IEEE Workshop on Video Everywhere 2015

LiTMaS: Live road Traffic Maps for Smartphones

S. M. Iftekharul Alam, Sonia Fahmy, Yung-Hsiang Lu
Purdue University
E-mail: {alams,fahmy,yunglu}@purdue.edu

***Abstract*—A smartphone application that displays a live view of road traffic can provide drivers with real-time information on traffic jams, flash floods or accidents along their planned routes. This information aids them in avoiding delays and uncertainties on the road, enhancing their navigation experience. Several US departments of transportation provide information from street cameras in the form of a snapshot of a particular location on a Google map. However, it is difficult for drivers to manually select cameras on the map to obtain consolidated information about road conditions along a particular route. In this paper, we design an energy-efficient mobile service that provides drivers with a convenient interface to observe live camera coverage along a route. Our proposed solution comprises an Android application and a cloud-based proxy service between smartphones and traffic cameras. The proxy provides an abstraction layer over different communication protocols adopted by traffic cameras, and transfers camera images for a specified route to the smartphone as a batch, thereby reducing communication overhead and improving user response time. By employing an in-memory cache, the proxy server maintains the most recently accessed camera images and reduces camera polling. We integrate our solution with traffic cameras from cities in Massachusetts, New York, and Washington DC, and demonstrate its reduced energy consumption and reduced response time.**

I. Introduction

The introduction of the GPS (Global Positioning System) [1] and the proliferation of small form factor motion sensors have made possible navigation systems that are compact and inexpensive enough to be used in consumer products. With the availability of high speed 3G/4G networks and the incorporation of GPS sensors into smartphones, navigation applications are gaining traction among automobile owners and smartphone users. A navigation application typically finds out the user's position via GPS, then loads the map of surrounding areas. The application gives the users information about their position and a route to the desired destination. However, these applications lack information about *real-time* traffic bottlenecks and accidents along the route. This limits the ability of navigation systems to be responsive to dynamic changes on the roadways. Dynamic information can increase safety and provide alternate routes, especially with increasing congestion and accidents on freeways [2].

Google [3] initially provided users with a static street view along a route and now adopts Waze [4] which displays alerts on a map about possible traffic jams and accidents based on crowd-sourced information. There are also several crowd-source based smartphone applications [5], [6] available which provide similar services. However, none of these provides a *live* traffic view along a route. Further, crowd-sourced information cannot be used to make reliable decisions on the road since it is notoriously unreliable.

978-1-4799-8461-9/15/$31.00

Live feeds from road-side cameras installed by the departments of transportation (DoT) are a reliable source of real-time traffic information. Traffic cameras placed at common congestion points on highways, freeways, interstates and major arteries help commuters observe traffic flow at various points and select a route at their discretion. Normally, traffic flows do not vary much from day to day, but in the event of an accident or road closure, a traffic alert can be extremely valuable for a time-crunched commuter. Footage from traffic cameras also allows drivers to be aware of imminent hazardous road conditions stemming from sudden weather changes (*e.g.*, heavy snow, freezing rain, flash floods). Augmenting camera images with map information on smartphones can provide users with a live traffic view along their entire route, and help them avoid unexpected situations on the road. While some applications have been developed in collaboration with different DoTs [7], [8], they only allow drivers to manually select webcams on the map one by one. Hence, these applications are inconvenient to use and do not provide a comprehensive view of a specific route.

In this paper, we propose LiTMaS, a system that provides a live traffic mapping service, sending users snapshots from cameras along their route. A user can analyze the traffic from the images through a convenient interface. The system also familiarizes users with landmarks along their route. LiTMaS includes a cloud-based proxy service between smartphones and street cameras to reduce energy consumption of the mobile application, compared to existing technologies. The proxy server handles different communication protocols with street cameras, and sends images collected from street cameras to the phones through a standard protocol. Thus, the mobile application does not need to undergo any changes when there are changes to communication protocols or standards of street cameras. Further, we pre-process images according to the display requirements of different smartphones, and send images as a batch, thereby reducing computational overhead on smartphones. Transferring images in a batch additionally reduces the overall 3G radio tail [12].

