Open Access **Baghdad Science Journal** **P-ISSN: 2078-8665**
2020, 17(1) Supplement (March):391-400 **E-ISSN: 2411-7986**

DOI: https://dx.doi.org/10.21123/bsj.2020.17.1(Suppl.).0391

Software Defined Network of Video Surveillance System Based on Enhanced Routing Algorithms

Sukaina R. Shaker ***Mustafa I. Salman****

Received 16/1/2019, Accepted 10/4/2019, Published 18/3/2020

Abstract:

Software Defined Network (SDN) is a new technology that separate the control plane from the data plane. SDN provides a choice in automation and programmability faster than traditional network. It supports the Quality of Service (QoS) for video surveillance application. One of most significant issues in video surveillance is how to find the best path for routing the packets between the source (IP cameras) and destination (monitoring center). The video surveillance system requires fast transmission and reliable delivery and high QoS. To improve the QoS and to achieve the optimal path, the SDN architecture is used in this paper. In addition, different routing algorithms are used with different steps. First, we evaluate the video transmission over the SDN with Bellman Ford algorithm. Then, because the limitation of Bellman ford algorithm, the Dijkstra algorithm is used to change the path when a congestion occurs. Furthermore, the Dijkstra algorithm is used with two controllers to reduce the time consumed by the SDN controller. POX and Pyretic SDN controllers are used such that POX controller is responsible for the network monitoring, while Pyretic controller is responsible for the routing algorithm and path selection. Finally, a modified Dijkstra algorithm is further proposed and evaluated with two controllers to enhance the performance. The results show that the modified Dijkstra algorithm outperformed the other approaches in the aspect of QoS parameters.

Key words: Bellman-Ford algorithm, Dijkstra algorithm, Software defined network (SDN).

Introduction:

Video surveillance is critical for different aspects of life. The main objective of surveillance system to keep people's care, or minimize human dangers associated with illegal or criminal activity. The video surveillance frameworks are very significant in our daily lives owing to the number of applications they make possible. The causes for using the video surveillancebenefit in such frameworks are differing, ranging from protection requests and military packages to scientific purposes (1). A video surveillance that uses the SDN comprises number of IP cameras, OpenFlow switches, a monitoring center and a controller. The objective of creating such a framework is to watch and monitor a predefined place.

IP cameras capture the video and send the video file through the network to the monitoring center. The policy of the controller over the network is responsible about finding the best path between the IP cameras and monitoring center.

Computer Engineering, University of Baghdad, Baghdad, Iraq.
*Corresponding author: mustafa.i.s@coeng.uobaghdad.edu.iq
*ORCID ID: 0000-0003-4454-9743

After that, the controller should send all Open Flow tables and the information about the path to Open Flow switches for chosen the best path (2).

The Open Networking Foundation(3) (ONF) defines the SDN as follows: " In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications."(4). The SDN architecture consist three layers. First layer (Infrastructure layer) consists of both physical and virtual network devices. Second layer (Control layer) involves of a centralized control plane. It provides centralized global view to entire network. Third layer (Application layer) contains of network services, application that used to interact with control layer(5). The SDN uses the OpenFlow protocol to interface with OpenFlow switches. It allows both the controller and all the switches to understand each other(6).

In Computer Networks, routing is performed by defining some flow rules in a routing table, these rules contain the source and destination IP-addresses and MAC-address. When a packet arrives at a device, the device checks the flow table if it is available or not, and take the action (forwards, reject, send to the controller) as per the rules set by the routing protocol (6). The routing time of SDN networks is lesser compared to traditional Networks. At increase N node the conventional networks is consume more time for change the path while SDN require less time(7). .SDN serve the routing algorithm Therefore, the The Bellman ford algorithm uses relaxation to select single source shortest paths on the graphs, it applied by (8). The time complexity for Bellman Ford is N^3. As a result, it consume more time for finding all the paths(9). Because the surveillance system should be fast and reliable, the routing algorithm requires less time to chose the path. Consequently, the Dijkstra algorithm is more suitable than bellman ford for video surveillance system. The time complexity of Dijkstra algorithm is N^2 (9) which is less than the Bellman Ford algorithm.

Related Works

Different theories exist in the literature regarding the evolution of video surveillance systems and their relation to routing techniques. A considerable amount of literature has been published on how the captured video can be transmitted over the traditional networks, There are relatively few published studies in the area of video transmission over the SDN.

Panwaree, et al. (10) proposed that the video send over two types of OpenFlow enabled network testbeds (Mininet emulated and Open-v-Switch PC cluster). The authors use a POX controller in both methods and the VLC media player in both server and client sides .The shortest path algorithm was used as routing algorithm.

Harold and Arjan (11) have achieved three contributions, to begin with, it shows the video over software defined network (V-SDN), a network construction that select the best path using a nwtwork wide-view. Then, it portrays the V-SDN protocols, which utilized by the designer to get information about QoS from the network. At last, it displays the results of applying a system model and calculate the behavior of system utilizing message complexity. The author did not show the type of controller that used in the system. A routing protocol was used to find the best path among the IP cameras and the checking center.

Martijn (12) proposed use of a Software-Defined Networking that can be use in a dynamically configurable multi-camera environment for the playground. The controller in the network teach the cameras nodes and their location on the surveillance system. Using an API, an application was developed such that it gives the location of a ball on the field to the controller. This controller active a flow between the cameras that are cooperating on the specific work. This thesis a trade-off is made between RYU and Floodlight. The default routing algorithm was used for these controllers.

Reza et al. (2) proposed a traffic engineering technique to calculate the best routes between the cameras (source) and checking center (destination) in a video-surveillance system. This approach is based totally on Constraint Shortest path (CSP) issues and calculates the least cost path. Because of negative path completeness of the CSP issues, Lagrange-Relaxation-based-Aggregate-Cost (LARAC) algorithm is used to solve it. The primary proposed traffic engineering technique that is based totally on kind (2) fuzzy-set for sending video packets over SDN-network. The main-contribution of the this method is works to implement type (2) and type (1) fuzzy logic for computing the link-cost for all network links according to QoE and QoS. The author uses a POX controller and MiniNet emulator with VLC media.

