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 amounts of computational resources that are usually intermittently available, damaged or unavailable within the geographic location of the incident scenes. Thus, data-intensive visual computing requires seamless processing of imagery/video at the network-edge and the core cloud platforms with resilient performance that caters to user Quality of Experience (QoE) expectations.

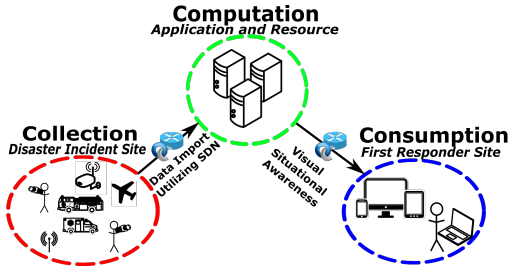


Fig. 1. Overview of the visual collection, computation and consumption (3C) ecosystem: fog computing resources are linked at the network-edge with core cloud computing.

To meet such network-edge data-intensive computations, in our earlier work [5] we have proposed a new paradigm of *Visual Cloud Computing (VCC)*, which augments elastic *cloud computing* property to the network-edge, closer to the location of users and data sensors. In contrast to the stream processing paradigm [6] which is tightly coupled with a computation site, VCC in general and fog computing in particular, allow an application to scale by using multiple computing sites. Such paradigms require an application-aware mechanism to decide on data import or computation offloading that needs to handle the abstraction of a typical VCC scenario shown in Figure 1, where *Collection*, *Computation* and *Consumption* (3C) sites could be in different geographical locations. Thus, we can see the importance of collected and pre-processed incident videos/images to be imported to the cloud for visual analytics, and further transferring of these results from the core cloud servers to a fog near first responders (with mobile devices or thin-clients) for crucial situational awareness over high-speed and reliable networks.

To build a 3C ecosystem, we leveraged emerging techniques in the field of mobile and desktop VCC for scalable processing of media-rich visual data [7], where overlay network paths are dynamically constructed using software-defined networking (SDN) [8], [9], which relies on non-traditional network protocols such as OpenFlow [10]. Coupled with SDN fog computing at the edge can rapidly compute and organize small instance processes locally and move relevant data from the incident geographical location to core cloud platforms such as Amazon Web Services or NSF Global Environment for Network Innovations (GENI) [11] for on-demand processing. Moreover, the overlay network paths can also be useful for moving cloud-processed data closer to the locations of first responders for content caching at fogs to enable low-latency access via thin-client desktops. Such on-demand computation and integration of thin-clients for visualization can enable

†Corresponding and primary author (calyamp@missouri.edu).
*Department of Computer Science, University of Missouri, USA.

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.

GENI motivation. On the one hand, deployment of your own cloud/fog testbed can give your privacy and large resource capacities at expense of a longtime and significant efforts as it involves collaboration between multiple geographically dispersed campuses to setup and configure computation clusters, remote storages, SDN switches and wireless edges with connected IoT devices and sensors. On the other hand, GENI platform today can offer all these features (i.e., compute, storage and wireless) provisioned at different geo-locations (by *stitching* different sites) at expense of privacy and large resource capacities. Thus, we have chosen GENI platform, as it significantly reduces time for deployment (e.g., from month or even years to several days or weeks), which is especially helpful when you need to validate new algorithms by earlier experiments (which can be easily reproduced) in real settings. It worth to mention, that obtained in GENI experimental results helped us to receive the NSF US Ignite award to continue our research in the public safety area.

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.

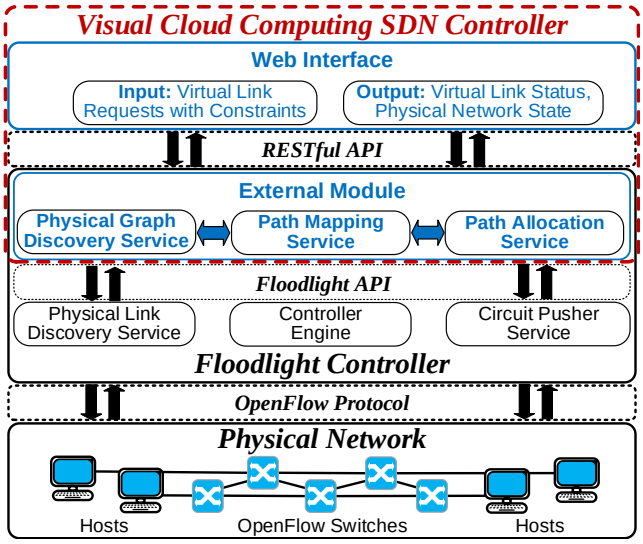


Fig. 2. System architecture of our *VCC* prototype (which is a module of the Floodlight OpenFlow controller [18]) includes four main logical components: a *physical graph discovery* service, a *path mapping* service, a *path allocation* service and a user *web interface* that interacts with the controller.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

