CMFR-CMQ: Congestion Management and Control Message Quenching Based on Flow Setup Requests in SDN-WISE

Achille Go, Mahamadi Boulou, Tiguiane Yélémou(✉), and Hamadoun Tall

Nazi BONI University, Bobo-Dioulasso, Burkina Faso
tyelemou@gmail.com

Abstract. Software-Defined Wireless Sensor Network (SDWSN) is an architectural solution that separates control plane from data plane of the sensor nodes and allows centralized management of the entire network. They are used in several application areas such as military, environmental, industrial, and medical. However, to ensure centralized management, and to facilitate the reconfiguration of the WSN, a significant amount of control messages are exchanged between the controller and the sensor nodes. These exchanges lead to the high energy consumption of the sensor nodes. In this paper, we propose a solution named CMFR-CMQ which is implemented on the SDN-WISE architecture. It avoids not only the duplication of Flow Request messages but also congestion of sensor nodes. Analytical performance of this solution shows that it significantly improves the network routing overhead and the packet loss rate compared to SDN-WISE.

Keywords: SDN-Wise · Routing overhead · Congestion avoidance · Energy consumption

1 Introduction

In last few years, Internet of Things (IoT) has been rapidly expanding and becoming an important part of our society. Several varieties have emerged, such as Wireless Sensor Networks (WSN). These are used in various fields of applications that include medicine, precision farming, smart home, environmental monitoring. Few years ago, sensor networks were composed of just a few hundred sensor nodes. Today, this number is increasing significantly. With traditional network management system, administrator must operate on the sensor nodes individually for any setting. This is much more time consuming and more likely to cause errors and is sometimes not possible because of difficult access to deployed nodes. To make this type of network easier to manage, the need to integrate the new paradigm of Software Defined Networking (SDN) or software networks into WSN has arisen. This technology allows decoupling control plane

Y. H. Sheikh et al. (Eds.): AFRICOMM 2021, LNICST 443, pp. 355–365, 2022.
https://doi.org/10.1007/978-3-031-06374-9_23

from data plane to reduce the rigidity of the network. SDN-WISE is one of the well-known architectures that aims to optimize energy consumption. However, in this architecture, when a node receives a packet for a given destination that does not have a match in its flow table for routing, it sends a request message to ask the controller for corresponding rule. If several packets are in the same situation with this node, it will send several request messages to ask for the corresponding flow rules for these packets. Number of these duplicate control messages may be high, and this may lead to important additional energy consumption. In this paper, we propose a new solution named CMFR-CMQ running on the SDN-WISE architecture. This approach prevents sending duplicate requests to the controller and manage sensor node congestion. Taking both aspects into account improves the energy management of the sensor nodes, thereby extending the network lifetime. The remainder of this paper is organized as follows. In Sect. 2, we present related work. In Sect. 3, we highlight our solution. The evaluation of this solution is exposed in Sect. 4. Then, we conclude in Sect. 5.

2 Related Works

SDN-WISE [4] is an approach in which network management logics are dictated by one or more controllers, one of them is the WISE-VISOR. The WISE-VISOR functions includes topology management (TM). The network elements of SDN-WISE are currently sink nodes and sensor nodes. Sink nodes communicate directly with the controller(s). They receive flow rules from controller(s). Indeed, between a sink node and the WISE-Visor, there is an adaptation layer that is responsible for formatting messages received from the sink node so that they can be processed by the WISE-Visor and vice versa. A flow table contains all the rules for processing an incoming packet. This table has three (03) sections: matching rules, action, and statistics. Thus, when a packet passes rules successfully, the actions can be executed, and then the corresponding statistics section is updated accordingly.

At a packet reception, an intermediate node starts a process of checking the flow table rules for that packet at the SDN cache. If a match is found, then the specific action related to that rule will be applied and thus it may be routed toward the receiver. If, no match was found in the flow table for the incoming packet, the node notifies the controller with a flow request message. The controller in turn decides, according to its configuration, what action to take for this packet, and then sends a new flow rule in the form of a flow rules message in order to update the concerned intermediate node flow table (Fig. 1).

