Performance Evaluation of Software Defined Network (SDN) Based Surveillance System

Asaad H. AbdAli [1]*Ameer H. Murad [2]

[1, 2] Department of Information and Communication Engineering, Al-Khwarizmi College of Engineering, University of Baghdad, Baghdad, Iraq
[1]*eng85asaad@gmail.com
[2] ameer@kecbu.uobagdad.edu.iq

Abstract. There are several challenges in traditional video surveillance systems such as storage size and power consumption. In this study, the software-defined network is used to find solutions to these issues. SDN is a new technology that simplifies the control of the network by data plane is separated from the control plane. Also, it enables the multi-camera networks in video monitoring systems to be reconfigured dynamically. The main idea of the proposed system is to find a way for activate the cameras recording when necessary only by taking advantage of the characteristics of SDN and without using motion detection sensors. As far as we know, we are the first to take advantage of SDN features to implement activity and sleep modes on cameras without the need for sensors. A use case is installing cameras on a roadside to monitor vehicle movement on the campus. We created an application that interacts with the controller through RESTful API web services. The application sends the time of active mode and the time of sleep mode for each camera to the controller. Also, the controller developed to activate the cameras or return them to standby mode according to the specified time by the network application. Finally, videos sent from the cameras are merged into a single video on the server, which reduces the time and effort spent when analysing received video streams. The system was simulated using the Mininet emulator and the POX controller. The results showed that the proposed system overcomes the traditional system in terms of reducing power consumption and storage size, thus reducing the cost of multi-camera systems.

Keywords: SDN, POX controller, storage size, power consumption.

1. INTRODUCTION

Todays, monitoring is becoming ever more important to protect the safety of many different environments – including road monitoring, border surveillance, immigration monitoring, staff monitoring, identification of suspects, robbery, vandalism, and crime prevention. Various techniques were developed in recent decades for the automatic monitoring of CCTV cameras and sensors[1].

The conventional multi-camera network has congestion in traffic because of the static configuration of its architecture. In this network, the difficulty is that to carry out any required changes, devices must be reconfigured manually or replaced; this leads to more cost and effort[2].

In traditional video surveillance systems, the size of video data is one of the most important challenges that need enough storage space to save it. Also, it would be a very tough task to process such huge data to extract the required information [3].

A huge amount of data is generated with increasing trends of camera installation in public as in private places, which required development the storage methods. Also, some of the developing world's energy crises (electricity) make it necessary to reduce power consumption in monitoring systems[4].

It is estimated that 15 million hard drives and $2B are the cost of storage video data from 10 million cameras a year. The authors estimate the Video traffic will be 100 times higher between 2020 and 2030. They also expect the data on video monitoring to be more than twice the total available storage size. This implies that all video surveillance data will infeasible stored in 2030 at the expected growth rate [5].

The monitoring camera has used up to 40 Watts per year, according to one single camera. This leads us to find solutions to overcome the problems of energy consumption. In addition, prevent failure of cameras when a blackout of power [6].

The power consumption of IoT and WSN has been a hot topic of research in the past years. The main focus of the work was on reducing uninterrupted energy consumption by introducing various sleep patterns. With the recent more widespread use of battery-powered surveillance systems without power line, reducing energy consumption and prolonging battery life is becoming more and more important[7].

Within the past few years, the SDN becomes one of the most attractive structures that enhance network performance and handle surveillance system problems. SDN is the new technology of network architecture. It aims to simplify network administration and to be adaptable more than conventional architecture by decoupling control and data planes. The control plane manages the network in a centralized manner. Having centralized management and programming entity make more efficiency and low-cost upgrading and reprogramming for any network [8].

A video surveillance system with SDN consists of multiple IP cameras, a controller, OpenFlow switches, and a monitoring center. IP cameras obtain the video and transmit the stream to the monitoring center through policy that determined by the SDN controller in the network[9].

The Open Networking Foundation (ONF) specifics the SDN architecture as three-layer[10]: (i)The Application Layer: is responsible for communicating with the controller and for directing it on how to control various devices in the physical network infrastructure. (ii) The control layer:

represented by the controller which acts as the manager for the entire system, and responsible for communicating between the application layer and infrastructure layer.(iii) Infrastructure Layer: consists of the network devices that perform data forwarding.

The control plane communicates via two interfaces with the other planes [11] :(i)Northbound interface: NBI is a bridge between the control plane and the application plane such as RESTful APIs. (ii) Southbound interface: SBI is an interface between the data plane and the control plane such as OpenFlow protocol.

The most commonly implemented protocol in SDN is OpenFlow. OpenFlow is a Southbound API, which provides a secure channel for communicating the controller with switches, enabling both the OpenFlow switches and controller to understand each other [12].