We implement the LiTMaS mobile application on Android and deploy our Java-based proxy server on Amazon EC2. We integrate LiTMaS with street cameras from Massachusetts, New York, and Washington DC, and demonstrate its effectiveness by comparing it with the state-of-the-art, *Beat the traffic* [8], which pulls camera images directly. We collect network packet traces from experiments with both systems. By fitting network packet traces to a standard energy model for a 3G network, we compute the energy consumption of communication. In addition, we collect CPU usage of the mobile application and average response time (time spent from the point of query to the reception of all camera images) per query. Our experimental results show that LiTMaS reduces en-

TABLE I. COMPARISON AMONG LITMAS AND EXISTING WORK.

System	Availability of live images of the route in question	Reporting real-time traffic incidents	Mobile service	Energy-efficient design	Source of real-time information
Google Map Street View [3]	No (a few years old images)	No	Yes	No	Images taken by specially adapted cars
Google Map Traffic (Waze [4] based)	No	Yes	Yes	No	Crowdsourcing
OpenStreetMap [5], WikiMapia [6]	No	Yes	No	No	Crowdsourcing
Augmented Aerial Earth Map [9], [10]	No (projected view)	No	No	No	Street cameras and their projections
Garmin Smartphone Link [11]	Yes	Yes	Yes	No	Street cameras, sensors, radio feeds
Beat the traffic [8]	Yes	Yes	Yes	No	Street cameras
LiTMaS	Yes	Yes	Yes	Yes	Street cameras

ergy consumption and reduces response time without incurring significant computation overhead.

Our contributions include the design of:

- A mobile application that provides drivers with a convenient interface to observe live camera feeds along a route.
- A cloud-based proxy service which handles communication with street cameras, *i.e., caching*, *preprocessing* and *batching* camera images before sending them to smartphones. These techniques reduce computation and communication overhead.
- A complete system integrated with street cameras from different cities. We demonstrate reduced energy consumption by at least 70%, 4× improvement in response time, and reduction in camera polling by at least 48% on the average, compared to the vanilla approach.

II. RELATED WORK

Table I compares traffic map applications and services developed over the past few years. Among these, Google Map [3] is the *de facto* standard for map service, providing users with street and traffic views along their planned routes. The street views are constructed based on images collected a few years ago and thus do not provide real-time information about road conditions. The traffic view feature provides information on the traffic on roads (*e.g.*, light or heavy traffic) and reports alerts about construction or accidents based on crowdsourced information. Crowdsourcing is, however, only as good as the crowd that provides the information. The application is susceptible to targeted, malicious attacks. OpenStreetMap [5] and WikiMapia [6] are two online services which also rely on crowd-sourced information to report real-time traffic situations and thus suffer from the same vulnerabilities. They do not currently run on mobile platforms.

The Garmin Smartphone Link [11] application works together with compatible GPS devices to fetch real-time traffic information. While such information mainly comes from road sensors and crowd-sourced data, this application has an additional feature enabling users to view live feeds from their selected set of traffic cameras. Beat The Traffic [8] is a stand-alone mobile application which provides similar service to get a webcam snapshot. However, it is inconvenient for users to manually select webcams on the map one by one to get consolidated information about road conditions along a particular route. This causes delays in displaying images to the user.

Other work [13], [14] exploits information from street cameras and floating car data to provide fleet management and traffic monitoring of a fixed location. These do not provide map services to drivers on the road. Augmented Aerial Earth Map [9], [10] provides a live view of a particular point on a route but not the entire route. Most recently, Kaseb *et al.* [15] developed a smartphone application with similar goals to ours, but they do not use a proxy service or focus on energy efficiency.

