Adaptive Sharing of IoT Resources Through SDN-Based Microsegmentation of Services Using Mininet

Angely Martínez[1], José D. Padrón[1(✉)], Jorge Luis Zambrano-Martinez[2], and Carlos T. Calafate[1]

[1] Computer Engineering Department (DISCA), Universitat Politècnica de València, Valencia, Spain
anmar74a@inf.upv.es, {jdpadper,calafate}@disca.upv.es
[2] University of Azuay, Cuenca, Ecuador
jorge.zambrano@uazuay.edu.ec

Abstract. As we gradually embrace Smart Cities and the many advantages they can potentially offer, several technical issues arise that should be properly addressed, including security, efficiency, and performance, among others. In this regard, the massive deployment of IoT devices to support the numerous potential applications can consume excessive resources if their use is not optimized; this includes the sharing of some IoT devices whose content has the potential to be shared by many potential applications. One such example are CCTV smart cameras, as their deployment is costly, has a significant impact on urban aesthetics, and also the traffic flow they generate is high. To address such issue, in this paper we propose a novel SDN framework that enables the seamless sharing of streamed CCTV camera contents among multiple users, while adequately accounting for security and privacy restrictions. In particular, we adopt the Zero Trust paradigm to have a fine granularity control of the data-sensitive contents streamed by CCTV cameras. Experiments performed in Mininet using the Ryu controller evidence the potential of our solutions, which is able to achieve the target goals in a resource efficient manner, while introducing a low network updating overhead.

Keywords: Software-Defined Networks · IoT · Microsegmentation · Resource sharing · Zero Trust security

1 Introduction

Smart Cities rely on Information and Communication Technologies (ICT) to create, deploy, and promote sustainable development practices that address the growing urbanization challenges. In this context, Internet of Things (IoT) solutions emerge as key enablers for smart city ecosystems [2,11], as they allow

This work is derived from R&D project PID2021-122580NB-I00, funded by MCIN/AEI/10.13039/501100011033 and "ERDF A way of making Europe".

J.-L. Guisado-Lizar et al. (Eds.): SIMUtools 2023, LNICST 519, pp. 229–242, 2024.
https://doi.org/10.1007/978-3-031-57523-5_18

collecting and analyzing data in real-time, thereby helping municipalities, enterprises, and citizens to make better decisions that improve their overall quality of life.

IoT devices are used in various domains of smart cities, such as infrastructure, mobility, public services, and utilities. For example, connected traffic lights [7] can adjust light cadence and timing to respond to real-time traffic, reducing road congestion. Connected cars can communicate with parking meters and electric vehicle charging docks, and thereby direct drivers to the nearest available spot [9]. Also, smart garbage cans can automatically schedule pick-up as needed, improving upon the existing pre-planned scheduling [23].

While IoT devices offer many benefits for smart cities, they also pose some challenges and limitations that need to be addressed. These challenges include the interoperability of IoT devices from different vendors, platforms, and standards, as well as the security and privacy of IoT devices and the data they generate. Moreover, IoT devices collect large amounts of personal and sensitive data from citizens, which raises concerns about data protection, consent, and ownership. Therefore, IoT devices need to have adequate security mechanisms and follow ethical principles to ensure the trust and safety of smart city stakeholders [1].

One possible solution to overcome some of the challenges of IoT devices in smart cities is to adopt a sharing economy model [17]. A sharing economy is a system where people share access to goods and services rather than owning them individually. Sharing economies can reduce the costs and environmental impacts of owning and maintaining IoT devices by optimizing their utilization and distribution. In the context of Smart Cities, it would be recommendable to share IoT resources that are more expensive to deploy and maintain, that occupy more physical space, or that consume more energy.

Nowadays, Closed-Circuit Television (CCTV) is employed in practically every area of the city, including public spaces, houses, shops, passageways, and even roadways outside the city. Although each of these cameras has a unique role, they are all employed to continuously monitor the area, capturing nearby events for later reference. Also, each of them is usually in a separate network. Currently, there is no collaborative utilization of these devices, which represent a large investment. Yet, in the long run, it appears from a macroeconomic perspective that the existence of parallel networks of CCTVs performing a same task will not be economically beneficial due to the high initial investment and ongoing maintenance expenses [22].