The network application communicates with the controller via the northbound APIs. It is the most critical component of the SDN controller's architecture. The most valuable benefit of SDN is its ability to support and enable innovative applications. Northbound APIs must support a broad range of applications due to their criticality. After establishing the OpenFlow protocol as the southbound interface, the ONF recently concentrated on the SDN API. They've formed a Northbound Working Group to write code, prototype solutions, and establish standards. REST is the most frequently used northbound interface at the moment and is implemented by the majority of controllers[13].

The REST API was used to provide the applications for data storage in the controller as an NBI. It provided the following options: post, put, delete, and get. Restconf's output can be in either XML or JSON format[14].

2. LITERATURE REVEIW

Different theories exist in the literature regarding the evolution of video surveillance systems. There are relatively few available studies in the area of video transmission over the SDN. The related works are clarified in this section.

In (2015), Martijn [15] proposed in his thesis using SDN to manage a multi-camera network for monitoring a soccer field. This requires coordination between cameras to successfully perform the required tasks. The controller needs to identify the locations of cameras and soccer. The locations are being sent from network application to the controller through RESTful API. The controller enables a flow between two cameras in the specific zone where the ball is located and passes the images to the client while the other traffic is blocked. The author made a trade-off between Floodlight and RYU controllers in this work. The default routing algorithm was used for these controllers.

In 2015, Ricardo Ramalho dos Santos [16] proposed to use an SDN controller application that calculates the routes between nodes by utilizing different route computation algorithms. This thesis presented that the usage of a constrained multiple path algorithm improves the QoS metrics. It uses Self-adaptive Multiple Constraints Routing Algorithm (SAMCRA) algorithm that content Single-path algorithms like Dijkstra's Algorithm and Multi-path algorithms like

Link-disjoint algorithms. In this thesis, use Opendaylight controller (ODL) and Mininet emulator.

In (2016), Mkwawa, et al. [17] proposed using SDN for a traffic intensity-based video quality control system under Adaptive Dynamic Streaming through HTTP. Mininet emulator and OpenDaylight controller are used to implementing the proposed scheme. A Python application that communicates via RESTful API web services with the controller was developed. This application analyzes the network flows, suggests optimal paths, and provides a directive on the best path to modify the traffic when congestion occurs.

In 2017, Corrado, et al. [18] proposed an SDN/NFV based surveillance system that allowing anyone to easily distribute a large number of IP cameras inside a smart city. The mobile app is developed, which allows: firstly, to record an IP camera to the Video Monitoring System. Secondly, connect the service. This platform allows easy deployment of functionality and simplified management of costs. Also, the platform reduces traffic by not replicating the video stream generated by a camera for each destination. The data flow from source to destination will be based on the type of service requested by the end-user. Mininet emulator, opendaylight controller, and the shortest path algorithm are used in this work.

In (2017) Reza, et al. [19] proposed the adaptive traffic engineering approach to find the best paths in the video surveillance system between the monitoring center and cameras. The LARAC algorithm is applied and the Dijkstra algorithm to select the shortest route between source and destination. This article uses Opendaylight controller for the application of traffic technology. Mininet emulator is also used in this work with the VLC media player. The results of the simulation demonstrate that applying the type-2 fuzzy for traffic engineering results- in better handling of traffic flow with lower delay.

In 2018, Chih-Heng, et al. [20] proposed an effective routing Algorithm based on SDN technology to select the best route for sending the video sreaming between sender and receiver. The authors determine the effectiveness of a genetic algorithm (GA-SDN) on the performance compared to the traditional Bellman-Ford algorithm. Mininet emulator and POX controller used in this work.

In (2019) Sharleen, et al. [21] developed An SDN-based framework for enhancing the performance of IP Video monitoring networks. They used two indicators for network levels to infer video quality measurements such as latency, packet loss, and jitter. The performance of the proposed technique was evaluated using the Mininet emulator and RYU controller. Also, the shortest path algorithm was used to specify the optimal path between sender and receiver.

In (2020)Tariq, et al. [22] proposed to execute load balancing jobs for traffic (LB) in software-defined data centers (SDN-DC) based on the protocol of open flow (OF) to overcome limitations of traditional data center networks (DC) caused by rapidly increasing in amounts of apps, websites, data-storage requirements. To test the traffic load balancing (LB) jobs, the opendaylight controller and ant-colony optimization algorithm are used. After LB implementation, the bandwidth, data transfer, and throughput increased, while a lowered average delay.

As shown in the literature survey, most of the related works have not considered the problems related to the power consumption and storage size in monitoring systems. The power consumption and storage size are two big challenges that need solutions. In this paper, we presented a solution to these challenges by taking advantage of the characteristics of SDN to implement active mode and sleep mode in traditional low-cost cameras instead of expensive modern cameras that depend on motion detection sensors mainly in their work. These sensors are also consuming power even in sleep mode because run all the time to wait for an incident. Therefore, this research will contribute to reducing the cost of multi-camera systems. Furthermore, it will improve the efficiency of these systems by lowering power consumption, reduction the storage space needed to save videos, and saving the time spent searching for and investigating these videos.

