University of Baghdad
College of Engineering
JOURNAL OF ENGINEERING

Journal of Engineering
journal homepage: www.joe.uobaghdad.edu.iq
Number 1 Volume 26 January 2020

Electrical, Electronics and communications, and Computer Engineering

Link Failure Recovery for a Large-Scale Video Surveillance System using a Software-Defined Network

Mustafa Ismael Salman
College of Engineering – University of Baghdad
E-mail: mustafa.i.s@coeng.uobaghdad.edu.iq

Sukaina R. Shaker *
College of Engineering – University of Baghdad
E-mail: sukaina_ridha@coeng.uobaghdad.edu.iq

ABSTRACT

The software-defined network (SDN) is a new technology that separates the control plane from data plane for the network devices. One of the most significant issues in the video surveillance system is the link failure. When the path failure occurs, the monitoring center cannot receive the video from the cameras. In this paper, two methods are proposed to solve this problem. The first method uses the Dijkstra algorithm to re-find the path at the source node switch. The second method uses the Dijkstra algorithm to re-find the path at the ingress node switch (or failed link). Based on simulation results, it is concluded that the second method consumes less time (lower transmission delay) than the first method. The delay consumed by the second method is half the delay in the first method. Also, the packet loss rate for second method is 14%, while 16% in the first method. The jitter for second method is almost similar to the jitter without link fail. Therefore, the second method led to select the path with small losses without impact on video quality. Finally, the results of two methods are compared in terms of end to end delay, packet loss rate, and jitter.
Keywords: Dijkstra algorithm, link fail, Software-defined network (SDN).

معالجة فشل الارتباط لنظام المراقبة الفيديوية واسع النطاق باستخدام شبكة معرفة برمجيا

مصطفى إسماعيل سلمان
مدرس
جامعة بغداد / كلية الهندسة / قسم هندسة الحاسبات

سكينة رضا شاكر
جامعة بغداد / كلية الهندسة /قسم هندسة الحاسبات

الخلاصة

تعتبرالشبكة المعرفة برمجيا (SDN) تقنية جديدة تفصل مستوى التحكم عن مستوى البيانات لأجهزة الشبكة. واحدة من أهم القضايا في نظام المراقبة بالفيديو هو انقطاع المسار .عند انقطاع المسار ، لا يمكن لمركز المراقبة من استقبال الفيديو من الكاميرات . في هذا البحث ، تم اقتراح طريقتين لحل هذه المشكلة. تستخدم الطريقة الأولى خوارزمية Dijkstra لإيجاد المسار في مفتاح التبديل الاقرب للمصدر . تستخدم الطريقة الثانية خوارزمية Dijkstra لإيجاد المسار في مفتاح التبديل القريب على

*Corresponding author
Peer review under the responsibility of University of Baghdad.
https://doi.org/10.31026/j.eng.2020.01.09

Article received: 30/1/2019
Article accepted: 20/2/2019
Article published: 1/1/2020

نقطة القطع (أو المسار الفاشل) . استناداً إلى نتائج المحاكاة ، تم استنتاج أن الطريقة الثانية تستهلك وقت اقل (تأخير الارسال اقل) أفضل من الطريقة الأولى. التأخير المستهلك في الطريقة الثانية نصف التاخير في الطريقة الاولى. بالاضافة الى معدل فقدان الحزم للطريقة الثانية هو14% بينما 16% في الطريقة الاولى. التردد للطريقة الثانية يكون مشابه للتردد بدون فشل الارتباط. لذلك ، أدت الطريقة الثانية إلى تحديد المسار بخسائر صغيرة دون التأثير على جودة الفيديو . وأخيرًا ، تم مقارنة نتائج الطريقتين في تأخير نهاية إلى آخر ، ومعدل فقدان الحزم والتردد.

الكلمات الرئيسية: خوارزمية Dijkstra ,فشل الارتباط , شبكة معرفة برمجيا(SDN).

1. INTRODUCTION

Video surveillance is critical for different aspects of life. The main objective of the surveillance system to keep people's care or minimize human dangers associated with illegal or criminal activity. Video-surveillance frameworks are very significant in our daily lives due to the number of applications they make possible. The reasons for benefit in such frameworks are differing, ranging from security requests and military applications to scientific purposes **(Licandro and Schembra, 2007)***.* Video surveillance over SDN comprises multi IP cameras, OpenFlow switches, a monitoring center, and a controller. The objective of creating such a framework is to watch and monitor an indoor or outdoor region. IP cameras capture the environment video information and then send the video to the monitoring center through the network. The policy of the controller in such system is to find the best path between IP cameras and checking center. After that, the controller should send the best path to Open Flow switches for routing video data **(Mohammadi and Javidan, 2017)**.

The Open Networking Foundation (ONF) **(Nunes et al., 2014)** 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." **(Cui et al., 2016)**. The SDN consists of three layers; the infrastructure layer, control layer, and application layer. Infrastructure layer consists of both physical and virtual network devices such as switches and routers. Control layer involves a centralized control plane. It provides centralized global view to entire network.

A major advantage of SDN is that it provides network applications and network services the ability to program the switches or any network devices. SDN has many more advantages, including the ability to automate network configuration, program the network, reduce network complexity, and increase the flexibility and security of the network **(Owens II and Durresi, 2015).**

The application layer contains network services, application, and orchestration tools that are used to interact with control layer **(Azodolmolky, 2013)**. The SDN uses the OpenFlow protocol to interface with OpenFlow switches. It allows both the controller and all the switches to understand each other **(Sumanth, 2016)**.

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 router, the router checks the routing table if it is available or not, and forwards it as per the rules set by the routing protocol used by the network operator **(Sumanth, 2016)**. The routing time of SDN networks is lesser compared to traditional Networks. An increase in N node the conventional networks are consuming more time to change the path while SDN requires less time **(Gopi et al., 2017).**

Failure management is one of the fundamental instruments that allows network operators to provide communication services that are much more reliable than individual network components (nodes and links). It allows reacting to failures of network components by reconfiguring the network devices to find a new path and make use of the surviving network infrastructure able to provide services **(Capone et al., 2015)**.