In this work, we focus on CCTV surveillance cameras in the context of IoT, where they are used as yet another IoT resource. In particular, we want to study how such resources can be shared in a controlled and secure manner between multiple users. To this end, we consider that the adoption of Software-Defined Networks (SDN), and specifically the service microsegmentation paradigm, can be helpful to achieve the target goals. Hence, our proposed solution leverages SDN microsegmentation to achieve the desired granular security based on the Zero Trust concept [10], which is applied in the context of IoT resource sharing.

In particular, we develop a full SDN architecture for computing on the edge whereby a web application instructs the Ryu SDN controller on how to enable CCTV camera flows only to authorized clients, being able to create dynamic rules for each CCTV element that benefit from a high-level of management granularity, including the possibility of pinpointing the exact days/times each client is allowed to access the video stream generated by each particular surveillance camera.

The remainder of the paper is organized as follows: in the next section, we provide an overview of some related works. Then, in Sect. 3, we provide an overview of our proposal, including also technical details regarding its implementation. Section 4 details the simulation framework we have set forth in order to undertake the desired study. Experimental results are then presented and discussed in Section 5. Finally, conclusions and future research directions are provided in Sect. 6.

2 Related Work

IoT resources that can be shared with multiple users in a controlled and secure manner are considered one of the main elements of smart city ecosystems. In this regard, several research works address SDN architectures that enable such IoT resource sharing. Li et al. [12] propose an SDN-based IoT architecture to provide data obtained from IoT devices, and their interoperability is supported at different levels, allowing the rapid creation of IoT applications to reuse data and applications. Similarly, Firouzi & Rahmani [6] discuss computational and data sharing among IoT devices, integrating transport network control with distributed cloud and edge resources to provide dynamic and efficient IoT services. On the other hand, Mukherjee et al. [13] propose an SDN-based distributed IoT network with a network functions virtualisation implementation for smart cities; in particular, a residential area uses ICT and IoT networks to improve performance and minimise round trip times through multiple distributed controllers and clusters, thereby improving load balancing, scalability, availability, integrity, and security. Bouloukakis et al. [4] propose a cross-layer middleware system that allows mission-critical data from IoT devices to be shared in a timely and reliable manner with relevant consumers by prioritizing messages.

Osman et al. [14] provide microsegmentation as a means to reduce smart home network attacks with the help of a cloud perimeter by implementing features that create an inventory of all devices and their vulnerabilities, dynamically assigning IoT devices to microsegments to isolate them using security policies at the network level. In [21], the author proposes implementing micro-segmentation to protect smart homes that contain IoT networks from internal attacks that involve lateral movements, thus automatically classifying non-malicious devices based on their functionality to assign them to confined network microsegments.

In the same way, there is a growing number of defenders of the Zero Trust security paradigm linked to micro-segmentation. Syed et al. [18] discuss the conventional micro-segmentation and automation approaches that are available.

In [5], the authors propose a two-layer access control architecture that is compatible with the Zero Trust model to provide automated support for dynamic re-configurations in IoT infrastructures with remote access. Instead, Basta et al. [3] analyze an analytical brand to characterize and quantify the effectiveness of Zero Trust micro-segmentation to improve network security.

In this paper we build upon this Zero Trust micro-segmentation concept to create an IoT-sharing solution whereby the video contents generated by CCTV cameras can be conveyed to multiple receivers (users) depending on their permission levels, which may vary dynamically. In this regard, we develop an application whereby the administrator can have fine grain control over these resources.

3 Proposed Solution

Computer networks are the backbone of our daily activity and of the Internet. The amount of data that circulates through them is uncountable, thus generating the concept and paradigm of "big data". Due to the static nature of the network architecture created by the different network devices (i.e.: routers, switches, firewalls, etc.), a bottleneck is often generated, establishing limitations in terms of scalability and automation. Regarding scalability, traditional networks are limited due to their dependence on the physical hardware of the devices themselves. In terms of automation, they have few capabilities and require a significant amount of manual intervention. Hence, the maintenance of such a large network, especially when it is growing and changing dynamically, is less and less in line with business needs and user demands, while at the same time become very complex to manage.

