, it uses a separate RTO value for each data source and then adjusts the congestion window size by AIMD mechanism. On the other hand, the authors of [17] propose predictive which maintains an RTO value for each Content Store to detect congestion and then uses the AIMD mechanism to adjust the congestion window size. In [18], the authors propose a Hop-by-hop Receiver-driven Interest Control Protocol (HR-ICP), which at the router level detects congestion using a virtual queue and then depending on the state of this queue, the consumer nodes use the AIMD mechanism to adjust the congestion window size. Other authors proposed CVUnion in [19] which detects congestion at intermediate nodes through the calculation of the average queue length of the interest packets, then once the consumer receives the feedback, it adjusts its congestion window size by the AIMD mechanism. In [20], the authors propose CHoPCoP (Chunk-switched Hop Pull Control Protocol) which detects congestion at intermediate nodes by monitoring the queue size of outgoing data packets, and then, based on the queue size, an explicit congestion notification is sent to the consumer to

adjust the congestion window size through the AIMD mechanism. The authors of [21] propose Stateful Forwarding which detects congestion by calculating the limit rate of interest packets. Stateful Forwarding generates a Negative-ACKnowledgment NACK packet that will be sent on downstream when congestion is detected. Once the downstream router receives this NACK packet, and depending on the received link state, it uses the AIMD mechanism to adjust the size of the congestion window. However, the authors of [22] [23] detected problems with the use of NACK, namely the delay in transmitting the NACK between two routers, which results in an excessive reduction in the sending rate of interest packets. Therefore, the authors propose to use three states for each interface which are, normal, congestion and check, and depending on the state of the network, the AIMD mechanism is used to adjust the congestion window size. Other authors propose to combine the multipath forwarding strategy with congestion control as in [24] where the authors deployed the forwarding strategy at the intermediate routers and the AIMD mechanism at the consumer nodes. In [25], the authors propose Standbyme which controls congestion in three steps: Accurate Local Congestion detection, Hob-by-hop congestion notification and Multipath strategy congestion avoidance and adjusts the congestion window size with AIMD mechanism.

In the congestion avoidance phase, AIMD increases the congestion window by 1/cwnd. In the case of short distance, cwnd is small, so the congestion window increase will be rapid and reasonable. However, in the case of long distance, cwnd is large and therefore the congestion window increase will be slow. In addition, in the case of congestion, AIMD uses a Multiplicative Decrease which divides the congestion window by 2 and moves to the next phase, where the congestion window will be increased by Additive Increase to reach again the cwnd maximum. In the case of short distance where the RTTs are small, this method provides acceptable throughput and reasonable bandwidth utilization. However, in the case of long distance where the RTTs are very large, this method takes too much time to reach again the maximal cwnd which results in low throughput and bandwidth utilization and consequently degrades the link performance.

To address these issues, this paper proposes EC-Elastic, a Hybrid congestion control mechanism to avoid congestion, increase bandwidth utilization on long delays and high-BDP networks and achieve efficient data delivery. EC-Elastic adapts the basic idea of Elastic-TCP [8] to control the rate at which the interest packets are sent to consumer nodes. EC-Elastic controls congestion in three phases; congestion detection at intermediate routers using CoDel-AQM, then explicitly signaling congestion to inform consumers to reduce their traffic rate, and finally adjusting the congestion window based on the type of packet received by consumers.

III. Principle of Congestion Control

To support high-speed applications (e.g., large-scale data transfer) and low-latency applications in NDN networks, we need a congestion control mechanism. This mechanism should contain the following steps: "Congestion detection", "Congestion signaling" and "Congestion window size adjustment". The description of each step is described above:

A. Congestion Detection

Data transfer can saturate queues, which degrades quality of service in the network. The deployment of an AQM strategy is necessary. In the literature, many AQM algorithms have been proposed such as Drop-Tail [26]., RED (Random Early Detection) [27], CoDel [28] or PIE (Proportional Integral controller Enhanced) [29]. The basic idea behind these algorithms is that the current queue length is not an indication of congestion as it can be caused by bursty traffic [30].

Drop-Tail [26] was proposed as the first algorithm to solve queue management problems. This algorithm works as follows: Each queue's length is fixed at a maximum value known as the maximum packet length, and user's incoming packets will be stored in this queue. When the length of the queue hits the maximum limit, the incoming packets will be dropped. Then, when the packets are removed from the queue and its length decreases, the incoming packets will be stored in the queue again. This method can fill up the queue quickly, resulting in a high loss rate for applications; the Drop-Tail queue increases delay since it can be full for a long period of time [26].

RED [27] is an algorithm that relies on the average queue length to drop packets, i.e., as the queue length increases, the probability of packet drop increases and vice versa. RED works according to two principles: the estimation of the queue length and the packet drop decision and uses two thresholds for this purpose. When the average queue length is lower than the minimum threshold, all incoming packets will be accepted. Otherwise, when the average queue length is higher than the maximum threshold, all incoming packets will be dropped. Finally, in the case of an average queue length between the two thresholds, the incoming packets will be marked by Pi probability. This probability is directly proportional to the bandwidth of the connection to the router. One of the problems with this algorithm is that it only works well when there is enough buffer space and it is properly parameterized. Thus, it requires a variety of parameters to cope with different types of congestion.

