CrowdWatch: Enabling In-Network Crowd-sourcing

Robin Kravets, Hilfi Alkaff, Andrew Campbell, Karrie Karahalios, Klara Nahrstedt
University of Illinois and Dartmouth College
{rhk,alkaff2,kkarahal,klara}@illinois.edu, campbell@cs.dartmouth.edu

ABSTRACT

Proliferation of mobile smartphones has opened up possibilities of using crowd-sourcing to gather data from and so monitor large crowds. However, depending on the size of the crowd, current solutions either put unpredictable stress on the infrastructure and energy-constrained smartphones or do not capture the crowd behavior accurately. In response, we present CrowdWatch, a scalable, distributed and energy-efficient crowd-sourcing framework. CrowdWatch achieves its goal through off-loading some of the processing to the devices and establishing a hierarchy of participants by exploiting devices with multiple radios (i.e. WiFi (high-power) and BlueTooth (low-power)). CrowdWatch can outperform traditional crowd-sourcing frameworks by reducing the stress on the infrastructures to 10% of that of a traditional crowd-sourcing solution, while only requiring each phone to use their Wi-Fi radios 15% of the time in a dense environment.

Categories and Subject Descriptors

C.2.1 [**COMMUNICATION NETWORKS**]: Network Architecture and Design

Keywords

Crowdsourcing, Smartphones, Energy-Management

1. INTRODUCTION

As people move through their everyday lives, they encounter and navigate through crowds of all sizes and compositions. While some of these crowds come from planned events, others pop up in unexpected places, and in many situations, the size and the behavior of the crowd simply cannot be predicted. Consider the 2012 Summer Olympics in London. While there was some expectation of the number of people attending a particular event, there was no way to predict how the people would move, putting unpredictable stress on many of the city's resources. Although London is well-known for its city-wide infrastructure-based cameras, the collected information collected requires complex and expensive image processing [1] and so cannot be used to react to real-time events. In response, the city deployed a smartphone-based crowd-sourcing app that allowed users to upload their location information via the cellular infrastructure to help determine how to manage the crowds and the associated city resources [2].

Although in such a crowd-sourcing solution the data is collected by the individuals, it ultimately needs to be uploaded and processed on a centralized server or in a cloud. However, the sheer size of a crowd and the demands it puts on any infrastructure, whether it is cellular or Wi-Fi, overstresses the networking resources needed to enable such a solution [3, 4]. Essentially, the more people in the crowd, the more demands they put on the network and the more unpredictable and unresponsive the network service becomes. Unfortunately for any services focused on crowds and crowd behavior, this is exactly when information about the crowd is needed but cannot be uploaded. Additionally, crowd formation and movement is unpredictable, making it difficult to provide sufficient infrastructure coverage in every possible location [5]. Finally, some crowds may form in an ad hoc manner and may not want to use the existing infrastructure.

While all of the data is already collected on the devices themselves, given the limited bandwidth to a server or cloud, it is interesting to consider offloading some of the processing to the devices as well. The resulting system would truly embody the idea of crowd-sourcing from *within the crowd itself*. To this end, we have designed CrowdWatch, a collaborative smartphone-based system that enables effective crowd monitoring and feedback services by monitoring the people, their behavior and the dynamics of the crowd from within the crowd. Through the use of local coordination and processing and probabilistic monitoring of the crowd, CrowdWatch limit the energy cost of processing on a user's phone, reducing the demand on the bandwidth to the server and allowing this limited resource to be more effectively used for aggregated data. The main challenge for such a service is the local monitoring of individuals, their behavior and their social interactions with the people around them in an effective and energy-efficient manner.

In this paper, we present the CrowdWatch architecture, which enables scalable, distributed crowd monitoring and detection via users' smartphones from the "inside-out". As such, CrowdWatch represents a radically different approach in contrast to external approaches that monitor the crowd

MCC'13, August 12, 2013, Hong Kong, China.