Regarding the IoT paradigm, it is being included both at a personal level, and in all kinds of industries, from manufacturing to healthcare or transportation; the multitude of Internet-connected devices that this implies contributes to the growth of network traffic, stressing the capacity of cloud data centers. According to Perwej et al. [16], the number of interconnected devices was projected to soar past 34 billion by 2021. Moreover, it is anticipated that these devices will uniquely identify objects, being most connected through an Internet of Things platform. In particular, the dominant areas of data generation are projected to be security systems and video surveillance due to their large file sizes.

This paper attempts to provide a solution based on SDN that allows to flexibly and efficiently manage access to different data sources from different IoT devices, which are to be shared conditionally with different service users. More precisely, we use the micro-segmentation technique, which allows security architects to logically divide the data center into different security segments down to the individual workload level. This allows administrators to program security policies based on where a workload may be used, what type of data it will access, and how important or sensitive the application is, to adequately manage the IoT devices that are part of that network.

Particularly, the network scenario to be implemented consists on: (i) IoT nodes with high data flows (i.e.: video surveillance cameras), and (ii) physically

connected nodes (i.e.: servers). The latter will have the role of clients, and will receive the traffic from these cameras according to the agreed service conditions. Thus, the main idea is to allow the operator to add and/or remove nodes dynamically and securely via a web interface, while controlling the flow of the entire network to the different clients.

To understand how our goal is achieved, Fig. 1 shows an example of what would be the communication process in the architecture if the administrator were to perform the operation of adding a client to a camera from the web application. First, the network administrator sends the POST request in a JSON format, to accept the multicast traffic from group 224.0.0.4 to the client. Then, the camera-client relationship table is updated in the database server. Third, the SDN controller processes the request, and sends the rule to the corresponding network devices through the OpenFlow protocol. Finally, when the Access Control List (ACL) table flows are updated, the controller sends an HTTP response with code 200 OK to notify of the success of the request.

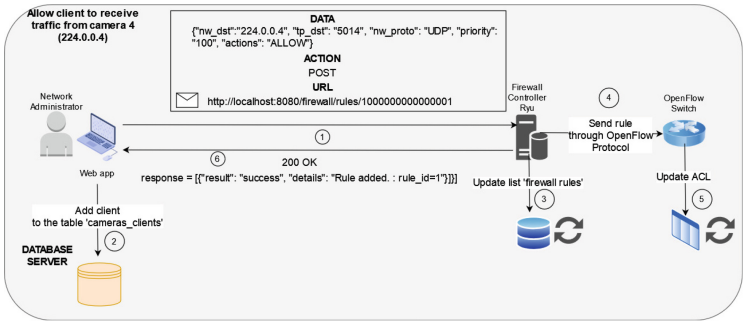


Fig. 1. Example of a triggered update in the context of our solution.

As final remarks regarding this solution, it is worth mentioning two crucial parts of the web app interface. In Fig. 2 we can observe how the allowance period for sending the information of a camera to a specific client is managed. As we can observe, we can set specific rules for a certain time period. Figure 3 presents the main view of the application, where the administrator can see the number of clients and cameras, and also which client(s) is related to the different cameras.

4 Simulation Framework

In this section, we describe the simulation framework used for our study, which involves two different tools:

1. Mininet-WiFi version 2.4.3 [8], in order to emulate the network.
2. Ryu version 4.34 [15], which serves as an external controller to the network.

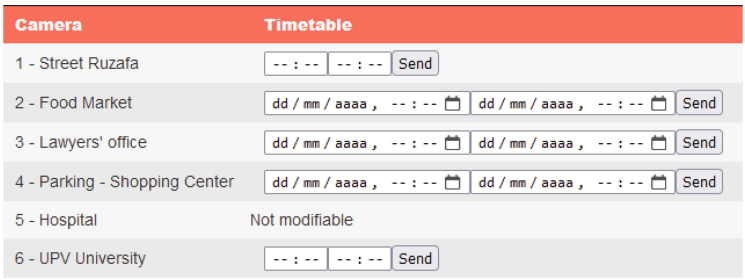


Fig. 2. Interface to detail the allowance period for specific cameras.

Firstly, Mininet-WiFi allows us to design and emulate the proposed network. To this end, we devise a tree topology consisting of 5 switches (Open vSwitch) and 4 Wi-Fi access points. Notice that Mininet-WiFi extends the functionality of Mininet [19] (working with the OpenFlow technology), by adding access points and virtualized Wi-Fi stations based on standard Linux wireless controllers and the 802.11 simulation driver 80211_hwsim.