CoDel [28] is an algorithm that has been proposed to manage the queue by calculating the sojourn time of packets in the queue. Based on this packet sojourn time, CoDel decides if the packet should be dropped or not. CoDel works as follows: It calculates the packet sojourn time (the time spent by every packet in the queue) and compares it to the threshold which is by default 5ms. If the minimum sojourn time is less than this threshold, the packet will be transmitted, otherwise if the sojourn time is greater than this threshold, the packet will be dropped. When the algorithm enters into the drop state, it starts sending congestion signals and drops packets that have a low and linearly increasing rate. CoDel starts the drop with the packet that is at the top of the queue and reduces the time interval of the next drop by a certain value. The packet drop increases if the sojourn time remains above the threshold. This algorithm can handle bursty traffic without causing packet loss. This algorithm is considered as a better predictor of congestion [30]. In EC-Elastic, we adopt the same congestion detection method as CoDel.

PIE [29] is an algorithm that controls the average latency of the queue to a target value. The PIE algorithm consists of three

components: a) Random dropping at enqueuing; calculates the dropping probability p. Based on this probability, the packets will be dropped randomly. The timestamp is not mandatory in this step. b) Latency based drop probability update; the calculation of drop probability uses the current estimate of the latency and the direction in which the latency is moving. Alternatively, the direction can be measured by subtracting the current delay from the old delay. There are two parameters used by PIE; (α) to determine the effect of the current latency on the fall probability and (β) to indicate the amount of additional adjustment based on increasing or decreasing latency. The probability of falling becomes stable at the point where the difference between the current and old latency is zero and the latency value equals the reference delay. The final balance between latency delay and latency jitter is determined by the relative weight between α and β. c) Dequeuing rate estimation; in a network, the queuing rate varies with the fluctuation of link capacity or queues that share the same link.

B. Congestion Signaling

After detecting congestion, the information of congestion should be transferred to the consumers and intermediate routers to react quickly to the congestion problem by decreasing the sending rate of interest packets. In NDN, several methods have been proposed to signal congestion to consumers and intermediate routers in order to regulate the sending rate of interest packets:

- Explicit congestion notification, which explicitly returns congestion level information in a NACK packet [22].
- Tagging data packets in the downstream direction, which allows downstream routers and consumers to reduce the sending rate of interest packets, thereby reducing congestion.
- The addition of congestion information in the Congestion Information Bits (CIB) to data packets. Adding a congestion tag to the data packet and sending it to the consumers [31].
- Random Early Marker algorithm REM [20], which explicitly marks data packets to signal the congestion state to downstream nodes.

C. Congestion Window Size Adjustment

The congestion window "cwnd" is used in all congestion control algorithms. It is used to control the quantity of sending packets between consumer and producer to avoid congestion. Despite this, congestion is not really avoidable, because the consumer always tries to maximize the available bandwidth by increasing its cwnd window, which could congest the network.

The congestion window adjustment of EC-Elastic borrows the basic idea of Elastic-TCP [1], which aims to increase the utilization of available bandwidth by using a Window Correlated Weighting Function (WWF), that handles large buffers, long delays and high-BDP networks [1].

IV. EC-Elastic Design Detail

Avoiding congestion is a major concern of all network architectures. In the following section, we present in detail our proposal EC-Elastic, which controls congestion in three steps: 1) Congestion detection based on packet sojourn time using CoDel. 2) Explicit congestion signaling. 3) Congestion window adjustment at the consumer node.

A. Motivation

As mentioned earlier, NDN is a new paradigm that is content-based rather than IP address-based. With this paradigm, data transfer evolves from host-based point-to-point transfer to more elaborate, efficient multipoint-to-multipoint transfer that is better suited for the massive and intensive use of content-based Internet. Thus, NDN adopts new features which are mainly receiver-based and connectionless transport mode, one-interest-one-data, multi-source, multi-path and caching. These new features have made TCP/IP's traditional congestion control mechanisms unable to act towards high performance in the emerging NDN paradigm. We present below, the limitations and motivations that led to our proposal:

- Congestion control in TCP/IP is based on delay and loss only at data senders, while NDN controls congestion at consumers and routers.
- The TCP/IP architecture uses end-to-end connected mode to transfer data between two endpoints and uses RTT (Round-Trip Time) and RTO (Retransmission Time Out) values as indicators of network congestion. These methods perform poorly in the NDN network, they don't provide accurate information about congestion levels because NDN is characterized by multi-path and multi-source transfer, i.e., data can be recovered from several sources and via several paths, which leads to large variations in RTT measurements.
- The use of caching in intermediate nodes allows the requested data to be retrieved directly from the intermediate nodes without needing to go through the producer. This technique minimizes data transmission time (RTT) and satisfies the interest packet when congestion losses occur on the producer route.
- In addition, NDN can aggregate interest packets having the same name into a single PIT (Pending Interest Table) entry and transmit the corresponding data packet to all the aggregated faces [10]. The recovery time of the interest packets that arrive after the first one will be shorter.
- These new features (multi-source, multipath, caching and PIT aggregation) can lead to short or long RTT measurements, which increases the detection time of packet losses (the case of long RTT) and consequently also increases the time of reaction to congestion. EC-Elastic avoids this problem by using explicit congestion signaling to react quickly to network congestion (see Section IV.3).
- If the cache is used, if it exhausts its data, the next requests will be handled by another more distant. If the route to the newer cache has a lower BDP and the number of interest packets in transit is higher, the new bottleneck queue may be overloaded before the consumer can adjust its interest packet sending rate.

Our mitigation of this problem is to use large buffers to manage temporary traffic bursts through detecting and signaling congestion using CoDel before that these buffers reach their limit (see Section IV.2).