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 include 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. Our source code is publicly available at [13]. In the rest of the section we describe with some more details each of the four components of our prototype.

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.

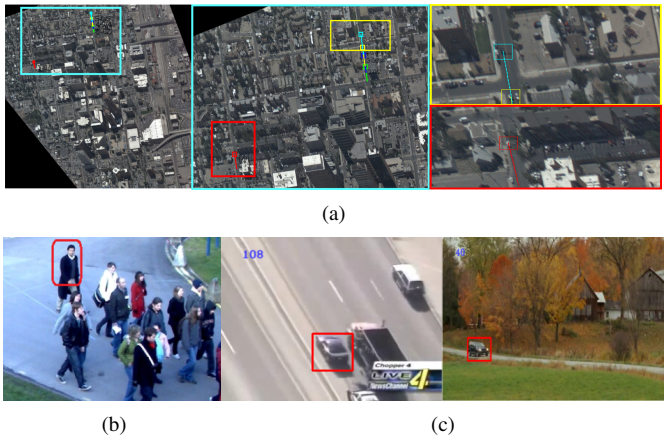
Path mapping. To map constrained virtual link requests, the path mapping module of our prototype uses information from the physical path discovery service and runs one of the following routing algorithm policies: NM, EDijkstra, IBF or EBFS (for more details refer to [17]). In the current version of our 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.



(a)

(b) (c)

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 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

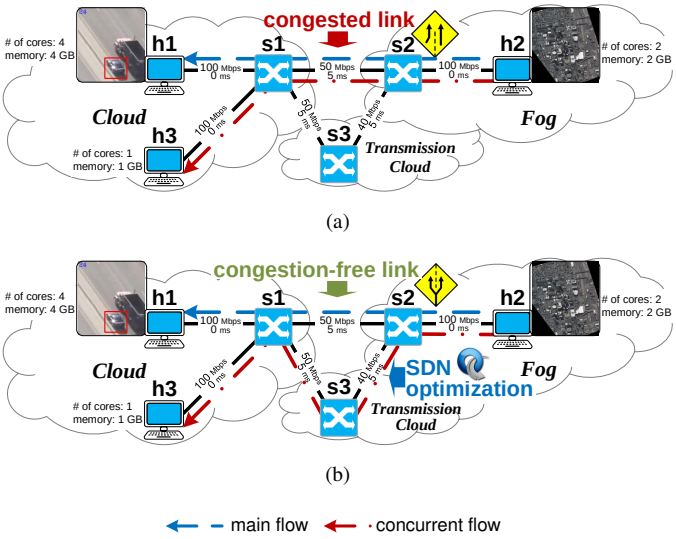


Fig. 4. Data flows in the allocated GENI topology: (a) VGA data flow interferes with concurrent flow on the $s2 \rightarrow s1$ link as regular network sends data through the best (the shortest) path; (b) with using SDN and our *VCC* controller we optimize network resources usage and redirect concurrent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

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 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

TABLE I
QOA IMPACT RESULTS COMPARISON FOR CORE CLOUD COMPUTING OVER IP NETWORK VERSUS UTILIZING SDN AND CLOUD/FOG COMPUTING BY DIVIDING THE APPLICATION INTO SMALL AND LARGE INSTANCE PROCESSING.