SDN-Wise architecture is used to optimise energy consumption in WSNs. For this purpose, several mechanisms improving this architecture have been proposed. Authors in [3] propose a traffic monitoring algorithm, SDN-TAP, based on SDN-WISE architecture. SDN-TAP operates in two (02) stages, namely network formation and network operation. In their architecture, they propose a new role, an adapter node that acts as a gateway between the controller and the sensor nodes. This algorithm signals congestion situation of the sensor nodes by sending

Matching Rule					Matching Rule					Matching Rule					Action					Statistics	
Op.	Size	S	Addr.	Value	Op.	Size	S	Addr.	Value	Op.	Size	S	Addr.	Value	Type	M	S	Addr.	Value	TTL	Counter
=	2	0	2	B	>	2	0	10	x_{Thr}	=	1	1	0	0	Modify	1	1	0	1	122	23
=	2	0	2	B	≤	2	0	10	x_{Thr}	=	1	1	0	1	Modify	1	1	0	0	122	120
=	2	0	2	B	-	0	-	-	-	-	0	-	-	-	Forward	0	0	0	D	122	143
=	2	0	2	A	=	1	1	0	0	-	0	-	-	-	Drop	0	0	-	-	100	42
=	2	0	2	A	=	1	1	0	1	-	0	-	-	-	Forward	0	0	0	D	100	32

Fig. 1. WISE flow table

an alarm message to the controller. This redefinex the flow rules and relievex the congested sensor node. This approach may improve network reliability in packet loss compared to Dijkstra's algorithm. To alleviate network congestion through load balancing, authors in [9] proposed mecanisms that exploit report message sent by the node to the controller by adding a field for status information. This information represents the congestion level of a node and the priority level of the packets that reach it. Rules can be transmitted to the nodes that define different drop probabilities for different flows depending on the congestion level. Thus, upon receiving this information, the controller can search for new paths where it is possible to divert the traffic flow and sends the corresponding new rules to the sensor nodes of the network so that the alternative paths are used. The results of their simulations attest the effectiveness of this QoS management mechanism. FTDP [1] is a routing solution implemented on the SDN-WISE architecture. It uses the fuzzy system to compute the cost associated with the neighboring nodes in order to choose the node with the best cost as the next hop. Sink node collects information from sensor nodes and extracts parameters such as the number of neighbors, the amount of residual energy and the number of packets in the buffer. This information is sent to the controller. Once at the controller level, the FTDP according to this information executes the fuzzy system. It consists of fuzzification, inference system and defuzzification to calculate the cost associated with the neighboring nodes. The node with the largest cost value is chosen as the most appropriate intermediate node. From their simulation results, they proved that FTDP improves the network lifetime and increases the packet loss rate compared to SDN-WISE. In [5], authors implemented the Trickle Timing mechanism within the SDN-WISE architecture to reduce network overhead and energy consumption. The Trickle Timing algorithm allows nodes in a lossy shared medium to exchange information in a very robust, energy-efficient, simple and scalable manner. The Trickle Timing algorithm establishes a density-aware local communication primitive with an underlying coherence model that guides when a node begins transmission. When the network is stable or coherent, Trickle Timing causes nodes to transmit very few packets to reduce network overhead. When an inconsistency is detected, the node automatically fills it by increasing the packet exchange rate. Trickle Timing shows interesting simulation results regarding energy reduction when implemented in SDN-WISE compared to RPL. A hierarchical architecture