To avoid the waste of network resources that are very expensive and important that can be caused by large buffers, we need to extend the congestion window to a large number of packets in order to fully utilize the available network bandwidth. In case of a network with high BDP (the number of interest packets in transit is higher), using RTT to increase the congestion window is not reliable because in these networks RTT is long which makes the increase of the congestion window very slow. In this case, the network spends a long period of time capturing the maximum link capacity, which underutilizes the network bandwidth. To avoid this problem, we propose to adopt the same Window-correlated Weighting Function (WWF) that was proposed by [1] to increase bandwidth utilization on TCP/IP high-BDP networks and try to prove its effectiveness on NDN networks to avoid congestion in the congestion avoidance phase and increase bandwidth utilization of NDN networks (see Section IV.4).

B. Congestion Detection based on Packet Sojourn Time using CoDel

In NDN networks, congestion detection based on packet loss or RTT (Round-Trip Time) is not reliable as in the current internet network TCP/IP because NDN is characterized by "multi-source" and "multi-path" transfer. In addition, the use of these features can increase bursty traffic that disrupts queue length and thus the production of congestion. Therefore, to absorb these bursty traffics, the buffer size must be larger than usual [30]. Active queue management (AQM) systems have been proposed to control the amount of data buffered to keep space available to absorb bursts and reduce queue delay. CoDel [28], as an AQM algorithm, is designed to control the queue by calculating the sojourn time of packets in the queue. This algorithm allows routers, which have a large buffer, to absorb traffic bursts and to reduce its queues through detecting congestion before the buffer is full [10]. In EC-Elastic, we adopt the congestion detection method proposed by Codel.

The CoDel algorithm, presented below, calculates the sojourn time of each packet in the queue "queuing delay" and compares the minimum sojourn time over a given period of time (default: 100ms) with a threshold, by default equal to 5ms. The first time the packet sojourn time exceeds the threshold, the current time will be recorded as FirstAboveTime and the packet sojourn time will be recorded as FirstSojourn. If the minimum sojourn time over a period of time (default: 100ms) exceeds the threshold (default: 5ms), the outgoing link in the queue is considered congested. The Codel code is presented in Algorithm 1.

Algorithm 1 CoDel algorithm

```
1: Function CheckSojournTime(Packet, Now)
2:     sojournTime ← Now – Tag.GetTime
3:     if sojournTime > Target then
4:         OverTargetForInterval ← False
5:       if FirstAboveTime == 0 then
6:           OverTargetForInterval ← False
7:           FirstAboveTime ← Now
8:       else
9:          if Now > (FirstAboveTime + Interval) then
10:             sojourn ← Now
11:             OverTargetForInterval ← True
12:         else
13:             FirstAbiveTime == 0
14:             OverTargetForInterval ← False
15:         end if
16:       end if
17:    end if
18:    return OverTargetForInterval;
19: end function
20: Function DoDequeue(Packet, Now)
21:    Now ← CoDelGetTime()
22:    OkToMark ← CheckSojournTime(Packet,Now)
23:    if OkToMark then
24:       if Now > NextMarkingTime then
25:          MarkNext ← True
26:          NextMarkingTime ← Now
27:    else
28:          MarkedCount ← 0
29:       end if
30:    end if
31: end function
```

C. Explicit Congestion Signaling

We use the same congestion signaling method that CoDel used, ECN marking (Explicit Congestion Notification) [32]. This signaling is done in the downstream direction by explicitly marking the concerned packets to notify the consumer of the link status, i.e., when a router detects congestion on one of its outgoing links, it marks the data packets and explicitly signals this state of congestion to the consumer nodes to reduce their sending rate of interest packets.

ECN marking is done as follows: When congestion occurs, the first packet is marked and the next packets are marked in a marking interval that corresponds to the CoDel drop spacing; This interval starts at "1.1 * the CoDel interval (100ms)" [33]. A congestion notification bit is used by ECN in the packet headers to provide feedback on network congestion. This bit is activated in the PIT entry of the packet when the packet sojourn time exceeds the threshold. Depending on the data packet received (Normal or Marked) at the consumer nodes, the authors adapt the sending rate of interest packets. An advantage of ECN marking is that consumers can be informed of congestion quickly and thus react quickly to the congestion problem.

D. Congestion Window Adjustment

This section describes in detail the algorithm used to adjust the congestion window of EC-Elastic, which is based on the Elastic-TCP mechanism. The principal purpose of this algorithm is to improve overall performance and bandwidth utilization while avoiding packet loss. Algorithm 2 describes the core functionality of EC-Elastic at the consumer node, where the congestion window cwnd is increased when the consumer receives a normal data packet and is decreased when the consumer receives marked packets or Timeouts.

Algorithm 2 Consumer Elastic Algorithm

```
On data reception do
  if no NACK received then
    if slow start then
        cwnd ← cwnd + 1
    else
        RTTcurrent ← (now – sendtime)
        if RTTcurrent > RTTmax then
          RTTmax ← RTTcurrent
        end if
        if RTTcurrent < RTTmin then
          RTTmin ← RTTcurrent
        end if
        WWF ← sqrt(RTTmax / RTT current * cwnd)
        cwnd ← cwnd + WWF / cwnd
    end if
  else
    if slow start then
      cwnd ← cwnd × β1
    else
      cwnd ← cwnd × β2
    end if
    ssthresh ← cwnd - 1
  end if
```

1) Design of the consumer window adjustment algorithm: The basic idea of algorithm 2 is to use the Window-correlated Weighting Function WWF which was proposed in [1] and aims to improve the bandwidth utilization. WWF is based on the variation of RTT (Round Trip Time) according to the following formula:

$$WWF = \sqrt{\frac{RTTmax}{RTT\ current} * cwnd} \quad (1)$$

Where, $RTT_{current}$ is the current RTT obtained from the last ACK, RTT_{max} is the maximum RTT and cwnd is the current congestion window. This function is used in the congestion avoidance phase to increase the congestion window by $cwnd + \frac{WWF}{cwnd}$. However, in the slow start phase, EC-Elastic increases its congestion window by cwnd+1.

