Poster: Enhancing AMBER Alert using Collaborative Edges

Qingyang Zhang
Anhui University
Hefei, Anhui, China
Wayne State University
Detroit, Michigan, USA
qyzhang@wayne.edu

Quan Zhang
Weisong Shi
quan.zhang@wayne.edu
weisong@wayne.edu
Wayne State University
Detroit, Michigan, USA

Hong Zhong
Anhui University
Hefei, Anhui, China
zhongh@ahu.edu.cn

ABSTRACT

AMBER alert systems are inefficient since object searching heavily relies on reports of witnesses, who might miss alerts and cannot search enough areas of city. Using automatic license plate recognition (ALPR) technique, city-wide video surveillance is of great improvement for vehicle searching. However, analyzing huge amount of video data in the cloud leads to vast cost of data transmission and high response latency. Edge computing as an emerging computing paradigm can significantly reduce the cost of data transmission and response latency for latency-sensitive applications due to the data processing at the proximity of data sources. In this poster, we propose an enhanced AMBER alert system using collaborative edges, called AMBER Alert Assistant (A3 in short), which can search the suspect vehicle by analyzing static and mobile cameras' data in real time fashion. We propose location-direction-related diffusion that effectively optimizes the searching area for vehicle searching. The evaluation results show that real-time video analytics can be achieved by collaboratively leveraging multiple edge nodes.

CCS CONCEPTS

•Computer systems organization →Distributed architectures;

KEYWORDS

Edge Computing; AMBER Alert; Public Safety; Video Analytics

ACM Reference format:
Qingyang Zhang, Quan Zhang, Weisong Shi, and Hong Zhong. 2017. Poster: Enhancing AMBER Alert using Collaborative Edges. In *Proceedings of SEC '17, San Jose / Silicon Valley, CA, USA, October 12–14, 2017,* 2 pages.
DOI: 10.1145/3132211.3132459

1 INTRODUCTION

AMBER Alert system is kind of system used to alert the public of worrying or life-threatening disappearances of children, and different countries have similar systems [1]. In the USA, when a kidnapping event occurs, an alert is sent to smartphones around the area immediately, which includes sufficient description about the event, such as kidnapper's vehicle license plate number. However, the suspect vehicle searching is inefficient due to the searching heavily relying on the reports of witnesses. Nowadays, video surveillance is widely used in most cities, such as traffic cameras, security cameras and mobile cameras installed on buses and taxis also can provide video data. This provides the opportunity to leverage ALPR technique to analyze video data and it is of great improvement for efficient vehicle searching in *AMBER Alert system* . However, pushing all the video data to the cloud is not suitable for this real-time video analytics, due to high data transmission cost, bandwidth requirement, and long response latency. Beside of cloud-based solutions, edge computing processes the data at the edge of the network, which will significantly reduce the data transmission cost and lower the requirement of network bandwidth [3, 4]. In this poster, we will introduce our application A3, which aims to build a real-time vehicle searching system in a distributed collaborative edge computing way. Our application mainly focuses on the collaborations of local edge nodes for a real-time video analytics and of city-wide edge nodes for vehicle searching.

2 APPLICATION DESIGN

In this application, we consider an edge computing network model as shown in Figure 1. Each camera, including static cameras (e.g., traffic cameras) and mobile cameras installed on buses and taxis, and its edge nodes connect to a router via wire or wireless links. Those edge devices will constitute a local area network (LAN) and the router is used to connect with wide area network (WAN). We defined three kinds of edge devices, called *Control Center*, *Task Receiver* and *Data Processor*. Noted that the *Task Receiver* and *Data Processor*s installed on mobile vehicles are not shown in figure.

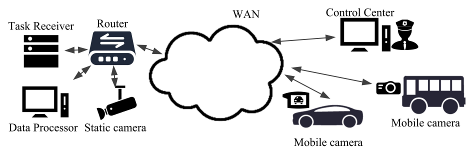

Figure 1: An overview of application.

A *Control Center* is used to publish AMBER alerts to *Task Receiver*s and collect the report from *Task Receiver*s. Once the *Task Receiver* receives an alert, it will automatically pull the video from the connected camera and then analyze the video. According to the system status, it dispatchs part of video analytics workloads (i.e., plate recognition) to neighboring *Data Processor*s. As time goes, the *Task Receiver* will extend the searching areas by forwarding the task to neighboring *Task Receiver*s according to task scheduling algorithm. When suspect vehicle found, it will automatically shrink the searching area by reporting this to other *Task Receiver*s for energy saving. The *Data Processor* in A3 only provides the video analytics services, but it is necessary for a real-time video analytics.

3 APPLICATION IMPLEMENTATION

We implement our A3 on the top of the *Firework* framework [5]. *Firework* is a framework for big data processing and sharing among multiple stakeholders in collaborative edge environment, in which data are owned by multiple stakeholders (i.e., cameras in our A3). Thought *Firework*'s interfaces, we can easily and fastly develop the application on the edge by only implementing the functions as services and setting up the configuring of the application.

