Poster: A Software-Defined Multi-Camera Network

Peer-reviewed author version

Chen, Po-Yen; Chen, Chien; Selvaraj, Parthiban & CLAESEN, Luc (2016) Poster: A Software-Defined Multi-Camera Network. In: ACM MobiSys 2016: The 14th ACM International Conference on Mobile Systems, Applications, and Services, Singapore, 25-30/06/2016.

DOI: 10.1145/2938559.2938599
Handle: http://hdl.handle.net/1942/23397

Poster: A Software-Defined Multi-Camera Network

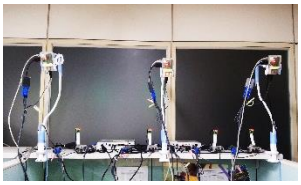
Po-Yen Chen, Chien Chen, Parthiban Selvaraj
National Chiao Tung University

Luc Claesen
Hasselt University

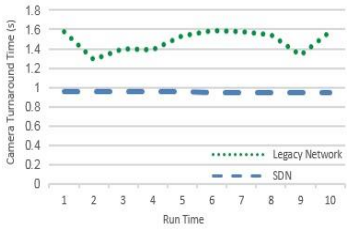
Keywords
OpenFlow; Software-Defined Network; Multi-Camera Network; Software-Defined Multi-Camera Network; Virtual Network

1. INTRODUCTION
The widespread popularity of OpenFlow leads to a significant increase in the number of applications developed in Software-Defined Networking (SDN). In this work, we propose the architecture of a Software-Defined Multi-Camera Network consisting of small, flexible, economic, and programmable cameras which combine the functions of the processor, switch, and camera. A Software-Defined Multi-Camera Network can effectively reduce the overall network bandwidth and reduce a large amount of the Capex and Opex when we set up the multi-camera system. Furthermore, we propose a centralized management system along with a Web User Interface for the user to view the information from the GPS sensor and the digital compass, and monitor the range through the Google map. The user also can see the real-time video stream easily via clicking the camera icon. We propose managing a large amount of cameras, owned by different enterprise customers, through the authentication mechanism and the function of network virtualization in Software-Defined Networking. The groups of customers from different enterprises can manage their cameras in their own Virtual Multi-Camera Networks, isolated from the other enterprise customer's network. Therefore, we can provide different qualities of service through the network virtualization to different enterprise customers. In addition, the forwarding path of the video stream can be dynamically changed by the SDN controller to achieve a seamless real-time video stream, and support an efficient multicast of the video stream and user mobility. Our goal is to address the problems which cannot be programmed, and that have been found difficult to solve in the legacy network over the years through our Software-Defined Multi-Camera Network platform. We designed a smart campus camera network application for real time video surveillance service on the top of the SDN controller, the video seen by school security in the Web User Interface will be dynamically updated if the tracked student moves from the monitoring range of one camera to another.

2. DESIGN AND IMPLEMENTATION
We use the Raspberry Pi as our Software-Defined Camera which is small, cheap and programmable. We connect camera module, GPS sensor and the digital compass to the Pi (see Figure 1(a)) to get the video stream, GPS coordinate, and the angle information. We also use OpenCV to make the Pi deliver the video stream. By default, all of the video stream on Pi will be delivered to a storage server. We use Ryu as our SDN controller and NA-340 (1U-type network appliance) as our OpenFlow switch. We design and implement three modules running on SDN controller including Simple Switch Video Routing module, Multicast module and DHCP module. First module will handle the video stream sent by the Pi and redirect the video stream to the user who has request the video stream. Multicast module will modify the rules installed in the OpenFlow switch to make a video stream came from a Pi forward to multiple ports and users. DHCP module will send the IP address to the Pi and store the GPS coordinate and the angle information sent by the Pi.



(a) Software-Defined Multi-Camera Network



(b) Performance Result

Figure 1: Software-Defined Multi-Camera Network and Performance Result

FlowVisor [1] can create the virtual network in the SDN, but it cannot build the virtual link which is not exist in the physical network. We use the OpenVirteX [2] to solve this problem and design an authentication mechanism running on the OpenVirteX to separate the user and the network device to the different group by their MAC addresses, and use the different Tenant ID to identify the different group.

We not only use the OpenCV on the Raspberry Pi to deliver the video stream to the storage server, but also use the OpenCV to run the motion detection at the same time.

3. PERFORMANCE EVALUATION
To compare the performance result between our system and the legacy network, we use two Raspberry Pi with the camera module running the motion detection and measure the camera turnaround time. We define the turnaround time as the time difference between the time when a tracked student enters the monitoring range of the second camera and the time for a user to see the latest real-time video from the second camera on the WebUI.

We measure the camera turnaround time for ten rounds and the performance result in each round is as showed in Figure 1(b). In legacy network, the results are around 1.2s to 1.6s. In our system, the results are around 0.9s to 1.0s. The reason why the results in our system are less than that in the legacy network is because the second camera will inform SDN controller to modify the rules installed in the OpenFlow switch to make the WebUI does not know that the video saw by the user is from different video sources when the second camera detects the tracked student. In legacy network, after the second camera detects the tracked student, WebUI need to connect to the second camera and disconnect with the first camera to see the video stream containing the tracked student inside. This is because that the camera and the switch are not flexible and programmable in the legacy network.

4. REFERENCES
[1] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. *FlowVisor: A Network Virtualization Layer*. OpenFlow Switch Consortium, Tech. Rep, 2009.

[2] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, G. Parulkar, E. Salvadori, and B. Snow. OpenVirteX: Make Your Virtual SDNs Programmable. In *HotSDN*, 2014.