Secondly, the Ryu controller allows the dynamic creation of the flow table entries, and the redirection of the various network messages. This allows us to dynamically redirect messages to the various clients associated with IoT security cameras based on their needs and permissions.

Figure 4 depicts how these tools are combined to generate the scenario for the evaluation of our solution. As it can be seen, the architecture is divided in three layers:

1. Infrastructure layer: formed by each of the end devices participating in the network. On the left side, 6 video surveillance cameras, connected each one to an AP (Access Point), which then are connected to an Open vSwitch device. Then, on the right side, 5 servers (clients) connected to 4 Open vSwitch devices.
2. Control layer: composed by the Ryu SDN controller.
3. Application layer: web application, which will give the network administrator the flexibility to make changes in the infrastructure. It also allows dynamic addition and deletion of clients, and establishes time-based rules for receiving traffic.

To have a complete view of how our implementation works, Fig. 5 shows the different flow associations and the network topologies involved.

Firstly, Fig. 5a represents the network architecture part related to the IoT Security Cameras, which act as nodes. At the node level, we have that all the cameras are wirelessly connected to their corresponding access point. For the APs, we can see that they are physically connected to a switch which manages the traffic on this network segment based on the flow tables.

Secondly, the client-side network topology is illustrated in Fig. 5b. Here, it becomes evident that the client nodes are not connected to the Access Points

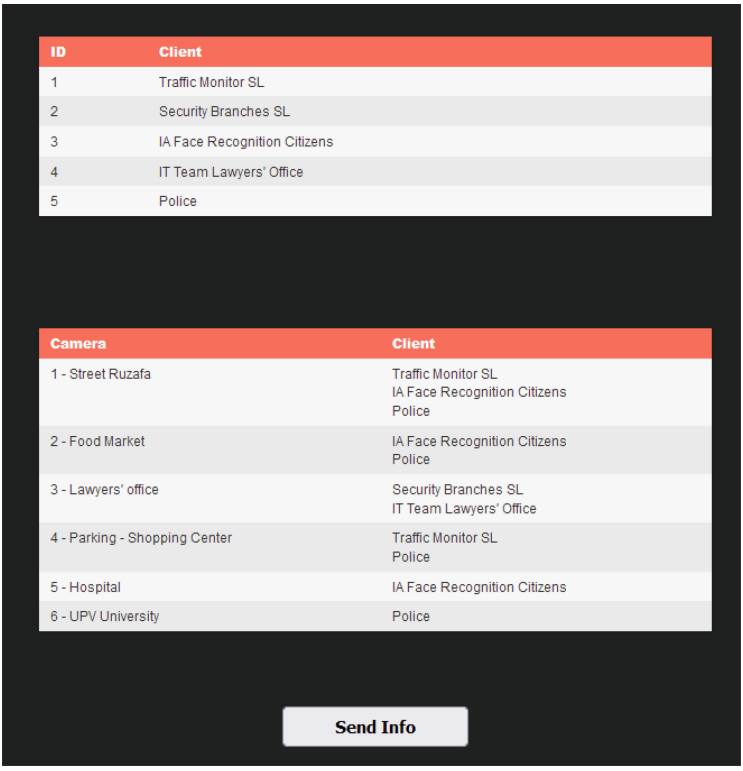


Fig. 3. Global view of the system status.

(APs), but are directly linked to switches. It is crucial to remember that the first level switch is connected to the equivalent first level switch within the camera topology, as we have depicted earlier in Fig. 4. This arrangement guarantees the visibility of both topologies and enables communication, even when the controller is located centrally within the control layer.

Finally, to illustrate our point of view, we present an example showing the data flow transmitted within the network from different cameras to clients 1 and 5, as seen in Fig. 5c. Obviously, this data flow previously traverses the respective network devices. This representation originates from the camera-client relationship that we have established for the initial basic solution. Ultimately, the multicast data stream that is sent follows the camera-client association shown in Table 1.