3. PROBLEM STATEMENT

One of the main problems in traditional multi-camera systems is the storage size due to the huge amount of data sent from cameras. Also, power consumption is another challenge in surveillance systems. The goal of this work is to present a monitoring system for the movement of cars inside the campus based on SDN to control the operation of cameras' recording in traditional systems only when necessary without using motion detection sensors.

In this paper, we built a system using SDN having an ability to reduce storage size through enabling only necessary data to be sent from cameras. Also, it reduces the power consumption in traditional multi- camera network by not making the cameras run continuously. The proposed system is emulated using the Mininet and the POX controller.

4. PORPOSED SYSTEM MODEL

The proposed system based on the software-defined network consists of three cameras, four OpenFlow switches, one SDN controller, and a monitoring center. The cameras used are the wireless IP cameras. Only the first camera is equipped with motion and speed detection features, while the other cameras are normal cameras that do not have these features.

The proposed system is emulated utilizing a Mininet emulator[23], which is used to emulate a virtual network consisting of controllers, switches, hosts, and links. The POX has been used as an SDN controller which uses RESTful API Web services to interact with network application and OpenFlow protocol to interact with Open virtual switches. The simulation environment performed using Ubuntu 20.04.2 LTS operating system. Figure 1 shows the proposed video surveillance system.

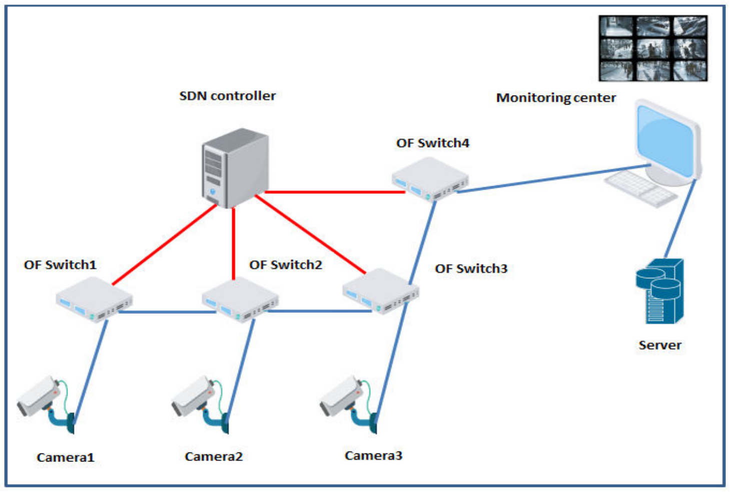

Figure 1.The proposed video surveillance system

5. SYSTEM COMPONENTS

A. **Application**: An application created that calculates the time of active mode and the time of sleep mode for each camera and sends them to the POX controller via the RESTful API Web services based on the speed of the car that sends from camera1, the distance between cameras, and the coverage distance for each camera.

B. **Controller**: The pox controller developed to enable or stop a flow between cameras and monitoring center based on the specific active mode or sleep mode.

C. **Server**: The main purpose of the server is to store the information that comes from the cameras and merges it into one movie.

D. **Switches**: The switches that are used are called Open Virtual switches (OVS). These switches receive camera images and transmit the images to the network. The rules in the flow table match the packets according to IP src/dst and forward them to ports.

E. **Cameras**: The used cameras are the wireless IP cameras that run on the battery. The only camera1 contains smart monitoring features like (motion detection, speed detection, etc.). These cameras transmit and receive data via the Internet and a computer network. We can access these cameras by specifying an IP address for each of them. Figure 2 shows block diagram for the work of these components.

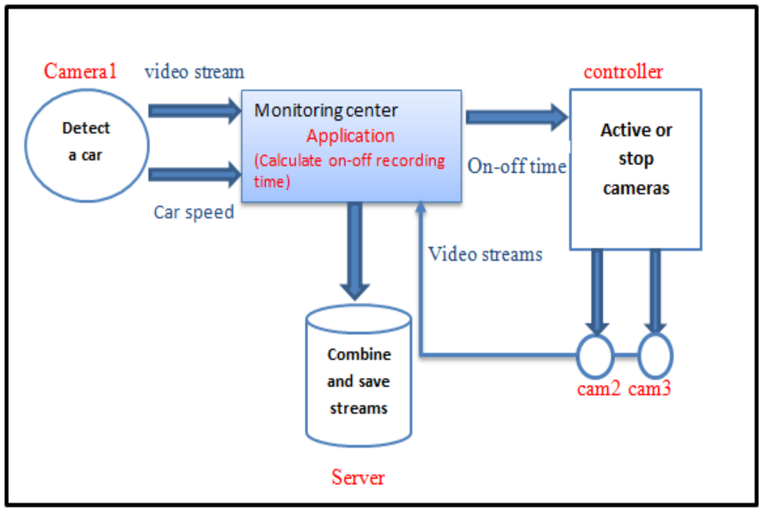

Figure 2.Block diagram for system components work.