The main contribution for this paper is proposing two methods to solve link failure problem:

1. Path recovery method1: this method uses the Dijkstra algorithm to re-find the path at source node switch (switch which near to transmitter source).
2. Path recovery method2: this method uses the Dijkstra algorithm to re-find the path at ingress node switch (or failed link).

The remainder of this paper is organized as follows. Section 2 compares video over SDN and link failure with related works. Section 3 discusses the system model that contains the SDN configuration, Dijkstra algorithm, network topology, and video file — followed by the implementation of two proposed methods. Section 4 explains the performance metrics and results. Finally, section 5 describes the conclusion.

2. 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., 2014) proposed that the video sent over two kinds of OpenFlow enabled network testbeds (Mininet-emulated and Open vSwitch PC-cluster) OpenFlow networks. 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.

(Rymen, 2015) proposed the use of a Software-Defined Networking that can be used in a dynamically configurable multi-camera environment for the playground. The network controller should be able to teach the camera nodes and their location on the 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.

(Santos, 2015) proposed to use an SDN controller application that calculates the path between network hubs by utilizing different path computation algorithms. This thesis presented that the usage of a constrained multiple path algorithm improves the QoS metrics. It uses a Self-adaptive Multiple Constraints Routing Algorithm (SAMCRA) algorithm that contains Single-path algorithms like Dijkstra's Algorithm and Multi-path algorithms like Link-disjoint algorithms. This thesis use Opendaylight controller (ODL) and MinNet emulator.

(Sumanth, 2016) presented the following i) the design and execution of the SDNcontroller framework, ii) Utility Proportional Fairness algorithm for bandwidth allocation, and how the QoS is achieved, iii) the emulation of the above algorithm in a virtual openflow network with MiniNet. iv) the bandwidth sharing algorithm is assessed with regard to the common situation where there is no QoS policy. The author of this thesis uses POX controller and bandwidth sharing algorithm with MiniNet emulator.

(Havlík, 2017) proposed new method for a video transmission quality monitoring. It consists of a client to server construction, in which the client records 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.

(Rametta et al., 2017) proposed a smart video surveillance platform to exploit the workplaces displayed by full 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 the ODL controller that depends on the number of hops.

(Yu and Ke, 2018) proposed an energetic routing technique, named GA-SDN, developed 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.

The proposed system uses the Dijkstra algorithm with POX controller for a large-scale network of video surveillance system. This paper proposed two methods solution for link failure issue. The first method uses the Dijkstra algorithm to find a new path between H1and H2 and update the flow table in source node switch. The second method uses the Dijkstra algorithm to find a new path between the switches where the failure occurred.

3. 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 controller that can work with OpenFlow switches. **Fig.1** shows the system model for the proposed system.

3.1 SDN Configuration

The SDN controller defines the data flows that happen in the SDN data -plane. Each flow must first get permission from the controller, which confirms that the communication is permissible by the network policy **(Azodolmolky, 2013)**. The SDN consists three main modules; the topology discovery module, statistics gathering module and route computation module**(Hosseini Seno, 2018)**. The controller asks switches to get information about configuration (topology discovery module). The information consists of operational ports and their MAC addresses using Ofpt-Features-Request-message. This message contains (Oftp-Packet-Out and Oftp-Packet-In). The controller (SDN) sends LLDP packets for each port in the switch using Oftp-Packet-Out. This message is sent with the link layer discovery protocol (LLDP) packet, which holds instructions to direct the packet to the connected port. The switches send LLDP packet with OFTP_PACKET_IN message to the controller. This packet contains the switch ID and entering port ID **(Pakzad et al.,**

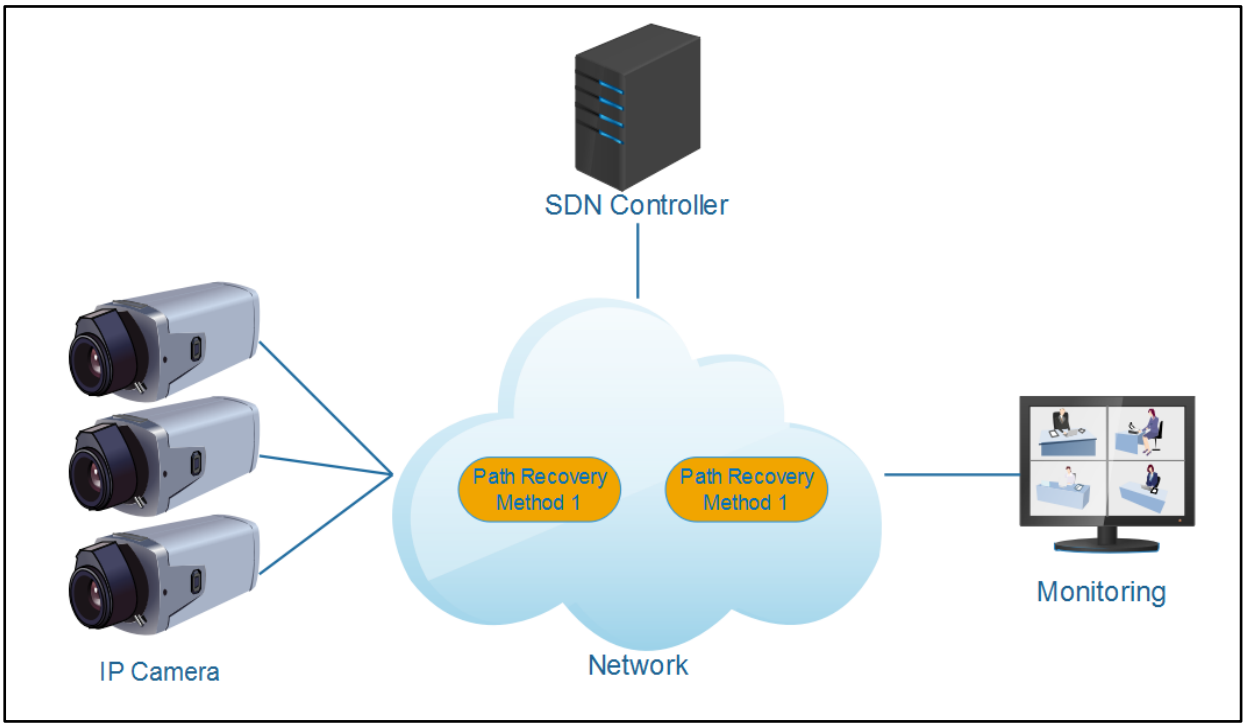

Figure 1. System model for video surveillance system.

