2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 979-8-3503-8199-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/TrustCom60117.2023.00325

Control Overhead Reduction using Length-based Same Destination Aggregation for Large Scale Software Defined Networks in Next Generation Internet of Things

Mohammad Shahzad, Lu Liu, Nacer Belkout
School of Computing and Mathematical Sciences, University of Leicester, UK
Department of Computer Science, University of Science and Technology Algiers, Algeria
mns14@leicester.ac.uk, l.liu@leicester.ac.uk, naceredine.belkout@etu.usthb.dz

***Abstract*— The Software Defined Networking (SDN) paradigm is rapidly emerging as a viable alternative in the realm of Next Generation Internet of Things (NG-IoT) due to the inherent challenge posed by traditional computer networks. SDN is revolutionizing traditional network architecture by significantly reducing the reliance on hardware. One of the key challenges of implementing SDN in NG-IoT is the extensive communication that occurs between the switches and the controllers due to the bursty nature of the traffic. Therefore, it is crucial to explore solutions aimed at minimizing this control overhead caused by excessive communication. To alleviate the burden on SDN controllers caused by switches, we propose a new method, named LSDA, which can effectively reduce the flow of control information. To promote fairness within the network, packets heading towards the same controller are aggregated, thereby eliminating the need for complex disassembly in the case of different destination controllers. The simulation results demonstrate that LSDA can effectively reduce the packet loss ratio while simultaneously improving network delay and jitter.**

Keywords— ***Control overhead, Internet of Things, Packet Aggregation, SDN***

I. Introduction

Next Generation Internet of Things (NG-IoT) encompasses the interconnectedness of various physical devices, vehicles, buildings, and other objects equipped with sensors, software, and network connectivity. This enables these devices to collect and exchange data seamlessly. NG-IoT [1-3] embraces the integration of intelligent solutions such as personalised healthcare [4], which draws inspiration from embedded intelligence at the edge. The effectiveness of this intelligence depends on faster processing capabilities, real-time analysis of information, and seamless high connectivity. Software Defined Networking (SDN) [5-6] plays a crucial role in enabling NG-IoT devices and transforming the telecom industry [7]. SDN controllers adeptly tackle the challenges experienced in both data and control planes by offering comprehensive solutions. Within an IoT ecosystem, SDN traffic experiences latency and jitter issues stemming not solely from contention and burstiness, but also from the control overhead occurring between switches and controllers. This control overhead has significant implications for performance.

SDN is a networking approach that empowers network administrators to manage network services using software, replacing the need for traditional hardware-based network devices like routers and switches. SDN empowers network administrators by giving them greater control, flexibility, and automation in managing and optimizing network resources, ultimately improving network performance and agility. Fig. 1 depicts the fundamental concept of SDN, which has revolutionised the perspectives of network researchers, network design, network management, and network operations. SDN has emerged as a fully programmable architecture for NG-IoT [8], offering enhanced dynamics, cost-effectiveness, adaptability, and an ideal solution for the dynamic and varied requirements of large-scale NG-IoT deployments.

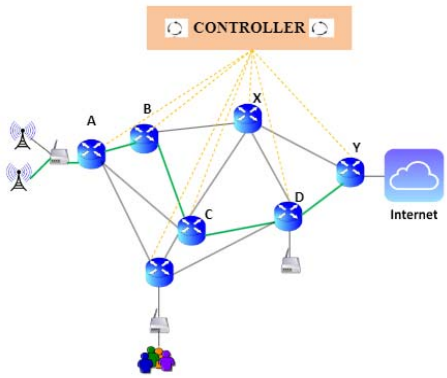

Fig. 1. Software Defined Networking

The implementation of a centralised SDN architecture [9] in IoT introduces challenges related to congestion, jitter, and low throughput. However, the control overhead between switches and controllers significantly impacts network performance. The primary source of control overhead is the communication between the controller and the SDN switches involving sending updates, state information, and control messages. The frequency and volume of these messages can contribute to control overhead. SDN switches typically

2324-9013/23/$31.00 ©2023 IEEE
DOI 10.1109/TrustCom60117.2023.00325

maintain flow tables that contain the rules for forwarding packets. When the controller sends flow rules to switches or updates existing rules, it can introduce control overhead, especially in large networks with many switches. Network policies implementation and access control involves sending policy instructions from the controller to the switches leading to control overhead particularly if policies are complex and frequently updated.