6. NETWORK TOPOLOGY

In (figure 3), topology created using miniedit which represents the graphical user interface (GUI) for mininet. Host4 represents the server while host1, host2, and host3 represent the cameras.

The IP configuration for H1, H2, H3, and H4 are 10.0.0.1, 10.0.0.2, 10.0.0.3, and 10.0.0.4 respectively and this work sets the controller's default OpenFlow port as 6633.The configuration for each system node is shown in Table 1.

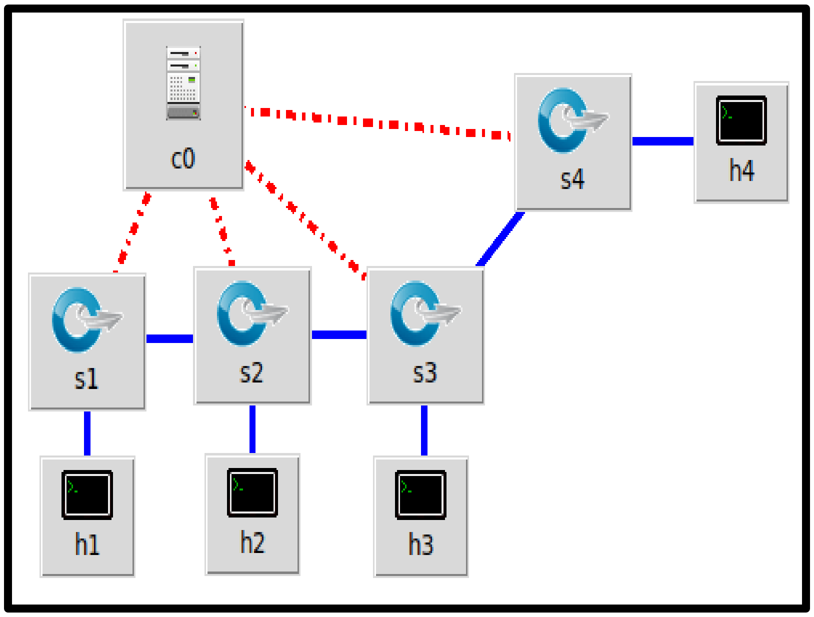

Figure 3.Network topology

Table 1.configuration set up for the system

Virtual devices	**IP or Port**
Nodes	c0 h1 h2 h3 h4 s1 s2 s3 s4
H1	10.0.0.1 (eth0)
H2	10.0.0.2 (eth0)
H3	10.0.0.3 (eth0)
H4	10.0.0.4 (eth0)
c0	127.0.0.1 (6633)
S1	s1-eth1:h1-eth0 s1-eth2:s2-eth3
S2	s2-eth1:h2-eth0 s2-eth2:s3-eth3 s2-eth3:s1-eth2
S3	s3-eth1:h3-eth0 s3-eth2:s4-eth2 s3-eth3:s2-eth2
S4	s2-eth1:h2-eth0 s2-eth2:s3-eth3 s2-eth3:s1-eth2

We used OpenCV to read, view, and merge the videos. Also, we used the TCP protocol to send the frames between the hosts. To make topology ready to run the simulation test, the following performed:

(i) We wrote Python code that call videos in MP4 format on each of the hosts.
(ii) We wrote Python code that through it the host4 listens for receiving the frames from the host1, host2, and host3. Then it merges the videos received from the hosts into a single video.
(iii) We wrote Python code that added to POX components, which enables the POX controller to turn on and off the hosts based on the times sent from the application.
(iv) We created a web page with HTML and JavaScript to implement the network application. Depending on the law of time: Time = distance/speed, the application calculates the time to turn on record and off it in cameras and sends it to controller via RESTful API.

7. TESTBED

We sent random speeds to the network application with the start of transmission from host1 (camera1) to host4 (server), this application calculates the on and off times of host2 (camera2) and host3 (camera3) and sends them to the POX controller based on a mechanism specified by us that will be clarified in the section of results and discussion later. We noticed that the controller was able to on and stop host2 and host3 according to the time specified by the application. Also, the videos sent from host1, host2, and host3 merged into one video and stored in host4 each time the network is tested by sending a different speed from host1. Figure 4 represents a graph of one of the tests where the packets were captured by the Wireshark analyzer.