Jan (13) proposed new method for a video transmission quality monitoring. It consists of a client to server construction, in which the client is record the video and passes the one's information to the server. The server updates Net-Flow information with those statistics. The project consists of video encoding, packet encapsulation and internet protocols associated with this topic. The structure is written in a c language.

Corrado et al. (14) proposed a smart video-surveillance applications to exploit the workplaces displayed by complete SDN_NFV networks. The author of this paper uses IP cameras that connected to the Video Surveillance System by using Mininet and Opendaylight (ODL) controller. The default routing algorithm (shortest path algorithm) was used in ODL controller that depends on the number of hops.

Chih-Heng et al. (8) proposed An energetic routing technique, called GA-SDN, is advanced based on software-defined-network (SDN) approach. the framework integrates the H.264 based on SVEF with the Mininet emulator. The author of this paper used a POX controller with Mininet emulator. The genetic algorithm had been used to select the route from sender to receiver.

All of the previous works mentioned above , other than (8), haven’t considered video transmission / or they haven’t considered H.264/SVC for video compression. This makes the QoS parameters used by (8) more appropriate to follow and compare with, because it clarifies the video performance parameters such as PSNR.

In this paper, we use POX and Pyretic controllers that run two routing algorithms Bellman-Ford and Dijkstra. First, implement the Bellman Ford algorithm with POX controller. Second, implement the Dijkstra algorithm with pyretic controller. Third, apply the Dijkstra algorithm with two controllers. Finaly, modify the Dijkstra algorithm The SVEF encoding should be used before sending the video to the monitoring system.

System Model

The proposed system is emulated by using Mininet emulator, which is a software emulator for prototyping and running the network topology. In particular, two SDN controllers are used; the POX and Pyretic controllers that can work with OpenFlow switches. The compression method and encoding method are applied to video before it has being transmitted. Fig.1 represents the block diagram of the proposed video-surveillance system.

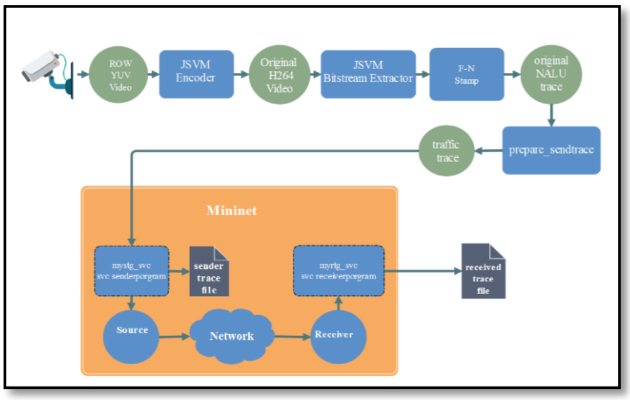


Figure 1. Block diagram of the proposed video surveillance system

SDN Configuration

The SDN controller describes the set of flows that happen in the SDN data-plane. Each flow in flow table must first get permission from the -SDNcontroller, that confirms the communication is permissible by the network rules(5). The SDN consist three main modules the topology discovery module, statistics gathering module and route computation module(15). The SDN-controller asks OF-switches for get information around configuration (topology discovery module). The information consist of operational ports and their MAC-addresses using Ofpt-Features-Request-message. This message contain (Oftp-Packet-Out and Oftp-Packet-In). The controller (SDN) sends link layer discovery protocol (LLDP-packets) for all ports in the OF-switch using Oftp-Packet-Out. This message send with the (LLDP-packet), which holds information to direct the packet to the connected port. The switches sends LLDP-packet with Oftp_packet_in message to -SDNcontroller. This packet contains the switch-ID and entering port-ID (16). The controller has complete information about the topology consequently the controller uses the routing algorithm to discovery the shortest-path for one switch to other switches. After that, the controller builds the flow tables for all switches and send it with OpenFlow protocol.

The OpenFlow switches contain three layers; the open flow protocol API, abstraction layer and the software layer. The OpenFlow is responsible for the communication between Op-switches and the SDN-controller. The abstraction layer contains the flow-table one or multiple tables. The last layer packet-processing function is the packet that treating in virtual switch(5).

The flow-tables are the essential data constructions in an Op-switches. These flow-tables allow the Op-switches to calculate received packets and apply the suitable decision (17). The Flow tables contain of a number of listed flow entries. Each entry consist three components rule, actions, and status. The rule component consists of many fields that use to compare with incoming packet (source IP, MAC and destination IP, MAC, etc.). These fields include the link-layer devices, network-layer devices and transport-layer. The action contain many decision:

1. Forwarding the received packet to a specific port.
2. Forwarding received packet to the controller.
3. Dropping the received packet.
4. Flooding the received packet for all available ports.
5. Send to normal processing pipeline.

The Network Topology and Video File

The network of proposed video surveillance system will be created. The switches should connect the hosts (prefer camera) to each other with active SDN controller. The switches that should be use called Open-v-Switch (OVS). The OVS is a manufacture quality that designed to enable huge network automation by way of programmatic extension, whilst still supporting standard interfaces and protocols. The proposed system will follow the same steps and the same method that used by (8) to evaluate the performance metrics. Figure-2 shown the network topology for video surveillance system. The author uses Host6 to sent video file to Host8 and uses Host7 to sent background traffic to Host9 to make the network congestions The video file sent frame size is (352 X 288) and encoded at

30-fps using an h.264/svc codec with 6 clips each clip 10 seconds (total 60-second video length 1800 frame and 5364 packets). The network setting information
can borrow from the paper to get the same result, so the Table-1 below explains emulation parameters such as the bandwidth, delay and the details of video format .