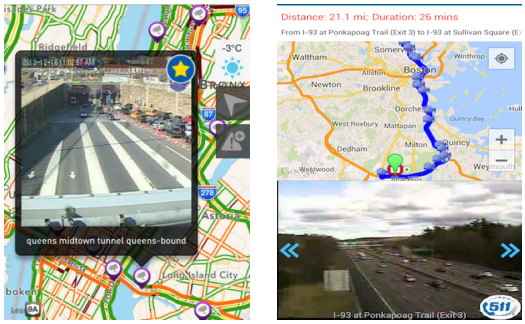
Our proposed solution, LiTMaS, employs a cloud-based proxy service between smartphones and street cameras to provide a real-time traffic view along a route with reduced communication and computation overhead on the smartphones.

III. LITMAS: A LIVE TRAFFIC MAPPING SYSTEM

As discussed above, the high penetration of smartphones and the deployment of a large number of street cameras by the DoTs are two enabling factors to realize a live traffic viewing application for smartphones. However, designing and implementing such an application is non-trivial due to the energy-constrained nature of smartphones and the interoperability issues of street cameras. It is cumbersome to continue updating the mobile application to adapt to the different communication protocols used by the street cameras of different DoTs. Further, the communication overhead between smartphone and street cameras should be optimized to reduce smartphone energy consumption. We therefore define the problem of realizing a live traffic mapping system as: *Providing mobile users with a service that collects real-time snapshots from street cameras and offers them a convenient interface to view live traffic along their specified routes with reduced communication overhead.*

We propose a live traffic mapping system, *LiTMaS*, consisting of a cloud-based proxy service and an Android mobile application. The proxy serves as the communication interface to street cameras and fetches images from cameras to satisfy a certain user query. By caching images during a period of time, our system reduces polling of the cameras and improves user response time. The mobile application allows users to query a particular route and provides them with a video or slideshow of images from the cameras available along that route. We show an example in Figure 1(b) where a video is provided along with the route information for a particular user query. As shown in Figure 1(a), in existing systems, a user has to manually select a camera icon and view one camera's view at a time. This incurs overhead when a large number of users try to view snapshot from the same camera at the same time, since no caching is performed.

Figure III illustrates the overall architecture of LiTMaS. In the subsequent sections, we describe the functional modules of LiTMaS in more detail.



(a) Beat the traffic (b) LiTMaS

Fig. 1. Existing systems require a user to manually select a camera and show one snapshot at a time; In contrast, LiTMaS provides the user with a video or slideshow of images found from the cameras available along the desired route.

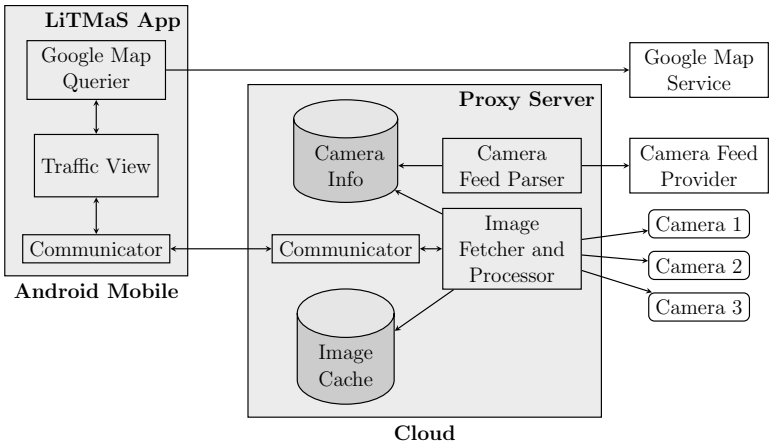


Fig. 2. Overall architecture of LiTMaS

A. LiTMaS Mobile Application

The Android application consists of three modules: (i) Google map querier, (ii) traffic viewer, and (iii) communicator.

The communicator module uses TCP for sending and receiving data to and from the proxy server. The traffic viewer module lets the user enter the start and end points for the journey and shows live traffic along the route. After obtaining user input, the Google map querier module interacts with the Google map service using the Android Google Map APIv2 to retrieve the route from the starting location to the destination.