and a new QoS-based routing protocol called QSDN-WISE are presented in [8]. QSDN-WISE consists of a clustering algorithm, a routing algorithm, and the maintenance of the local network. The clustering mechanism is based on a dual cluster head called DCHUC. It avoids energy hostage and reduces the workload of a single cluster head. The centralized routing algorithm builds two heterogeneous routing paths for the nodes, which meets the requirements of different data levels. Finally, the LAN maintenance optimised the number of control messages in the network. Their simulation results indicate that QSDN-WISE may provide QoS support for data with different requirements, balance power consumption, and extend the network lifetime compared to SDN-WISE. Schaerer et al. [7] proposed a dynamic traffic aware routing protocol (DTARP). For this purpose, they extended the SDN-Wise framework by adding statistical information to the reports, which are periodically sent to the controller. This protocol takes into account the centrality between nodes and the dynamic traffic statistics of the nodes when calculating the path. With this additional information, it is possible to increase the cost of links for more active and central nodes. Therefore, routes to less active nodes are chosen to carry traffic even if they have a slightly higher hop count. Their simulation results prove that DTARP improves the lifetime of the entire network by reducing congestion, optimizing energy consumption, and distributing traffic more evenly compared to the RPL protocol. For a better balance of energy consumption, authors have proposed DEARP (Dynamic Energy Aware Routing Protocol) [2]. This protocl selects shortest path that respects a minimum threshold of residual energy at the nodes that constitute this path. Thus, a relatively longer path may be chosen for data transmission at the expense of a shorter path. To allow an efficient balancing of the energy consumption and to achieve a quasi-simultaneous exhaustion of the energy of the different nodes of the network, this minimum required (threshold) to be competitive in the choice of the path is progressively reduced. The analysis of this solution leads to a better balance of the overall energy consumption of the network while making the best use of the shortest path for data transmission.

3 Motivations

Like OpenFlow, when there is no match in the WISE flow table of a sensor node, a request is instantly sent to the controller to give the instruction to forward data. However, this request process might lead to duplicate control packets inducing network overhead. According to an approach called FR-CMQ [6], duplicate requests can be avoided by considering the source/destination address pair of the received packet. This approach states after detecting a mismatch in the WISE flow table checking phase. Instead of sending a request directly to the controller, the system first performs an intermediate step. This consists of checking a list of source/destination pair references stored in the buffer to find out if a previous request had been generated by the same type of packet (with the same source/destination pair). If a request has already been made, then the packet is queued until the associated processing procedure arrives, and then it

is processed accordingly. However, we know that every wireless sensor node has a limited memory capacity of just a few bytes. It is used to store packets while waiting for routing instructions from the controller. The FR-CMQ algorithm could lead to a buffer overflow, causing packets discarding as long as the conditions of this mechanism are met. Furthermore, in the approach proposed by Paolo et al. [9], we find that the buffer usage level could reveal the degree of congestion of each node in the network. Thus, thanks to the periodically generated report message, the controller would be notified of the congestion situation of the nodes. As a result, it would react accordingly by making the nodes use alternative paths during the time of decongestion. Once decongested, they will be able to participate again to the packet forwarding. For the best of our knowledge, these techniques are not yet applied on SDN-Wise architecture and there is not yet a solution that combines the FR-CMQ technique [6] with a congestion management system [9]. It would be worthwhile to consider the implementation of a solution that combines the strengths of these two techniques.

4 Proposed Approach: CMFR-CMQ

The proposed algorithm consists of both reducing the number of duplicate flow rule request messages and avoiding node congestion due to queued packets waiting for these controller's flow rules. The different steps are conditioned by the checking of the current congestion state of the node. Indeed, the controller determines the state of a node by calculating its congestion rate thanks to the information obtained through the *Report* message. In the rest of this section, we will first present the parameters and variables used, then we explain the algorithm.

4.1 Parameters and Variables

The following variables are used in our algorithm.

Capacity (C): this is the size in bytes of node's buffer.

Data Packet Quantity (QP): This is the amount of data packets waiting to be processed in the buffer.

Congestion rate (T): It is expressed in percentage and refers to the ratio of the number of data packets waiting in the buffer memory, over the memory capacity. T=QP/C*100

Congestion status (E): E is a Boolean value that controls the level of congestion. This information allows a node to signal to the controller whether it can receive more packets or not. E is set to 0 by default and changes to 1 when the congestion rate (T) is greater than or equal to 80% and returns to 0 if T becomes 20% or less.

Source/Destination Pair (SDi): This is the source and destination addresses pair of the packet being processed.

Source/Destination List (LSD): This is the list of source and destination addresses pairs stored in the node's buffer for verification purposes.