2014). The controller has complete information about the topology consequently the controller uses the routing algorithm to find 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 open flow switches contain three layers; the open flow protocol API, abstraction layer and the software layer. The open flow is responsible for communication with the controller. The abstraction layer contains the flow table one or multiple tables. The last layer packet processing function is packet processing in virtual switch **(Azodolmolky, 2013)**.

The flow tables are the essential data structures in SDN switches. These flow tables allow the switches to evaluate received packets and apply the suitable action **(Goransson et al., 2016)**. The Flow tables contain a number of listed flow entries. Each entry consists three components rule, actions, and status. The rule component consists of many fields used to compare with incoming packet (source IP, MAC, and destination IP, MAC, etc.). These fields include link layer devices, network layer devices, and transport layer. The action contains many decisions:

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.

3.2 Dijkstra algorithm

Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1959, is a graph algorithm that works with single Source shortest path problem for a graph with nonnegative edge path costs **(Rochan Mehrotra, 2014)**. The Dijkstra‘s Algorithm gets a wide interest as it solves an important problem of graph theory, by finding the Shortest-Path (SP) for a graph that each edge having a weight or path length. **Fig. 2** shows the pseudocode of Dijkstra algorithm. The proposed system uses python language to build Dijkstra algorithm's file then fed the python file in Pox controller's files. The steps for applying the Dijkstra algorithm is explained in flowchart as shown in **Fig.3**.