The route consists of a large number of latitude/longitude points, and is divided into small boxes called *route boxes*. A *route box* is a rectangular box specified by four pairs of latitude/longitude points that covers a number of points on the path. The dimensions of the route box are configurable. The latitude/longitude coordinates of the boxes are then sent over to the proxy server. The server retrieves a list of all cameras within a specified radius of these coordinates, and takes snapshots of live feeds from the cameras. Once the communicator module receives these snapshots, it informs the traffic viewer which displays a slideshow of the received images. The traffic viewer also allows viewing live feeds of a particular point or location.

We note that there is a non-negligible overhead for sending route box information from the smartphone to the proxy, especially when the route is long. By performing the Google map query on the proxy, this overhead can be avoided; however, Google Map has a limit on the number of API requests per IP address: 100,000 requests are allowed per 24 hours and a maximum of 23 waypoints per request [16]. Hence, it is likely that the proxy will be blocked by Google since it has to handle queries on behalf of a large number of users. To reduce this route box-related overhead, we consider caching the location (latitude, longitude) of cameras on the smartphone. Since the route box information is only necessary to find relevant cameras, cached camera information can help locate appropriate cameras on the route without sending route boxes to the proxy. Once cameras are identified, the smartphone can simply send the identifiers of the cameras to the proxy to fetch images, thereby reducing the size of the messages exchanged between the smartphone and the proxy.

B. LiTMaS Proxy Server

The proxy server runs in the cloud and is the bridge between the mobile application and street traffic cameras. It has four main functions: (a) parsing data from the DoTs regarding camera locations, (b) preprocessing images before transferring them to the smartphone, (c) fetching images (if not in cache) and transferring them as a batch to the smartphone, and (d) reducing polling of street cameras by caching frequently used images.

1) Traffic Camera Feed Parser: This module collects information about street cameras from the website of DoTs or via manual input. The DoTs provide information such as the name of the camera, location of the camera (latitude, longitude), hyperlink where the corresponding camera image can be downloaded, and direction of the camera. However, the format of the information (*e.g.*, XML schemas) varies from US state to US state. We define a *Parser* interface that takes raw information provided by DoTs and calls a *parse()* function that outputs a common *CameraObject* for each camera. For each state in the US, we need to implement a separate parser that overrides the *parse()* function and still outputs camera information in the form of *CameraObject*s. If we need to add camera information for new states or modify existing formats, we only need to change the corresponding *parse()* function, keeping other parts of the system intact. After parsing, camera information is stored into a database for future reference.

2) Image Fetcher and Processor: This module takes a set of route boxes along a specific route as input, and finds a list of cameras from the database that are located inside these boxes. Then it searches the cache (described below) to check for the availability of images for the listed cameras. If there is no active image found for a camera, the module fetches the image directly from that camera by retrieving a file from the hyperlink provided with the camera information. The retrieved image is also stored into the cache.

Before sending the images to the mobile application, we adjust their resolution. Preprocessing images for the best viewing experience is important, and we perform this on the proxy server to save smartphone energy. When the LiTMaS mobile application requests the list of cameras available on a route, it also informs the proxy of the dimensions of its

screen. The proxy resizes the images according to the specified dimensions. Once image resizing is complete, the resized images are composed into a slideshow or video in order, and sent to the mobile application.

3) Caching: We use an in-memory Least Recently Used (LRU) cache in the cloud to store the images. The LRU cache moves an item to the head of a queue when it is accessed. When an item is added to a full cache, the item at the end of that queue is evicted and becomes eligible for garbage collection. We start with a single cache node of default size and incrementally add more cache nodes if the total acquired space of the available cache nodes nears their capacity. The advantage of using cloud-based cache nodes is that they can be replaced automatically if there is a failure, thereby reducing the overhead associated with self-managed systems. The ability to add new cache nodes on-demand alleviates the risk of overloaded databases or caches, which ensures smooth operation on the mobile application side.