5 Simulation Results

In this section, we present our simulation results based on the simulation framework described in the previous section. Our goal is to determine the degree of

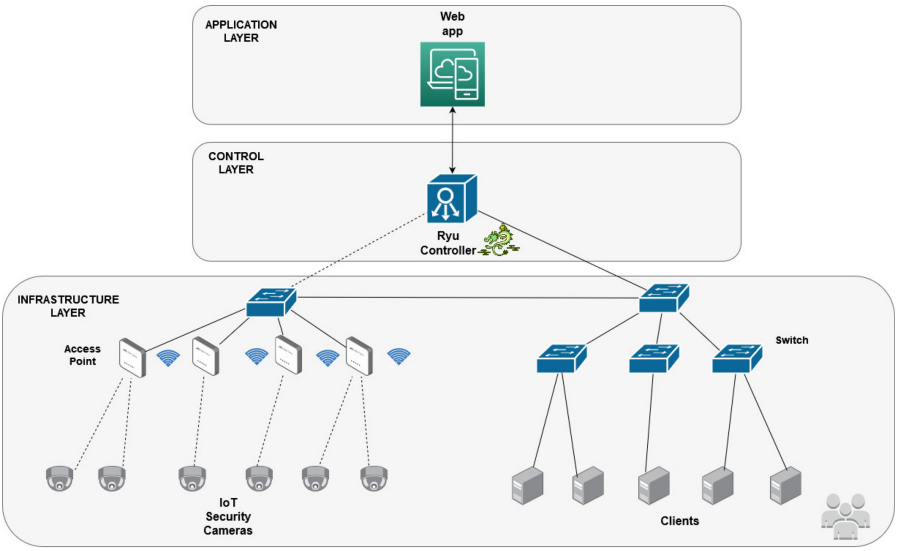
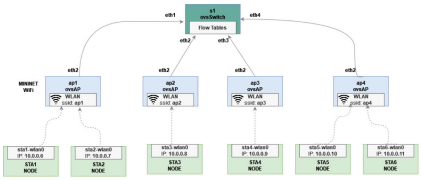
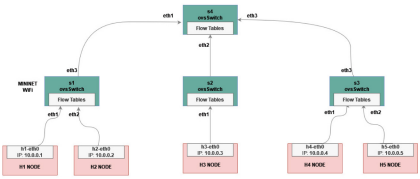


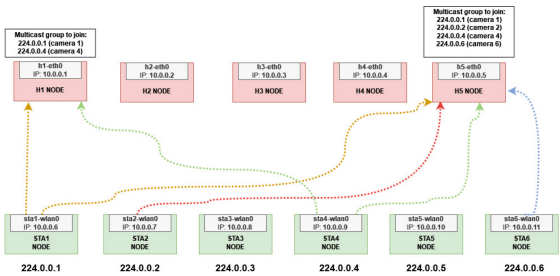
Fig. 4. Scenario used for evaluation.



(a) Network topology on IoT side.



(b) Network topology on the client side.



(c) Multicast flow definitions.

Fig. 5. Overview of the different topologies involved, along with the end flow associations.

effectiveness of our SDN-based solution, where the proposed workflow of Fig. 1 is used to share the data flows of the CCTV cameras for the sake of efficiency, while maintaining adequate levels of privacy. To this end, we perform different experiments that measure the network's efficiency. In particular, we assess the performance of UDP multicast connections when a client is receiving data from a camera, along with the efficiency of deploying OpenFlow rules when clients are added or removed. It is important to note that the number of clients and cameras used in this experiment are shown in Table 1.

Regarding resource consumption metrics, we perform a comparative analysis between various packet loads (180, 360, 720, and 1000 pkt/s). The goal of this test is to evaluate whether sending flow rules to network devices substantially raises the CPU load of these devices, and to know their status during traffic redirection. More precisely, our goal is to measure the performance of the primary switch, which is the gateway for all rules and the resulting traffic flow.

Having said this, we start our analysis by examining the performance of the UDP multicast connection when a client receives 1,130 KBytes of data from the camera over a span of 10 s. To facilitate this, we employed the iperf tool [20], which enables the execution of client/server bandwidth tests, and provides various performance parameters of the connection outcome.