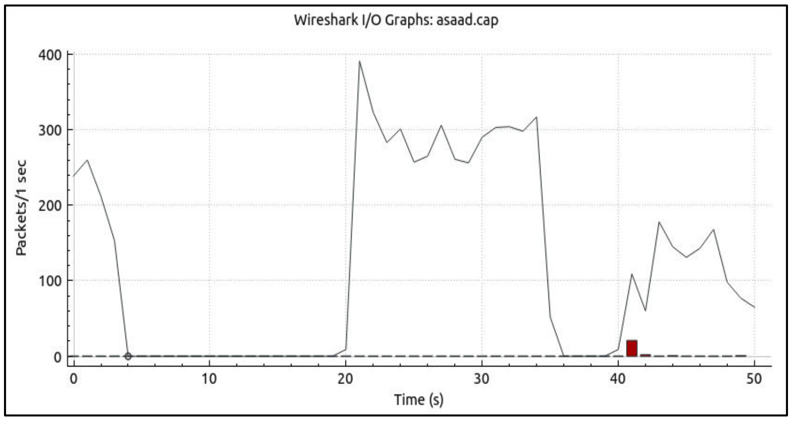

Figure 4. Wireshark I/O graph

The analysis of network packets is shown in Table 2 below. Only packets that are important for this application are kept in this table.

Table 2.Output of Wireshark

Time	Source	Destination	Protocol	Info
0.000	10.0.0.1	10.0.0.4	TCP	Src port: 54592→ Dst port: 65432
20.175	10.0.0.2	10.0.0.4	TCP	Src port: 53444→ Dst port: 65432
35.414	10.0.0.2	10.0.0.4	TCP	Src port: 36808→ Dst port: 5001
40.359	10.0.0.3	10.0.0.4	TCP	Src port: 60658→ Dst port: 65432

50.587	10.0.0.3	10.0.0.4	TCP	Src port: 56846→ Dst port: 5002

As can be seen in Table (4.1), (0.000) it is the time when the transmission starts from host 1. (20.175) it is the time when activate host2 (i.e. 20 seconds after the transmission started from Host1) and (35.414) it is the time when return host2 to sleep mode. Also, (40.359) it is the time when activate host3 (i.e. 40 seconds after the transmission started from Host1) and (50.587) it is the time when return host3 to sleep mode.

8. RESULTS AND DISCUSSIONS

To evaluate the results of the proposed system, we assumed that the cameras were installed on a street at a distance of 1 km inside the campus. As we know that the car's speed should not exceed 40 km / h inside the campus. We took the different speeds of cars randomly. Also, we assumed the distance between camera1 and camera2 is 300 meters, the distance between camera1 and camera3 is 1 km, and we took the following wireless IP camera (model C3A) specifications and we assumed that it does not have a motion detection sensor.

1. Power Supply: (5500) mA/h rechargeable battery and 5V DC.
2. Power consumption: (1000) mA/h, (0.277) mA/sec.
3. Frame Rate: 15 fps.
4. Resolution: 1920 x 1080, FHD.
5. Coverage distance: 15 m.

8.1. calculate active and sleep modes

Depending on speed formula[24] we find the time related to it:

Time=distance/speed (1)

The application we build can determine the time to turn recording on and off in camera 2 and sends it to controller via RESTful API. The distance between camera 1 and camera 2 is 300 meters and between camera 1 and the beginning of the coverage distance of camera 2 is 292.5 meters (The coverage distance of the cameras used is 15 meters). So, to start activating recording video in camera 2 when the vehicle speed for example is 40 km/h, the application calculates the specific time as follow.

$$\begin{aligned} \text{Tcam2-on} &= \text{distance / speed} \\ &= 292.5\text{m} / 40 \text{ km/h} \\ &= 26.1 \text{ sec} \end{aligned}$$

This means that the POX controller will activate the recording in camera 2 after 26.1 seconds from camera 1 starts transmitting.
Also, the application calculates the time to turn off the recording in Camera 2 by the distance of the camera coverage and the speed of the vehicle.

Tcam2-off = distance / speed
= 15 m / 40 km/h
= 1.35 sec

So, the POX controller will activate the recording in camera 2 for only 1.35 seconds and then return it to standby mode.
The application then calculates the time to turn on and off the recording in Camera 3 in the same way as calculated in camera 2.Then we calculate the total time for the car to cross a distance of 1km (Cameras deployment space) in a velocity of 40 km / h.

Total time = Total distance / Total speed
= 1 km / 40 km/h
= 90 sec

Based on the above data, we conclude the following:
In the conventional system, each camera will run for 90 seconds, while in the proposed system, the same camera will only run for 1.35 seconds for the same distance and speed of the car.
We took different speeds randomly inside the campus and analyzed them. Also, we compared the recording run time for each camera in the proposed system with its counterparts in the traditional system (for the same distance and speed). The results showed that the proposed system outperforms the traditional system in reducing the recording run time for cameras. Table 3 and figure 5 show the recording runs durations in both systems.