In addition to the LRU strategy, we evict an image from the cache if it resided in the cache more than a specified time (called *expiry time*). As future work, we plan to investigate the trade-off between accuracy and communication overhead over cameras by varying the *expiry time*.

4) Communicator: The communicator module is responsible for all communication with the mobile application. Traditional blocking I/O spawns a thread for each user, and processes all input/output corresponding to that user in the thread. The thread lives until the user terminates the connection. This approach lacks scalability as the number of users increases, making the system unusable. LiTMaS overcomes this problem by using a single threaded communicator and a pool of worker threads. When there is new data, the communicator reads it from the channel and submits it to a pool of worker threads which perform further processing. If a worker thread is somehow terminated while it is still in use, it is automatically replaced by a new thread.

IV. Evaluation

We test the LiTMaS Android application on an HTC One mobile (32 GB internal storage, 2 GB RAM, Qualcomm APQ8064T Snapdragon 600 processor and Androidv4.4 Kitkat). Additionally, since we do not have the administrative access necessary to analyze energy consumption, we use an emulator of the HVGA slider phone (3.2 inch, Android API 15, ARM 64 bit CPU, 512 MB RAM and 1024 MB SD card) to conduct our experiments. We implement our proxy server using Java, and deploy it on Amazon EC2 with ElastiCache of size 555 MB. We use Amazon Simple DB to implement the database for camera information. The emulator runs on a computer connected to Purdue WiFi network.

We compare the performance of LiTMaS to that of *Beat the traffic* [8], which pulls an image directly from a camera upon a request from the user. For a route query, this application displays the route on a Google map and shows camera icons along the route. The user has to select camera icons one by one to view the live traffic along the route. We implement this approach and port it to the emulator to compare the two approaches.

A. Performance Metrics

In order to analyze the performance of the proxy server, we consider two performance metrics: (1) Cache hit rate, *i.e.*, percentage of camera access requests that are served from the cache of the proxy server, and (2) Average reduction in the number of accesses per camera, compared to the strategy of *Beat the traffic* where each request for an image requires an access to a camera.

We also compute three performance metrics to compare the performance of LiTMaS and *Beat the traffic* mobile applications: (1) Response time: Time between the user query and the reception of an image, (2) CPU usage: Percentage of CPU usage of the mobile application, and (3) Energy consumption: Energy expended for network data transfer over the 3G network on the smartphone.

B. Datasets

We use three datasets from the DoTs of Massachusetts (MA), Washington DC, and New York (NY), which have 68, 299 and 668 cameras, respectively. We generate 500, 1100, and 1000 routes in MA, DC, and NY, by randomly choosing a pair of camera locations within each city. The cumulative distributions of the number of cameras along a route are shown in Figures 3(a), 3(c), and 3(b) for the three datasets. The MA dataset has at most 33 cameras on a route. In contrast, cameras in DC and NY are more densely deployed, with more than 50% of the routes with more than 25 cameras. The maximum number of cameras on a route in DC and NY are 70 and 80, respectively.

