Average Delay-based early Congestion Detection in Named Data of Health Things

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*—The Internet of Health Things (IoHT) is receiving more attention from researchers because of its wide use in the healthcare field. IoHT refers to medical devices whose main purpose is to transmit health data in a secure and lossless manner between them and healthcare personnel. However, in a medical emergency, sensors transmit vital patient data simultaneously and frequently, increasing the risk of congestion and packet loss. This problem is highly undesirable in an IoHT system, leading to undesirable results. To address this issue, a new approach based on Named Data Networking (NDN) (which is considered as the most appropriate internet architecture for IoT systems) is proposed to control congestion in IoHT systems. The proposed approach, Average delay-based early congestion Detection (ADCD), detects and controls congestion at consumer nodes by calculating the average queuing delay based on the one-way delay similar to that proposed in Sync-TCP. Then according to the calculated value, ADCD divides the network into three states: no-congested state, less congested state, and heavily congested state. The adjustment of the congestion window size is done according to the state of the network. ADCD was implemented in ndnSIM and compared to the Interest Control Protocol ICP. The results show that ADCD maximizes bandwidth utilization compared to ICP and maintains a reasonable delay.

Keywords*—*Named data networking; internet of health things; congestion control; congestion detection

I. INTRODUCTION

The Internet of Things (IoT) is a collection of intelligent devices that can communicate, interact and exchange information with each other [1]. It interconnects billions of small devices to deliver information at any time and across the world [2]. IoT has been primarily applied in several domains like healthcare, smart home, smart cities, industries, transportation and logistics, etc. In healthcare, IoHT has been proposed as an extended version of IoT to connect people and smart objects in the medical field [3]. It refers to medical devices, healthcare sensors and intelligent biomedical applications that enable the exchange and treatment of various types of healthcare data with each other and with healthcare people [3], whose main objective is to transmit health data securely and without loss. This data can be patient monitoring data, patient analyses, consultations or even sensitive data such as heart rate and breathing.

In the case of a medical emergency or patient vital signs monitoring, sensors implanted on patients detect and transmit vital patient data simultaneously and frequently, which increases the risk of congestion and consequently packet loss. In IoHT, congestion is highly undesirable and can lead to undesirable outcomes such as patient death. However, ensuring that packets arrive at their destination in time guarantees patients' safety and survival [4]. Due to the importance and sensitivity of the transmitted data, congestion should be avoided as much as possible and controlled if this is not possible.

However, the IP protocol stack on which the IoT is based was designed for a different purpose and cannot handle the important challenges caused by the heterogeneity of the devices and the importance of the traffic generated, highlighting the limitations of IP. However, the current IP solutions are working hard to support IoT systems, but the gaps in IP are still hard to hide. Along with efforts to adapt IP to the IoT and other content-based architectures, Information Centric Networking (ICN) [5] promises to support IoT systems and emerging internet applications natively. It is a switch from a host-centric communication model to a content-centric system, caching at intermediate nodes, and the use of multiple paths and sources. With its features, ICN has the potential to be a reliable framework for IoT by connecting billions of heterogeneous constrained objects. Indeed, ICN provides easy access to data and reduces data recovery time and the load on data producers. NDN [6] is considered as the appropriate ICN architecture for IoT systems among several ICN architectures.

NDN is a new internet architecture based on the content of data rather than its IP address. NDN defines three roles: Consumer, Router and Producer and offers two types of packets; the interest packet, containing the required content name and the data packet, containing the required content. There are three elements to every NDN node as shown in Fig. 1; Content Store (CS), Pending Interest Table (PIT) and Forwarding Interest Base (FIB). The CS stores copies of the content that passes through it so that future demands for the same content can be satisfied. The PIT is a table that preserves the records of incoming and outgoing interest and data packets. The FIB transfers interest and data packets between nodes through routing protocols [7]. The data transfer process between NDN nodes is as follows, the consumer asks for content by sending an interest packet.

*Corresponding Author.