3.3 Network Topology and Video File

The network of the 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 used is called Open-v-Switch (OVS). The OVS is a manufacturing quality designed to enable huge network automation by way of programmatic extension, while still supporting standard interfaces and protocols. The proposed system uses the network topology is explained in **Fig. 4**. The topology contains 37 switches and two hosts; H1 represents the IP camera and H2 represents the monitoring system. Host1 sends video file to Host2, the video file is sent (frame size is 352 × 288) and encoded at 30 fps using an H.264/SVC codec with six clips each clip 10 seconds (total 60-second video length 1800 frame and 5364 packets).

```
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 2 .Pseudocode of Dijkstra algorithm.

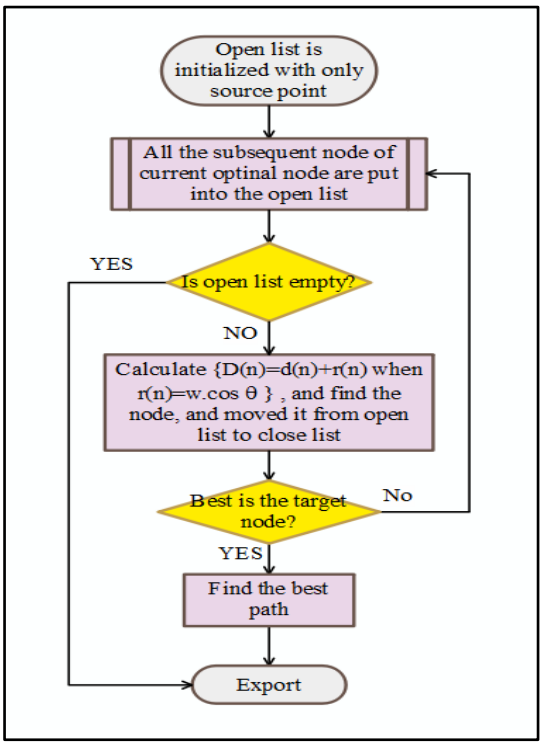

Figure 3. Flowchart for Dijkstra algorithm.

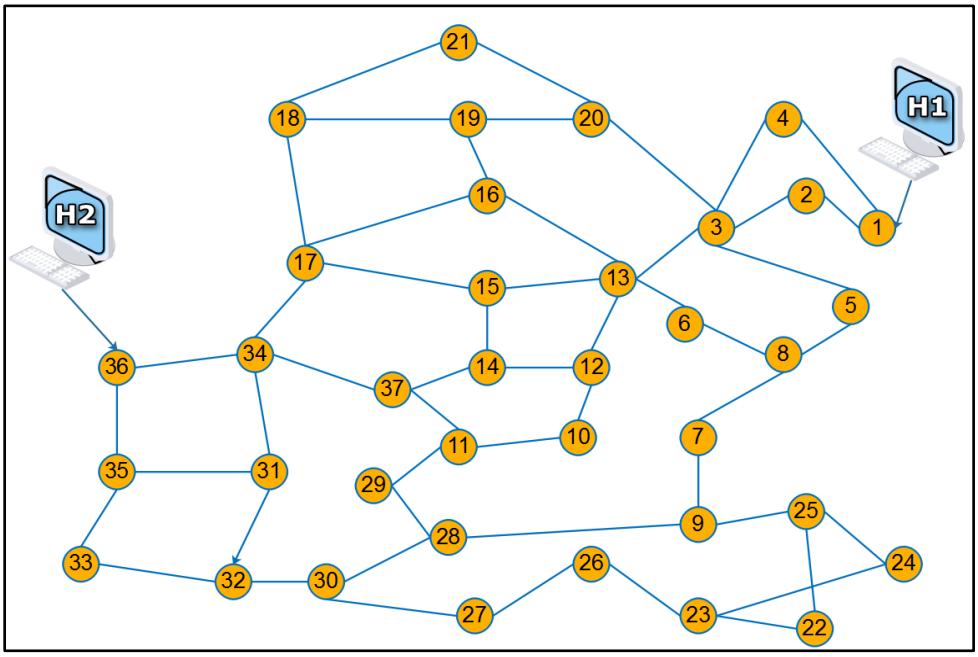

Figure 4 .Network Topology.

3.4 The Network without Link Fail

The first scenario is running the network and transmits the video without any link fail as shown in **Fig.5**. The framework of this system consists of the topology and SDN controller. The two parts are running in Mininet emulator. First, the network topology is running and check the connection between H1 and H2 by uses pingall instruction. The POX controller uses the Dijkstra algorithm to select the shortest path between H1 and H2. The selected path is H1, S1, S2, S3, S13, S15, S17, S34, S36, H2.

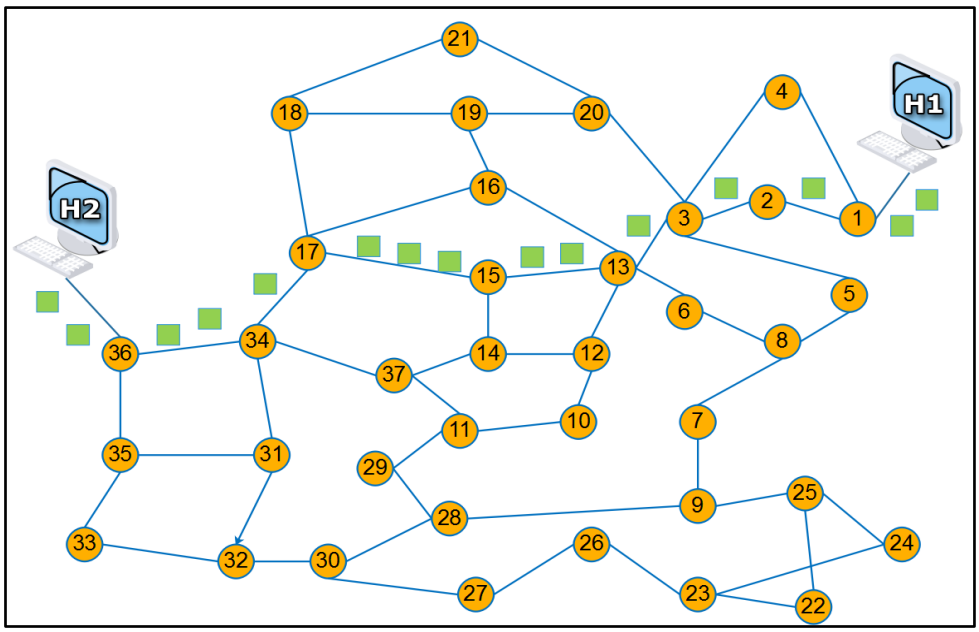

Figure 5. Network without link fail.

The video file is transmitted over this path. Any failure that happens in one link for entire path causes problem for transmitting the video to monitoring system. Therefore, this paper proposed methods to solve this problem.

3.5 Path Recovery- Method1