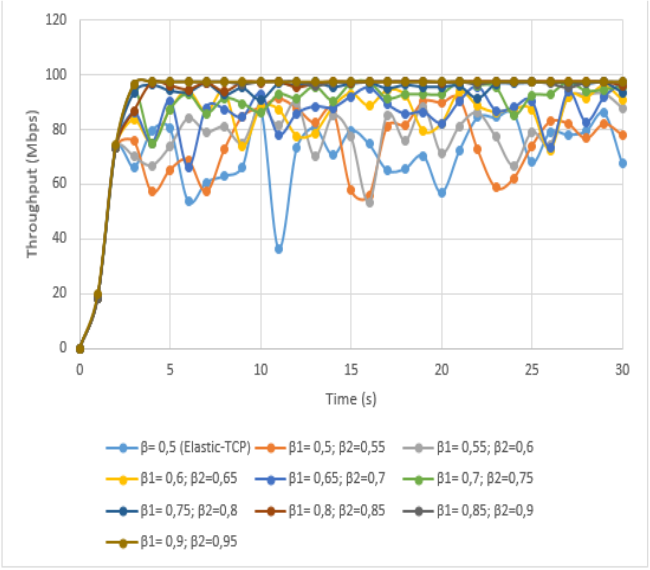

Fig. 2. Throughput of EC-Elastic by Varying the Parameter β.

In case of congestion detection or timeouts, Elastic-TCP [1] applies a multiplicative decrease that halves the cwnd after each loss detection regardless of the phase in which the loss is detected. In contrast, EC-Elastic uses two ways to decrease the cwnd, after any congestion detection. This decrease varies depending on the phase where the loss is detected. As shown in Algorithm 2, if the loss is detected in the slow start phase, EC-Elastic decreases its cwnd to cwnd*β1 of the last cwnd. If the loss is detected in the congestion avoidance phase, EC-Elastic decreases its cwnd to cwnd*β2 of the last cwnd and the ssthresh (the threshold) is reduced to cwnd -1 after any degradation to avoid switching to an undesirable slow start. Since the loss that occurs in the slow start phase is more severe than the loss that occurs in the congestion avoidance phase [34], the value of β1 should therefore always be less than β2 (β1 and β2 are two parameters used for adjusting the size of the congestion window, their values vary between 0 and 1).

Fig. 2 presents the simulation results we conducted on the first scenario (Fig. 3 and Table II) in order to find the most optimal values for choosing the coefficients β1 and β2. This figure shows a comparison between using a multiplicative decrease (as in Elastic-TCP which uses β=0.5 to decrease its congestion window ", a multiplicative decrease is usually equal to 1/2") and using two parameters β1 and β2 in both congestion control phases. According to Fig. 2, with the increase of β_1 and β_2, the throughput also increases and when β_1= 0,9 / β_2=0,95, the throughput is almost the same as that of β_1= 0,85 / β_2=0,9 which indicates that the throughput does not change when the value of β1 is greater than 0,85 and β_2 is greater than 0,9. EC-Elastic performs better in terms of link utilization with the use of the two parameters β_1 and β_2 than the use of multiplicative decrease. Based on the experimental result (Fig. 2), we set β_1 = 0,85 and β_2 = 0,9 in our algorithm.

2) General behavior of EC-elastic: EC-Elastic uses a slow start phase to increase the congestion window at consumer nodes. Then, intermediate routers calculate the sojourn time of each packet in the queue (using CoDel). If this sojourn time exceeds a well-defined threshold, the router marks data packets and sends them explicitly to the consumers to react to this situation. At the consumer nodes, once the first marked packet is received; EC-Elastic reduces its congestion window cwnd by the factor β1 and enters into the congestion avoidance phase which is characterized by using the Window-Correlated Weighting Function (WWF). In this phase, EC-Elastic increases its congestion window cwnd by WFF/cwnd and decreases it by the factor β2 (by receiving a marked packet). However, if a timeout is detected in any phase, EC-Elastic resets its congestion window cwnd to the initial value. The main objective of EC-Elastic in NDN is the same as that of Elastic-TCP in TCP/IP network, to improve bandwidth utilization in NDN networks, where RTTs are long, buffers are very large, and packet losses are very frequent.

V. Performance Evaluation of EC-Elastic

This work focuses on developing a new congestion control mechanism named EC-Elastic that has the capability to increase bandwidth utilization in high-speed NDN networks. Using ndnSIM [35], based on NS-3 and designed specifically for the numerical study of NDN networks, the performance of EC-Elastic is evaluated and compared to three other congestion control algorithms: Agile-SD [34] , CUBIC [36] and STCP [37]. These algorithms have been implemented in NDN, in the same scenarios as EC-Elastic.

A. Simulation Scenarios

1) Scenario 1: one consumer - one producer

Fig. 3 shows the first topology which contains a consumer, a router and a producer.

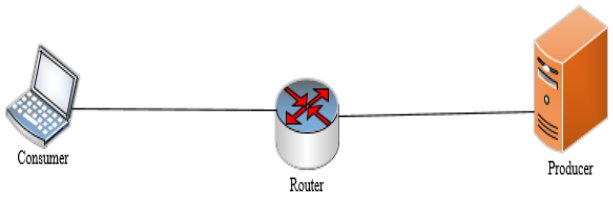

Fig. 3. Simulation Topology 1.