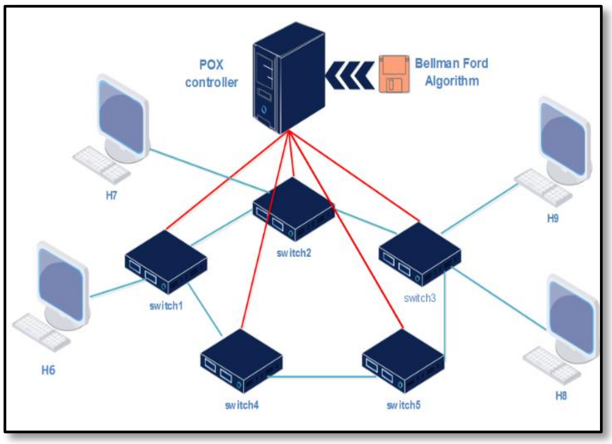


Figure 2. Network topology of video surveillance system

Table 1. Experimental parameters (8)

Parameter	value
The bandwidth of each link	10 Mbps
The propagation delay of each	1 ms
Output queue length of each link	20 packets
bandwidth of UDP background traffic in first experiment	9000 kbps
bandwidth of UDP-based background traffic second experiment	9500 kbps
Number of the frame of foreman video file	1800 frames
The average rate of foreman video file	574 kbps
The peak rate of foreman video file	942 kbps
Sending rate of foreman video file	30 frames/s
Video length of foreman video file	60 s
Software of controller	POX
Software of switch	OpenvSwitch

Validation of The Video Surveillance Over The SDN With Bellman-Ford Algorithm

To validate the results of the Bellman-Ford algorithm for comparison with (8), the same specifications (the type of controller and the topology) are used. ‹3.Fig From H6 divides a large video frame into many fragments, so the total packets are 5364 video packet. The controller executes the Bellman-Ford algorithm to find the shortest-path for each transmitted. The Bellman-Ford algorithm finds the shortest-path with careless of the link utilization status for both the background-traffic and the video-flow. Therefore, the path for video-flow is H6, switch-1, switch-2, switch-3, H8 and for background traffic is H7, switch-2 , switch-3 and H9. Consequently, the path between switch 2 and switch 3should become more congested.

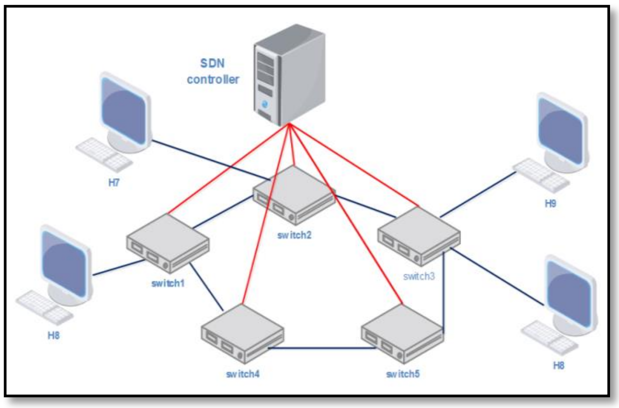
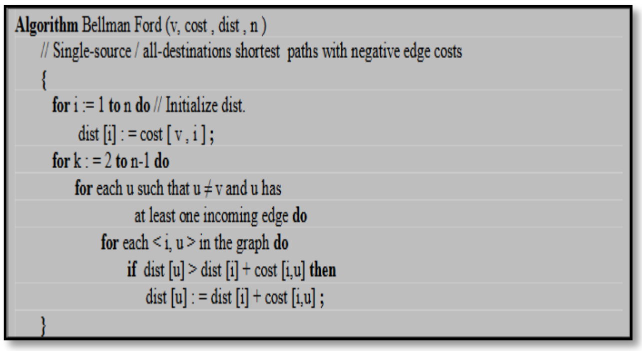


Figure 3. Bellman Ford over SDN controller

This congestion may cause to slow the network Because of this congestion, the end-to-end delay and Packet Loss Ratio (PLR) should be increased and decreases the peak signal to noise ratio (PSNR). All these factors may affect the efficiency and the effectiveness of a video surveillance system. A possible solution is to find another algorithm that enhances sending and receiving videos for video surveillance system. Fig.4 represent the Pseudocode for this algorithm.



```
Algorithm Bellman Ford (v, cost , dist , n )
    // Single-source / all-destinations shortest  paths with negative edge costs
    {
     for i := 1 to n do // Initialize dist.
          dist [i] : = cost [ v , i ] ;
     for k : = 2 to n-1 do
          for each u such that u ≠ v and u has
                      at least one incoming edge do
               for each < i, u > in the graph do
                    if  dist [u] > dist [i] + cost [i,u] then
                        dist [u] : = dist [i] + cost [i,u] ;
    }
```

Figure 4. Pseudocode of Bellman algorithm(18)

The Dijkstra Algorithm Over One SDN Controller (Pyretic Controller)

The first scenario that uses the Dijkstra algorithm instead of the Bellman-Ford algorithm with one controller in same network and hosts and link settings (Fig.5). The important step in the algorithm based on data structure storage is to utilize an appropriate data structure to store the network information(19). This factor can lead to change the path for transmission, so if the path is congested it can switch to another path for fast transmission. When the links are under huge-load, their associated-weights will be increased, so the links should have a lower probability to be selected for data transmission. The logs of the SDN-controller, which runs the Dijkstra algorithm and display the original path of the video-flows is H6, switch-1, switch-2, switch-3, H8 before the insertion of background traffic. After that when, the background-traffic should be inserted into the network the link between switch-2, switch-3 become congestion. Because this congestion the Dijkstra algorithm will change the path transmission to H6, switch-2, switch-4, switch-5, switch-3, H8 as new path transmission to solve the congestion.