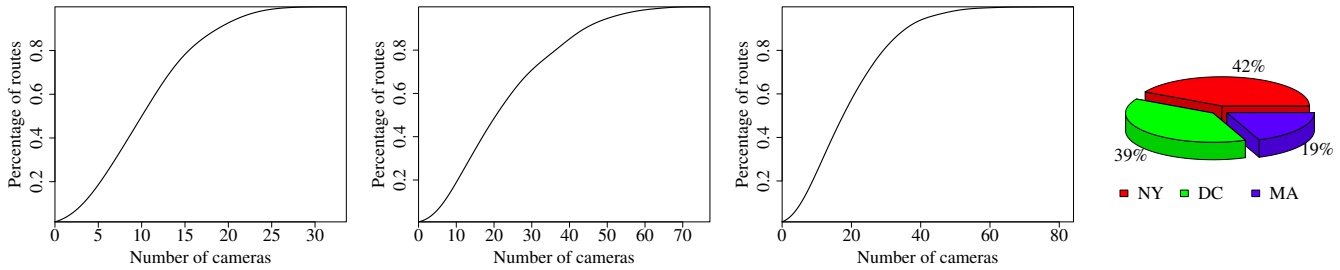
C. Performance of Proxy Server

We run our proxy server on an Amazon EC2 Linux instance. In order to simulate a large number of clients, we rewrite the mobile applications of both LiTMaS and *Beat the traffic* in C using Android's native development kit. These native versions are lightweight since they exclude graphical user interfaces and just handle network communication. We run 2600 clients on a Purdue University server. Out of these 2600 clients, 42% are from NY, 39% are from DC, and the rest are from MA as shown in Figure 3(d). The arrival of clients is controlled according to a Poisson process with $\lambda = 0.3$ s. The duration of each experiment is 13 minutes. We reduce the burden on the DoT servers by grabbing images from their servers once, saving them on the proxy server and then serving them from the proxy, adding the appropriate round trip time (RTT) of each camera link.

TABLE III. Reduction in number of accesses per camera compared to *Beat the traffic*

Expiry time (s)	Average reduction rate	Maximum reduction rate
30	46.80%	91.5%
60	59.49%	95.59%
120	69.70%	98.4%

We vary the *expiry time* for each cache entry (*i.e.*, image) from 30 s to 120 s and measure the cache hit rate at the end of each experiment. Table II shows the output of these experiments. With 30 s, 60 s and 120 s *expiry time*, out of a total 44249 requests, 70.13%, 80.08%, and 88.11% are served



(a) CDF of Massachusetts dataset (b) CDF of Washington DC dataset (c) CDF of New York dataset (d) Distribution of 2600 users

Fig. 3. Cumulative distribution functions (CDFs) of number of cameras along different routes for three datasets and distribution of users considered in the experiment.

TABLE II. PERFORMANCE OF THE CACHE MODULE OF THE PROXY SERVER

Expiry time (s)	Number of requests	Number of hits	Number of misses	Number of items reclaimed	Hit rate
30	44249	31036	13213	10837	70.13%
60	44249	35757	8492	6913	80.08%
120	44249	38989	5260	1379	88.11%

from the cache, respectively. The cache hit rate increases as the *expiry time* increases since the longer an item lives in the cache, the more client requests can be served from the cache. We also find that the number of reclaimed items is 10837 for 30 s *expiry time* which is higher, since a larger number of write or set commands use memory from the expired items as they have a relatively shorter life spans.

We further measure how many times a camera link would be accessed without a proxy (the strategy of *Beat the traffic*) and how many times a camera link is accessed in the presence of our proxy. Then we calculate the reduction in the number of accesses per camera for different *expiry time* values as listed in Table III. With caching, we reduce the average number of accesses per camera by 46.80% to 69.70% where the maximum reduction can be as large as 91.5-98.4%.

D. Performance of Mobile Application

We use `tcpdump` [17] to capture network packets and feed the packet traces to the AT&T tool ARO [18] to analyze energy usage. We use the standard Android *top* to get the percentage of CPU usage for each mobile application every second.

TABLE IV. AVERAGE RESPONSE TIME FOR DIFFERENT ROUTES

Route	# of cameras	Time (milliseconds)	
		Beat the traffic	LiTMaS
1	5	586.76	135.6
2	10	557.78	121.5
3	30	461.24	94.6

We assume that a user needs 4 s to view an image, *i.e.*, images found from the cameras on the route are displayed every 4 s. To compare our approach and *Beat the traffic*, we consider three different routes in Massachusetts, consisting of 5, 10, and 30 cameras, and display all images.. The comprehensive list of all the cameras present on a route is obtained by querying the camera information database on the proxy server. For each route, we run every experiment 5 times and the results are averaged over the 5 runs.