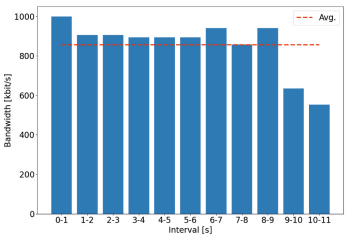
Figure 6 displays the results in terms of bandwidth and jitter for the UDP datagram client/server test. More specifically, Fig. 6a represents the bandwidth achieved at each one-second interval. As we can observe, it begins at its peak of 1,000 Kbit/s, then stabilizes at approximately 900 Kbit/s. Subsequently, from the ninth second until the conclusion, it declines by 33%, down to 600 Kbit/s. This primarily occurs because the majority of the file has been transferred, which in turn impacts the bandwidth. So, in general, the bandwidth is maintained consistently stable.

Table 1. Initial client-camera relationship.

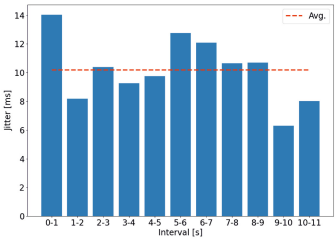
Client (IP)	Camera (IP)
1 (10.0.0.1)	1 (224.0.0.1)
	4 (224.0.0.4)
2 (10.0.0.2)	3 (224.0.0.3)
3 (10.0.0.3)	1 (224.0.0.1)
	2 (224.0.0.2)
	5 (224.0.0.5)
4 (10.0.0.4)	3 (224.0.0.3)
5 (10.0.0.5)	1 (224.0.0.1)
	2 (224.0.0.2)
	4 (224.0.0.4)
	6 (224.0.0.6)

Regarding jitter, Fig. 6b shows its performance during the test. As it can be seen, it varies between 6.3 and 12.8 ms. A high jitter has a negative impact on the quality of the video transmission, being more susceptible for applications that process video in real time. When performed on our simulation, we cannot say that the transmission result is poor; however, if implemented on a real environment, it would be convenient to implement measures including network congestion management and adequate allocation of resources, to mitigate jitter as much as possible.

In terms of average results, the test showed an average bandwidth of 854 Kbit/s and a jitter of 10.19 ms, which are acceptable results for this application.



(a) Bandwidth results.



(b) Jitter results.

Fig. 6. UDP client/server traffic test.

In terms of OpenFlow rules performance, Table 2 shows two important parameters to be measured: the time required by the network to fully update when a new rule is added, and the time to delete a client with the use of an OpenFlow rule. In particular, the first time value is the measured time that the client nodes took to receive the traffic from the camera, measured from the time when we send the instructions from the application level. As can be observed, this time is 37.876 ms. This represents the interval required to implement the changes, and ensures that the network reflects the new configurations. It is important to mention that this will vary depending on various factors, such as the size or complexity of the network topology. However, the time obtained for our case shows a high efficiency in terms of updating the network, despite Mininet resource virtualization in the scope of a single standard machine.

Regarding the second time, this measures the time it takes for network devices to update the rules received from the application level, in this case to deny a client from receiving traffic. For this test, client number 5 will be used as the player, since in the initial scenario (see Table 1) it is the one with the most cameras allowed. As shown, the time value achieved is 67.052 ms. This is quite acceptable since the procedure is performed in less than 1 tenth of a second, and was performed on the client that was receiving the most traffic. These results highlight the agility of the SDN architecture in this solution, allowing the overall

Table 2. OpenFlow rules performance.

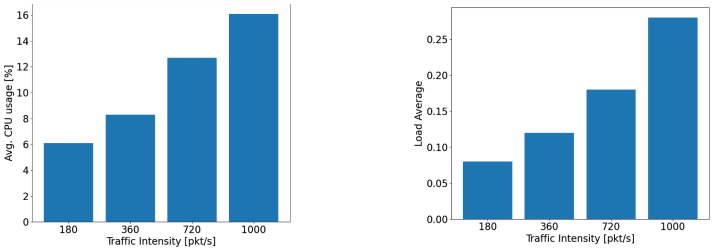
Time to update network	37.876 ms
Time to delete a client	67.052 ms

system to quickly adapt to the changes and demands of the network environment, while offering efficiency and low latency for this type of requests.

Finally, the question arises as to the solution's ability to manage resource consumption for high network traffic intensities. As depicted in Fig. 7, the results indicate a more than acceptable performance within the tested system. In terms of CPU usage for the main switch, Fig. 7a shows that it remains relatively low, having a peak value of only 16.1% despite an increase in the load, suggesting an efficient utilization of resources. Similarly, the system utilization during the test (load average), as illustrated in Fig. 7b, stays at reasonably low levels, under 0.3, suggesting that the system is not subjected to any excessive load. In particular, this means that the CPU was idle 70% of the time on average for the maximum traffic intensity.