This scenario discusses network behavior when the path is failed. First, the network topology runs without any link fail. Then, the connection between H1 and H2 is tested using pingall instruction to check the connection between them. The path between the source and destination before the fail is H1, S1, S2, S3, S13, S15, S17, S34, S36, H2. After that, the link between S13 and S15 is failed by using [Link S13 S15 down] instruction as shown in **Fig. 6.** The red link represents the link fail. The procedures to solve link-fail problem as the following:

1. Switch 13 Sent request to the controller to inform it about the link-fail.
2. The controller uses the Dijkstra algorithm to find a new path between H1 and H2.
3. The controller updates the flow table in the switch 1.
4. The new path is H1-S1-S2-S3-S13-S16-S17-S34-S36-H2.

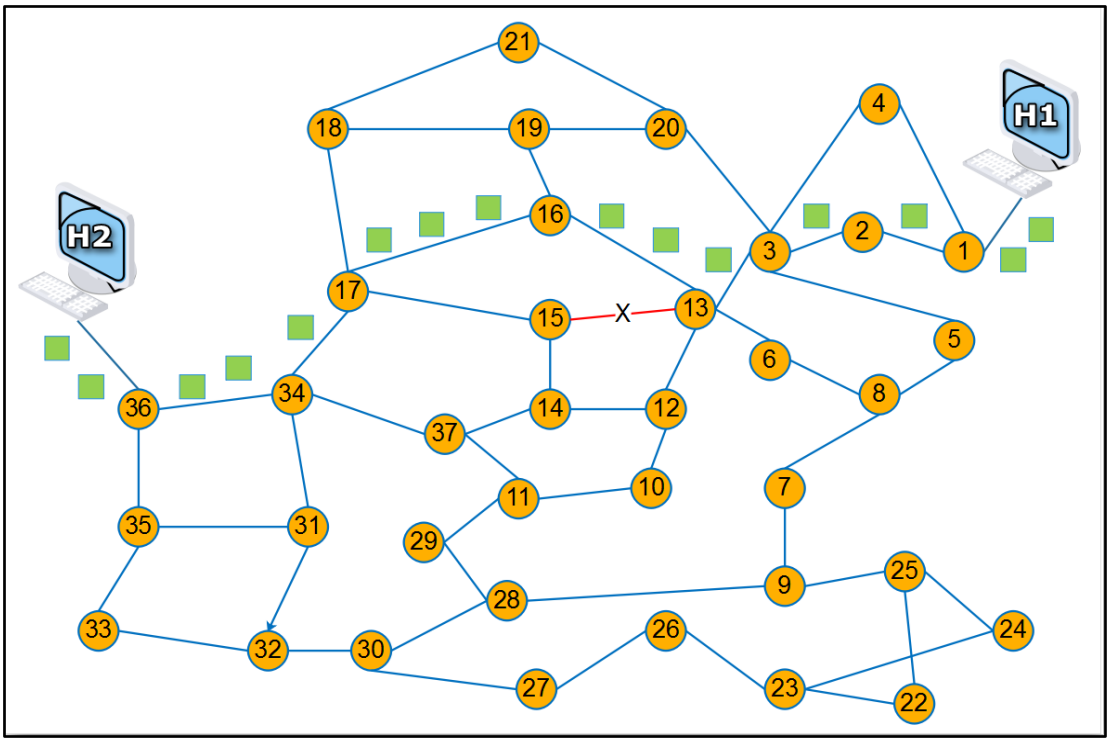

Figure 6. Path recovery method1.

Fig. 7 shows the steps to re-find a new path at source node switch. Consequently, H1 sends the video file over the new path and calculate the performance metrics (delay-method1, packet loss rate-method1, and jitter-method1).

3.6 Path Recovery-Method2

This scenario discussed the solution to same problem (link fail). First, running the network topology and check the connection between H1 and H2. Then, using the same instruction to fail the link between S13 and S15 as shown in **Fig.8.** The red link represents the link fail. Consequently, this method finds a new path [H1, S1, S2, S3, S13, S12, S14, S17, S34, S36, H2] to transmit the video. The procedures to solve link-fail problem as the following:

1. Switch 13 sends a request to the controller to inform it about the link-fail.
2. The controller uses the Dijkstra algorithm to find a new path between S13 and S15.
3. The controller updates the flow table in the switch 13.
4. The new path is H1-S1-S2-S3-S13-S12-S14-S16-S17-S34-S36-H2.

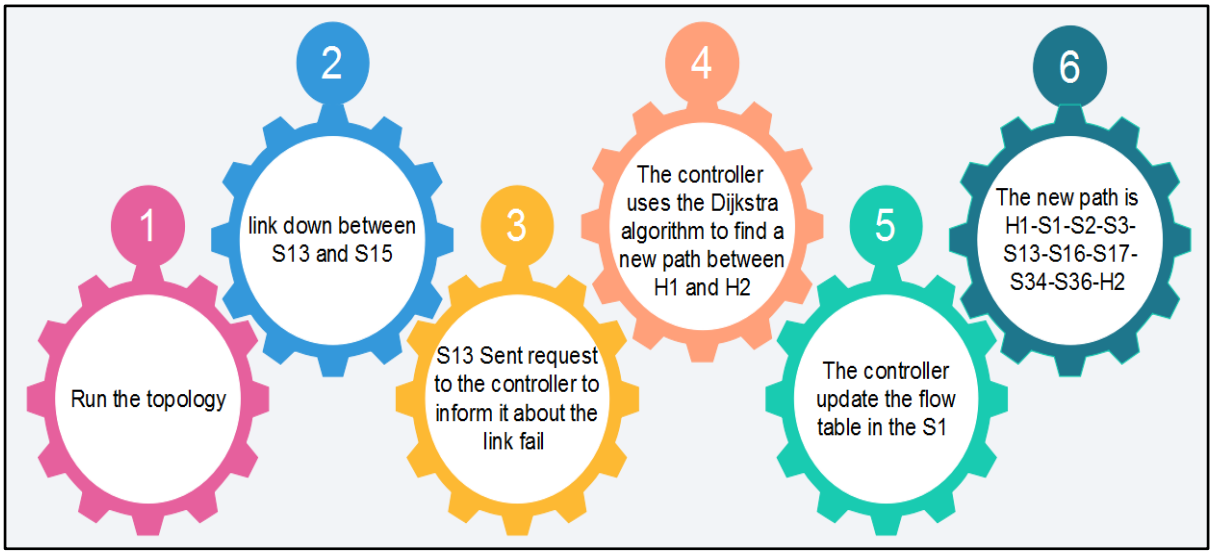

Figure 7. Procedures method 1.

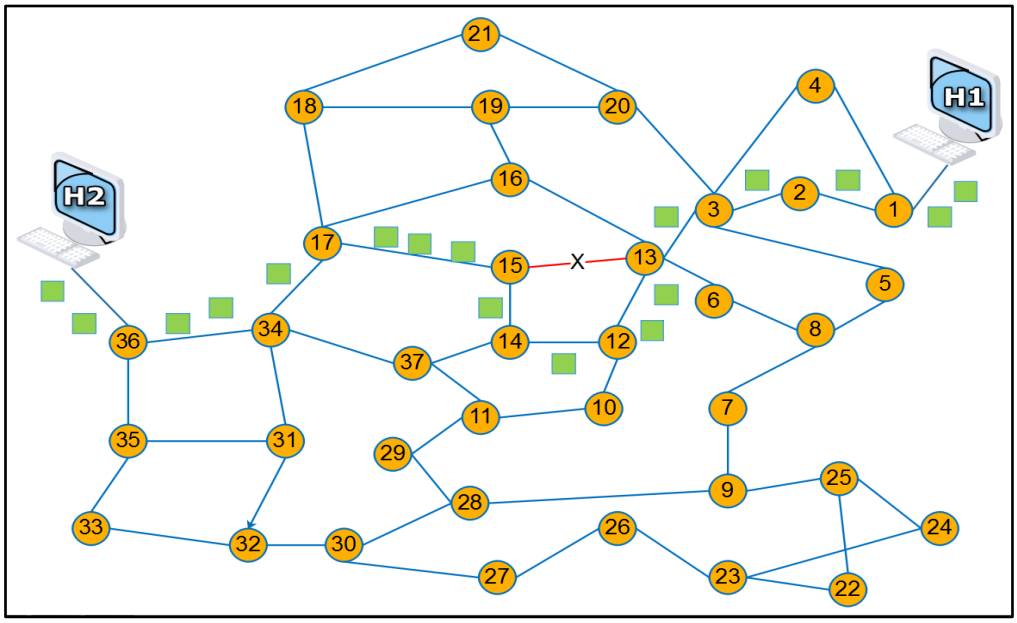

Figure 8. Path recovery method 2.

Fig.9 shows the steps for re-finding a new path at failure node switch. Consequently, H1 sends the video file over the new path and calculate the performance metrics (delay-method2, packet loss rate-method2, and jitter-method2).