1) Response Time: Table IV lists the response times (defined in Section IV-A) of LiTMaS versus *Beat the traffic* for 5, 10, and 30 cameras. Before polling the first camera, both approaches generate the route and obtain the list of cameras along the route. Hence, polling the first camera image takes much more time compared to polling other cameras. Since we use an intermediate proxy server to fetch the images from the cameras as opposed to polling them directly, LiTMaS is much faster. On the average, LiTMaS is around 4 times faster.

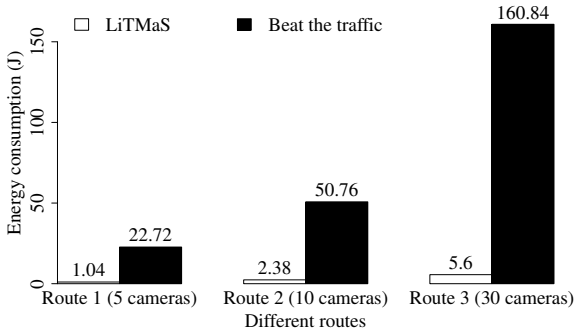


Fig. 4. Energy consumption of LiTMaS and Beat the traffic

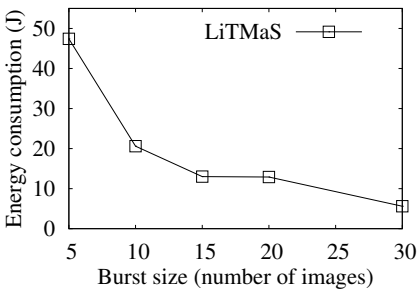


Fig. 5. Energy consumption with varying burst size

2) Energy Consumption: Figure 4 depicts the energy consumption of LiTMaS versus *Beat the traffic* for routes with 5, 10, and 30 cameras. As the number of cameras on the route increases, there is an exponential rise in energy consumed with *Beat the Traffic*, whereas there is a roughly linear rise for LiTMaS. The difference in energy consumption is due to the

fact that the images are directly fetched by the smartphone with *Beat the Traffic*. Every 4 seconds, the user queries each camera on the route, and hence the radio remains in the high power state (DCH) until the user asks for the last camera image on the route. In contrast, the energy intensive job of preparing the list of cameras along the route and fetching the images from them is done in the cloud in LiTMaS. Images are batched, which does not keep the radio in the DCH state unnecessarily, leading to almost negligible energy consumption for the mobile client. We also compute CPU usage of both LiTMaS and *Beat the traffic* applications and find their performance comparable (3 to 4% CPU usage on average).

3) Varying Burst Size: The *burst size* is the number of camera images transferred at a time from the proxy server to the mobile application. Figure 5 shows the energy consumption for the route with 30 cameras with varying burst size. Energy consumption is reduced as the burst size increases. A burst occurs when the user requests a new image (not from the previous burst) and the radio remains on until the transfer of the last burst is complete. This is why energy consumption becomes higher for smaller burst sizes. In contrast, energy consumption is lowest when the burst size is 30 since the radio of the mobile can go to the idle state as soon as all the images on the route (30 images in this case) are downloaded. We note that even with a burst size of 5, LiTMaS reduces energy consumption by 70% compared to *Beat the traffic*.

E. Discussion

Since LiTMaS serves images from a cache, the images may be stale in cases of rapidly changing traffic. By tuning the *expiry time* parameter of each cache entry to its corresponding camera refresh rate, the stale image problem can be avoided. While existing live traffic systems update traffic information every two minutes [11], LiTMaS can provide more rapid updates with an *expiry time* of 30 s, while still reducing the average number of accesses per camera by at least 46.80%. We certainly acknowledge that this benefit stems from leveraging additional resources such as the proxy and cache, which may be hosted on the same infrastructure as the camera database. *Beat the traffic* also interacts with a database server to obtain information about traffic cameras on a given route. As future work, we plan to measure the redundancy across subsequent images of the same camera in cases of frequent user queries about a route. By exploiting this redundancy, LiTMaS can reduce the amount of data exchanged between the smartphone and proxy. We also plan to add personalized features to LiTMaS, enabling users to get live feeds of their favorite routes quickly. For example, the user may prefer to view quantitative traffic information over raw camera images, and the proxy can automatically analyze images and report status in this case.