4.2 Data Packet Management Process with CMFR-CMQ

The different stages of packets processing in our solution are presented in Algorithm 1.

a) A congested node can only receive flow rules from the controller. Thus, the flow rule coming from the controller will be inserted in the node's WISE table. This update of its WISE table allows the processing of the concerned data packets waiting in the buffer to be started, if any.
b) A non-congested node is subject to several verification mechanisms. When a packet is received, it is first checked for the type of packet to determine what action should be taken:
   b.1) If the packet matches a controller flow rule, it automatically updates the node's WISE table for processing.
   b.2) If the packet is a data packet, then the node performs a match check in its WISE table. Once a match is found, the appropriate action is taken accordingly. Otherwise, the node checks the source/destination address of the packet being processed:

– If the source/destination address is already in the list of sources/destinations address pairs, then the packet is queued for processing in the buffer. A duplicate request is avoided, unlike the classic SDN-Wise mechanism.
– Otherwise, the new source/destination address is inserted in the list of sources/destinations addresses pairs, then a *request* message is sent to the controller for a flow rule request to process the packet(s) concerned and the packet is queued for processing in the buffer.

For simplicity of the algorithm, we exclude the cases of packet dropped despite a match exist in the WISE flow table. The different steps of our algorithm can be summarized in Fig. 2).

5 Analytical Evaluations of the Proposed Solution

For the performance evaluation of our algorithm, we consider three sensor nodes A, B and C. We assume that the node buffer capacity is five data packets. Also, this experimentation is done in a context where a node sends a *request* message to the controller when it has not yet received a response for the source/destination pair concerned by the received data packet. Node A could pass through B or C to reach the sink. We describe this communication scenario with the three packet management mechanisms: SDN-WISE, FR-CMQ and CMFR-CMQ. We assume that initially the controller commands the node A to send its data packets (20 packets) through the node B.