SDN community has made efforts to optimise the control plane operations and reduce control overhead. This optimization includes the use of efficient communication protocol, caching, and distributed controllers. Nevertheless, control overhead remains a consideration when designing and managing SDN networks, especially in situations where network responsiveness and scalability are critical factors.

The primary focus of the paper is on addressing the issue of control overhead between SDN switches and controllers, which leads to problems such as link congestion, packet loss, and subpar network performance. Recognising the formidable obstacle posed by the elevated control overhead within the realm of SDN, this paper introduces a novel approach known as Length-based Same Destination Aggregation (LSDA). LSDA can address this challenge and subsequently enhance the overall system's performance. The central emphasis of this proposed method is the reduction of control overhead between switches and controllers, coupled with the development of a novel data assembly strategy that centres around data destined for the same controller. This proposed method can effectively manage controller workloads and achieve equitable outcomes by means of TCP and UDP aggregation.

The paper is organised as follows: Section II provides a comprehensive literature review on control overhead and its reduction. Section III discusses the proposed design and its approach. Section IV highlights the implementation details. Finally, Section V concludes the paper.

II. Related Works

In an IoT environment, SDN traffic encounters delays, and jitter caused not only by contention and bursty nature but also by the control overhead between switches and controllers. This control overhead has serious repercussions on the performance. In their work cited as [10], the authors put forth algorithms aimed at achieving flow table resource balance and reducing the number of flow table entries through path aggregation. Performance metrics such as total flow entries, percentage of rejected flows, and percentage of overhead load were utilised to assess the impact. The results justify the effectiveness of minimizing communication overhead through the reduction of control messages.

The framework proposed in [11] focuses on reducing control overhead in SDN-based data centre networks by addressing both the controller and switch components. The paper examines the utilization of OpenFlow to reduce control communication by incorporating an out-of-band controller and bypassing the hybrid architecture. The architecture proposed in [12] for 5G VANET leverages SDN, offering enhanced flexibility, improved resource allocation, and increased programmability. By ensuring devices remain within their respective clusters or zones, the proposed approach achieves minimised controller overhead, leading to improved throughput and reduced transmission delay. This control communication management between vehicles and the controller effectively optimises network performance. Communication between devices and controllers occurs only when required, minimizing unnecessary interactions. However, this work does not address optimised resource sharing and protocols at SDN controllers, which could potentially enhance the overall efficiency and performance of the system.

The work presented in [13] focuses on the management of SDN flow entries using reinforcement learning and highlights the network configuration overhead as a key argument. The authors aim to maintain forwarding rules in the forwarding table and process them through the controller to reduce control plane overhead. This objective is accomplished through two stages: firstly, using a traditional reinforcement learning algorithm, and secondly, employing deep reinforcement learning. The results validate a significant 60 percent improvement in control plane overhead. It is important to note that these results were obtained using a simpler topology rather than a realistic network setting.

The work by Ziyong Li et al. [14] tackles the problem of switches neglecting flow characteristics when connected to controllers. The study analyses the consumption of control resources caused by flow requests and highlights the potential reduction in consumption achieved through controller association. To address control overhead, a greedy algorithm employing a coalitional game approach is utilised. VERO-SDN [15] offers an open-source SDN solution tailored for IoT, enabling programmable routing control to mitigate control overhead. It incorporates a modular controller and utilizes a protocol such as OpenFlow to enhance communication quality. The paper successfully achieved a 47 percent reduction in end-to-end delay and achieved a packet delivery ratio of over 99 percent.