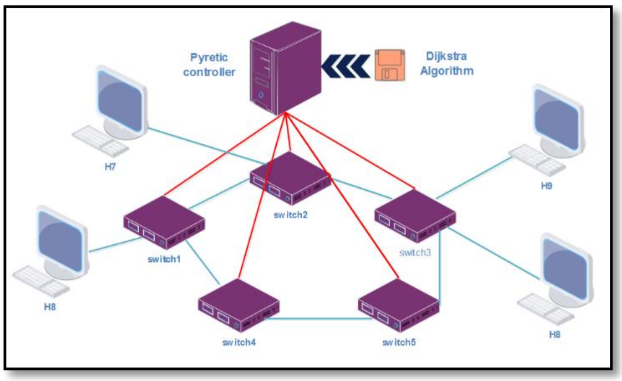
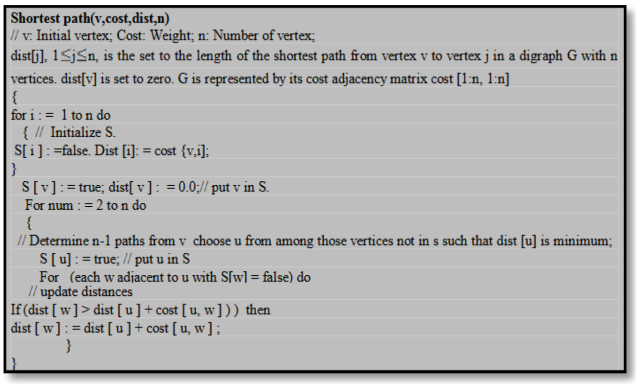


Figure 5. Dijkstra algorithm with one controller

Although fewer video packets are still lost during the path change processes، the acquired results are still recorded better than the Bellman-Ford algorithm. Fig.6 shown the pseudocode of Dijkstra algorithm



```
Shortest path(v,cost,dist,n)
// v: Initial vertex; Cost: Weight; n: Number of vertex;
dist[j], 1≤j≤n, is the set to the length of the shortest path from vertex v to vertex j in a digraph G with n
vertices. dist[v] is set to zero. G is represented by its cost adjacency matrix cost [1:n, 1:n]
{
for i : =  1 to n do
  {  //  Initialize S.
 S[ i ] : =false. Dist [i]: = cost {v,i];
}
  S [ v ] : = true; dist[ v ] :  = 0.0;// put v in S.
   For num : = 2 to n do
   {
 // Determine n-1 paths from v  choose u from among those vertices not in s such that dist [u] is minimum;
      S [ u] : = true; // put u in S
      For   (each w adjacent to u with S[w] = false) do
    // update distances
If (dist [ w ] > dist [ u ] + cost [ u, w ] ) )  then
dist [ w ] : = dist [ u ] + cost [ u, w ] ;
             }
}
```

Figure 6. Pseudocode of Dijkstra algorithm(18).

The Dijkstra Algorithm With Two SDN Controllers (Pox And Pyretic)

This section, discuss the Dijkstra algorithm with two controllers (POX, Pyretic). One reason why two controllers have been used. The main cause is to divide the jobs between POX controller and Pyretic controller. The Pyretic controller responsible for routing (Dijkstra algorithm) and POX controller responsible for the monitoring jobs.

The network performance improved when using two controllers to speed up the path selection process. Fig.7 show the topology when using two SDN controllers.

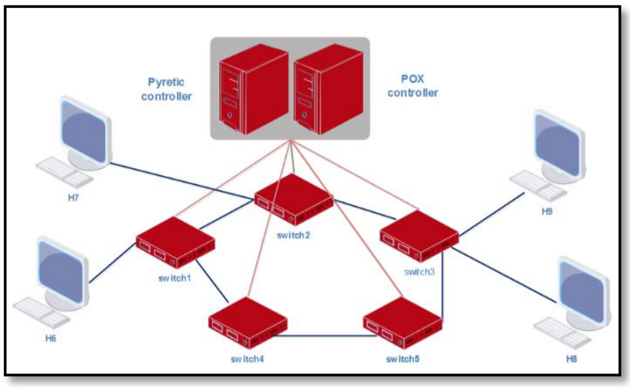


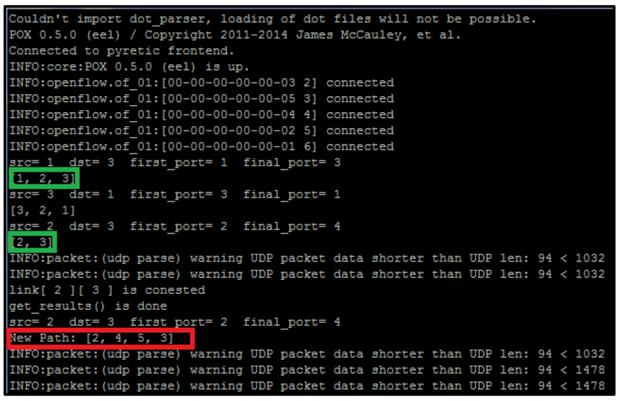
Figure 7. Dijkstra algorithm with two controller

To achieve some level of performance and scalability it will use a multi-controller architecture that contains the set of controllers working together. The multi-controller can be designed in two architectures a flat or a hierarchical design. In a flat or horizontal architecture, the SDN-controllers are located horizontally on one-level. In addition, the control plane consists of one layer, and each controller has the same responsibilities at the same time and has a partial view of its network. In a hierarchical or vertical architecture, the SDN-controllers are located vertically(20) .The proposed system used flat or horizontal architecture because the flat architecture has several advantages such as reduced control latency and improved resiliency (21). First, when H6 starts to a transmit video file to H8, the pyretic controller use the Dijkstra algorithm to find the shortest-path from source to destination. The algorithm finds S1-S2-S3 is the shortest path. The when starts H7 to transmit background traffic to H9 the algorithm chooses S2-S3 as the shortest path from H7 to H9. The link between S2 and S3 has become congested. At the same time, the POX controller is checking the link status by checking the link bandwidth. If the POX controller found the bandwidth is less than 1Mbps consequently POX send command for the pyretic controller to find a new path.

The Modified Dijkstra Algorithm with Two SDN Controllers

In this section, a new approach to modify the Dijkstra algorithm is discussed. This approach is implemented using same topology proposed. The proposed system will follow the same steps and the same method to evaluate the performance metrics.
The pyretic controller uses the Dijkstra algorithm to find the shortest-path from source to destination.