The ALPR functions in A3 are implemented based on Openalpr [2]. We define an application-defined topology for A3, which defines connections between *Task Receiver*s and between a *Task Receiver* and its *Data Processor*s. Based on this topology, we present two diffusion algorithms for task scheduling, distance-related diffusion (DD) algorithm, and location-direction-guided diffusion (LD) algorithm, which also are defined by configuration files.

A DD algorithm, as the simplest algorithm, will diffuse the task according to the distance from the incident location. For example, the radius of tracking area will increase by a fixed number as the time goes. In the actual scenario, the DD algorithm has many disadvantages. The speed limitation on different roads is varying, hence, we should set up different diffuse speed for different cameras. Therefore, we propose a location-direction-guided diffusion algorithm for our application. In this algorithm, the edge node will transfer the task to edge nodes according to the road topology.

4 EVALUATION

First of all, we evaluate the performance of collaboration of local edge nodes. Then, we evaluate the two task scheduling algorithms in our A3. All of those experiments compare the performances in terms of latency and count of working threads over time.

To demonstrate the collaboration of local edge nodes, we set up two edge nodes to collaboratively analyze the video, and evaluate the performance of collaboration in our A3 in terms of frame latency, which is defined as the time duration between a video frame is generated and recognized. In this experiment, all the edge nodes are Amazon EC2 t2.xlarge. For case#1 and case#2, the *Task Receiver* will implement a two-thread plate recognition service and *Data Processor* will implement a single-thread service and three-thread service, respectively. For comparison, we use a three-thread service as case#3 comparing with case#1 and use a five-thread service as case#4 comparing with case#2. Figure 2 illustrates the average frame latency of each second. In general, a collaborative solution is better and it can break through the limitation of cores count. In case#2, it can process the video in real time.

We evaluate the performance of task scheduling part in our A3 on the testbed. We deploy 80 *Task Receiver*s to simulate 80 edge nodes connected with traffic cameras. All of these *Task Receiver*s are hosted on the Amazon EC2 VMs with a high performance (i.e., 8 cores CPU) to achieve a real-time video analytics by itself. We also deploy one *Control Center* on a low-performance Amazon EC2 VM (i.e., one core vCPU and 512MB memory). We deploy the applicable video data in *Task Receiver*s, to make sure they can locate the vehicle

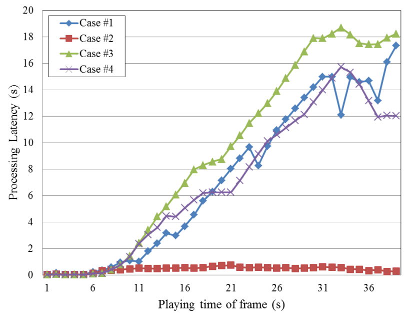

Figure 2: Latency for the collaboration of local edge nodes.

four time, around at 37th second, 75th second, 125th second and 150th second. In this experiment, we record the count of working threads per 100 milliseconds to quantize the workloads.

Figure 3 shows the results of this experiment. From the figure, we can see that the workload of all edge nodes using LD algorithm is less than the case using DD algorithm before 200s, and once the targeted vehicle is located, the workload will reduce immediately. Then, as the searching area extends again, the workload increases. After the last time locating the vehicle, the workload of two cases will increase to the same value. It is because that all of 80 edge nodes we used will participate in tracking vehicle as the time goes.

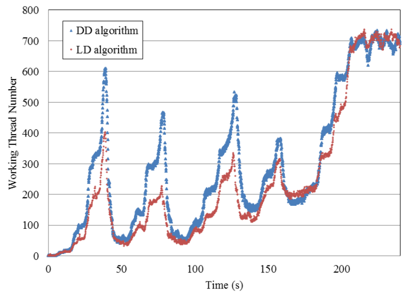

Figure 3: The result of two task scheduling algorithms.

5 CONCLUSION

In this poster, on the top of *Firework* , we implement A3, which can analyze the video streams from cameras in real time by local edge nodes, collaboratively, and control the tracking area effectively. We also proposed two task scheduling algorithms for tracking. Evaluation results show that the LD task scheduling algorithm is very effective in controlling the searching area for vehicle tracking.

REFERENCES

[1] 2017. AMBER Alert. (March 2017). Retrieved March 20, 2017 from https://en.wikipedia.org/wiki/AMBER_Alert
[2] 2017. OpenALPR. (March 2017). Retrieved March 16, 2017 from https://github.com/openalpr/openalpr
[3] Mahadev Satyanarayanan. 2017. The Emergence of Edge Computing. *Computer* 50, 1 (2017), 30–39.
[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. 2016. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646.
[5] Q. Zhang, X. Zhang, Q. Zhang, W. Shi, and H. Zhong. 2016. Firework: Big Data Sharing and Processing in Collaborative Edge Environment. In *2016 Fourth IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. 20–25.