Table 3.Comparison of the recording runs durations in both systems.

Speed of car (km/h)	distance (km)	Proposed system	Traditional system
		active time of camera (sec)	active time of camera (sec)
5	1	10.8	720
10	1	5.4	360
15	1	3.6	240
20	1	2.7	180
25	1	2.16	144

30	1	1.8	120
35	1	1.54	103
40	1	1.35	90

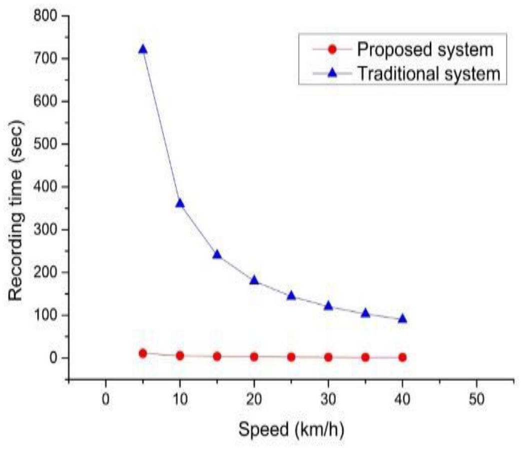

Figure 5.Intervals of recording for each camera.

8.2. Power consumption

8.2.1 Power consumption in the traditional system

For example when the speed of the vehicle passing on the road is 40 km / h, the recording in each camera of the conventional system works for 90 seconds when the vehicle velocity is 40 km/h and the distance is 1km.

This means the following:

Power consumption for each camera = 0.277 * 90
= 25 mA per 90 sec
= (25/1000) * 5
= 0.125 Watt

8.2.2 Power consumption in the proposed system

For example when the speed of the vehicle passing on the road is 40 km / h, the recording in each camera of the proposed system works for 1.35 seconds when the vehicle velocity is 40 km/h and the distance is 1 km. This means the following:

Power consumption for each camera = 0.277 * 1.35
= 0.375 mA per 90 sec
= (0.375/1000) * 5
= 0.0019 Watt

Based on the results shown in table (3), a comparison of the power consumption of each camera was made during the same time (the same travelled distance and speed of the car) in both systems. The results showed that the proposed system overcomes the traditional system in terms of reducing the power consumption by cameras. Table 4 and figure 6 show the results of the comparison.

Table 4.Power consumption in both systems

Speed of car (km/h)	Time (second)	Power consumption/ proposed system (Watt)	Power consumption/ traditional system (Watt)
5	720	0.015	1
10	360	0.0075	0.5
15	240	0.005	0.333
20	180	0.00375	0.25
25	144	0.003	0.2
30	120	0.0025	0.166
35	103	0.0021	0.143
40	90	0.0019	0.125

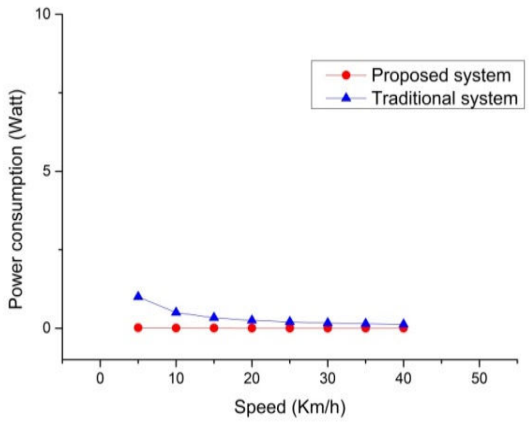

Figure 6.Power consumption in both systems.

8.3. Storage size

We calculated the frame size and the video size using the equations (2) and (3) below[25].

Frame size (Bytes) = (Camera Resolution x Color Depth)/8 (2)
= (1920 x 1080 x 24) / 8
= 6.2 MB
Video size (Bytes/Sec.) = Frame size x Frame rate (3)
= 6.2 x 15 fps
= 93 MB/sec

8.3.1 Storage size in the traditional system

For example when the speed of the vehicle passing on the road is 40 km / h, the total time the car takes to cross a distance of 1 km (Cameras deployment distance) in a velocity of 40 km / h as follow:

Total time = Total distance / Total speed
= 1 km / 40 km/h
= 90 sec

Since the recording in the cameras works continuously in the traditional system. This means that the recording of camera works for 90 seconds in a velocity of 40 km / h. Therefore, the video size sends from each camera in the conventional system within 90 seconds is as follows:

Video size = 93 MB/s x 90 sec
= 8370 (MB per 90 sec)

8.3.2 Storage size in the proposed system

The pox controller makes recording of cameras works for a specific time in the proposed system. The recording playback time for each camera in the proposed system with a vehicle passing at 40 km/h is 1.35 seconds. Therefore, the video size transmitted from each camera is as follows:

Video size = 93 MB/s x 1.35 sec
= 125.55 (MB per 90 sec (

We notice the big difference between the video sizes sent from the same camera in the two systems during the same time (90 seconds). In traditional video surveillance applications, the size of video data is one of the most important challenges for Video content analysis. The proposed system based on SDN will contribute to process such huge data to extract the required information.

