Incident-Supporting Visual Cloud Computing Utilizing Software-Defined Networking in GENI

D. Chemodanov*, P. Calyam†

Abstract—In the event of natural or man-made disasters, providing rapid situational awareness through video/image data collected at salient incident scenes is often critical to first responders. However, computer vision techniques that can process the media-rich and data-intensive content obtained from civilian smartphones or surveillance cameras require large amounts of computational resources or ancillary data sources that may not be available at the geographical location of the incident. In this paper, we evaluate proposed in our earlier works the incident-supporting visual cloud computing (VCC) architecture (which supports computing at the network-edges and software-defined networking (SDN)) that can enhance public safety application performance to near real-time in comparison with common core cloud computing over IP networks in real settings. To this end, we design and develop a set of reproducible experiments in NSF Global Environment for Network Innovations (GENI) platform by first developing the VCC SDN controller and then creating GENI Custom Images of both developed SDN controller and the public safety application, as well as all corresponding GENI Resource Specifications (RSpecs) and bash scripts to enhance experiment reproducibility.

Index Terms—Visual Cloud Computing, Software-Defined Networking, GENI platform

I. INTRODUCTION

In the event of natural or man-made disasters, videos and photographs from numerous incident scenes are collected by security cameras, civilian smart phones, and from aerial platforms. This abundance of media-rich video/image data can be extremely helpful for emergency management officials or first responders to provide situational awareness for law enforcement officials, and to inform critical decisions for allocating scarce relief resources (e.g., medical staff/ambulances, hazard material or search-and-rescue teams). Information technology resources that are in distributed locations will be needed to interpret and update information about the disaster, coordinate team decision-making for crisis management, rapid response operational resource allocation and geospatial monitoring to redeploy resources. Computer vision methods can play a central role in improving situational awareness and rapid assessment of risk to the public and infrastructure damage by fusing crowd-sourced and surveillance imagery from the geographical region of the incident [1]. Tracking the movement of people, vehicles and objects at the regional-scale will provide analytics for assisting victims, monitoring refugees and coordinating wide-area relief activities. Face recognition is particularly useful to reunite families [2], [3] and assist law enforcement post disaster [4].

However, the computer vision techniques needed to process this media-rich and data-intensive content require large

1 Corresponding and primary author (calyamp@missouri.edu).
2 Department of Computer Science, University of Missouri, USA.
3 To ease user Quality of Experience assessment, observed computation speedup is also demonstrated on the video - https://youtu.be/f4-BdvrI0_g.
access via thin-client desktops. Such on-demand computation and integration of thin-clients for visualization can enable large data processing within the cloud and deliver high user Quality of Experience (QoE). Visual cloud computing is now scalable and capable of processing petabytes of imagery in actionable time frames [12]. However, these paradigms need to be made more resilient, adaptive and fault tolerant to enable post disaster field support for operational use.

**Motivation to use GENI Infrastructure.** Deployment of a private cloud/fog testbed for a researcher can be a time-consuming and expensive proposition at a single site. The deployment can be even more challenging if the testbed involves geographically distributed sites with whom one has to establish collaborations and setup: computation clusters, remote storage, SDN switches and mobile edge networks with IoT devices and sensors. Fortunately, the GENI infrastructure solves these challenges for researchers today and can offer a diverse set of federated resources (e.g., compute, storage and wireless) provisioned at different geo-locations. Our decision to use the GENI platform has not only enabled us to pursue cutting-edge experiments on various cloud/fog architectures, but also has significantly reduced our time for testbed deployment (on the order of even several months if we had to do it in a private infrastructure). In addition, the services and tools in GENI have made our experiments reproducible, which is vital for the future of science and engineering fields. Last but not the least, preliminary results presented in this paper have helped our research lab obtain NSF US Ignite (Focus Area 2) funding in 2016, and we are continuing to build on these experiments to improve disaster communities. Our GENI experiments are essential to engender a next-generation “Incident Command System” that can provide visual situation awareness to public safety organizations, and could potentially save lives during disaster incidents.

**Our contribution.** In this paper, we evaluate how our incident-supporting VCC architecture enhances public safety application performance [5] in real settings by designing and developing reproducible experiments in GENI platform. To this aim, we first have developed a new SDN controller to run our novel traffic steering algorithms adopted for real-time public safety application needs [5], and have created a public GENI Custom Image of it to ease VCC controller deployment in future. The VCC controller prototype source code is also publicly available under a GNU license at [13]. We then have created a public GENI Custom Image of our regional-scale data application with LOFT (Likelihood of Features Tracking) technology [14], [15], [16] we developed to track and recognize objects of interest in aerial motion imagery. Finally, we have created GENI Resource Specifications (RSpecs) and bash scripts needed to further ease resource allocation and the experiment reproducibility.