SDN controllers effectively address the challenges encountered in both the data and control planes by providing comprehensive solutions. These challenges manifest in various forms, such as congestion, limitations in flow entries, communication issues, throughput concerns, packet loss, and security considerations [16-18]. When SDN switches receive incoming traffic, they utilize the Ternary Content Addressing Memory (TCAM) to search for matching entries in the flow table. If a match is found, the corresponding details are attached as headers, and the packet is forwarded accordingly.

However, if an entry is missing in the table, the switches reach out to the controller to request the necessary flow information. This process is repeated for each packet that encounters a missing entry. As the number of clients and traffic increases, the frequency of such requests also rises, leading to an escalation in communication overheads. As a recent work in this context [19], the focus has been on efficiently controlling packets by exploring different strategies. It discusses the limitations of existing control plane approaches in traditional wireless sensor networks and proposes a distributed control system architecture for software-defined wireless sensor networks (SDWSNs).

The paper provides insight and directions towards the development of a distributed control system for SDWSNs, offering potential advancements in the management and control of wireless sensor networks. A research study focused on developing a distributed control system for SDWSNs using a fragmentation-based approach is presented in [20]. The proposed system divides the control plane into fragments, each responsible for managing a specific subset of sensor nodes. These fragments work collaboratively to achieve global control and management of the network. The paper discusses the design and implementation details of the fragmentation-based control system, including fragmentation algorithms, control message dissemination, and coordination mechanisms.

In conventional network setups, server load balancing is employed to meet the demand for handling large data volumes. This approach necessitates significant capital investment and offers limited scalability and adaptability, making it challenging to accommodate the highly dynamic workload requirements of extensive mobile user populations. In order to address these issues, [21] examines the principles of SDN and introduces a novel probabilistic load balancing method based on variance analysis. This method can dynamically manage traffic flows to support extensive mobile user populations within SDN networks. The paper proposes an alternative solution using OpenFlow virtual switching technology instead of the traditional hardware switching approach. Within this framework, an SDN controller leverages variance analysis to monitor data traffic on each port and employs a probability-based selection algorithm to dynamically reroute traffic using OpenFlow technology. When compared to existing load balancing methods designed for conventional networks, this solution boasts reduced costs, heightened reliability, and improved scalability, effectively meeting the requirements of mobile users.

SDN stands out as a highly promising future internet development model, owing to its programmability and centralised administration advantages. However, relying on a single centralised controller can introduce issues related to reliability and scalability. While employing multiple controllers can address scalability and reliability concerns associated with a single centralised controller, it becomes crucial to implement a flexible mechanism for load balancing. Uneven distribution of traffic loads between controllers can easily arise. In response to this challenge, [22] proposed a prediction-based SDN load balancing dual-weight switch migration scheme for scenarios involving multiple distributed controllers.

This scheme leverages historical traffic load data to predict future traffic load patterns. By employing predictive technology, one can anticipate instances of controller overload, allowing for proactive switch migration operations. To mitigate the additional processing and communication overhead in the control plane, which is typically associated with periodic active load information exchange between distributed controllers, it introduces a triggered load information algorithm.

Taking both past and future load data into account, the proposed scheme recommends migrating specific switches between controllers. This approach utilizes historical and projected switch load information, incorporating a dual-weight switch migration algorithm that reduces the frequency of switch migrations. Empirical experiments have demonstrated the effectiveness of this scheme in rapidly balancing the load between controllers and minimizing the number of switch migrations.

Our approach focuses on the aggregation of data with the same destination to reduce complexity and tackle related challenges. The primary objective is to investigate the impact of data aggregation on crucial performance metrics, including packet loss, end-to-end delay, and jitter.

III. Design of Algorithm

The widespread adoption of IoT applications [23], [24] leads to a significant influx of data generated by sensors in end devices, posing a challenge to the SDN model. Moreover, as the SDN infrastructure is typically located remotely from the edge layer, the growing IoT phenomenon can potentially overload system resources, resulting in sluggish response times and increased network latency. This situation is particularly unsuitable for delay-sensitive IoT services.