Based on the results shown in table (3), the video size that sends from each camera in both systems was compared (during the same time). The results showed that the proposed system outperforms the traditional system in reducing storage size for the surveillance system. Table 5 and figure 7 show the results of the comparison.

Table 5.Comparison of the video size in both systems

Speed of car (km/h)	Total time (sec)	Video size / Traditional system (MB)	Video size / Proposed system (MB)
5	720	66960	1004.4
10	360	33480	502.2
15	240	22320	334.8
20	180	16740	251.1
25	144	13392	200.88
30	120	11160	167.4
35	103	9579	143.22
40	90	8370	125.55

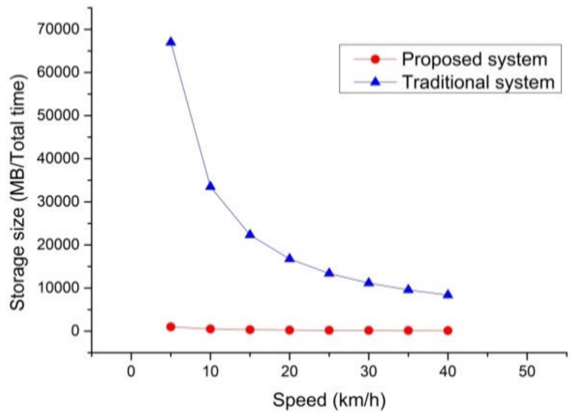

Figure 7.Video size in both systems.

9. CONCLUSION

- Traffic congestion and Storage size were reduced in a large percent, and we expect this to be very beneficial to real-time traffic monitoring for smart cities.
- Power consumption was also reduced in a large percent, so the cameras will work for longer hours when using SDN compared to cameras operating on a regular network.
- By using the proposed surveillance system, the cost will be reduced because we do not need all the surveillance cameras to be professional. We only need one camera of this type while the rest of the cameras will be normal.
- Among the capabilities provided by the Software Defined Network is the scalability and flexibility to manage the networks dynamically. So the camera's network can be expanded and new cameras can be dealt with easily.
- When the car passes in front of the first camera and sends the speed to the monitoring center, there is a possibility of the car’s speed changing or stopping. Therefore, the control process of the cameras will not be as planned. This requires some editing in the codes of the controller and application to address this situation.
- Since the continued operation of our system depends mainly on the first camera, it is necessary to have a backup one for it that runs in a standby mode in case of any damage or failure in the first camera.

REFERENCES

[1] M. Chowdhury, J. Gao, and R. Islam, “Human surveillance system for security application,” *Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng. LNICST*, vol. 164, pp. 711–724, 2015, doi: 10.1007/978-3-319-28865-9_47.

[2] J. Sree, “Virtualization of Traditional Networks using,” vol. 7, no. July, pp. 6–11, 2019, doi: 10.13140/RG.2.2.12079.89767.

[3] B. N. Subudhi, D. K. Rout, and A. Ghosh, “Big data analytics for video surveillance,” *Multimed. Tools Appl.*, vol. 78, no. 18, pp. 26129–26162, 2019, doi: 10.1007/s11042-019-07793-w.

[4] G. K. Adam, P. A. Kontaxis, L. T. Doulos, E. N. D. Madias, C. A. Bouroussis, and F. V. Topalis, “Embedded microcontroller with a CCD camera as a digital lighting control system,” *Electron.*, vol. 8, no. 1, 2019, doi: 10.3390/electronics8010033.

[5] A. Mohan, K. Gauen, Y. H. Lu, W. W. Li, and X. Chen, “Internet of video things in 2030: A world with many cameras,” *Proc. - IEEE Int. Symp. Circuits Syst.*, pp. 2–5, 2017, doi: 10.1109/ISCAS.2017.8050296.

[6] M. Ahmad, I. Ahmed, K. Ullah, I. Khan, A. Khattak, and A. Adnan, “Energy efficient camera solution for video surveillance,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 3, pp. 522–529, 2019, doi: 10.14569/IJACSA.2019.0100367.

[7] H. Kim and H. J. Lee, “A low-power surveillance video coding system with early background subtraction and adaptive frame memory compression,” *IEEE Trans. Consum.*

Electron., vol. 63, no. 4, pp. 359–367, 2017, doi: 10.1109/TCE.2017.015073.