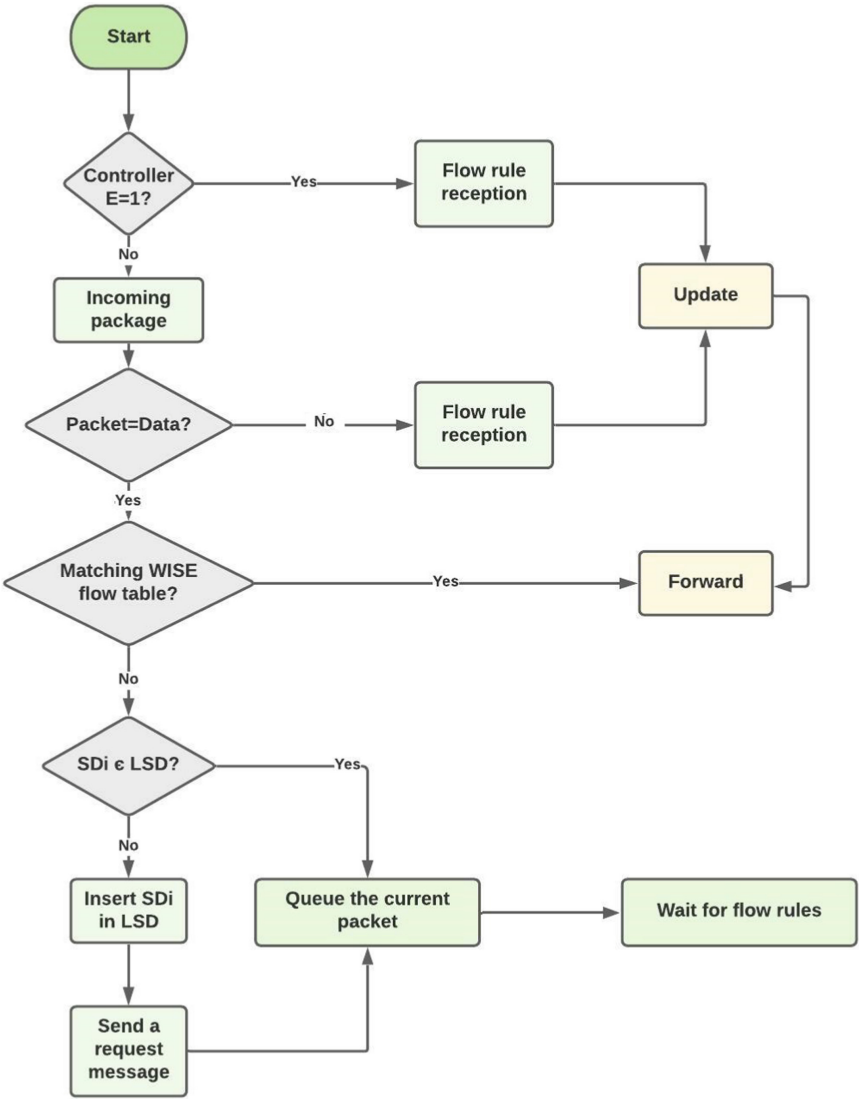

Fig. 2. CMFR-CMQ algorigram

- The SDN-WISE operating mode
  Upon receiving a packet from the node A, node B generates a *request* message and sends it to the controller, and places the data packet in its buffer, awaiting instructions. This process is repeated until the number of packets in the buffer reaches five, hence its saturation. Thus, if node A sends twenty packets to node B, five request messages would be generated, five packets would be stored in its memory and fifteen packets would be lost (Fig. 3).

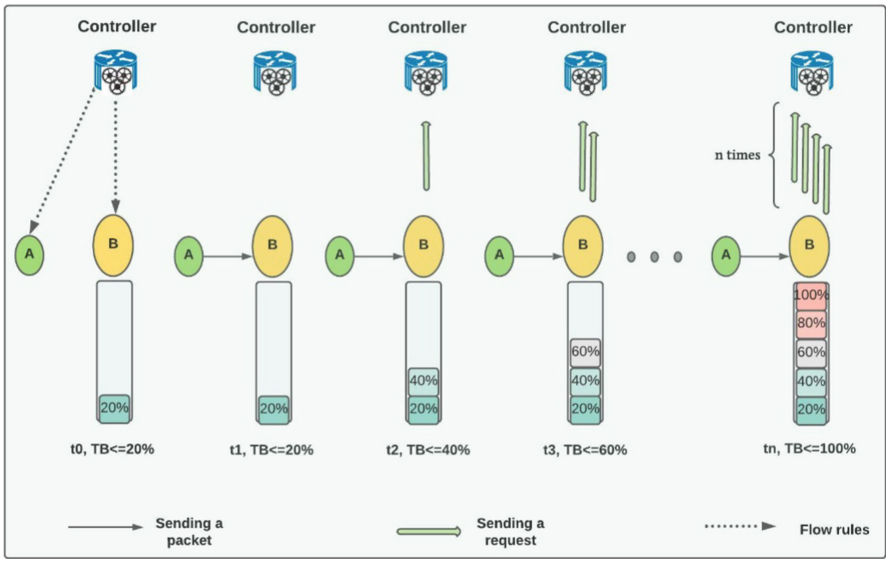

Fig. 3. Data packet process in SDN-WISE

- FR-CMQ operating mode
  Unlike SDN-WISE, in FR-CMQ, when Node B receives data packets with the same source/destination, it produces a single request message. It then stores the packets in its buffer until it is full. Therefore, if node A sends twenty packets to node B, we get a single request message, five packets would be stored in memory and fifteen packets would be discarded.
- The CMFR-CMQ operating mode
  As in FR-CMQ, when node B receives several data packets from node A, it sends a single request message. However, CMFR-CMQ considers the level of congestion of the nodes. Thus, if node A sends twenty packets to node B, only one request message is issued, four packets would be stored in memory and sixteen packets would be forwarded to another non-congested node (C in our case) (Figs. 4 and 5).

Figure 6 provides a summary of performance results for the three algorithms in terms of packet loss and routing load (number of control messages) when 20 packets of data are sent from node A to node B.

Based on these results in Fig. 6, we conclude that SDN-WISE produces more control messages compared to FR-CMQ and CMFR-CMQ. This induces a high energy consumption in the network. Also, FR-CMQ and SDN-WISE have a very high packet loss rate compared to CMFR-CMQ in case of congestion. This high packet loss rate is due to the lack of a congestion management mechanism in the first two (02) solutions. We note that the estimated packet loss in our study is only related to the congestion of the nodes and we do not refer to ones resulting from link quality.