The algorithm finds S1-S2-S3 is the shortest path. Then when H7 starts to transmit background traffic to H9 the algorithm chooses S2-S3 as the shortest path from H7 to H9. The link between S2 and S3 becomes congested. At the same time, the POX controller detects that the available bandwidth of the link in the video path is less than 1Mbps. Then the link weight is increased consequently, POX controller re-helps the video by removing the link that causes congestion and use the Dijkstra algorithm to help H7-H9 find a new path. If finds a new path, it move the traffic flow of H7-H9 to the new path. In addition to, the original video is still the original Transmission path (from H6- H8) so the congestion problem is solved. Fig.8 shows execution for this modification. The green rectangle represents the original path for H6-H8 (S1- S2- S3) and background traffic H7- H9 (S2- S3). The red rectangle represents a new path for background traffic is the path (S2- S4- S5- S3).



```
Couldn't import dot_parser, loading of dot files will not be possible.
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
Connected to pyretic frontend.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-03 2] connected
INFO:openflow.of_01:[00-00-00-00-00-05 3] connected
INFO:openflow.of_01:[00-00-00-00-00-04 4] connected
INFO:openflow.of_01:[00-00-00-00-00-02 5] connected
INFO:openflow.of_01:[00-00-00-00-00-01 6] connected
src= 1  dst= 3  first_port= 1  final_port= 3
[1, 2, 3]
src= 3  dst= 1  first_port= 3  final_port= 1
[3, 2, 1]
src= 2  dst= 3  first_port= 2  final_port= 4
[2, 3]
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1032
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1032
link[ 2 ][ 3 ] is conested
get_results() is done
src= 2  dst= 3  first_port= 2  final_port= 4
New Path: [2, 4, 5, 3]
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1032
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
```

Figure 8. Execution of modifying the Dijkstra algorithm

Performance Metrics

In this section, the results for all previous scenarios are discussed. The performance metrics that will be used for comparison between these scenarios are:

1. End-to-End delay.
2. Packet Loss Rate.
3. Peak Signal to Noise Ratio.

- **End-to-End Delay**

The End-to-End packets delay can be calculated by: ***Delay [Packet Number] = Receiving Time – Sending Time***

The Receiving Time can be found in the file received by destination host. For example, when sending from H6 to H8 the received file found in H8 contain receiving time column. In addition, the Sending Time can be found in the sent file in H6. The proposed system uses file written in C language for subtracting the sending time from receiving time.

- **Packet Loss Rate**

The packet Loss Rate is the second metrics, which is used to compare the results with (8). It is calculated by***: PLR = ((Total Packets-Received Packet)/Total packets)*100%***

The total packet from comparison paper is 5364 packets. The packets number column found in receiving a file in the destination. Therefore, it can calculate the number of packets that arrive from the network and subtract it from total packets to get the missing packet that was the loss in the network. Then divide it by the total packets.

- **Peak Signal to Noise Ratio**

The Peak Signal to Noise Ratio is the third metric that will be used for performance evaluation and comparison. The definition of PSNR "is the ratio between the maximum possible signal in the video frame and the noise, which corrupts the signal accuracy" (13). PSNR is calculated as follows:

$$PSNR = 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE)$$

Use prepare-received trace1 to convert the received-file to the format necessary for SVEF-files. The result from prepare-received trace is frame level-received trace. Then the frame-level received-trace and the original NALU-trace and traffic trace are processed by prepare-received trace2 to received NALU-trace (Fig.9).

The received NALU trace file was fed into nalufilter filter file that will remove the late-frames and the frames, which cannot be decoded depend on frame dependencies. The JSVM version (9.19.8) cannot decode video packets-affected by out of order, corrupted, or missing NALUs (22). Therefore, SVEF uses filtered packet trace-file to extract the corresponding packets in the original H.264 video-file by means of Bit Stream Extractor Static. The result from Bit Stream Extractor Static is used by H264decoder to create file has YUV extension.

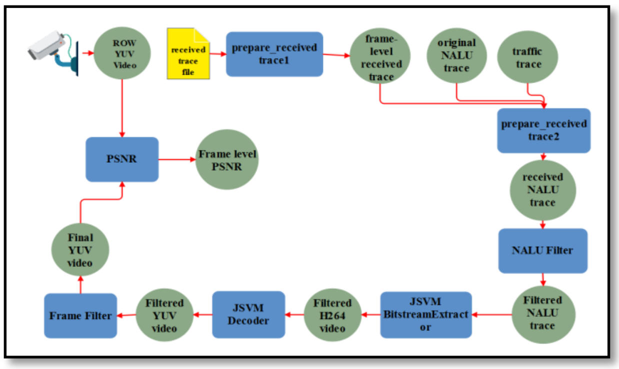


Figure 9. Steps for calculate the PSNR

The PSNR calculation of original-YUV and receiving-YUV file need the same number of video frames. Therefore, this method will hide the missing frames by copying the previous frame. The copied frame was done by a file written in C language called frame filter .Finally, the original YUV and receiving YUV file (output from frame filter) are used to calculate PSNR.

Results

In this section, the results of four scenarios are discussed. First, applied Bellman Ford in SDN. Second, applied the Dijkstra algorithm with one SDN controller. Third, run the Dijkstra algorithm with two SDN controllers. Finally, modify the Dijkstra algorithm with two controllers.

- First performance metrics is end to end delay:

The delay that shown in Fig.10, part A represent the delay with Bellman Ford algorithm. When the video start to transmit the delay is reach to 0.06 sec. after that when the background-traffic is starts transmission the network became congestion. Therefor the delay is rising up to 0.1 sec and continues in this value to the end transmission.

The delay in the part B reveals that the delay of Dijkstra algorithm is less than the delay obtained by Bellman-Ford algorithm. This enhancement in the delay is achieved because of using Dijkstra algorithm that choose another path for transmission when the path congestion occurred. The delay is starting with 0.04 sec when the video file is transmitting over the network. In packet 1200 the background traffic starts transmission. The congestion has caused the rising in the end to end delay up to 0.08 sec in many packets. After that the end to end delay decreases down to 0.03 sec in packet 4000 due to the reroute capability of the Dijkstra algorithm .