[8] S. Chourasia and K. M. Sivalingam, "Experimental study of SDN-based evolved packet core architecture for efficient user mobility support," *Resour. Alloc. Next-Generation Broadband Wirel. Access Networks*, pp. 273–298, 2017, doi: 10.4018/978-1-5225-2023-8.ch012.

[9] S. R. Shaker, "2989-Article Text with Author names and affiliation-21391-1-10-20200318.pdf," vol. 17, no. 1, pp. 391–400, 2020.

[10] D. Gopi, S. Cheng, and R. Huck, "Comparative analysis of SDN and conventional networks using routing protocols," *IEEE CITS 2017 - 2017 Int. Conf. Comput. Inf. Telecommun. Syst.*, pp. 108–112, 2017, doi: 10.1109/CITS.2017.8035305.

[11] F. Laassiri, M. Moughit, and N. Idboufker, "Evaluation of the QoS parameters in different SDN architecture using Omnet 4.6++," *2017 18th Int. Conf. Sci. Tech. Autom. Control Comput. Eng. STA 2017 - Proc.*, vol. 2018-Janua, pp. 690–695, 2018, doi: 10.1109/STA.2017.8314976.

[12] S. Yamashita, A. Yamada, K. Nakatsugawa, T. Soumiya, M. Miyabe, and T. Katagiri, "Extension of OpenFlow protocol to support optical transport network, and its implementation," *2015 IEEE Conf. Stand. Commun. Networking, CSCN 2015*, pp. 263–268, 2016, doi: 10.1109/CSCN.2015.7390455.

[13] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "SDN controllers: A comparative study," *Proc. 18th Mediterr. Electrotech. Conf. Intell. Effic. Technol. Serv. Citizen, MELECON 2016*, no. 978, pp. 18–20, 2016, doi: 10.1109/MELCON.2016.7495430.

[14] E. Husni and A. Bramantyo, "Design and Implementation of MPLS SDN Controller Application based on OpenDaylight," *2018 Int. Symp. Networks, Comput. Commun. ISNCC 2018*, 2018, doi: 10.1109/ISNCC.2018.8530900.

[15] M. Rymen, "Software-Defined Networking for Multi-Camera Systems," no. January, 2015.

[16] R. Santos, "Development of an OpenFlow controller application for enhanced path computation," 2015, [Online]. Available: https://estudogeral.sib.uc.pt/handle/10316/35573.

[17] I. H. Mkwawa, A. A. Barakabitze, and L. Sun, "Video quality management over the software defined networking," *Proc. - 2016 IEEE Int. Symp. Multimedia, ISM 2016*, pp. 559–564, 2017, doi: 10.1109/ISM.2016.142.

[18] C. Rametta, G. Baldoni, A. Lombardo, S. Micalizzi, and A. Vassallo, "S6: A Smart, Social and SDN-based Surveillance System for Smart-cities," *Procedia Comput. Sci.*, vol. 110, pp. 361–368, 2017, doi: 10.1016/j.procs.2017.06.078.

[19] R. Mohammadi and R. Javidan, "An adaptive type-2 fuzzy traffic engineering method for video surveillance systems over software defined networks," *Multimed. Tools Appl.*, vol. 76, no. 22, pp. 23627–23642, 2017, doi: 10.1007/s11042-016-4137-0.

[20] Y. S. Yu and C. H. Ke, "Genetic algorithm-based routing method for enhanced video delivery over software defined networks," *Int. J. Commun. Syst.*, vol. 31, no. 1, pp. 1–13,

2018, doi: 10.1002/dac.3391.

[21] S. J. Y. Go, C. A. M. Festin, and W. M. Tan, “An SDN-based framework for improving the performance of Underprovisioned IP video surveillance networks,” *Proc. - 2018 15th Int. Symp. Pervasive Syst. Algorithms Networks, I-SPAN 2018*, no. 1, pp. 154–161, 2019, doi: 10.1109/I-SPAN.2018.00033.

[22] T. E. Ali, A. H. Morad, and M. A. Abdala, “Traffic management inside software-defined data centre networking,” *Bull. Electr. Eng. Informatics*, vol. 9, no. 5, pp. 2045–2054, 2020, doi: 10.11591/eei.v9i5.1928.

[23] K. Kaur, J. Singh, and N. S. Ghumman, “Mininet as Software Defined Networking Testing Platform,” *Int. Conf. Commun. Comput. Syst.*, no. December, pp. 3–6, 2014.

[24] J. Walker, D. Halliday, and R. Resnick, *Fundamentals of Physics Halliday & resnick 10ed*. 2014.

[25] S. M. Qaisar, D. Sidiya, M. Akbar, and A. Subasi, “An event-driven multiple objects surveillance system,” *Int. J. Electr. Comput. Eng. Syst.*, vol. 9, no. 1, pp. 35–44, 2018, doi: 10.32985/ijeces.9.1.2.