An LSDA algorithm, deployed at the controller as illustrated in Fig. 2, minimises computational resources and decreases communication overhead. LSDA was accomplished by grouping packets with varying thresholds that share the same destination controller. The clients generate TCP packets and transmit them to the servers. At the controller, the incoming packets are observed, along with the monitoring of the queue size. By extracting the required packet information, efforts were made to collect the packets destined for the same location in a queue list. The primary obstacle lies in accommodating the diverse nature of data and the fluctuating number of clients, while effectively consolidating the packets into bursts to minimize control information. LSDA successfully tackles this challenge. Fig. 2 provides a visual representation of the functioning of LSDA.

Algorithm: Length-based same Destination Aggregation

Input: *switchList*; burst assembly threshold *BAT*;
Output: burst packet *BP*;

```
1 Initialization:
    netQueue ← ∅; netDev ← ∅;
    assembledPacket ← ∅; queueLoad ← ∅;
2 for each TimeUnit in SimulationTime do
3   for each switch in switchList do
4     netDev ← getNetDevice(switch)
5     netQueue ← getNetQueue(netDev)
6     queueLoad ← getQueueLoad(netQueue)
7     if queueLoad = BAT then
8       assembledPacket ← getFirstPacket(netQueue)
9       for each packet in netQueue do
10        assembledPacket ← addToEnd(packet)
11        netQueue ← removePacket(packet)
12      end for
13      BP ←assembledPacket
14    end if
15  end for
16 end for
```

Fig. 2. The pseudo code for LSDA

A. Formulation

Formula (1) was employed to determine the overall packet loss by considering various parameters such as packet size, bandwidth, and packet rate. The total number of sent and received packets was used in this calculation. Formula (2) was utilised to calculate the packet loss ratio, while formula (3) was used to determine the packet delivery ratio. Different parameters were considered in these equations to assess the performance in terms of packet loss and delivery.

$$PL = T_x - R_x \tag{1}$$

$$PLR = \frac{(T_x - R_x)}{T_x} \tag{2}$$

$$PDR = \frac{(T_x - R_x)}{R_x} \tag{3}$$

Formulas (4) to (6) were utilised to evaluate the end-to-end delay, total end-to-end delay, and average end-to-end delay. In these calculations, the packet transmission time (TTx) and packet reception time (RTx) were taken into account, considering a total of k transmitted packets. These equations provided insights into the overall delay experienced in the end-to-end communication, as well as the average delay per packet.

$$EED = (TT_x - RT_x) \tag{4}$$

The average end-to-end delay within a network represents the average delay encountered by all transmitted packets across the network. By considering the total end-to-end delay and the total number of packets, it becomes possible to calculate the average end-to-end delay. It's worth noting that the average end-to-end delay can fluctuate due to factors such as network topology, traffic load, and network conditions. As a result, it is crucial to consistently measure and monitor the end-to-end delay in the network to ensure it meets the desired performance requirements.

$$TEED = \sum_{i=0}^{i=k} (TT_{xi} - RT_{xi}) \tag{5}$$

$$AEED = \sum_{i=0}^{i=k} \frac{(TT_{xi} - RT_{xi})}{k} \tag{6}$$

The calculation of end-to-end jitter involved subtracting the current delay (CDx) from the last delay (LDx). This difference provided the measure of end-to-end jitter.

$$EEJ = (LD_x - CD_x) \tag{7}$$

Equations 8 and 9 were employed to calculate the total end-to-end jitter and average jitter, respectively. These calculations took into account the number of transmitted packets, represented by the variable 'k'.

$$TEEJ = \sum_{i=0}^{i=k} (LD_{xi} - CD_{xi}) \tag{8}$$

$$AEEJ = \sum_{i=0}^{i=k} \frac{(LD_{xi} - CD_{xi})}{k} \tag{9}$$

To determine the throughput, the number of bytes received was divided by the difference between the last packet receive time and the first packet transmission time. This calculation allowed for the estimation of the data transfer rate in terms of bytes per unit of time.

$$TPT = (RB_x * 8 / (LPRT_x - FPTT_x) \tag{10}$$

IV. Evaluation Results

Fig. 3 depicts the utilised topology for evaluating the LSDA by combining the packets routed towards the same controller. The employed topology comprises four controllers, seven routers, and three destination servers. Each server is positioned at varying distances from the clients and hosts UDP and TCP applications on specific ports. The packets generated by the varying number of clients are routed from R0 to the corresponding servers through the routers. The performance results encompass latency, jitter, and the loss ratio. The code was developed in C++ and evaluated within the ns-3 environment, utilizing the identical topology depicted in Fig. 3. The LSDA has been deployed on router R0 with the aim of minimizing computational resources and decreasing communication overhead.