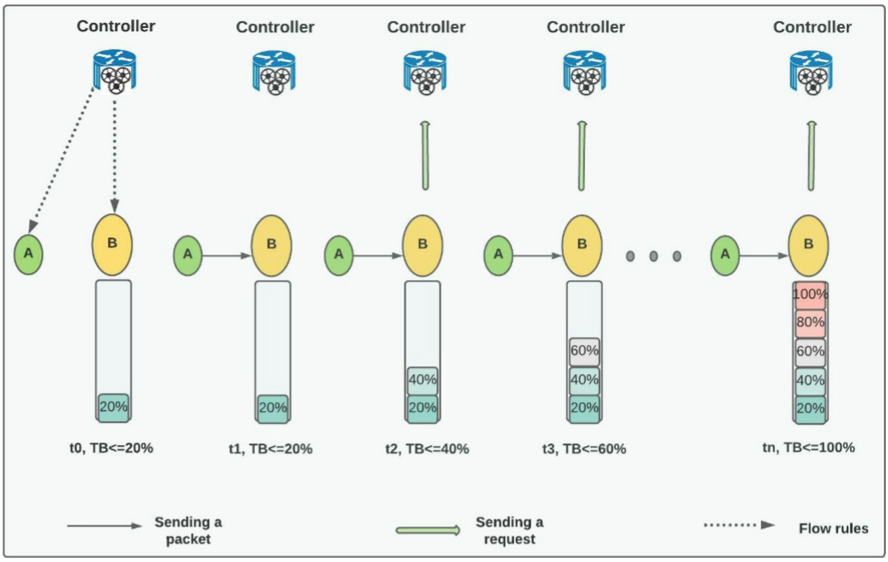

Fig. 4. Data packet process in FR-CMQ

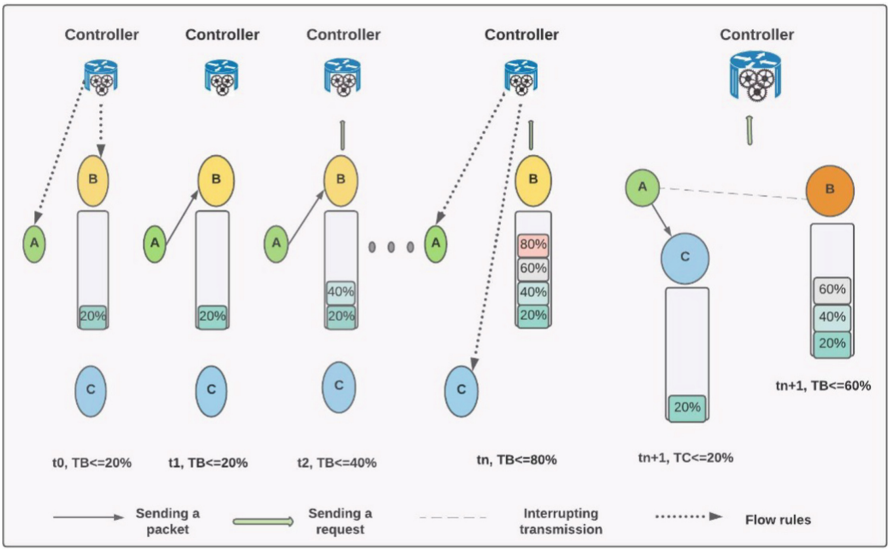

Fig. 5. Data packet process in CMFR-CMQ

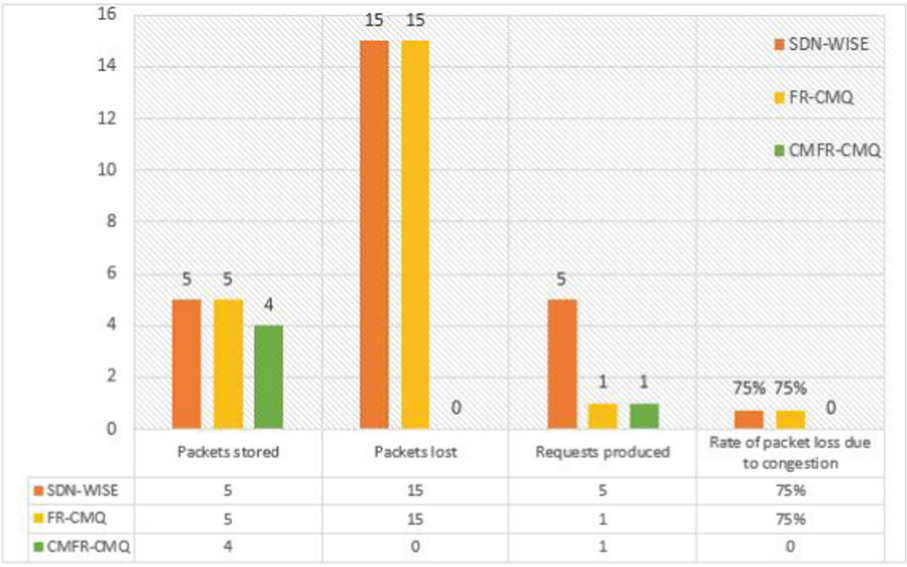

Fig. 6. Number of different packets issued or dropped according to the three algorithms.

6 Conclusion

Faced with energy consumption problem in SDN-WISE, we have proposed in this paper the CMFR-CMQ solution. This solution, on the one hand, avoids duplicate flow rule requests. On the other hand, it exploits an extended report message to warn controller of congestion situations of the forwarding nodes. According to the analytical results we obtained, this solution improves SDWSN routing load and packet loss rate in SDN-Wise architecture. As future work, we plan to conduct an intensive tests of this solution and we will compare its performance with the best results of energy consumption optimization approaches in SDN-Wise.

References

1. Abdolmaleki, N., Ahmadi, M., Malazi, H.T., Milardo, S.: Fuzzy topology discovery protocol for SDN-based wireless sensor networks. Simul. Model. Pract. Theor. **79**, 54–68 (2017). https://doi.org/10.1016/j.simpat.2017.09.004
2. Boulou, M., Yélémou, T., Rollande, D.A., Tall, H.: Dearp: dynamic energy aware routing protocol for wireless sensor network. In: 2020 IEEE 2nd International Conference on Smart Cities and Communities (SCCIC), pp. 1–6 (2020). https://doi.org/10.1109/SCCIC51516.2020.9377331