The part C show the delay of the Dijkstra algorithm with two controllers. Theend to end delay was improving when using two controllers because the flat architecture for multiple controllers reduces the controller latency. The delay is almost steady at 0.025 sec when sending only video files. Then at packet 2500, the background traffic starts transmission. Therefore, the link [2-3] becomes congested. The end to end delay is rising to 0.06 sec. After that, the delay reduces when the controller reroutes the path for video file to another path. The highest point in this figure is 0.06 sec is less than the Dijkstra algorithm with one controller. As a result, two controllers.

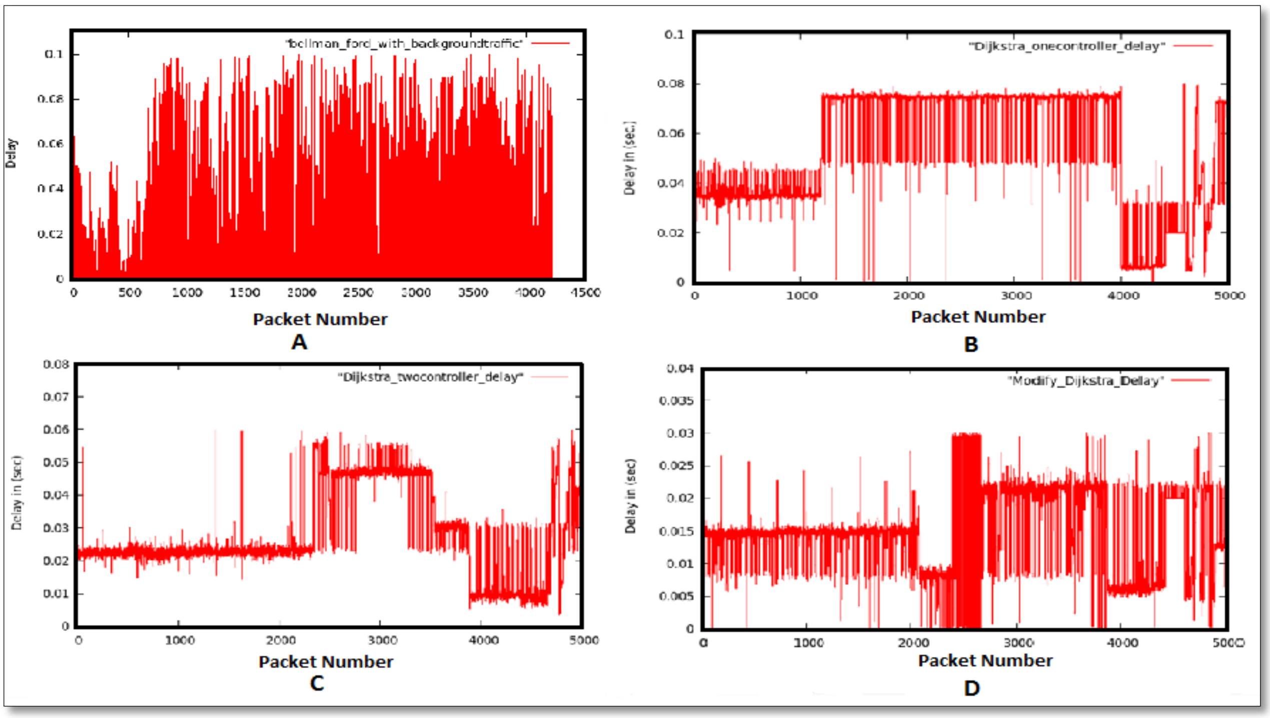


Figure 10. Mininet Emulation results of four scenarios over the SDN

The part D represent the delay of modify Dijkstra algorithm that begins with 0.015 sec. When the network is congested the delay increase to 0.03 sec. then the controller removes the path that causes congestion from the routing table for background traffic and adds a new path from switch 2 to switch 4 to solve the congestion problem and reduce the delay to 0.02 sec. The highest value in the figure is 0.03 sec represents the least peak value for all previous method. lain the delay comparison 2-Table .sfor all previous scenario

Table 2. Delay comparison

Methods	Starting time	Congestion time	After reroute	High point
Bellman-Ford	0.07	0.1	Do not has to reroute	0.1
Dijkstra-with one controller	0.04	0.075	0.03	0.075
Dijkstra-with two controller	0.02	0.055	0.03	0.06
Modify Dijkstra with two controller	0.015	0.03	0.022	0.031

- The second performance metrics is PLR:

The PLR comparison is discussd in the Table-3 for all scenarios. The Bellman-Ford algorithm has made a high loss rate of 21%, which is bad approach to select the path for video surveillance system. At first, the PLR obtained by [8] is validated. According to Eq. (3), the PLR is calculated by subtracting the number of packets arrived to the destination through the network and subtract it from total sent packets to get the missing packets that was the loss in the network. This approach is applied for all scenarios such that PLR values is shown in Table -3. The Modify Dijkstra algorithm with two controllers is good approach to select the path for video surveillance system. It has loss rate 3% with loss 168 packets.

Table 3. Packet loss rate

Methods	Sending packets	Received Packets	Loss packets	Loss rate
Bellman-Ford	5364	4227	1137	21%
Dijkstra-with one controller	5364	4666	698	13%
Dijkstra-with two controller	5364	4999	365	6%
Modify Dijkstra with two controller	5364	5196	168	3%

- The third performance metrics is PSNR

The PSNR of Bellman-Ford is equal to 35 dB in the starting of transmission and reduces to 10 dB after the frame-154 when the congestion occur, as shown in (figure-11 A). The PSNR performance when Dijkstra algorithm is used with the pyretic controller is show in (figure-11 B) it is obvious that the PSNR is improved compared to Bellman-Ford algorithm. The figure shows that the frames of PSNR is equal to 35 dB in the starting and reduces to 10 dB after happening the congestion. The Dijkstra algorithm has improved the PSNR to15 dB and has average PSNR equal to 15 dB. The PSNR for the Dijkstra algorithm with two controllers is show in (figure-11 C).