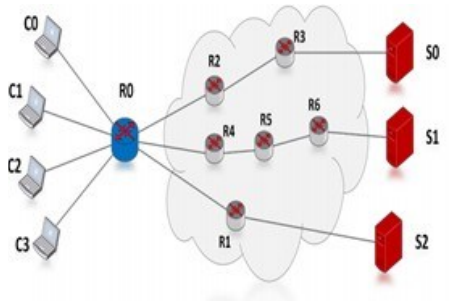

Fig. 3. Network Topology

By expanding the client base and conducting multiple simulation iterations, we were able to examine the scalability and effectiveness of the algorithm across various scenarios. The simulation served as a means to assess the performance of the LSDA when confronted with a growing clientele, and to determine its capability to accommodate larger networks. In order to assess how the LSDA performs as the client count rises, we can gauge key metrics such as average end-to-end delay, throughput, and packet loss rate.

Utilizing these metrics can aid in evaluating the algorithm's performance and pinpointing potential bottlenecks or challenges that might emerge as the network expands in size. Through the completion of 13 simulation rounds, a significant volume of data was collected to facilitate the analysis of the algorithm's performance under various conditions. This approach aided in the identification of performance trends and patterns within the metrics, empowering data-driven decision-making regarding algorithm optimization. In summary, the augmentation of client numbers and the execution of multiple simulation rounds offer a valuable means of assessing the scalability and performance of the algorithm, guaranteeing its ability to meet the demands of intricate and expansive networks. Additionally, the increased client count contributed to a larger packet volume, facilitating thorough testing of diverse thresholds.

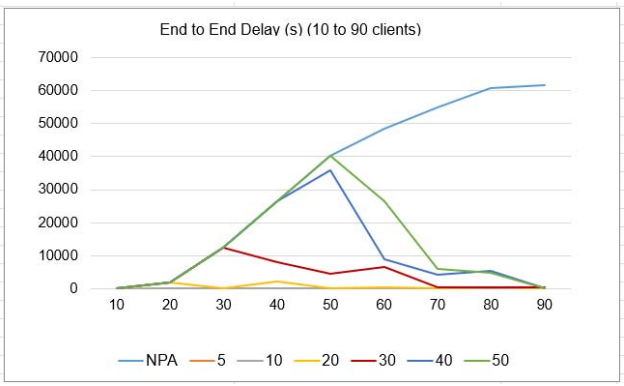

Fig. 4. End to End delay – Clients up to 90

Initially, the attention was directed towards the packet delivery ratio, end-to-end delay, and jitter concerning the No Packet Assembly (NPA) method, as depicted in Fig. 4. To explore the impact of packet assembly for the same destination controller, the assembly threshold is incremented from 5 to 50, indicating that every 5 packets in the queue were assembled using the algorithm. Subsequently, we tested thresholds of 10, 20, 30, 40, and 50 packets, respectively. Notably, the end-to-end delay for NPA exhibited an upward trend due to the presence of a substantial number of packets in the queue without a length-based assembly algorithm. However, as soon as the LSDA commenced operation, a significant reduction in delay was observed.