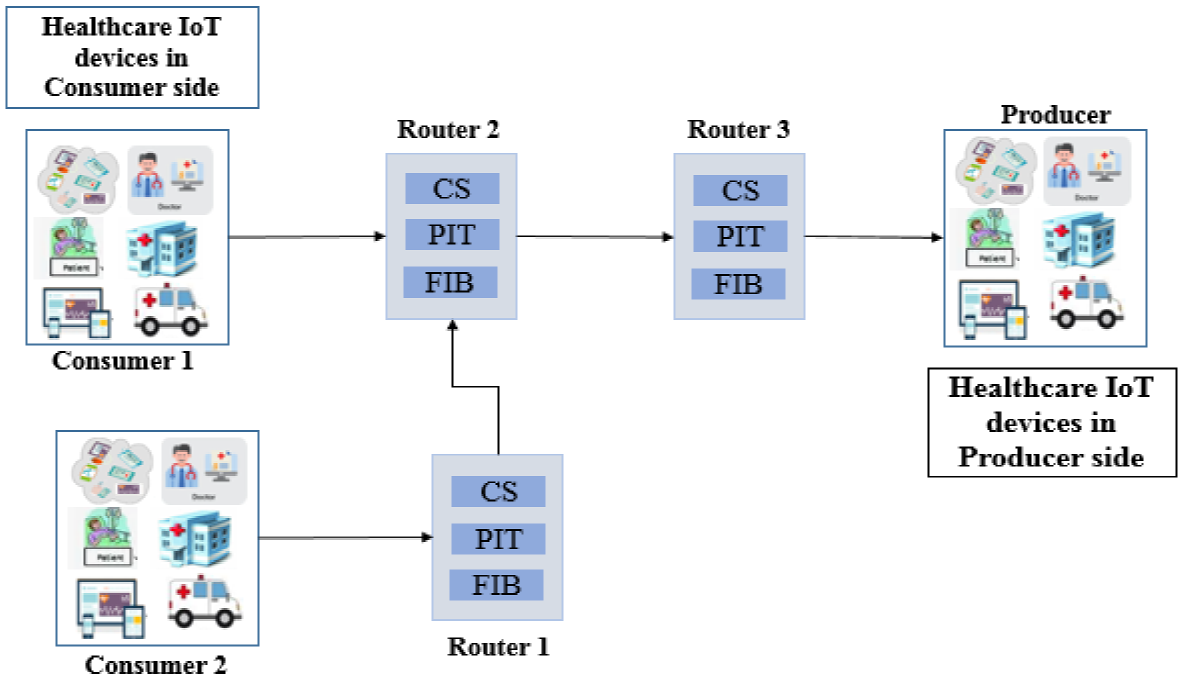
Healthcare IoT devices in Consumer side
Consumer 1
Router 2
CS
PIT
FIB
Router 3
CS
PIT
FIB
Producer
Healthcare IoT devices in Producer side
Consumer 2
CS
PIT
FIB
Router 1

Fig. 1. NDN Information Distribution in Case of IoHT Environment.

The routers use FIBs to forward this packet to the content producer and create a PIT entry list on each router to define the reverse path. Then, the producer transfers the related content via the reverse path to the consumer, and the CS saves the content that traverses it for future utilization [8].

An example of the data transfer process between NDN nodes is shown in Fig. 1. Suppose that consumer one starts this process and sends an interest packet containing the requested name. This interest packet is transferred to the producer using the FIB services. The requested content is put into a data packet and is stored in the CS of router three and router two for future utilization. Subsequently, if consumer two requires the same content, its interest packet is locally satisfied from the CS of router two without the need to forward this packet to the original content producer. In the case of IoHT, the consumer can be a doctor, a patient, a nurse, a medical application, a hospital or any medical or IoT device that desires to obtain data/information concerning the health field. The producer can also be a doctor, a patient, a nurse, a medical application, a hospital or any medical or IoT device that has data/information concerning the health field. The router's mission is to route the data/information concerning the health field between the consumer and the producer.

To address the problem of congestion in healthcare field, and as NDN congestion control mechanisms that are based on the estimation of RTTs as the main indication of congestion are not reliable in NDN-IoHT because the RTT value changes frequently due to caching or multipath. This paper proposes a new approach based on Sync-TCP [9] (which has been proposed to control congestion based on measurements of the One-way Transit Time (OTT) between senders and receivers of packets) to control congestion in NDN-IoHT. OTTs are more accurate in reflecting queue delay resulting from network congestion than RTTs.

In our approach, congestion is detected by calculating the average queuing delay based on the one-way delay similar to that proposed in Sync-TCP [9]. Then according to the calculated value, the network is divided into three states; no-congested state, less-congested state and heavily congested state. The adjustment of the congestion window size is made according to the state of the network. Increased in case of a no-congested network and decreased in a less-congested and heavily congested network. The choice of congestion control at the consumer nodes comes from the fact that the method used for congestion detection is based on variations of OTTs which is more reliable at the consumer node than at the router node because consumers are able to estimate the one-way delay of the data packet using the transmission time available in its header and the reception time of this packet. The rest of the paper is organized as follows: Section II presents background and related work, Section III describes the proposed method, Section IV presents results and discussion and Section V concludes the article.

II. Background and Related Work

A. NDN for IoT

ICN has been proposed as a promising future internet architecture to fill the gaps in the CoAP/RPL/6LowPan/ 802.15.4 protocol stack on which the IoT system is based and improve its deployment and data distribution [10]. Among the ICN architectures proposed in the literature, NDN is considered as the most appropriate ICN architecture for IoT systems. In effect, the characteristics of NDN, namely hierarchical naming of unique content that is independent of its location, caching in intermediate routers, multipath, multisource, name-based routing, support for user mobility, the use of encryption for better access control, make it a very suitable platform for IoT system traffic and applications.