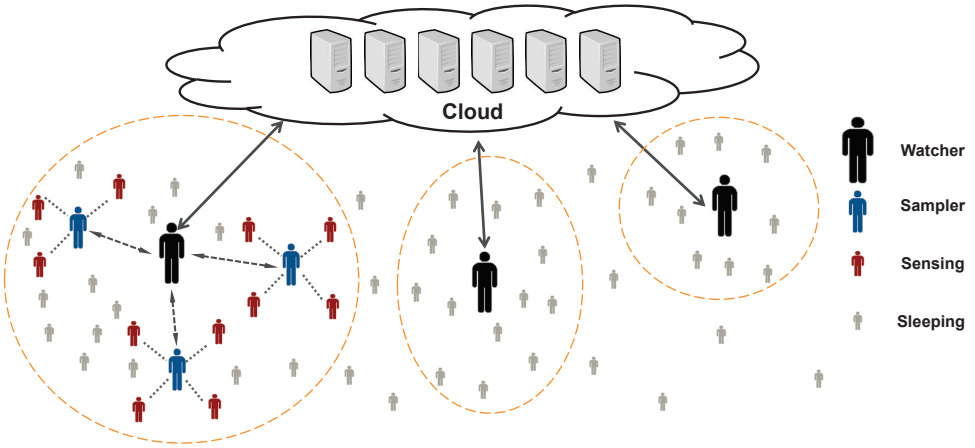

Figure 1: The Hierarchy of a Crowd

from the "outside-in". CrowdWatch operates in an automated manner and requires no specialized devices or infrastructure other than standard off-the-shelf smartphones and scales with the number of phones in the crowd. As a result, CrowdWatch offers users access to unprecedented information to guide them in a crowd.

In the rest of this paper, we first discuss support for large-scale crowd monitoring. In Section 3, we present the hierarchical design of CrowdWatch. Our evaluation of the CrowdWatch algorithms and their impact on energy consumption and backend load is presented in Section 4. Finally, we conclude in Section 5 and discuss our future directions.

2. CROWD MONITORING

Many applications could benefit from crowd detection as well as monitoring of the behavior of a crowd. For example, public safety applications with crowd-monitoring could watch for overcrowding or fights as well as disseminate information about which gates to enter a stadium or where to get medical attention [6].

While crowd-sourcing [7] and crowd computing [8, 9, 10] have been proposed to leverage the information collected by individuals within a crowd, bandwidth and access to servers/clouds over cellular or local Wi-Fi networks might be very unreliable and hence access to servers might not be available. Additionally, these solutions treat a user as a stand-alone entity. The user is required to use significant resources to connect to servers to participate, even if the services, data or people they are trying to find are entirely local.

While infrastructure-based CCTV camera solutions [11, 1, 5] for crowd monitoring and control require no resources from the people in the crowd or from the wireless networking infrastructure and are in place in many cities (e.g., London), these solutions cannot detect the social interactions within a crowd and even lose crucial information about an individual due to the limited image/video resolution.

In response to these limitations, we next present the design of the CrowdWatch Architecture and then evaluate its effectiveness in varying crowd conditions.

3. CROWDWATCH FRAMEWORK

While crowd detection and management can be successful given all individual information all of the time, collecting such information from energy constrained mobile devices is not only overly-demanding of the device's resources, but likely unnecessary in such an environment. Instead, solutions need enable each device to collect sensor information, location/proximity and surrounding context in an energy-efficient manner.

CrowdWatch achieves distributed energy efficient crowd monitoring by exploiting three key observations. First, since people in crowds tend to cluster in groups, there is the potential for effective sampling of data. Essentially, it is not necessary for all users to collect data at all of the times. Second, crowds are dynamic and change constantly. Therefore, it is not necessary to have a perfect snapshot of the crowd at all times. Third, most current mobile devices have multiple radios, Wi-Fi (high power) and BlueTooth (low power). Although the Wi-Fi radio has a longer communication range, it has a higher idle energy cost. Therefore, it is not desirable for all devices to turn on their Wi-Fi radios all of the time. To enable connectivity, CrowdWatch builds a hierarchy of devices, leveraging the distant Wi-Fi connectivity for creating a dynamic backbone in the crowd and the local Bluetooth connectivity to support local sampling and data collection. In this section, we describe the CrowdWatch for energy-efficient and scalable crowd monitoring.

3.1 Architecture

Given our goal and the challenges of the on-line in-crowd monitoring, we have designed CrowdWatch as a distributed, hierarchical cross-layer architecture, with a small functional service residing on each smartphone.

To reduce redundant data collection and processing, and so reduce energy consumption, CrowdWatch organizes the smartphones within a crowd to enable cooperative, scalable, energy-efficient participatory sensing by managing the expected participation of a given phone. To this end, a given node may be actively participating as a *watcher*, a *sampler*

Figure 2: Watcher and Sampler contention management slots.

or a *sensor*, or may be conserving energy in a sleep state (see Figure 1).

At the highest level of the hierarchy, CrowdWatch first selects a small group of *watchers* that act to aggregate local and information from other CrowdWatch nodes in their Wi-Fi neighborhood, with the expectation that the collection of watchers covers the majority of the crowd as needed. Since it is not always necessary to collect sensor data from all devices in a crowd, each watcher targets a number of representative sample areas in its region by probabilistically selecting a set of *sampler* nodes in its Wi-Fi neighborhood. Sampler's then collect sensing and activity information from the devices in their BlueTooth neighborhood, which is a small subset of the watcher's Wi-Fi neighborhood, and pass it back to the watcher. Watchers can then run probabilistic inference algorithms on the collected data and convey the results to other watchers and/or to the cloud server for further analysis. All other nodes not actively participating sleep to conserve their energy. While this is a simple hierarchy, the challenges come from managing roles in a constantly changing environment.