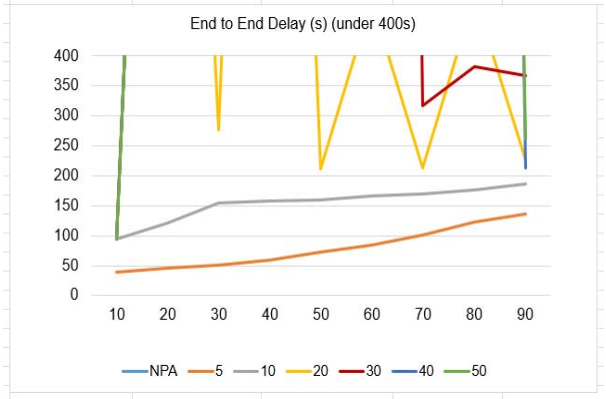

Fig. 5. End to End delay – under 400 seconds

At the initial stages, the delay experienced for thresholds 40 and 50 is slightly higher due to the fact that the LSDA initiates its operation when the packet count reaches 40 or 50. Consequently, during periods of low load, some delays may be encountered. However, as the number of clients increases, the algorithm demonstrates improved performance. Fig. 5 provides a close-up view of the same graph depicted in Fig. 4, focusing on a time frame of 400 seconds. In this graph, we observe that thresholds 5 and 10 promptly begin aggregating packets upon arrival, resulting in optimal performance at a smaller number of clients. However, as the client count increases, a slight deterioration in performance occurs, accompanied by a slight increase in delay.

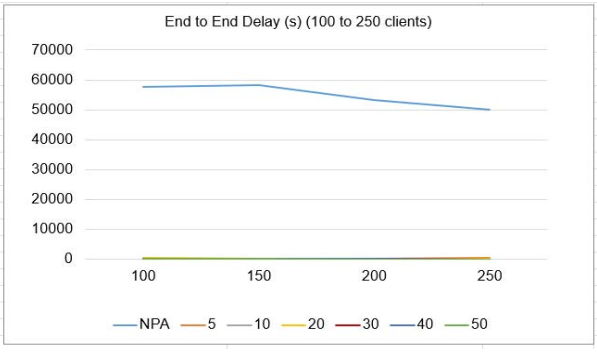

Fig. 6. Delay – Clients 100 to 250

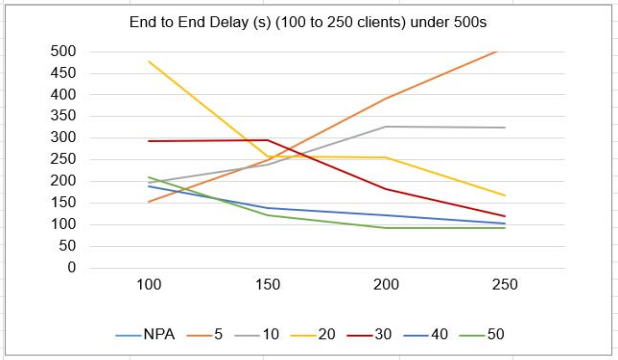

Fig. 7. Delay – Under 500 seconds

Fig. 6 and 7 offer valuable insights into the realm of end-to-end delay, with a particular focus on the range of 100 to 250 clients. In this scenario, the increased number of clients places greater demands on the algorithm as it generates a larger volume of data. Fig. 6 clearly illustrates that the NPA approach is unequivocally outperformed by the assembly thresholds, underscoring the superior performance of LSDA.

To gain a more comprehensive understanding of how the various assembly threshold schemes perform, we present a closer examination in the zoomed-in graph of Fig. 7, which specifically depicts the delay over a 500-second timeframe. Under the heightened workload, it becomes evident that the assembly threshold with the higher value exhibits superior performance, characterised by lower delay when compared to the other thresholds.