Several works have been done on this topic to show that NDN is the most suitable and promising architecture for IoT. In [11], the authors suggested that NDN can meet IoT requirements and is the most suitable architecture for IoT scenarios. In [12], the authors showed that the semantics of NDN meets the requirements of IoT applications and its main challenges. They also showed that the communication model used by NDN "based on the content name" allows IoT networks an easy deployment and configuration. In [13], the authors proposed a comparative study of ICN architectures in an IoT context and concluded that NDN is an architecture most suitable for IoT systems. In [2], the authors proposed an NDN integration in IoT devices and then evaluated this proposal in a Smart Farming application scenario to prove that NDN is an architecture most suitable for IoT systems. In [14], the authors described the advantages of the NDN architecture compared to the current IP internet architecture for IoT systems and then explained how NDN can be included in an IoT architecture. In [15], the authors discussed the main characteristics of NDN in IoT, then they proposed an IoT architecture via NDN named IoT-NDN for various IoT domains. Some studies have discussed the requirements of IoT and how ICN architectures support them without examining which of these architectures is most appropriate for IoT.

B. *Related Work*

In NDN architecture, the problem of network congestion is the subject of much active research focused on minimizing transit delays and reducing packet loss caused by the transfer capability of routers. In NDN architecture, data packets are the leading cause of congestion because they are much larger than interest packets. Therefore, to control congestion, many research works propose controlling the sending rate of interest packets to limit the returning rate of data packets. Several congestion control mechanisms have been proposed in the literature, classified as receiver-based control, Hop-by-hop control, and Hybrid control [16]. In this paper, we present some NDN congestion control works. Among the first interest control protocols proposed for NDN is the Interest Control Protocol (ICP) [17], which detects congestion at the consumer node by measuring delay and timer expirations. The window size adjustment is made at the receiver level using the AIMD (Additive-Increase Multiplicative-Decrease) mechanism. If RTO (Retransmission Time-Outs) is triggered, the consumer decreases its congestion window by MD (Multiplicative Decrease). Otherwise, it increases its congestion window by AI (Additive Increase). The authors of [18] proposed DCP "Delay-based Congestion Control Protocol" based on the window and the receiver. DCP detects congestion based on the value of the queue delay. If this delay is below a given threshold, DCP considers that the link is not congested. Otherwise, DCP considers the link is congested. The calculation of queuing delay is done by measuring the delay returned by the producer or intermediate nodes along the path of the transmitted data packets. DCP uses a linear controller to adjust the congestion window. In [19], the authors proposed a hop-by-hop congestion control mechanism (HCCM) based on explicit notification of interest packet rates. This mechanism detects congestion by calculating the queue length of the interest packets of the output interface. According to the value found, each router between the congested node and the consumer can adjust the sending rate of interest packets. The authors used two levels in the queue, qmax and qmin, representing the maximum and minimum queue occupancy thresholds respectively. Once the queue reaches one of the levels (qmax or qmin), the router sends a notification to the downstream node to inform it of the congestion state and the regulation of the sending rate of interest packets. In [7], the authors proposed EC-Elastic, an Explicit Congestion Control Mechanism that detects congestion at the routers by measuring the sojourn time of packets in the queue using CoDel-AQM algorithm and then, according to this value, the router marks the concerned data packets to inform the consumer nodes to reduce their sending rate of interest packets. At the consumer node, if the phase is a slow start, the congestion window is increased by one and decreased by $\beta_{1,}$ and if the phase is the congestion avoidance, the congestion window is increased by $\frac{WWF}{cwnd}$ using the Window-correlated Weighting Function WWF of Elastic-TCP [20] and decreased by β2. In [21], the authors proposed the MPCC Multipath Congestion Control mechanism, which is based on two principles: Multipath discovery that tags each sub-path with a path tag in the forwarding process and then based on these tags, a tag-aware forwarding strategy has been proposed to discover and manage sub-paths. For multipath congestion control, the authors proposed a Multipath Window Adaptation Control (MWAC) scheme to control the congestion window. In [22], the authors proposed DPCCP Delay-based Path-specified Congestion Control Protocol based on three modules, namely: The congestion estimation module, which aims to measure the number of backlogged packets for every sub-flow using RTT and baseRTT, where the number of backlogged packets measured for a sub-flow is the product of the sub-flow queuing delay and the sub-flow rate. The fairness control module which is used to calculate the target number of backlogged packets for each sub-flow to equalize the aggregate queuing delay [23] and the flow control module, which aims to adjust the rate based on the queuing delay and the target number of packets in the queue using the Adaptive Additive Increase Additive Decrease (A-AIAD) algorithm.

Different from prior works, this paper proposes a new approach that controls congestion by calculating the average queuing delay based on the one way delay similar to that proposed in Sync-TCP [9]. Then according to the calculated value, the network is divided into three states; no-congested state, less-congested state and heavily congested state for efficient and lossless deployment in an IoHT environment. The adjustment of the congestion window size is made according to the state of the network. Increased in case of a no-congested network and decreased in a less-congested and heavily congested network.

III. The Proposed Method

A. *Motivation*

The NDN paradigm has several features, such as caching in intermediate routers that serve future requests for the same content without going through the content producer. This feature reduces the content retrieval time and the charge on the content producer. The use of multiple paths and multisource to avoid congestion of one path over another and to divert requests in the event of a congested path. However, the use of RTT (Round-Trip Time) estimates as a primary indication of congestion are unreliable in NDN-IoHT because the value of RTT frequently changes due to caching and multipath causing problems for congestion control mechanisms that are affected by RTT variation particularly in the event of large RTTs which are treated as losses despite having no packet losses, thus halving the rate of sending packets of interest and consequently more time to fully utilize the bandwidth.