It can be observed that the PSNR value is 35 dB when video file starts to transmit over the network. After that, the background traffic is starts to transmit over the network and causes the network congestion. Therefore, PSNR is reduced to 15 dB. After that, the controller reroutes the path for background traffic to another path and improve the PSNR to 25 dB .The PSNR for modifying the Dijkstra algorithm is show in (figure-11 D).

The PSNR start with 35 dB and reduce to 15 dB when the network congestion. Then PSNR reaches to 35 dB again as result the controller reroutes the background traffic to a new path to solve the congestion. Therefore, this method is more suitable for video surveillance. The PSNR comparison is explain in Table-4.

Table 4. PSNR comparison

Methods	Starting PSNR	Congesti on region	After reroute	Average PSNR
Bellman-Ford	35 dB	10 dB	Do not has reroute	10 dB
Dijkstra-with one controller	35 dB	10 dB	20 dB	15 dB
Dijkstra-with two controller	0.02	15 dB	30 dB	27 dB
Modify Dijkstra with two controller	0.015	15 dB	35 dB	33 dB

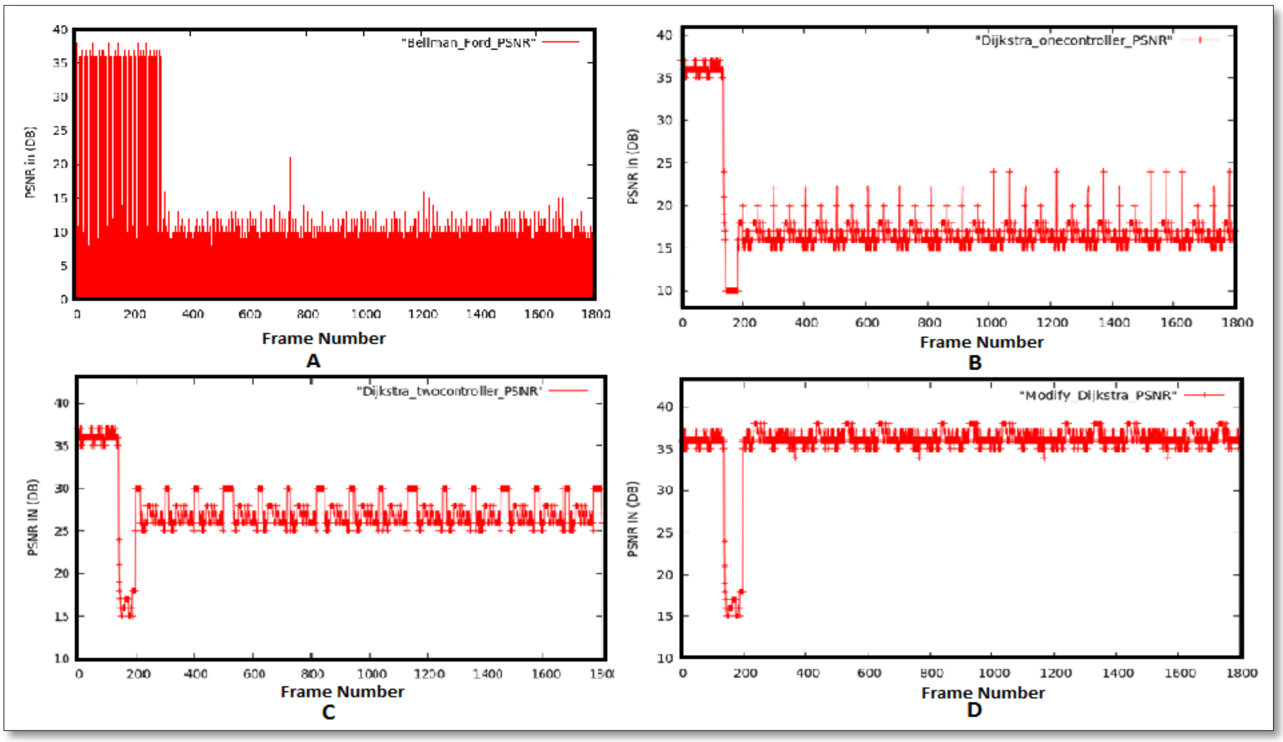


Figure 11. PSNR of four scenario

Conclusion:

The proposed system improves the transmission for video in many steps. The following conclusions can be drawn from the current study:

With traditional networking, networking functionality is generally carried out via hardware devices consisting of a router, switches, firewalls. each of which ought to be manually-configured through an IT-administrator who is chargeable for making sure every tool is up to date with the trendy configuration settings. consequently, the software defined networking is more quickly to find the solution for those issues. in addition, the SDN has no problem in overcoming the limitations of traditional networking. The SDN separating the hardware from the software i.e. separating the control plane from the forwarding plane.

The main disadvantage of the Bellman-Ford algorithm that it does not consider weightings and slower update for the paths. Therefore, the Dijkstra algorithm is used to enhance the video transmission. The reason for using the Dijkstra algorithms is the link status that considers by the algorithm. The controller detects the congestion then the link weight is increased. Consequently, the controller re-helps the video by changing the path for video while Bellman-Ford still the video transmission in the same path.

The proposed system uses one controller with the Dijkstra algorithm. This scenario for applicate the Dijkstra with SDN controller. From the result, the Dijkstra algorithm enhances the video transmission than Bellman-Ford. After the congestion occurs the Dijkstra algorithm attempt to found a new path to solve this problem. Therefore, the delay, PRL, and PSNR improved.

The video surveillance system uses two controllers with Dijkstra algorithm to improve the video performance by reducing the latency and make the network management more flexible. The controllers are designed in flat architecture to achieve the scalability. The reason for this setup is that the two controllers are cooperating with each other (the POX controller responsible for monitoring the network and the pyretic responsible for selecting the path according to the used algorithm).

The main contribution of this study is to enhance the performance by modifying the Dijkstra algorithm. It solves the congestion problem from the flow tables that locate inside the devices (switches, routers). The modification is done by removing one of the paths that sent on the same delivery path then established a path new and adds it in the flow tables.

Conflicts of Interest: None.