In our experiments, we demonstrate the use of SDN for on-demand compute offload with congestion-avoiding traffic steering for our LOFT application. Particularly, we show how VCC enhances LOFT application performance to near real-time in comparison with common core cloud computing over traditional IP networks. This result is due to improvements in data throughput and tracking time when remotely analyzing imagery data by utilizing SDN and by dividing the application into small and large instance processing.

**Smart cities of the future will use and have more critical reliance on distributed information technologies especially sensor streams and media-rich video and imagery. The broad scale societal impact of these technologies will be to coordinate the rapid response of government, civil society and business resources in order to facilitate providing timely assistance for disaster victims, save lives, manage and restore damaged infrastructure. The remainder of this paper is organized as follows. Section II presents our SDN controller prototype implementation. Section III describes our testbed setup in GENI platform and details our experimental performance analysis. Section IV concludes the paper.**

**II. SDN Controller Implementation**

In this section we describe the prototype implementation of our VCC data transferring algorithm that is built using SDN principles. The VCC data transferring algorithm solves the constrained shortest path problem to optimize physical resource utilization for real-time application needs [17]. In particular, we have implemented our prototype as a module of the Floodlight OpenFlow controller [18] as shown in Figure 2. Our prototype includes four main logical components: a physical graph discovery service, a path mapping service, a path allocation service and a user web interface that interacts with the controller. The prototype source code is publicly available under a GNU license at [13].

![Fig. 2. System architecture of our VCC prototype which is a module of the Floodlight OpenFlow controller.]()Fig. 2. System architecture of our VCC prototype which is a module of the Floodlight OpenFlow controller (18) includes four main logical components: a physical graph discovery service, a path mapping service, a path allocation service and a user web interface that interacts with the controller. The prototype source code is publicly available under a GNU license at [13].

Smart cities of the future will use and have more critical reliance on distributed information technologies especially sensor streams and media-rich video and imagery. The broad scale societal impact of these technologies will be to coordinate the rapid response of government, civil society and business resources in order to facilitate providing timely assistance for disaster victims, save lives, manage and restore damaged infrastructure. The remainder of this paper is organized as follows. Section II presents our SDN controller prototype implementation. Section III describes our testbed setup in GENI platform and details our experimental performance analysis. Section IV concludes the paper.

**Physical graph discovery.** Upon bootstrapping an SDN setup, all configured OpenFlow switches connect to the controller allowing a dynamic switch-to-switch link discovery. After this phase, the VCC module tracks all unknown incoming packets to detect hosts and their corresponding host-to-switch links. We specify the main physical link properties, such as capacity and cost (e.g., delay) through an XML configuration file. The XML configuration file indirection allows our prototype to easily interact with foreign measurement services, for example for real-time awareness of its link properties. The information
collected by the path discovery service is then used in the path mapping and the path allocation steps.

Path mapping. To map constrained virtual link requests, the path mapping module of our prototype uses information from the physical path discovery service and runs one of the following routing algorithm policies: NM, EDijkstra, IBF or EBFS (for more details refer to [17]). In the current version of our prototype the routing policy is static and has to be set via our XML configuration file before starting the Floodlight controller.

Path allocation. In this last phase of the path embedding, the VCC module sets all appropriate flow rules in all switches via the OpenFlow [10] protocol, along with the computed path obtained during the path mapping phase. Specifically, using OpenFlow protocol messages the module assigns active flows to corresponding queues, i.e., applies an ENQUEUE action, for guaranteed bandwidth provisioning. In our XML configuration file, users may specify also the type of timeout for each flow i.e., the flow duration time can start from either the previously matched packet or from the first flow instantiation. To estimate the available bandwidth on each physical link, the VCC module uses both the capacity information derived from the XML configuration file, and its allocated bandwidth queried from the flow stored in the OpenFlow switch.

Web interface. To request virtual links and/or to monitor physical network states, we have implemented a web user interface, which uses a RESTful API to communicate with our NM prototype module. The user interface uses technologies such as HTML, PHP, AJAX, JavaScript, and CSS.

III. GENI Testbed Setup and Performance Evaluation

In this section, we discuss our GENI testbed setup and show improvements in data throughput and tracking time when using our VCC implementation that utilizes SDN and divides the LOFT application into small and large instance processing for cloud/fog computation, versus complete compute offloading to a core cloud over best-effort IP networks.

![Diagram](image_url)

Fig. 4. Data flows in the allocated GENI topology: (a) VGA data flow interferes with concurrent flow on the s2 → s1 link as regular network sends data through the best (the shortest) path; (b) with using SDN and our VCC controller we optimize network resources usage and redirect concurrent flow through longer path s2 → s3 → s1 to avoid congestion. Further, moving image pre-processing to the fog (to h2 instead h1) enables near real-time tracking with LOFT.

experiments, we consider common resolutions in surveillance video belonging to the broad categories of: (a) Full-resolution WAMI (7800 x 10600) (see Figure 3(a)), (b) Large-scale aerial video (2560 x 1900), and (c) Ground surveillance video (640 x 480) (see Figure 3). To demonstrate VCC (cloud/fog) benefits without large computation demands, we choose the VGA resolution data for the experiment to track pedestrians [19] in a crowd (see Figure 3(b)). The pre-processing step cloud/fog is performed for every image that needs to undergo adaptive contrast enhancement with Imagemagick tool before being used for tracking in the core cloud. The adaptive contrast enhancement requires global image information and thus needs to read in image data into memory, and operates on every pixel. All images are pyramidal tiled TIFF (Tagged Image File Format) and the pre-processing retains the tile geometry.