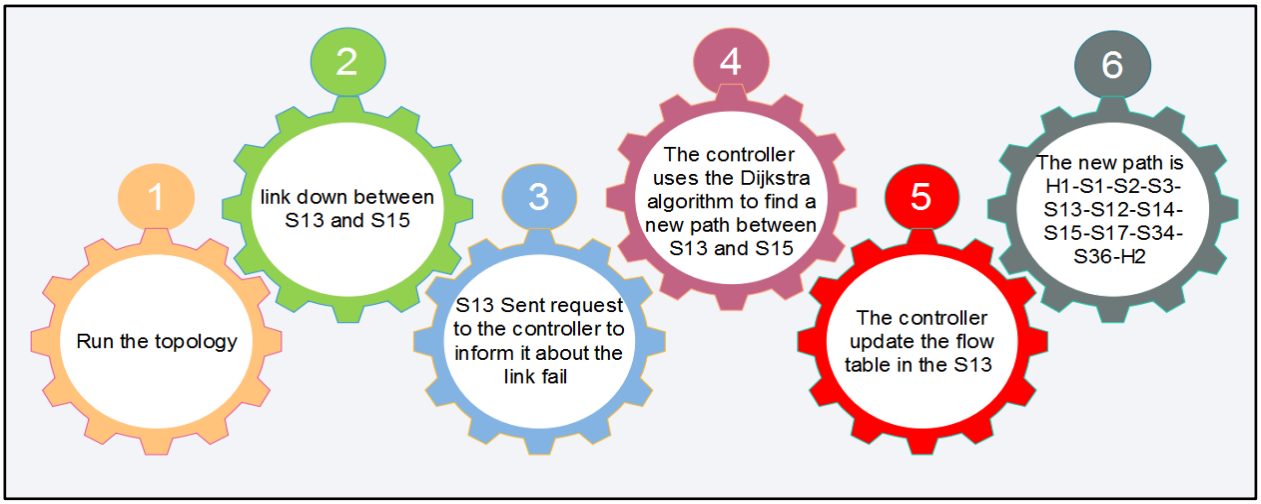

Figure 9. Procedures method 2.

4. PERFORMANCE METRICS AND RESULTS

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. Jitter.

4.1 End to End Delay

The end-to-end packets delay can be calculated by:

$$\text{Delay [Packet Number]} = \text{Receiving Time} - \text{Sending Time} \quad (1)$$

In Eq. (1), the Receiving Time can be found in the file received by the destination host. For example, when sending from H1 to H2 the received file found in H2 contains receiving time column. In addition, the Sending Time can be found in the sent file in H1. The proposed system uses file written in C language for subtracting the sending time from receiving time.

The delay shown in **Fig. 10** part A represents the delay without link fail. When the video starts to transmit the delay, it reaches to 0.15 sec and ranges around this value till the end of transmission. The high point in the delay figure is 0.28 sec. Part B represents the delay with path recovery method1. When the video starts to transmit the delay, it reaches to 0.18 sec. After that, the link between the S13 and S15 becomes fail, and the procedure is applied to solve this problem; the delay is rising to 2 sec. Consequently, the end to end delay decreases down to 0.15 sec. Part C represents the delay with path recovery method2. When the video starts to transmit the delay, it reaches to 0.21 sec. After that, the link between the S13 and S15 becomes fail and the procedure is applied to solve this problem, the delay is rising up to 1 sec. Then, the end to end delay decreases down to 0.23. **Table 1.** shows the delay comparison of three scenarios.

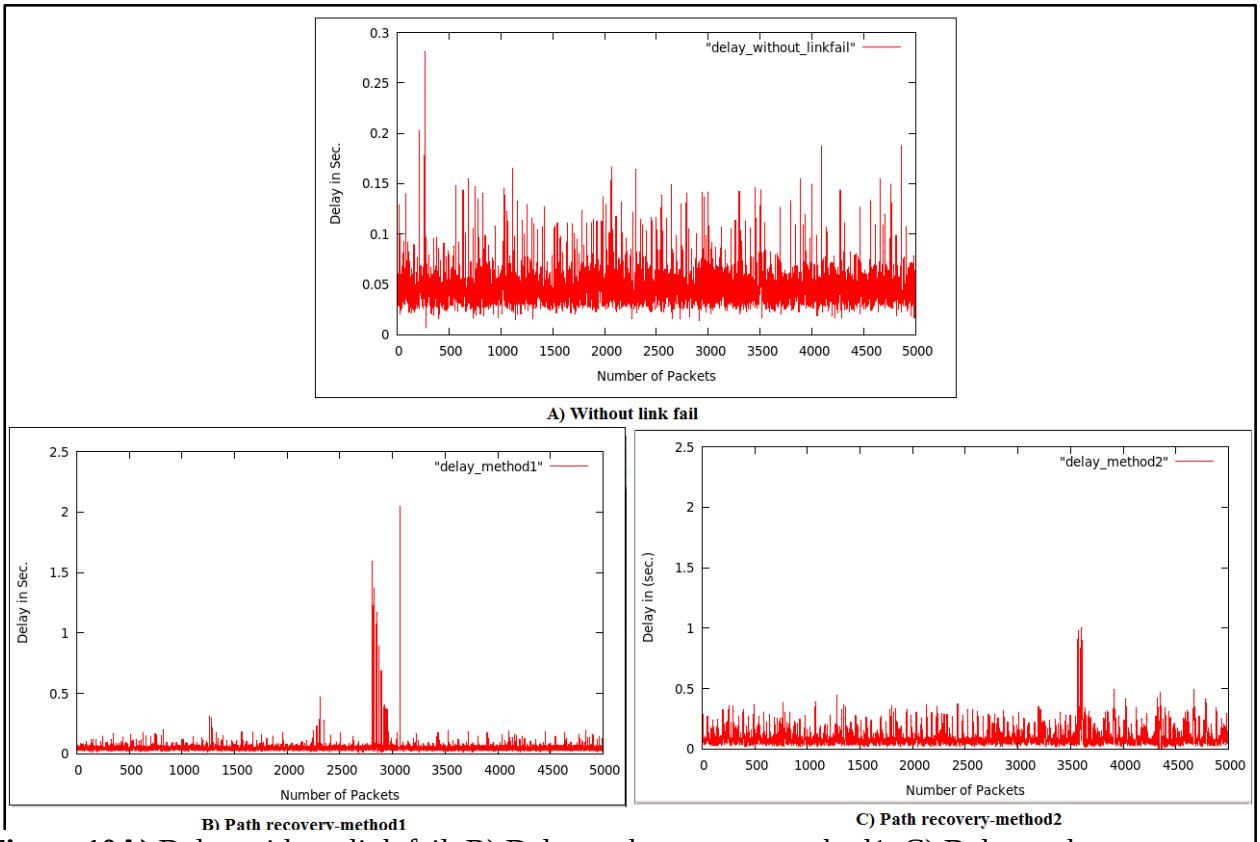

Figure 10A) Delay without link fail, B) Delay path recovery-method1, C) Delay path recovery-method2.

Table 1. Delay comparison.

Methods	Starting time	Procedure time	After reroute	High point
Network without link fail	**0.15 sec**	**Do not has Procedure**	**Do not has to reroute**	**0.28 sec**