References:

1. Licandro F, Schembra G. Wireless mesh networks to support video surveillance: architecture, protocol, and implementation issues. EURASIP. 2007;2007(1):031976.

2. Mohammadi R, Javidan R. An adaptive type-2 fuzzy traffic engineering method for video surveillance systems over software defined networks. Multimed Tools Appl. 2017;76(22):23627-42.
3. Open Networking Foundation. Available from: https://www.opennetworking.org/, last online : 2019/3/4.
4. Cui L, Yu FR, Yan Q. When big data meets software-defined networking: SDN for big data and big data for SDN. IEEE network. 2016;30(1):58-65.
5. Azodolmolky S. Software defined networking with OpenFlow: Packt Pub, Birmingham, UK. 2013.
6. Sumanth B. Designing an Openflow Controller for data delivery with end-to-end QoS over Software Defined Networks: Computer Science and Engineering; Conference in Hollywood, CA, USA 2016.
7. Gopi D, Cheng S, Huck R, editors. Comparative analysis of SDN and conventional networks using routing protocols. Computer, Information and Telecommunication Systems (CITS), 2017 International Conference on; 2017: IEEE.
8. Yu YS, Ke CH. Genetic algorithm-based routing method for enhanced video delivery over software defined networks. International Journal of Communication Systems. 2018;31(1):e3391.
9. Magzhan K, Jani HM. A review and evaluations of shortest path algorithms. International journal of scientific & technology research. 2013;2(6):99-104.
10. Panwaree P, Kim J, Aswakul C, editors. Packet delay and loss performance of streaming video over emulated and real OpenFlow networks. Proceedings of 29th International Technical Conference on Circuit/Systems Computers and Communications (ITC-CSCC); 2014.
11. Owens II H, Durresi A. Video over software-defined networking (vsdn). Computer Networks. 2015;92:341-56.
12. Rymen M. Software-Defined Networking for Multi-Camera Systems 2015. Thesis of master degree, National Chiao Tung University, Taiwan.
13. HAVLÍK J. video quality monitoring using netflow [BACHELOR'S THESIS]. Brno University Of Technology, 2017, Bachelor's thesis, Brno University of Technology, Faculty of Information Technology, Prague.
14. Rametta C, Baldoni G, Lombardo A, Micalizzi S, Vassallo A. S6: a Smart, Social and SDN-based Surveillance System for Smart-cities. Procedia Computer Science. 2017;110:361-8.
15. Hosseini Seno SA. Dynamic Routing Method over Hybrid SDN for Flying Ad Hoc Networks. Baghdad Science Journal, vol. 15, no. 3, pp 361-368
16. Pakzad F, Portmann M, Tan WL, Indulska J, editors. Efficient topology discovery in software defined networks. Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on; 2014: IEEE.
17. Goransson P, Black C, Culver T. Software defined networks: a comprehensive approach: Morgan Kaufmann; 2016.
18. Sanan S, Jain L, Kappor B. Shortest path algorithm. IJAIEM. 2013;2(7):316-20.
19. Rochan Mehrotra SB. A Comparative Study between Bellman-Ford Algorithm and Dijkstras- Algorithms IJIRST. 2014;1(5).
20. Blial O, Ben Mamoun M, Benaini R. An overview on SDN architectures with multiple controllers. IJCNC, 2016, 1(5).
21. Bannour F, Souihi S, Mellouk A. Distributed SDN control: Survey, taxonomy, and challenges. IEEE Commun. Surv. Tutor. 2017;20(1):333-54.
22. Tantisarkhornkhet, Piyawit, and Warodom Werapun. "Qlb: Qos routing algorithm for software-defined networking." 2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS). IEEE, 2016.

شبكة معرفة برمجيا لمنظومة مراقبة فديوية اعتمادا على تحسين خوارزمات التوجيه

سكينة رضا شاكر **مصطفى اسماعيل سلمان**

كلية الهندسة، جامعة بغداد، بغداد، العراق.

الخلاصة :

تعتبر شبكة معرفة برمجيا (SDN) تقنية جديدة التي تفصل مستوى التحكم عن مستوى البيانات. توفر SDN خيارًا في التشغيل التلقائي والبرمجة أسرع من الشبكة التقليدية. وهو يدعم جودة الخدمة (QoS) لتطبيق المراقبة بالفيديو. واحدة من أهم القضايا في المراقبة بالفيديوية هي كيفية العثور على أفضل مسار لتوجيه الحزم بين المصدر (كاميراتIP) وجهة التسليم (مركز المراقبة). يتطلب نظام المراقبة بالفيديوية إرسالًا سريعًا وتسليمًا موثوقًا به وجودة خدمة عالية. لتحسين جودة الخدمة (QoS) ولتحقيق المسار الأمثل ، يتم استخدام بنية SDNفي هذه الورقة. بالإضافة إلى ذلك، يتم استخدام خوارزميات التوجيه المختلفة مع خطوات مختلفة. أولاً، نقوم بتقييم نقل الفيديو عبر SDNباستخدام خوارزمية Bellman Ford. بعد ذلك ، نظرًا لمساوئ خوارزمية Bellman ford ، يتم استخدام خوارزمية Dijkstra لتغيير المسار عند حدوث ازدحام. علاوة على ذلك، يتم استخدام خوارزمية Dijkstra مع وحدتي تحكم لتقليل الوقت المستغرق في وحدة التحكم SDN .يتم استخدام وحدات التحكم POX و Pyretic SDN بحيث تكون وحدة تحكم POX مسؤولة عن مراقبة الشبكة ، بينما تكون وحدة التحكم Pyretic مسؤولة عن خوارزمية التوجيه واختيار المسار. وأخيرًا ، يتم أيضًا اقتراح وتقييم خوارزمية Dijkstra معدلة باستخدام وحدتي تحكم لتحسين الأداء. وأظهرت النتائج أن خوارزمية Dijkstra المعدلة تفوقت على الطرق الأخرى في جانب معلمات جودة الخدمة(QoS).

الكلمات المفتاحية: خوارزمية Bellman ford و خوارزمية Dijkstra و شبكة معرفة برمجيا(SDN) .