3. Fotouhi, H., Vahabi, M., Ray, A., Björkman, M.: SDN-TAP: an SDN-based traffic aware protocol for wireless sensor networks. In: 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services, Healthcom 2016, p. n (2016). https://doi.org/10.1109/HealthCom.2016.7749527

4. Galluccio, L., Milardo, S., Morabito, G., Palazzo, S.: SDN-WISE: design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks. In: Proceedings - IEEE INFOCOM, vol. 26, pp. 513–521 (2015). https://doi.org/10.1109/INFOCOM.2015.7218418
5. HHieu, N.Q., Thanh, N.H., Huong, T.T., Thu, N.Q., Van Quang, H.: Integrating trickle timing in software defined WSNs for energy efficiency. 2018 IEEE 7th International Conference on Communications and Electronics, ICCE 2018, pp. 75–80 (2018). https://doi.org/10.1109/CCE.2018.8465747
6. Ndiaye, M., Abu-Mahfouz, A.M., Hancke, G.P., Silva, B.: Exploring control-message quenching in SDN-based management of 6LoWPANs. In: IEEE International Conference on Industrial Informatics (INDIN), vol. 2019, pp. 890–893 (2019). https://doi.org/10.1109/INDIN41052.2019.8972067
7. Schaerer, J., Zhao, Z., Braun, T.: DTARp: a dynamic traffic aware routing protocol for wireless sensor networks. In: RealWSN 2018 - Proceedings of the 7th International Workshop on Real-World Embedded Wireless Systems and Networks, Part of SenSys 2018, pp. 49–54 (2018). https://doi.org/10.1145/3277883.3277885
8. Tan, X., Zhao, H., Han, G., Zhang, W., Zhu, T.: QSDN-WISE: a new QoS-based routing protocol for software-defined wireless sensor networks. IEEE Access **7**, 61070–61082 (2019). https://doi.org/10.1109/ACCESS.2019.2915957
9. Wsns, S.d., et al.: Exploiting State Information to Support QoS in (2016)