An explicative example of the inefficiency of this congestion detection method was discussed in [18]. In this article, the authors shows that the increase in RTT along the path can be caused by the PIT congestion of one of the routers and not by the data path congestion. In addition, when using caching, the content retrieved from the cache has a shorter RTT. This situation can lead to an erroneous increase in the congestion window size because a short RTT value is perceived as a sign of bandwidth availability, which can cause packet loss [18].

Our approach is motivated by these problems of considering RTT as an indication of congestion, which is not tolerated in an IoHT environment because the transmitted data are important and critical and the recovery time plays a crucial role in this domain. In an NDN-IoHT environment, a congestion control mechanism should achieve the following goals: Avoid congestion if possible and, in cases where congestion cannot be avoided, control it; Ensure delivery of healthy data transmitted in the network and minimize packet loss; Utilize available bandwidth and maintain low packet delivery delay.

B. Congestion Detection

The congestion detection method of ADCD is inspired by the Sync-TCP algorithm proposed in [9]. It detects congestion based on One-way Transit Time (OTT) measurements. OTTs are more accurate in reflecting queue delay resulting from network congestion than RTTs. In NDN, each node (router or producer) adds the time of its transmission in the header of each data packet so that the consumer can estimate the one-way delays. When the consumer receives this packet, it can estimate the one-way delay of the data packet using the transmission time available in its header and the reception time of this packet by subtracting the reception time of the packet from the sending time of that packet as follows:

$$OTT = A_T - T_T \quad (1)$$

where, A_T refers to the data packet arrival time and T_T refers to the transmission time of this data packet. Similar to DCP [18], we are focused only on the measurement of the relative delay between data packets to know whether the content is served from a new data producer or not. ADCD uses changes in queuing delay of a path to detect congestion. For each data packet received, the consumer ADCD obtains a new estimate of the queuing delay and then calculates it average. The average queuing delay is based on the changes in the current queuing delay measure as follows:

$$AvgQDelay = 0.875 * AvgQDelay + 0.125 * CurrQDelay \quad (2)$$

Where CurrQDelay is the current queuing delay, it is estimated by taking the minimum observed OTT and subtracting it from the current delay as follows:

$$CurrQDelay = \text{Current\_delay} - OTT_{min} \quad (3)$$

Where Current_delay is the one-way delay incurred by data packets and OTT_{min} is the minimum one-way delay. The current queuing delay CurrQDelay is calculated for each received data packet once the minimum OTT and current_delay are updated, as shown in Algorithm 1. Once AvgQDelay is calculated, ADCD situates it in one of the network states.

Algorithm 1 AvgQDelay Estimation Algorithm

```
1: **Function** ONDATA(DataPacket)
2: Current_delay 🡠 CurrentTime
3: $OTT_{min}$ 🡠 min ($OTT_{min}$, Current_delay)
4: $OTT_{max}$ 🡠 max ($OTT_{max}$, Current_delay)
5: $OTT_{mid}$ 🡠 ($OTT_{min}$ + $OTT_{max}$) /2
6: CurrQDelay 🡠 Current_delay - $OTT_{min}$
7: AvgQDelay 🡠 0.875 *AvgQDelay + 0.125 * CurrQDelay
8: **end function**
```

ADCD considers three network congestion states; no-congested state, less-congested state and heavily congested state. ADCD determines these three states based on the changes in the received OTTs and adjusts the congestion window size cwnd according to the degree of congestion in the network. The role is as follows:

$AvgQDelay \leq OTT_{mid}$ no-congested state

$OTT_{mid} < AvgQDelay < OTT_{max}$ less-congested state

$AvgQDelay \leq OTT_{max}$ heavily congested state

If $AvgQDelay \leq OTT_{mid}$, ADCD considers the network non-congested. In this status, ADCD has a low queuing delay, a sign of bandwidth availability. In this case, the ADCD consumer increases its congestion window according to the network phase (Slow Start or Congestion Avoidance) to utilize the available bandwidth.

If $OTT_{mid} < AvgQDelay < OTT_{max}$, ADCD considers the network low congested. As the risk of packet loss is low, the congestion window is decreased by the factor $\beta_{1.}$

If $AvgQDelay > OTT_{max}$, ADCD considers the network heavily congested and decreases its congestion window by the factor β2.

Where, OTT_{max} is the maximum one-way delay and OTT_{mid} is the mid-point between OTT_{max} and OTT_{min}. The values of OTT_{max}, OTT_{min} and OTT_{mid} are updated whenever a data packet is received by the consumer according to the following formulas:

$$OTT_{min} = \min(\text{Current\_delay}, OTT_{min}) \quad (4)$$

$$OTT_{max} = \max(\text{Current\_delay}, OTT_{max}) \quad (5)$$

$$OTT_{mid} == (OTT_{min} + OTT_{max})/2 \quad (6)$$

C. Adjustment of the Congestion Window

When the consumer calculates AvgQDelay and locates the status of the network, it proceeds to the adjustment of the congestion window size. ADCD controls the congestion window size in two phases Slow Start and Congestion Avoidance. ADCD adopts the Additive Increase mechanism in Slow Start phase and increases its congestion window by one while cwnd <= ssthresh, where ssthresh is a predefined threshold. In the congestion avoidance phase, which corresponds to cwnd > ssthresh, ADCD adopts the Window-correlated Weighting Function WWF of [20] and increases its congestion window by $\frac{WWF}{cwnd}$. The objective of using this function is to improve bandwidth utilization. Its formula is as follows:

$$WWF = \sqrt{\frac{\text{RTTmax}}{\text{RTT current}} * \text{cwnd}} \tag{7}$$

where $\text{RTT}_{\text{current}}$ is the current RTT, RTT_{max} is the maximum RTT and cwnd is the last congestion window size.