As a sample of the traffic generated from the cameras for the evaluation of the CPU load, Fig. 8 shows the volume of data traversing the network; some traffic flow peaks can be spotted, occurring concurrently with falls, yet maintaining a good stability overall. This is to be expected due to the virtualized nature of all network elements, and performance should be better in actual deployments.

All in all, the results indicate that, as network traffic increases, the parameters assessed tend to increase correspondingly. However, this increase is not very significant. Therefore, we can draw the conclusion that the system demonstrates an acceptable efficiency within the given context, proving that it can manage a higher data flow without a performance drop.



(a) Average CPU usage in the main switch when varying the traffic load.

(b) Load Average comparison.

Fig. 7. Metrics when traffic flows through the network.

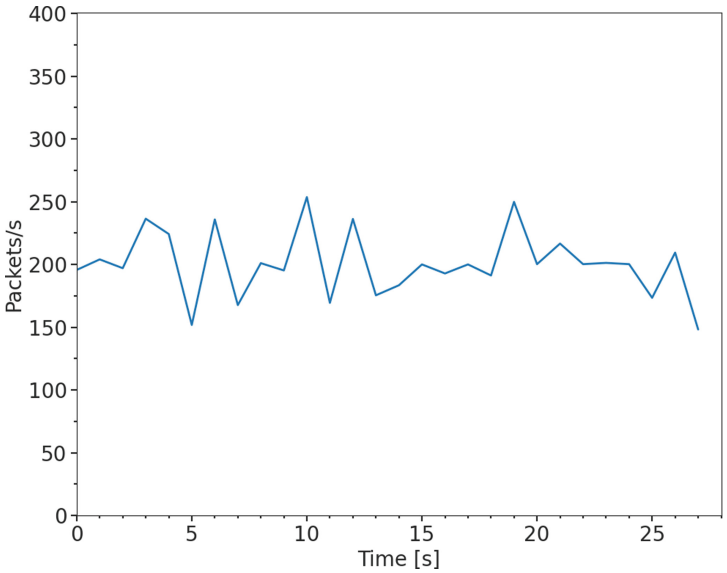


Fig. 8. No. of packets over time for 180 packets/s traffic load.

6 Conclusions and Future Work

As we move towards Smart City scenarios embracing thousands of IoT devices, avoiding further device proliferation is recommendable for the sake of efficiency, energy consumption, economy and aesthetics, among others. Hence, the sharing of data flows generated by IoT devices is recommendable, although it may generate new issues related to security and privacy of such data flows, especially when such devices generate sensitive data like video flows.

In this paper we addressed the aforementioned issue by proposing an SDN framework that allows sharing video flows generated by CCTV cameras. Due to the highly sensitive nature of such data, we devise a solution based on the Zero Trust paradigm that manages such flows with a very high granularity, to have full control on which clients can access the flows of individual cameras, and their allowed time span.

Experimental results using the Mininet platform show that our application, when combined with the Ryu SDN controller, is able to achieve the target goals in a straightforward and yet effective manner. In addition, we find that the time overhead for introducing network updates is maintained rather low: under 40ms for updating the network with new clients, and under 70ms to remove an existing client (receiving multiple flows). In addition, we show that traffic performance is maintained at good levels using our multicast-based solution, to efficiently reach multiple clients simultaneously.

As future work we plan to extend our solution to other types of IoT devices/data flows, and to deploy our solution in a testbed to have more realistic performance data, especially for the wireless part of the network.