TABLE I. Parameters of Simulation Topology 2

Parameters	Delay	Bandwidth
Consumer - Router	10ms	100Mbps
Router - Producer	10ms	1Gbps

In this scenario, the link bandwidth from the consumer to the router is fixed at 100 Mbps with a 10 ms delay while the link bandwidth from the router to the producer is fixed at 1 Gbps with a 10 ms delay, as illustrated in Table I.

2) Scenario 2: Multiple consumers - multiple producers

In this second topology (Fig. 4), six consumer nodes are connected to six producer nodes via a bottleneck link, consisting of two routers (Router 1 and Router 2).

In this scenario, each link consumer-router is set to 100Mbps with different values of link delay between different nodes in the studied topology (1ms, 10ms, 15ms, 20ms, 25ms and 30ms). The link Router1-Router2 is set to 5Mbps with a delay of 15ms. From Router 2 to producers 1/3/5, the link is set to 20Mbps with delays of 10ms, 5ms and 1ms respectively and from Router 2 to producers 2/4/6, the link is set to 10Mbps with delays of 10ms, 5ms and 1ms respectively as presented in Table II. In this scenario, the consumers request the same content. The time for both simulations is set to 30 seconds.

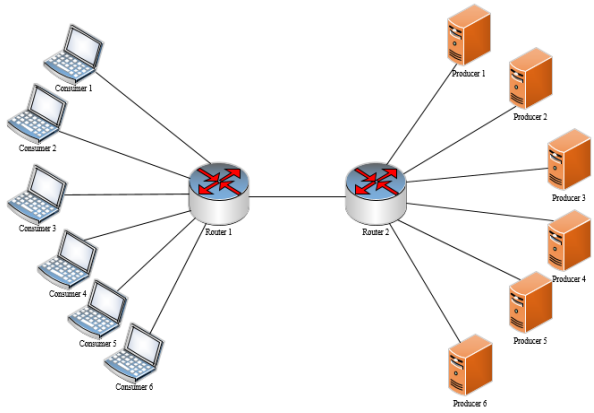

Fig. 4. Simulation Topology 2.

TABLE II. Parameters of Simulation Topology 2

Parameters	Delay	Bandwidth
Consumer1 - Router1	1ms	100Mbps
Consumer2 - Router1	10ms	100Mbps
Consumer3 - Router1	15ms	100Mbps
Consumer4 - Router1	20ms	100Mbps
Consumer5 - Router1	25ms	100Mbps
Consumer6 - Router1	30ms	100Mbps
Router1 - Router 2	15ms	50Mbps
Router2 - Producer1	10ms	20Mbps
Router2 – Producer2	10ms	10Mbps
Router2 – Producer3	5ms	20Mbps
Router2 – Producer4	5ms	10Mbps
Router2 – Producer5	1ms	20Mbps
Router2 – Producer6	1ms	10Mbps

B. Simulation Results

We evaluate the performance of EC-Elastic in both study scenarios described above and are primarily interested in throughput, packet loss rate, and delay. In this study, delay is the time from sending an interest packet to receiving the corresponding data packet. Similarly, throughput, measured in bits per second, designates the number of successfully transmitted data packets from source to destination and changes with the amount of packets transmitted and the amount of packets dropped in the network [38]. Packet loss rate, designates the number of dropped packets per second and is measured as the difference between the amount of packets sent by a node and the amount of packets received by the same node, over a given period of time. In order to have reliable values, all simulations were repeated several times and the results presented in the following are an average of the obtained values.

1) Throughput measurement: Fig. 5 shows a comparison of throughput between EC-Elastic and the three algorithms (Agile-SD, CUBIC, and STCP) in the first study scenario, and Fig. 6 shows a comparison of throughput between EC-Elastic and the three algorithms (Agile-SD, CUBIC, and STCP) in the second study scenario while, Table III shows the throughput of both scenarios.

The objective of the first scenario (Fig. 5) is to study the ability of these mechanisms to fully utilize the available bandwidth. EC-Elastic outperforms the other mechanisms because of its fast cwnd growth resulting from the use of the Window-correlated Weighting Function WWF. It is clear that EC-Elastic can fully utilize the bandwidth and is more stable than Agile-SD, CUBIC and STCP algorithms. In the second scenario (Fig. 6), increasing consumer and producer numbers shows better performance, in terms of throughput, for EC-Elastic than those obtained by the other three algorithms Agile-SD, CUBIC and STCP. In addition, EC-Elastic has a more stable throughput than the other three algorithms in both scenarios.

EC-Elastic achieves the best and most stable throughput performance compared to the other algorithms and this is due to the use of the Window-correlated Weighting Function WWF which aims to maximize the bandwidth usage of the network. The result of this study is that EC-Elastic has the capability to perceive and predict rapidly, deal with the variation of bandwidth and adapt to NDN characteristics.

The necessity of the proposed mechanism was raised because of the incapacity of the existing mechanisms to fully utilize the available bandwidth on high speed networks where RTTs are very long and large buffers are used.

TABLE III. THROUGHPUT OF SCENARIOS 1 AND 2

Algorithms	EC-Elastic	CUBIC	Agile-SD	STCP
Scenario 1	93,778	24,282	74,116	61,366
Scenario 2	113,62	37,26	108,96	88,37

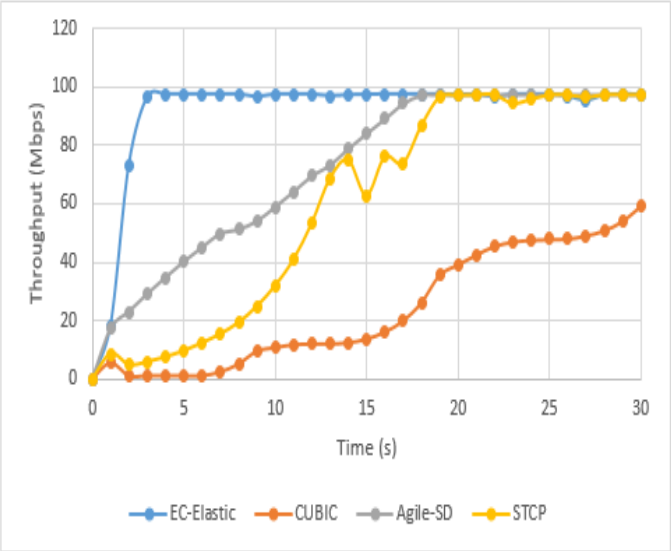

Fig. 5. Throughput of Scenario 1.