ADCD starts the communication with the Slow Start phase and increases its congestion window by one by sending an interest packet to the network. Once the corresponding data packet is received, the consumer extracts the sending time and calculates AvgQDelay according to equation 2. If AvgQDelay is less than or equal to OTT_{mid}, the network is considered not congested. In this case, the consumer continues to increase the congestion window exponentially to use the available bandwidth fully. Otherwise, the network is considered less congested if AvgQDelay is between OTTmid and OTTmax. In this case, the consumer decreases its congestion window by the factor $\beta_1 = 0.8$. Otherwise, if AvgQDelay exceeds OTT_{max}, the network is considered heavily congested, and the consumer decreases its congestion window by the factor $\beta_2 = 0.5$ and enters into the congestion avoidance phase to slowly increase its congestion window. In this phase, the congestion window increases by using the WFF function. If the consumer receives a NACK packet or TimeOut, The congestion window is divided by two using a multiplicative decrease. Algorithm 2 summarizes the congestion window adjustment steps at the consumer node.

Algorithm 2 CWND Adjustment Algorithm

```
1: On data reception do
2: if slow start then
3: cwnd ← cwnd + 1
4: else
5: cwnd ← cwnd + $\frac{WWF}{cwnd}$
6: end if
7: if NACK or TimeOut received then
8: cwnd ← cwnd / 2
9: end if
```

IV. Results and Discussion

In this section, we present the experiment's parameters and then evaluate the performance of the proposed mechanism ADCD in different scenarios using ndnSIM [24], an NS3-based simulator that has been proposed for NDN networks. We compare the performance of ADCD to that of ICP [17] in terms of throughput, delay, and packet loss rate.

A. Simulation Parameters

The first scenario is shown in Fig. 2. It represents the ideal case of a non-congested network for evaluating the performance of ADCD to demonstrate the high bandwidth utilization that can be achieved under the optimal conditions. It contains a consumer, a router, and a content producer. The bandwidth of the consumer-router path is 60 Mbps with a delay of 10ms, and the bandwidth of the router-producer path is 100 Mbps with a delay of 10ms. The second scenario is illustrated in Fig. 3. This scenario contains three consumers who request the same content, two routers, and two content producers. The three consumers have an equal bandwidth of 50Mbps with different delays, 10ms for consumer1, 5ms for consumer2, and 10ms for consumer3. The size of the transmitted data packet is 1024 bytes, and the simulation time was in the 30s.

B. Throughput

Throughput refers to the total number of packets successfully transmitted between the source and the destination every second. Fig. 4 compares the throughput between ADCD and ICP in the first scenario, while Fig. 5 compares ADCD and ICP in the second scenario. In both figures, time in seconds is defined on the x-axis, and throughput in Mbps is defined on the y-axis.

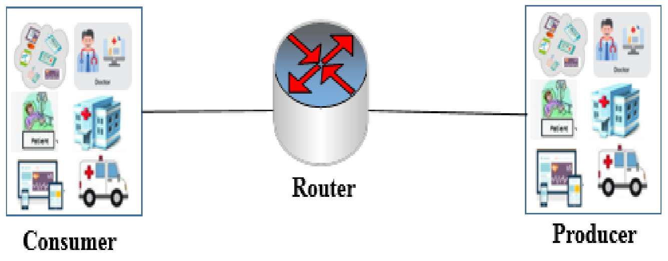

Fig. 2. Topology of the 1st Scenario.

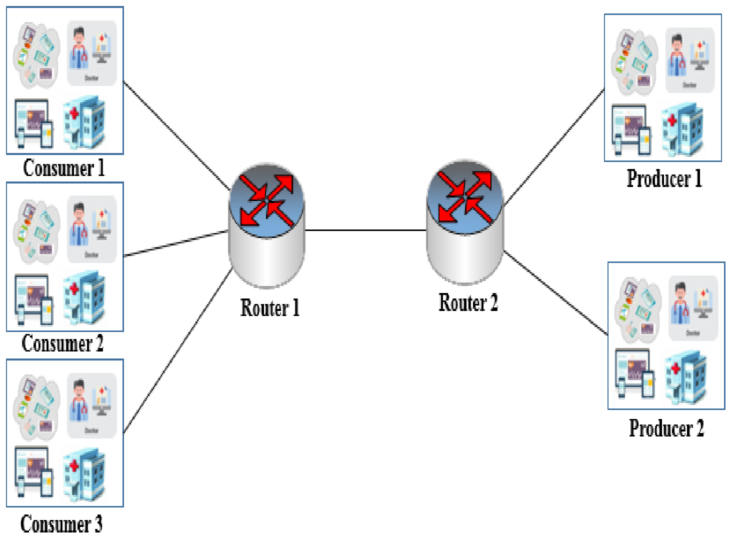

Fig. 3. Topology of 2nd Scenario.