V. Conclusions

We have presented the design and implementation of a live traffic mapping system for smartphones, LiTMaS, that provides drivers with a convenient interface to observe live camera coverage along a route. Our proposed solution includes a proxy service between smartphones and street cameras to make it more energy efficient. The proxy server handles different communication protocols with street cameras and sends images collected from street cameras to smartphones through a well-defined uniform protocol. The proxy also pre-processes camera images according to the displays of different smartphones, and sends images in a batch, thereby reducing overhead on the smartphones. We implement our mobile application on Android and evaluate it with street cameras from Massachusetts, New York, and Washington DC. Experimental results show that LiTMaS significantly reduces energy consumption and improves user response time.

Acknowledgments

The authors would like to thank the members of the CAM2 (continuous analysis of many cameras) project: `cam2.ecn.purdue.edu`, Wenyi Chen, Ganesh Gingade, Pranjit Kalita, Ahmed Kaseb, and Youngsol Koh for helping identify the traffic cameras used. This work has been sponsored in part by NSF grant CNS-1319924.

References

[1] R. Guha and W. Chen, "A distributed traffic navigation system using vehicular communication," in *Proc. of the IEEE Vehicular Networking Conference (VNC)*, Oct 2009, pp. 1–8.

[2] H. Abbott and D. Powell, "Land-vehicle navigation using GPS," *Proc. of the IEEE*, vol. 87, no. 1, pp. 145–162, Jan 1999.

[3] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver, "Google street view: Capturing the world at street level," *Computer*, vol. 43, no. 6, pp. 32–38, June 2010.

[4] "Waze map," https://www.waze.com/editor/.

[5] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *Pervasive Computing, IEEE*, vol. 7, no. 4, pp. 12–18, Oct 2008.

[6] "Wikimapia," http://wikimapia.org/#lang=en&lat=40.424900&lon=-86.916200&z=12&m=b.

[7] "California transportation department," http://www.dot.ca.gov/dist10/cctv/map.htm.

[8] "Beat the traffic," http://www.beatthetraffic.com.

[9] K. Kim, S. Oh, J. Lee, and I. Essa, "Augmenting aerial earth maps with dynamic information," in *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, 2009.

[10] A. D. Abrams and R. B. Pless, "Webcams in context: Web interfaces to create live 3d environments," in *Proc. of the International Conference on Multimedia*, 2010.

[11] "Garmin smartphone link," http://sites.garmin.com/en-US/smartphonelink/.

[12] F. Qian, Z. Wang, Y. Gao, J. Huang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Periodic transfers in mobile applications: Network-wide origin, impact, and optimization," in *Proc. of the International Conference on World Wide Web*, 2012.

[13] A. Efentakis, S. Brakatsoulas, N. Grivas, G. Lamprianidis, K. Patroumpas, and D. Pfoser, "Towards a flexible and scalable fleet management service," in *Proceedings of the ACM SIGSPATIAL International Workshop on Computational Transportation Science*, 2013.

[14] O. Sidla, M. Rosner, M. Ulm, and G. Schwingshackl, "Traffic monitoring with distributed smart cameras," in *Proc. SPIE*, vol. 8301.

[15] A. S. Kaseb, W. Chen, G. Gingade, and Y.-H. Lu, "Worldview and route planning using live public cameras," in *Imaging and Multimedia Analytics in a Web and Mobile World*, 2015.

[16] "The google geocoding API," https://developers.google.com/maps/documentation/geocoding/.

[17] "Tcpdump," http://www.tcpdump.org.

[18] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Profiling resource usage for mobile applications: A cross-layer approach," in *Proc. of the International Conference on Mobile Systems, Applications, and Services*, 2011.