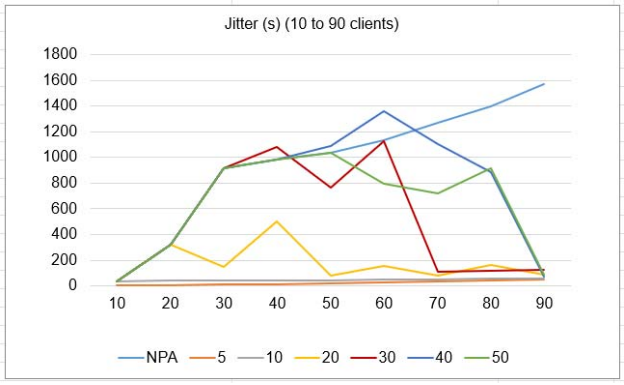

Fig. 8. Jitter – 90 Clients

Similar patterns can be observed for jitter, as illustrated in Fig. 8. The simulation was over 13 rounds, each with varying numbers of clients and assembly thresholds. In Fig. 9, the graph demonstrates that as the load increases for the NPA approach, the packets start dropping, resulting in a packet delivery ratio of 17 percent when the number of clients reaches 250. However, for the assembly thresholds, the delivery ratio remains above 90 percent, except for thresholds 30, 40, and 50. In the case of these thresholds, the delivery ratio initially drops as packets fail to meet the assembly threshold. However, once the threshold is met, the delivery ratio jumps to over 99 percent.

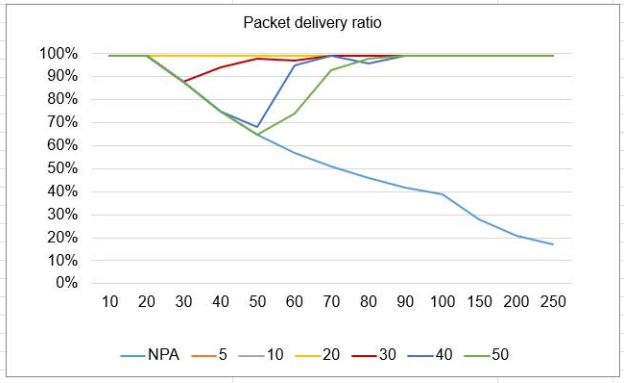

Fig. 9. Packet Delivery Ratio

V. Conclusion

This research paper primarily focused on addressing the issue of control overhead between SDN switches and controllers, which leads to problems such as link congestion, packet loss, and subpar network performance. A proposed solution, referred to as the length-based same destination aggregation algorithm, utilizes assembly thresholds ranging from 5 to 50 to aggregate data packets and forward them to the appropriate controllers. Through a comparison with the traditional methods, the burst aggregation approach was found to effectively mitigate link congestion, reduce the number of packets on the links, and enhance the packet delivery ratio. Moreover, this approach demonstrated improvements in network latency and jitter.

The experiments conducted in this study evaluated the performance of the burst aggregation approach across various conditions, including different assembly thresholds and network sizes. The results of these experiments showcased the approach's ability to significantly reduce control overhead and enhance overall network performance. Consequently, the findings highlight the burst aggregation approach as a promising solution for addressing control overhead concerns in SDN networks. Additionally, this approach exhibits the potential to enhance network scalability, reliability, and performance, particularly in larger and more complex networks.

References

[1] Zikria, Y.B., Ali, R., Afzal, M.K. and Kim, S.W., 2021. Next-generation Internet of Things: Opportunities, challenges, and solutions. *Sensors*, *21*(4), p.1174.

[2] Mukhopadhyay, S.C., Tyagi, S.K.S., Suryadevara, N.K., Piuri, V., Scotti, F. and Zeadally, S., 2021. Artificial intelligence-based sensors for next generation IoT applications: a review. *IEEE Sensors Journal*, *21*(22), pp.24920-24932.

[3] Chen, C.W., 2020. Internet of Video Things: Next-generation IoT with visual sensors. *IEEE Internet of Things Journal*, *7*(8), pp.6676-6685.

[4] Althobaiti, K., 2021. Surveillance in Next-Generation Personalized Healthcare: Science and Ethics of Data Analytics in Healthcare. *The New Bioethics*, *27*(4), pp.295-319.

[5] Amin, R., Reisslein, M. and Shah, N., 2018. Hybrid SDN networks: A survey of existing approaches. *IEEE Communications Surveys & Tutorials*, *20*(4), pp.3259-3306.

[6] Alsaeedi, M., Mohamad, M.M. and Al-Roubaiey, A.A., 2019. Toward adaptive and scalable OpenFlow-SDN flow control: A survey. *IEEE Access*, *7*, pp.107346-107379.