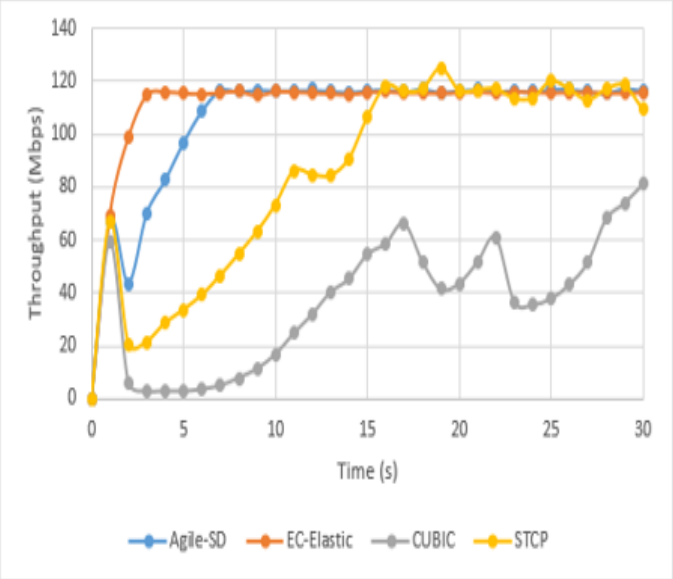

Fig. 6. Throughput of Scenario 2.

2) Packet loss rate measurement: The main objective of each congestion control mechanism is to maximize throughput and minimize packet loss rate. Table IV shows the packet loss rate in both scenarios. The results obtained from our numerical study, show almost identical performance for the four algorithms compared, with almost negligible packet loss rate because the use of CoDel queueing reacts before the queue reaches its limit and also reacts quickly to the congestion problem by marking packets and notifying the consumer to reduce their sending rate of interest packets. In addition, explicit congestion marking reduces retransmissions, because by notifying the consumer of the link status in case of congestion, the packet received by the consumer also contains the requested data.

TABLE IV. PACKET LOSS RATE OF SCENARIOS 1 AND 2

Algorithms	EC-Elastic	CUBIC	Agile-SD	STCP
Scenario 1	0,001	0,032	0,055	0,032
Scenario 2	0	0	0	0

3) Delay measurement: Fig. 7 and 8 show the delay measurement for Scenarios 1 and 2, respectively, and Table V shows the delay measurement for both scenarios.

For the first scenario (Fig. 7), we observe that EC-Elastic, Agile-SD, and Cubic show a lower delay measure than STCP and this measure becomes almost the same for all the mechanisms when we exceed 5s of the simulation. In the second scenario (Fig. 8), we observe that EC-Elastic and CUBIC have a lower average delay measure than that measured by STCP and Agile-SD.

The exponential increase in delay between seconds 1 and 5 is due to all algorithms rapidly increasing their congestion window at the end of the slow start phase to ensure full utilization of the available bandwidth before switching to the congestion avoidance phase, resulting in problems such as packet loss, and thus increasing the delay between sending a packet of interest and receiving its corresponding data packet. As a result, our mechanism EC-Elastic ensures a reasonable packet transmission delay.

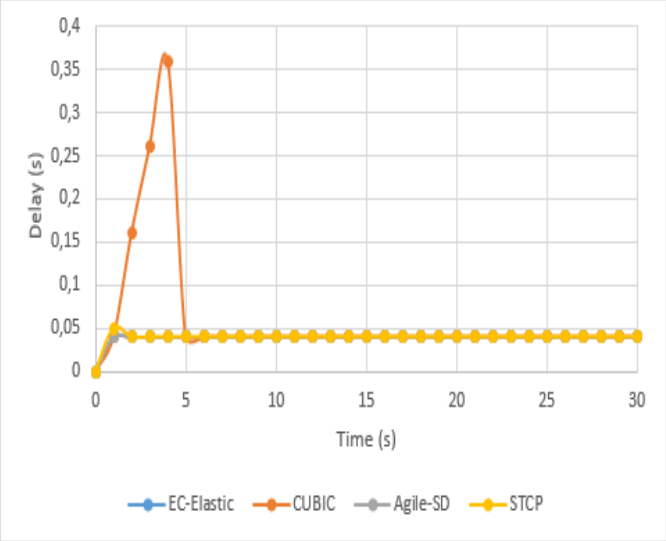

Fig. 7. Delay Analysis of Scenario 1.