Path recovery method1	**0.18 sec**	**2 sec**	**0.15**	**2 sec**
Path recovery method2	**0.21 sec**	**1 sec**	**0.23**	**1 sec**

4.2 Packet Loss Rate

The packet loss rate is the second metrics. It is calculated by**:**

$$\textbf{PLR} = \textbf{((Total Packets-Received Packet)/Total packets)*100\%} \quad \textbf{(2)}$$

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, hen dividing it by the total packets as shown in Eq (2).

The PLR comparison is discussed in **Table 2** for all scenarios. The network without link fail made a less loss rate of 2% with loss 107 packets the path recovery-method1 has 16% with loss 751 packets which are bad approach to solve link fail problem for video surveillance system. The path recovery-method2 is good approach to solve link fail problem for video surveillance system. It has loss rate 14% with loss 751 packets.

Table 2. Packet loss rate comparison.

Methods	Sent packets	Received Packets	Loss packets	Loss rate
Network without link fail	**5364**	**5256**	**107**	**2%**
Path recovery method1	**5364**	**4505**	**859**	**16%**
Path recovery method2	**5364**	**4613**	**751**	**14%**

4.3 Jitter

The jitter is the latency variant and does not depend on the latency. For example, the high response time can be obtained with very low jitter. The jitter is important factor for the network that supporting the

Quality of **Iperf -c 10.0.0.2 -u -b 100K -t 500**

Service. Specifically, the network that transmits the voice over IP (VoIP). The high jitter may be the cause to break the call (video, voice); as a result, the video surveillance system will calculate this factor. To calculate the Jitter in SDN topology, the IPerf (Network Performance Measurement). Open the client side in H1 and server side in H2 as the following command:

On host H1 >>

On host H2 >> **Iperf -s -i 1 –u**

The symbols n these commands represent the following (-c the client side, –u UDP packets, -b bandwidth, -t timing, -s server side and -i interval).