[7] Anerousis, N., Chemouil, P., Lazar, A.A., Mihai, N. and Weinstein, S.B., 2021. The origin and evolution of open programmable networks and SDN. *IEEE Communications Surveys & Tutorials*, *23*(3), pp.1956-1971.

[8] Zeb, S., Mahmood, A., Khowaja, S.A., Dev, K., Hassan, S.A., Qureshi, N.M.F., Gidlund, M. and Bellavista, P., 2022. Industry 5.0 is coming: A survey on intelligent nextG wireless networks as technological enablers. *arXiv preprint arXiv:2205.09084*.

[9] Chahlaoui, F. and Dahmouni, H., 2020. A taxonomy of load balancing mechanisms in centralized and distributed SDN architectures. *SN Computer Science*, *1*(5), p.268.

[10] Maleh, Y., Qasmaoui, Y., El Gholami, K., Sadqi, Y. and Mounir, S., 2022. A comprehensive survey on SDN security: threats, mitigations, and future directions. Journal of Reliable Intelligent Environments, pp.1-39.

[11] Singla, A., 2020. A systematic literature survey: Development of smart city based on various internet of things architectures. Integration of WSN and IoT for Smart Cities, pp.65-78.

[12] Zhao, L., Sun, W., Shi, Y. and Liu, J., 2018. Optimal placement of cloudlets for access delay minimization in SDN-based Internet of Things networks. IEEE Internet of Things Journal, 5(2), pp.1334-1344.

[13] Alghamdi, K. and Braun, R., 2020. Software defined network (SDN) and OpenFlow protocol in 5G network. Communications and Network.

[14] Adbeb, T., Wu, D. and Ibrar, M., 2020. Software-defined networking (SDN) based VANET architecture: Mitigation of traffic congestion. International Journal of Advanced Computer Science and Applications, 11(3).

[15] Jia, X., Li, Q., Jiang, Y., Guo, Z. and Sun, J., 2017. A low overhead flow-holding algorithm in software-defined networks. Computer Net- works, 124, pp.170-180.

[16] Pranata, A.A., Jun, T.S. and Kim, D.S., 2019. Overhead reduction scheme for SDN-based data center networks. Computer Standards and Interfaces, 63, pp.1-15.

[17] Khan, A.A., Abolhasan, M. and Ni, W., 2018, January. 5G next generation VANETs using SDN and fog computing framework. In 2018 15th IEEE Annual Consumer Communications and Networking Conference (CCNC) (pp. 1-6). IEEE.

[18] Mu, T.Y., Al-Fuqaha, A., Shuaib, K., Sallabi, F.M. and Qadir, J., 2018. SDN flow entry management using reinforcement learning. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 13(2), pp.1-23.

[19] Li, Z., Hu, Y., Hu, T. and Wei, P., 2019. Dynamic SDN controller association mechanism based on flow characteristics. IEEE Access, 7, pp.92661-92671.

[20] Theodorou, T. and Mamatas, L., 2020. A versatile out-of-band software-defined networking solution for the Internet of Things. IEEE Access, 8, pp.103710-103733.

[21] Zhong, H., Lin, Q., Cui, J., Shi, R. and Liu, L., 2015. An efficient SDN load balancing scheme based on variance analysis for massive mobile users. *Mobile Information Systems*, *2015*.

[22] Zhong, H., Xu, J., Cui, J., Sun, X., Gu, C. and Liu, L., 2022. Prediction-based dual-weight switch migration scheme for SDN load balancing. *Computer Networks*, *205*, p.108749.

[23] Sood K., Yu, S. and Xiang, Y., 2015. Software-defined wireless networking opportunities and challenges for Internet-of-Things: A review. IEEE Internet of Things Journal, 3(4), pp.453-463.

[24] Pritchard, S.W., Hancke, G.P. and Abu-Mahfouz, A.M., 2017, July. Security in software-defined wireless sensor networks: Threats, challenges and potential solutions. In 2017 IEEE 15th International Conference on Industrial Informatics (INDIN) (pp. 168-173). IEEE.