TABLE V. DELAY MEASUREMENT OF SCENARIOS 1 AND 2

Algorithms	EC-Elastic	CUBIC	Agile-SD	STCP
Scenario 1	0,04	0,062	0,04	0,04
Scenario 2	0,06	0,054	0,06	0,056

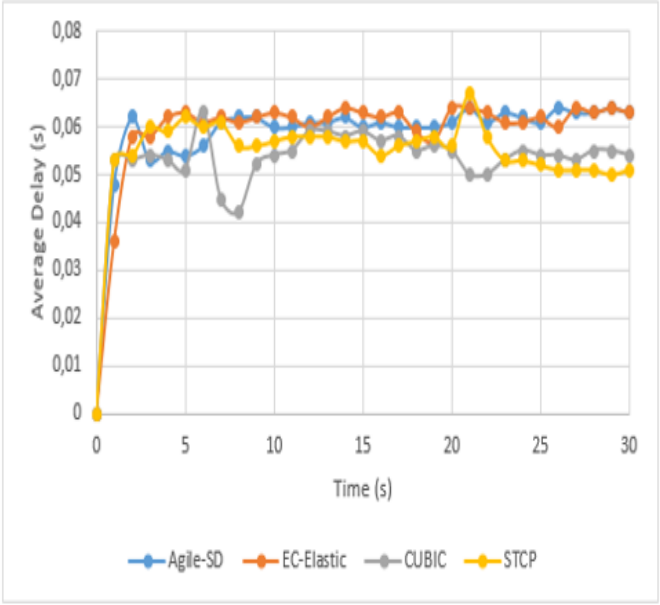

Fig. 8. Avg Delay Analysis of Scenario 2.

The numerical study performed in this work reveals that our algorithm EC-Elastic seems to give better performance, in terms of throughput, compared to those of Agile-SD, CUBIC and STCP algorithms. Thus, our algorithm can continuously fully utilize the bandwidth of the sources while keeping the delay and packet loss rate lower.

VI. CONCLUSION

This paper proposes EC-Elastic, a hybrid congestion control mechanism for NDN, to avoid congestion, increase bandwidth utilization and achieve efficient data delivery. EC-Elastic is based on the use of Window Correlated Weighting Function (WWF) which aims to improve bandwidth utilization in the network. At the intermediate routers, EC-Elastic uses AQM-CoDel queue to measure the packet sojourn time and explicitly signals congestion to inform the consumer to decrease its sending rate of interest packets, and then at the consumer nodes, EC-Elastic adopts the basic idea of Elastic-TCP to control the sending rate of interest packets. The conducted numerical study of the performance of EC-Elastic compared to Agile-SD, CUBIC and STCP in terms of throughput, packet loss rate, and delay shows that EC-Elastic can significantly improve bandwidth utilization while maintaining lower delay and packet loss rates. EC-Elastic can be a promising solution to enhance bandwidth utilization on high speed networks where RTTs are very long and large buffers are used. As a future work, we plan to implement EC-Elastic in the Internet of Health Things (IoHT) to evaluate its performance in more complex scenarios where sensitive patient data becomes a critical component of healthcare that requires ensuring its timely delivery while avoiding congestion and data loss.

REFERENCES

[1] Alrshah MA, Al-Maqri MA, Othman M. Elastic-TCP: Flexible Congestion Control Algorithm to Adapt for High-BDP Networks. IEEE Syst J. 2019;:1–11.

[2] Rhee I, Xu L, Ha S, Zimmermann A, Eggert L, Scheffengger R. CUBIC for Fast Long-Distance Networks. 2019;1:105–12.

[3] Dong M, Li Q, Zarchy D, Godfrey PB, Schapira M. PCC: Re-architecting congestion control for consistent high performance. Proc 12th USENIX Symp Networked Syst Des Implementation, NSDI 2015. 2015;:395–408.

[4] Khelifi H, Luo S, Nour B, Moungla H. LQCC: A Link Quality-based Congestion Control Scheme in Named Data Networks. IEEE Wirel Commun Netw Conf WCNC. 2019;2019-April:1–6.

[5] Ahlgren B, Dannewitz C, Imbrenda C, Kutscher D, Ohlman B. A survey of information-centric networking. IEEE Commun Mag. 2012;50:26–36. doi:10.1109/MCOM.2012.6231276.

[6] Zhang L, Estrin D, Burke J, Jacobson V, Thornton J, Diana K, et al. Named Data Networking (NDN) Project Named Data Networking (NDN) Project. 2010; May.

[7] El-bakkouchi A, Bouayad A, ELBekkali M. A hop-by-hop Congestion Control Mechanisms in NDN Networks – A Survey. 2019 7th Mediterr Congr Telecommun. 2019;:1–4.

[8] El-Bakkouchi A, Ghazi M El, Bouayad A, Fattah M, Bekkali M El, Mazer S. Packet Loss and Delay Measurement Analysis of TCP Variants in NDN Congestion Control. 2020 1st Int Conf Innov Res Appl Sci Eng Technol IRASET 2020. 2020;:2–6.

[9] Ren Y, Li J, Shanshan Shi, Lingling Li, Guodong Wang. An explicit congestion control algorithm for Named Data Networking. In: 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE; 2016. p. 294–9. doi:10.1109/INFCOMW.2016.7562089.

[10] Schneider K, Yi C, Zhang B, Zhang L. A Practical Congestion Control Scheme for Named Data Networking. doi:10.1145/2984356.2984369.

[11] Ren Y, Li J, Shi S, Li L, Wang G, Zhang B. Congestion control in named data networking – A survey. Comput Commun. 2016;86:1–11. doi:10.1016/j.comcom.2016.04.017.

[12] Mejri S, Touati H, Kamoun F. Hop-by-hop interest rate notification and adjustment in named data networks. IEEE Wirel Commun Netw Conf WCNC. 2018;2018-April:1–6.

[13] Carofiglio G, Gallo M, Muscariello L. ICP: Design and evaluation of an Interest control protocol for content-centric networking. In: 2012 Proceedings IEEE INFOCOM Workshops. IEEE; 2012. p. 304–9. doi:10.1109/INFCOMW.2012.6193510.