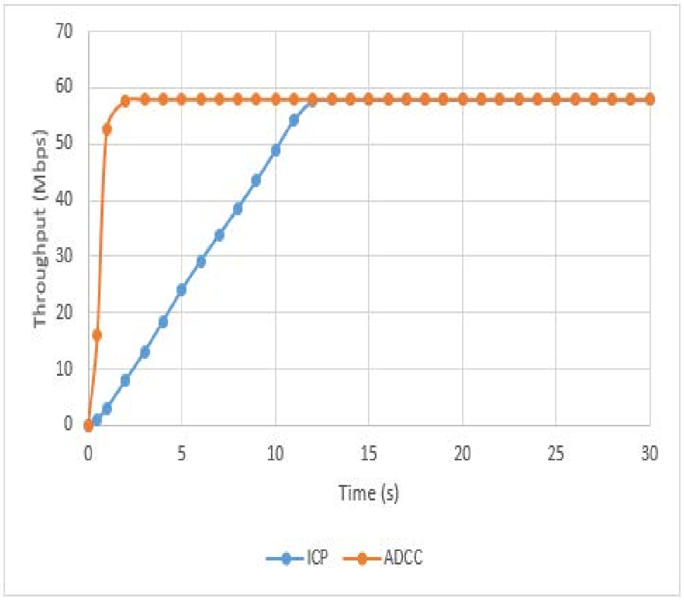

Fig. 4. Throughput of the 1st Scenario.

Fig. 4 shows the ability of the proposed mechanism to use the available bandwidth fully in the case of a non-congested network. This figure clearly shows that ADCD outperforms ICP in terms of throughput. The ADCD consumer could use an average throughput of 56.33 Mbps of the link capacity, while ICP used 45.63 Mbps. The increase in throughput with ADCD compared to ICP is due to the algorithm used by ADCD for congestion window adjustment, which rapidly increases its congestion window in the slow start phase, and then in the congestion avoidance phase, uses a window correlated weighting function (WWF) which aims to increase the use of the available bandwidth. However, ICP uses the AIMD algorithm to increase its congestion window, which rapidly increases the congestion window in the slow start phase, but in the congestion avoidance phase, it increases the congestion window by 1/cwnd, which increases the window slowly and consequently, a lower throughput than ADCD.

Similarly, in Fig. 5, which represents the throughput of the second scenario, we observe that the ADCD mechanism achieves better throughput than ICP. This is explained by the fact that ADCD divides the network into three states; no-congested state, less congested state, and heavily congested state. When it sees an increase in average queuing delay, it situates this value in one of the three states and reacts quickly by adjusting the size of the congestion window by β_1 or β_2. However, in ICP when a timeout is detected, it divides the congestion window by two, dividing the amount of data to be transmitted in the network by two, consequently decreasing the throughput.

These results demonstrate the capability of ADCD to manage the NDN network by transferring packets over the network bandwidth without causing network congestion or queue overflow. In the case of a congested network, ADCD responds quickly to the congestion problem while maintaining an optimal throughput.

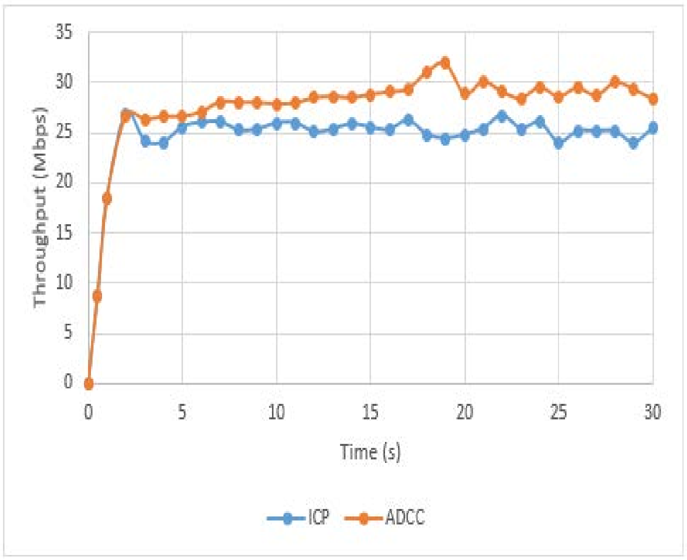

Fig. 5. Throughput of 2nd Scenario.

C. Delay Measurement

Delay is an essential factor for healthcare applications. It is the time taken for a packet to reach its destination from the source. Fig. 6 presents the delay measurements of the first scenario, Fig. 7 presents the delay measurements of the second scenario, and the average delay of scenarios 1 and 2 is presented in Table I.