```
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[904] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 32781
[ ID]  Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[904]  0.0- 1.0 sec  1.17 MBytes  9.84 Mbits/sec  1.830 ms  0/ 837  (0%)
[904]  1.0- 2.0 sec  1.18 MBytes  9.94 Mbits/sec  1.846 ms  5/ 850  (0.09%)
[904]  2.0- 3.0 sec  1.19 MBytes  9.98 Mbits/sec  1.802 ms  2/ 851  (0.04%)
[904]  3.0- 4.0 sec  1.19 MBytes  10.0 Mbits/sec  1.830 ms  0/ 850  (0%)
[904]  4.0- 5.0 sec  1.19 MBytes  9.98 Mbits/sec  1.846 ms  1/ 850  (0.12%)
```

Figure 11. Use IPref to calculate the Jitter.

The result for this instruction is shown in **Fig. 11.** The fifth field represents the jitter values. The jitter comparison is discussed in **Table 3.** The jitter value without link failure at starting time is 100 ms while in the path recovery method1 is starting at 300 ms. The path recovery method2 is starting at 100 ms. Therefore, this method is better than method 1 as shown in **Fig. 12**.

Table 3. Jitter comparison.

Methods	Starting Jitter	At 10 Sec	At 50 Sec
Network without link fail	**100 ms**	**15 ms**	**25 ms**
Path recovery method1	**300 ms**	**50 ms**	**40 ms**
Path recovery method2	**100 ms**	**15 ms**	**30 ms**

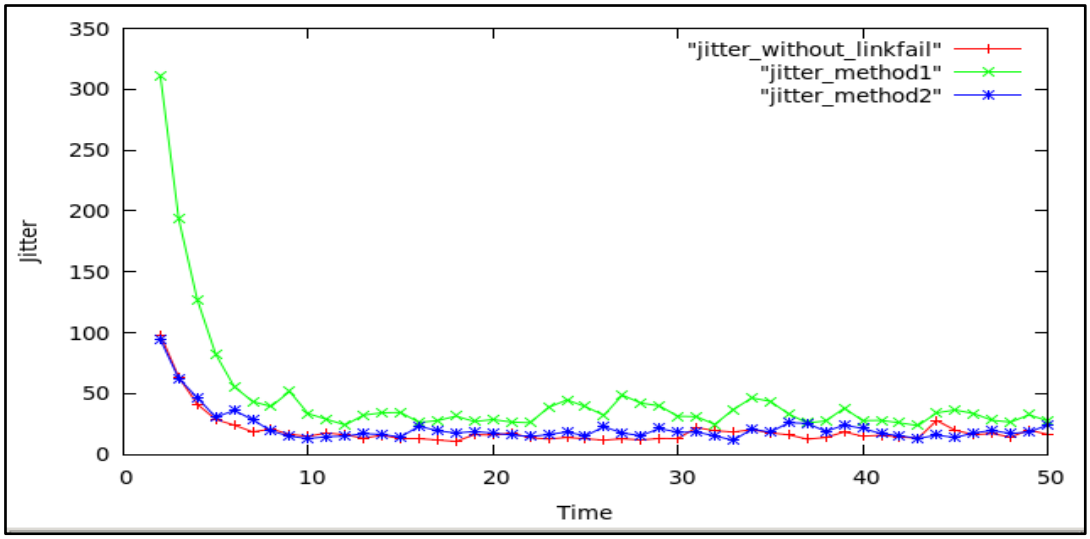

Figure 12. Jitter for three scenario.

5. CONCLUSIONS

With traditional networking, networking functionality is usually employed through dedicated hardware devices such as a router, switches, and firewalls. Each of which much is manually configured by an IT administrator who is responsible for ensuring each device is updated with the latest configuration settings. Therefore, software-defined networking is rapidly becoming a proper solution for those problems. Also, SDN has no difficulty in overcoming the limitations of traditional networking. The SDN decoupling hardware from software i.e. separating the control plane from the data forwarding plane. It enables the hardware to be controlled/managed from a centralized software application (controller) that is separated from the hardware itself. The purpose of this study is to solve the link fail problem instead of continuously dropping the video traffics and cannot reach to the monitoring system**.** The system requires the speed in processing the solution. Therefore, it is proposed to use two-controller in horizontal architecture instead of using one controller to enhance network performance.

NOMENCLATURE

CSP = constraint shortest path
LLDP = with the link layer discovery protocol
ODL = opendayligh
ONF = open networking foundation
OVS = Open vSwitch
QoS = quality of service
SAMCRA= self-adaptive multiple constraints routing algorithm
SDN = software defined network
SVEF = scalable video coding evaluation framework
VLC =video lan client

REFERENCES

- Azodolmolky, S. 2013. Software defined networking with openflow. Packt Pub, Birmingham, UK.
- Capone, A., Cascone, C., Nguyen, A. Q. & Sanso, B. (2015) ' Detour planning for fast and reliable failure recovery in SDN with OpenState '. 11th International Conference on the Design of Reliable Communication Networks (DRCN), Kansas City, MO, USA, 24-27 March 2015, IEEE.
- Cui, L., Yu, F. R. & Yan, Q, 2016. 'When big data meets software-defined networking: SDN for big data and big data for SDN'. IEEE Network, vol 30**,** no. 1, pp 58-65.
- Gopi, D., Cheng, S. & Huck, R. (2017), 'Comparative analysis of SDN and conventional networks using routing protocols'. International Conference on Computer, Information and Telecommunication Systems (CITS), Dalian, China 21-23 July 2017, IEEE, 108-112.
- Goransson, P., Black, C. & Culver, T. 2016. Software defined networks: a comprehensive approach, Morgan Kaufmann, California USA.
- Havlík, J. 2017. Video quality monitoring using netflow. Bachelor's thesis, Brno University of Technology, Faculty of Information Technology, Prague.
- Hosseini Seno, S. A. 2018. 'Dynamic routing method over hybrid SDN for flying ad hoc networks'. Baghdad Science Journal, vol. 15, no. 3, pp 361-368.
- Licandro, F. & Schembra, G. 2007. 'Wireless mesh networks to support video surveillance: architecture, protocol, and implementation issues'. EURASIP Journal on Wireless Communications and Networking-Springer, vol 2007**,** no.1, page 031976.
- Mohammadi, R. & Javidan, R. 2017. 'An adaptive type-2 fuzzy traffic engineering method for video surveillance systems over software defined networks'. Multimedia Tools and Applications-Springer, vol. 76**,** no. 22, pp 23627-23642.
- Nunes, B. A. A., Mendonca, M., Nguyen, X.-N., Obraczka, K. & Turletti, T. 2014. 'A survey of software-defined networking: Past, present, and future of programmable networks'. IEEE Communications Surveys & Tutorials, vol. 16**,** no. 3, pp 1617-1634.
- Owens Ii, H. & Durresi, A. 2015. 'Video over software-defined networking (vsdn) '. Computer Networks- Elsevier, vol. 92**,** pp 341-356.
- Pakzad, F., Portmann, M., Tan, W. L. & Indulska, J. (2014). 'Efficient topology discovery in software defined networks'. 8th International Conference on Signal Processing and Communication Systems (ICSPCS), 2014, IEEE, 1-8.
- Panwaree, P., Kim, J. & Aswakul, C. (2014). '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. 777-779.
- Rametta, C., Baldoni, G., Lombardo, A., Micalizzi, S. and Vassallo, A. 2017. S6: 'A Smart, Social and SDN-based Surveillance System for Smart-cities'. Procedia Computer Science- Elsevier, vol. 110**,** pp 361-368.

- Rymen, M. 2015. 'Software-Defined Networking for Multi-Camera Systems'. M Sc Thesis, National Chiao Tung University, Taiwan.
- Santos, R. R. D. 2015. 'Development of an OpenFlow controller application for enhanced path computation', Thesis of M Sc Thesis, University of Coimbra, Coimbra.
- Sumanth, B. 2016. 'Designing an openflow controller for data delivery with end-to-end qos over software defined networks'. M Sc Thesis, Computer Science and Engineering, Conference in Hollywood, CA, USA.

- Yu, Y. S. & Ke, C. H. 2018. 'Genetic algorithm-based routing method for enhanced video delivery over software defined networks'. International Journal of Communication Systems-Wiley, vol. 31**,** no. 1, page e3391.