Performance Metrics	Core Cloud Computing over IP network	Cloud/Fog Computing utilizing SDN	Perceived Benefits
(Imagemagick QoA) Pre-processing time (sec/fr)	0.1955 ± 0.0011	0.202 ± 0.023	No significant difference
(LOFT-Lite QoA) Estimated throughput (Mbps)	10.50 ± 0.34	41.85 ± 0.24	Avoiding congestion with SDN traffic steering in addition to fog computation maximizes application throughput
(LOFT-Lite QoA) Tracking time (sec/fr)	0.4097 ± 0.0022	0.4229 ± 0.0024	No significant difference
(LOFT-Lite QoA) Waiting time (sec/fr)	0.902 ± 0.032	0 ± 0	Achieving maximum application throughput avoids waiting time and supports real-time computation
(LOFT-Lite QoA) Total time (sec/fr)	1.912 ± 0.034	0.4229 ± 0.0024	Cloud/fog computation of small and large instances can produce 4X speedup over core cloud computation

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 throughout, 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

[1] N. Schurr, J. Marecki, M. Tambe, et al., "The future of disaster response: Humans working with multiagent teams using DEFACTO", *Proc. of AAAI Symposium*, 2005.

[2] G. Thoma, S. Antani, M. Gill, et al., "People locator: A system for family reunification" *IT Professional*, No. 3, pp. 13-21, 2012.

[3] S. Chung, M. Shannon, "Reuniting children with their families during disasters: A proposed plan for greater success", *American Journal of Disaster Medicine*, Vol. 2, 2007.

[4] J. Klontz, A. Jain, "A case study of automated face recognition: The Boston marathon bombings suspects", *Proc. of IEEE Computer*, 2013.

[5] D. Chemodanov, B. Morago, R. Pelapur, et al., "Incident-supporting visual cloud computing utilizing software-defined networking", *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[6] M. Stonebraker, U. Cetintemel, S. Zdonik, "The 8 requirements of real-time stream processing", *ACM SIGMOD*, Vol. 34, No. 4, pp. 42-47, 2005.

[7] H. Agrawal, C. Mathialagan, Y. Goyal, et al., "CloudCV: Large scale distributed computer vision as a cloud service", *arXiv:1506.04130*, 2015.

[8] N. McKeown, T. Anderson, H. Balakrishnan, et al., "OpenFlow: Enabling innovation in campus networks", *Proc. of ACM Computer Communication Review*, 2008.

[9] S. Seetharam, P. Calyam, Beyene, "ADON: Application-driven overlay network-as-a-service for data-intensive science", *Proc. of IEEE Conference on Cloud Networking*, 2014.

[10] OpenFlow Switch specification - https://www.opennetworking.org/sdn-resources/openflow/57-sdn-resources/onf-specifications/openflow; Last accessed February 2017.

[11] M. Berman, J. S. Chase, L. Nakao, et al., "GENI: A Federated Testbed for Innovative Network Experiments", *Elsevier Computer Networks*, Vol. 61, pp. 5-23, 2014.

[12] W. Thissell, R. Czajkowski, F. Schrenk, et al., "A scalable architecture for operational FMV exploitation", *Proc. of IEEE International Conference on Computer Vision Workshops*, 2015.

[13] Visual Cloud Computing SDN Controller - https://bitbucket.org/naas_cloud_computing_project/nm-of-controller/overview; Last accessed February 2017.

[14] K. Palaniappan, F. Bunyak, P. Kumar, et al., "Efficient feature extraction and likelihood fusion for vehicle tracking in low frame rate airborne video", *Proc. of IEEE Conference on Information Fusion*, 2010.

[15] R. Pelapur, S. Candemir, F. Bunyak, et al., "Persistent target tracking using likelihood fusion in wide-area and full motion video sequences", *Proc. of IEEE Conference on Information Fusion*, 2012.

[16] R. Pelapur, K. Palaniappan, G. Seetharaman, "Robust orientation and appearance adaptation for wide-area large format video object tracking", *Proc. of IEEE Conference on Advanced Video and Signal based Surveillance*, 2012.

[17] D. Chemodanov, P. Calyam, F. Esposito, A. Sukhov, "A general constrained shortest path approach for virtual path embedding", *Proc. of IEEE International Symposium on Local and Metropolitan Area Networks*, 2016.

[18] Floodlight - http://floodlight.openflowhub.org/; Last accessed February 2017.

[19] A. Ellis, A. Shahrokni, J. Ferryman, "Pets2009 and winter-pets 2009 results: A combined evaluation", *Proc. of IEEE Workshop on Performance Evaluation of Tracking and Surveillance*, 2009.