[14] Salsano S, Detti A, Cancellieri M, Pomposini M, Blefari-Melazzi N. Transport-layer issues in information centric networks. ICN'12 - ACM Proc Information-Centric Netw Work. 2012; August:19–24.

[15] Arianfar S, Nikander P, Eggert L, Wong W. ConTug: A Receiver-Driven Transport Protocol for Content-Centric Networks. http://users.piuha.net/blackhawk/contug/contug.pdf. Accessed 12 Mar 2018.

[16] Saino L, Cocora C, Pavlou G. CCTCP: A Scalable Receiver-driven Congestion Control Protocol for Content Centric Networking. https://pdfs.semanticscholar.org/c539/1eb884f234a40ffcff8a7ad2a8989ee13b79.pdf. Accessed 11 Mar 2018.

[17] Braun S, Monti M, Sifalakis M, Tschudin C. An Empirical Study of Receiver-Based AIMD Flow-Control Strategies for CCN. In: 2013 22nd International Conference on Computer Communication and Networks (ICCCN). IEEE; 2013. p. 1–8. doi:10.1109/ICCCN.2013.6614106.

[18] Carofiglio G, Gallo M, Muscariello L. Joint Hop-by-hop and Receiver-Driven Interest Control Protocol for Content-Centric Networks. https://conferences.sigcomm.org/sigcomm/2012/paper/icn/p37.pdf. Accessed 11 Mar 2018.

[19] XIA Y, WANG L, HOU F, WANG Y. Congestion Control for Content-Centric Networking Based on Protocol-Oblivious Forwarding. DEStech Trans Comput Sci Eng. 2017; wcne.

[20] Zhang F, Zhang Y, Reznik A, Liu H, Qian C, Xu C. A transport protocol for content-centric networking with explicit congestion control. In: 2014 23rd International Conference on Computer Communication and Networks (ICCCN). IEEE; 2014. p. 1–8. doi:10.1109/ICCCN.2014.6911765.

[21] Yi C, Afanasyev A, Moiseenko I, Wang L, Zhang B, Zhang L. A case for stateful forwarding plane. Comput Commun. 2013;36:779–91. doi:10.1016/j.comcom.2013.01.005.

[22] Kato T, Bandai M. Congestion control avoiding excessive rate reduction in named data network. 2017 14th IEEE Annu Consum Commun Netw Conf CCNC 2017. 2017;:108–13.

[23] Kato T, Bandai M. Avoiding excessive rate reduction in rate based congestion control for named data networking. J Inf Process. 2018;26:29–37.

[24] Nguyen D, Fukushima M, Sugiyama K, Tagami A. Efficient multipath forwarding and congestion control without route-labeling in CCN. 2015 IEEE Int Conf Commun Work ICCW 2015. 2015; Scpa:1533–8.

[25] Muchtar F, Al-Adhaileh MH, Alubady R, Singh PK, Ambar R, Stiawan D. Congestion Control for Named Data Networking-Based Wireless Ad Hoc Network. Springer Singapore; 2020. doi:10.1007/978-981-15-3369-3_10.

[26] Alwahab DA, Laki S. A simulation-based survey of active queue management algorithms. ACM Int Conf Proceeding Ser. 2018;:71–7.

[27] Floyd S, Jacobson V. Random Early Detection Gateways for Congestion Avoidance. IEEE/ACM Trans Netw. 1993;1:397–413.

[28] Nichols K, Jacobson V. Controlling queue delay. Commun ACM. 2012;55:42. doi:10.1145/2209249.2209264.

[29] Pan R, Natarajan P, Piglione C, Prabhu MS, Subramanian V, Baker F, et al. PIE: A lightweight control scheme to address the bufferbloat problem. IEEE Int Conf High Perform Switch Routing, HPSR. 2013;:148–55.

[30] Wang M, Yue M, Wu Z. WinCM: A Window based Congestion Control Mechanism for NDN. 2019; HotICN:80–6.

[31] Liu Y, Piao X, Hou C, Lei K. A CUBIC-Based explicit congestion control mechanism in named data networking. Proc - 2016 Int Conf Cyber-Enabled Distrib Comput Knowl Discov CyberC 2016. 2017;:360–3.

[32] Floyd S. TCP and explicit congestion notification. ACM SIGCOMM Comput Commun Rev. 1994;24:8–23.

[33] Nichols K, Pollere I, Jacobson V, A. McGregor E, J. Iyengar E. Controlled Delay Active Queue Management draft-ietf-aqm-codel-03. J Chem Inf Model. 2016;53:1689–99.

[34] Alrshah MA, Othman M, Ali B, Hanapi ZM. Agile-SD: A Linux-based TCP congestion control algorithm for supporting high-speed and short-distance networks. J Netw Comput Appl. 2015;55 August 2018:181–90. doi:10.1016/j.jnca.2015.05.011.

[35] Mastorakis S, Afanasyev A, Moiseenko I, Zhang L. ndnSIM 2 . 0 : A new version of the NDN simulator for NS-3. NDN Proj. 2015;:1–8.

[36] Ha S, Rhee I, Xu L. CUBIC: A new TCP-friendly high-speed TCP variant. Oper Syst Rev. 2008;42:64–74.

[37] Kelly T. Scalable TCP: Improving performance in highspeed wide area networks. Comput Commun Rev. 2003;33:83–91.

[38] El-Bakkouchi A, El Ghazi M, Bouayad A, Fattah M, El Bekkali M. Performance analysis of TCP variants in named data networking. Int J Adv Trends Comput Sci Eng. 2020;9 1.5 Special Issue:13–20.