In Fig. 6, we observe that ICP has a lower delay than ADCD, the consumer ICP has an average delay of 0.054s while the consumer ADCD has an average delay of 0.072s. In Fig. 7, we observe that ADCD has a lower delay than ICP. The consumer ADCD has an average delay of 0.26s while ICP is 0.42s. This is explained because ICP slowly increases its congestion window to transmit packets with lower throughput, fewer packets circulate in the network, and a lower transmission delay. However, the method used by ADCD to adjust the congestion window aims to increase the use of available bandwidth, thus allowing medical data to be transmitted with higher throughput and a reasonable transmission delay.

In the second scenario, the number of consumers has increased, so more requests for content are circulating in the network. ADCD handled this situation by calculating the average queuing delay for early congestion detection and reacting before overflowing the queue, consequently an optimal data transmission delay.

TABLE I. AVERAGE DELAY OF BOTH SCENARIOS

Mechanisms	ADCD	ICP
Scenario 1	0,072	0,054
Scenario 2	0,26	0,42

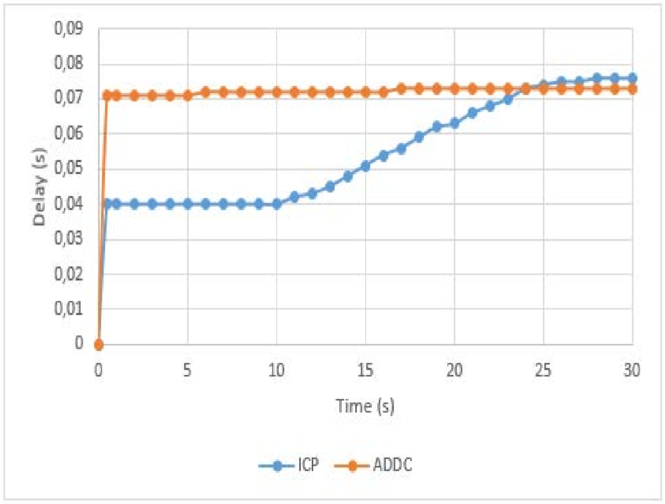

Fig. 6. Delay of the 1st Scenario.

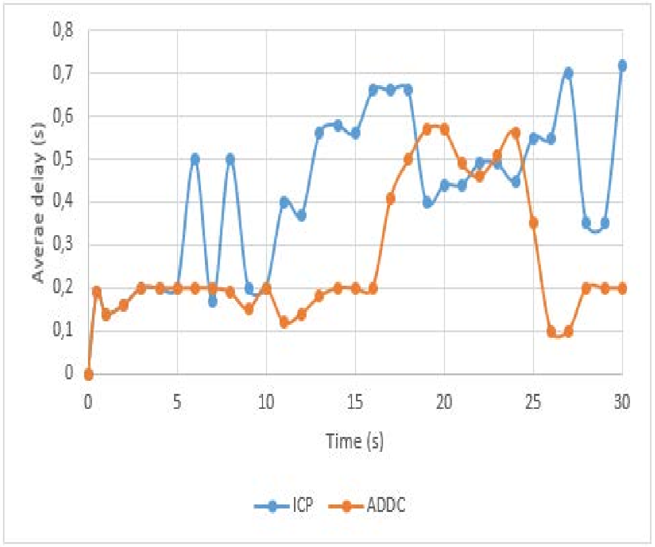

Fig. 7. Delay of 2nd Scenario.

D. Packet Loss Rate

The packet loss measurement is the difference between the number of packets transmitted and the number of packets received by the same node. It represents the number of packets abandoned per second. Congestion control mechanisms aim to reduce packet loss rate while increasing throughput and the use of available bandwidth. The simulation results of packet loss rate of the two scenarios are presented in Table II, where it is observed that the performance of ADCD and ICP are practically similar with a negligible packet loss rate.

TABLE II. PACKET LOSS RATE OF BOTH SCENARIOS

Mechanisms	**ADCD**	**ICP**
Scenario 1	0	0
Scenario 2	0,042	0,087

V. CONCLUSION

This paper proposes a new NDN-based approach, an Average delay-based early congestion Detection (ADCD), to control congestion in IoHT systems in which congestion is highly undesirable and can lead to undesirable results. In this approach, congestion is detected and controlled at consumer nodes by calculating the average queuing delay based on the one-way delay similar to that proposed in Sync-TCP, and then according to the calculated value, ADCD divides the network into three states: no-congested state, less congested state and highly congested state. The adjustment of the congestion window size is made according to the network state. The different simulations performed show the effectiveness of ADCD for early detection and control of congestion while improving network bandwidth utilization, maintaining optimal delay, and low packet loss rate.

In future work, we envisage extending the multipath analysis to independently managed paths to exploit the capability of NDN "multipath" to manage complex communication scenarios in IoHT systems.

REFERENCES

[1] K. Das, S. Zeadally, and D. He, “Taxonomy and analysis of security protocols for Internet of Things,” Futur. Gener. Comput. Syst., vol. 89, pp. 110–125, 2018.

[2] A. Abane, M. Daoui, S. Bouzefrane, S. Banerjee, and P. Muhlethaler, “A realistic deployment of named data networking in the internet of things,” J. Cyber Secur. Mobil., vol. 9, no. 1, 2020.