References

1. Al-Turjman, F., Zahmatkesh, H., Shahroze, R.: An overview of security and privacy in smart cities' IoT communications. Trans. Emerg. Telecommun. Technol. **33**(3), e3677 (2022). https://doi.org/10.1002/ett.3677
2. Arasteh, H., et al.: IoT-based smart cities: a survey. In: 2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC), pp. 1–6 (2016). https://doi.org/10.1109/EEEIC.2016.7555867
3. Basta, N., Ikram, M., Kaafar, M.A., Walker, A.: Towards a zero-trust microsegmentation network security strategy: an evaluation framework. In: NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, pp. 1–7. IEEE (2022)
4. Bouloukakis, G., et al.: PrioDeX: a data exchange middleware for efficient event prioritization in SDN-based IoT systems. ACM Trans. Internet Things **2**(3), 1–32 (2021)
5. Federici, F., Martintoni, D., Senni, V.: A zero-trust architecture for remote access in industrial IoT infrastructures. Electronics **12**(3), 566 (2023)
6. Firouzi, R., Rahmani, R.: A distributed SDN controller for distributed IoT. IEEE Access **10**, 42873–42882 (2022)
7. Han, J., Shen, D., Karbowski, D., Rousseau, A.: Leveraging multiple connected traffic light signals in an energy-efficient speed planner. IEEE Control Syst. Lett. **5**(6), 2078–2083 (2021). https://doi.org/10.1109/LCSYS.2020.3047605
8. INTRIG: Mininet-WiFi. emulation platform for software-defined wireless networks. (2023). https://mininet-wifi.github.io/. Accessed 29 June 2023
9. KC, Y., Kang, C.S.: A connected car-based parking location service system. In: 2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), pp. 167–171 (2019). https://doi.org/10.1109/IoTaIS47347.2019.8980443
10. Keeriyattil, S.: Microsegmentation and zero trust: introduction. In: Zero Trust Networks with VMware NSX, pp. 17–31. Apress, Berkeley (2019). https://doi.org/10.1007/978-1-4842-5431-8_2
11. Kim, T., Ramos, C., Mohammed, S.: Smart city and IoT. Future Gener. Comput. Syst. **76**, 159–162 (2017). https://doi.org/10.1016/j.future.2017.03.034. https://www.sciencedirect.com/science/article/pii/S0167739X17305253
12. Li, Y., Su, X., Riekki, J., Kanter, T., Rahmani, R.: A SDN-based architecture for horizontal internet of things services. In: 2016 IEEE International Conference on Communications (ICC), pp. 1–7. IEEE (2016)
13. Mukherjee, B.K., Pappu, S.I., Islam, M.J., Acharjee, U.K.: An SDN based distributed IoT network with NFV implementation for smart cities. In: Bhuiyan, T., Rahman, M.M., Ali, M.A. (eds.) ICONCS 2020. LNICST, vol. 325, pp. 539–552. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-52856-0_43
14. Osman, A., Wasicek, A., Köpsell, S., Strufe, T.: Transparent microsegmentation in smart home IoT networks. In: HotEdge (2020)
15. Pemberton, D., Linton, A., Russell, S.: Ryu OpenFlow controller. University of Oregon, Technical report (2014)

16. Perwej, Y., Haq, K., Parwej, F., Mumdouh, M., Hassan, M.: The internet of things (IoT) and its application domains. Int. J. Comput. Appl. **975**(8887), 182 (2019)
17. Rahman, M.A., Rashid, M.M., Hossain, M.S., Hassanain, E., Alhamid, M.F., Guizani, M.: Blockchain and IoT-based cognitive edge framework for sharing economy services in a smart city. IEEE Access **7**, 18611–18621 (2019). https://doi.org/10.1109/ACCESS.2019.2896065
18. Syed, N.F., Shah, S.W., Shaghaghi, A., Anwar, A., Baig, Z., Doss, R.: Zero trust architecture (ZTA): a comprehensive survey. IEEE Access **10**, 57143–57179 (2022)
19. Team, M.: Mininet releases (2021). https://github.com/mininet/mininet/releases. Accessed 10 Sept 2022
20. Tirumala, A.: Iperf: The TCP/UDP bandwidth measurement tool (1999). http://dastnlanr.net/Projects/Iperf/
21. Wasicek, A.: The future of 5G smart home network security is micro-segmentation. Netw. Secur. **2020**(11), 11–13 (2020)
22. Yeganegi, K., Moradi, D., Obaid, A.J.: Create a wealth of security CCTV cameras. J. Phys. Conf. Ser. **1530**(1), 012110 (2020). https://doi.org/10.1088/1742-6596/1530/1/012110
23. Zhou, Z.: IoT-based smart garbage system for efficient food waste management. Sci. World J. (2014). https://doi.org/10.1155/2014/646953