Our setup for the cloud/fog computation experiments includes 6 virtual machines (VMs) in the GENI platform testbed as shown in Figure 4, where three of these VMs emulate OpenFlow switches (s1, s2 and s3) and others are regular hosts (h1, h2 and h3). To consider disaster network scenarios that impact data transfer, we assume a 4G-LTE network configuration at the fog. Hence, each host-to-switch link has 100 Mbps bandwidth without delay, each switch-to-switch link has only 40 or 50 Mbps bandwidth and 5 ms transmission delay (which is crucial for the LOFT real-time tracking) to emulate congested and damaged network infrastructure in a disaster scenario. Our LOFT application runs on h1 (quad-core CPU and 4GB of RAM) which acts as a computation cloud site, whereas h2 (double-core CPU and 2GB of RAM) acts as a collection fog site, and h3 (single-core CPU and 1GB of RAM) consumes raw data from h2 by acting as a storage consumption cloud site. The h3 is configured with cross-traffic flows consumption such that it interferes with the main data traffic for the LOFT application. We call this cross-traffic as the ‘concurrent flow’, and the application traffic for LOFT as the ‘main flow’. Finally, the thin-client (local PC) acts as a

![Image](image_url)

Fig. 3. LOFT results on (a) WAMI (b) standard and (c) Full-Motion surveillance video. Each frame in (a) is tiled TIFF aerial image with a resolution of 7800x10600 pixels and ≈80 MB size, and each frame in (b) and (c) video datasets is about 2MB compressed.

1) GENI Testbed Setup and Input Data: Multiple video resolutions in practice need to be processed because the input source imagery in surveillance typically spans a wide variety sensor technologies found in mobile devices. In our
data consumer at the user end. LOFT runs on a thread with a backoff timer which sleeps for a specified delay while querying the local folder for the image stream. To transfer data between hosts, we use the SCP utility.

To differentiate between the cloud/fog and the core cloud computation, our experiment workflow is as follows: (i) start sending concurrent traffic from h2 to h3; (ii) start sending main traffic (Imagery) from h2 to h1; (ii.a) while performing cloud/fog computing, start pre-processing concurrently with step (ii) (we assume here that pre-processing is faster than data transfer); (iii) wait till at least the first frame has been transferred; (iii.b) in case of core cloud computing, start pre-processing before step (iv) (in this case LOFT has to wait for each frame when its pre-processing ends); (iv) start LOFT; (v) wait until all main traffic has been transferred; and (vi) terminate both the applications and data transfers.

2) GENI Cloud/Fog Computation Experiment Results: Table I shows the final timing results averaged over 10 trials to estimate 95% confidence intervals for the cloud/fog and core cloud computation cases. For each trial, we used a 300 frame video sequence and measured several QoA performance metrics such as estimated throughput, tracking time, waiting time and total time. We can pre-process frames faster in the core cloud computation case in comparison with cloud/fog computation. However, due to congestion in best-effort IP network and the absence of raw data at the computation cloud site, we cannot track with LOFT application in real-time (with 0 waiting time) in the core cloud computation case. Whereas in the cloud/fog computation utilizing SDN, LOFT can be run in near real time at 3 – 4 Hz.

IV. CONCLUSIONS

In this paper, we have evaluated our visual cloud computing architecture for media-rich data transfer utilizing the benefits of SDN for collection, computation and consumption (3C) in handling incidents due to natural or man-made disasters. Particularly, we have shown how our 3C architecture enables optimization of fog and cloud computation location selection utilizing SDN to connect the network-edges in the fog with the cloud core. Using realistic GENI environment testbed experiments for a regional-scale application for tracking objects in full motion from large scale aerial imagery, we showed how the optimization between fog computing at the network-edge with core cloud computing for managing visual analytics improves data throughput and tracking time when remotely analyzing imagery data. Our experiment results indicate almost 4X speedup of cloud/fog computation of small and large instances utilizing SDN in comparison with common core cloud computation over traditional IP networks. To ease GENI experiments reproducibility, we have designed and developed the VCC SDN controller (publicly available under GNU license at [13]) and then have created corresponding GENI Custom Images of both developed SDN controller and our LOFT tracking public safety application. In addition, we have prepared all needed GENI Resource Specifications (RSpecs) and bash scripts to further enhance experiment reproducibility.

REFERENCES