[3] Aroosa, S. S. Ullah, S. Hussain, R. Alroobaea, and I. Ali, “Securing NDN-Based Internet of Health Things through Cost-Effective Signcryption Scheme,” Wirel. Commun. Mob. Comput., vol. 2021, no. April, 2021.

[4] A. A. Rezaee, M. H. Yaghmaee, A. M. Rahmani, and A. H. Mohajerzadeh, “HOCA: Healthcare aware optimized congestion avoidance and control protocol for wireless sensor networks,” J. Netw. Comput. Appl., vol. 37, no. 1, pp. 216–228, 2014.

[5] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, “A survey of information-centric networking,” IEEE Commun. Mag., vol. 50, no. 7, pp. 26–36, 2012.

[6] P. Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., claffy, k., Crowley, P. and B. C., Wang, L., Zhang, “Named data networking,” ACM SIGCOMM Comput. Commun. Rev., vol. 44, no. 3, pp. 66–73.

[7] A. EL-BAKKOUCHI, M. EL GHAZI, A. BOUAYAD, M. FATTAH, and M. EL BEKKALI, “EC-Elastic an Explicit Congestion Control Mechanism for Named Data Networking,” Int. J. Adv. Comput. Sci. Appl., vol. 12, no. 11, pp. 594–603, 2021.

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

[9] M. C. Weigle, K. Jeffay, and F. D. Smith, “Delay-based early congestion detection and adaptation in TCP: Impact on web performance,” Comput. Commun., vol. 28, no. 8, pp. 837–850, 2005.

[10] B. Alahmri, S. Al-Ahmadi, and A. Belghith, “Efficient Pooling and Collaborative Cache Management for NDN/IoT Networks,” IEEE Access, vol. 9, pp. 43228–43240, 2021.

[11] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, “Named data networking for IoT: An architectural perspective,” EuCNC 2014 - Eur. Conf. Networks Commun., no. July 2015, 2014.

[12] W. Shang et al., “Named data networking of things (invited paper),” Proc. - 2016 IEEE 1st Int. Conf. Internet-of-Things Des. Implementation, IoTDI 2016, pp. 117–128, 2016.

[13] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, and S. Al-Ahmadi, “Named data networking: A promising architecture for the internet of things (IoT),” Int. J. Semant. Web Inf. Syst., vol. 14, no. 2, pp. 86–112, 2018.

[14] S. K. Datta and C. Bonnet, “Integrating Named Data Networking in Internet of Things architecture,” 2016 IEEE Int. Conf. Consum. Electron. ICCE-TW 2016, 2016.

[15] M. A. Hail, “IoT-NDN: An IoT architecture via named data netwoking (NDN),” Proc. - 2019 IEEE Int. Conf. Ind. 4.0, Artif. Intell. Commun. Technol. IAICT 2019, no. July, pp. 74–80, 2019.

[16] Y. Ren, J. Li, S. Shi, L. Li, G. Wang, and B. Zhang, “Congestion control in named data networking – A survey,” Comput. Commun., vol. 86, pp. 1–11, Jul. 2016.

[17] G. Carofiglio, M. Gallo, and L. Muscariello, “ICP: Design and evaluation of an Interest control protocol for content-centric networking,” in 2012 Proceedings IEEE INFOCOM Workshops, 2012, pp. 304–309.

[18] A. A. Albalawi and J. J. Garcia-Luna-Aceves, “A Delay-Based Congestion-Control Protocol for Information-Centric Networks,” 2019 Int. Conf. Comput. Netw. Commun. ICNC 2019, pp. 809–815, 2019.

[19] S. Mejri, H. Touati, N. Malouch, and F. Kamoun, “Hop-by-hop congestion control for named data networks,” Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl. AICCSA, vol. 2017-Octob, pp. 114–119, 2018.

[20] M. A. Alrshah, M. A. Al-Maqri, and M. Othman, “Elastic-TCP: Flexible Congestion Control Algorithm to Adapt for High-BDP Networks,” IEEE Syst. J., pp. 1–11, 2019.

[21] F. Wu, W. Yang, M. Sun, J. Ren, and F. Lyu, “Multi-Path Selection and Congestion Control for NDN: An Online Learning Approach,” IEEE Trans. Netw. Serv. Manag., vol. 18, no. 2, pp. 1977–1989, 2021.

[22] Y. Ye, B. Lee, R. Flynn, J. Xu, G. Fang, and Y. Qiao, “Delay-Based Network Utility Maximization Modelling for Congestion Control in Named Data Networking,” IEEE/ACM Trans. Netw., pp. 1–14, 2021.

[23] Y. Cao, M. Xu, and X. Fu, “Delay-based Congestion Control for Multipath TCP,” 20th IEEE Int. Conf. Netw. Protoc., pp. 1–10, 2012.

[24] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, “ndnSIM 2 : An updated NDN simulator for NS-3,” Dept. Comput. Sci., Univ. California, Los Angeles, Los Angeles, CA, USA, Tech. Rep. NDN-0028, no. November, pp. 1–8, 2016.