3.2 Hierarchical Monitoring

At the highest level, each *watcher* is responsible for monitoring the area covered by its Wi-Fi radio, including any sensed data from the nodes in its neighborhood collected by *samplers* and *sensing* nodes. Samplers are responsible for collecting sensing information from all active sensing nodes in their BlueTooth neighborhood. In general, this hierarchy is location based, aimed at probabilistic coverage of an area.

To provide coverage of the crowd, the crowd is initially "seeded" with a set of watchers and additional watchers selected pro-actively as need. The optimal selection of watchers and samplers in CrowdWatch is very challenging. If the same set of watchers and samplers is used all of the time, those nodes would quickly run out of energy. However, in the absence of a centralized coordinator, load-balancing is very hard to achieve.

Watcher Selection: To enable efficient load balancing with respect to energy usage, every node should have an equal probability of being a watcher and this role should eventually cycle through all nodes. Ideally, there should be one watcher for every Wi-Fi neighborhood, therefore, watcher selection is be done using the Wi-Fi radio. As a preliminary approach, we assume that the nodes are loosely time synchronized through some periodic use of GPS or other central clock. During watcher selection, all nodes are expected to keep their Wi-Fi interface on without any duty-cycling. During a short time period, nodes independently decide whether or not they want to become a watcher based on knowledge of the number of nodes or other watchers in their neighborhood and how recently they have been a watcher.

CrowdWatch uses a common contention-based approach to selecting watchers where the contention window size determines when the nodes send a claim for being the watcher in their neighborhood. As shown in Fig 2, time is divided into slots. Each node randomly picks a slot from its contention window to broadcast its claim. To insure all nodes hear the claim, which includes the initial time the claim was sent, the nodes repeats the claim for *wContention* seconds. Initially, nodes start with a `windowSize` equal to `maxWindowSize`. Every time a node fails to win a contention phase (i.e., fails to become a watcher), it decreases its `windowSize` by half, until it reaches `minWindowSize`. If a node wins a contention phase and becomes a watcher, it resets its `windowSize` to `maxWindowSize`.

Once watchers have been selected, the watchers broadcast Wi-Fi beacons to inform other nodes of their role and identity. If the network is asynchronous, the broadcast message enable other nodes to synchronize their clocks with the watcher, which supports tighter scheduling of communication between the watchers and its samplers. Over its time as a watcher, as defined by *wLifeTime*, a node may see multiple cycles of sampler selection and neighborhood active sensing. However, if we are relying on the use of smartphones, maintaining synchronization can consume significant amounts of energy and in the end may even be foiled by the limitations of the hardware to stay synchronized. To overcome these problems, we are currently integrating asynchronous solutions using asynchronous neighbor detection [12] and lightweight group management [13].

Sampler Selection: CrowdWatch's probabilistic sensing focuses on data collection at different parts of the crowd at different times. To determine the samplers, and so the sample regions, a watcher sends a beacon over Wi-Fi to all nodes in its neighborhood that specifies a probabilistic parameter for sampler selection, *samplerReq*. Essentially, only a specific fraction of the nodes (e.g., 10%) select themselves as samplers. To insure no overlap of samplers in a given Bluetooth neighborhood, the samplers perform a similar contention-based selection process, resulting in one sampler per neighborhood. Once the samplers have been selected, Wi-Fi is not needed until the samplers need to send their collected data back to the watcher.

The selection of samplers is completely random. This probabilistic approach results in good coverage. As changes are detected in the crowd, the watchers can dynamically adapt the number of samplers needed in their neighborhood. In essences, CrowdWatch monitors the crowd to learn how best to configure and use its resources.

Bluetooth Neighborhood Sensing: The first task of a sampler is to discover all nodes in its Bluetooth neighborhood. During this process, the presence of multiple samplers in the same Bluetooth neighborhood is dealt with using a tie-breaking mechanism. Based on the results of discovery, a sampler selects a percentage of neighbors to connect to over Bluetooth, defined by *sensorReq*, and cycles through these neighbors one by one and asks them for their sensing results. Once the sampler is done collecting samples from its selected subset of neighbors, it switches on its Wi-Fi interface and transmits the data to the watcher during some predefined period based on the watcher's clock.

Data Collection: After watchers and samplers have been selected, all sensing nodes enter the data collection phase. Sensing nodes send the data they collect to the samplers, which aggregate and send the data to the watcher. During data collection, watchers and samplers send beacons at intervals of *wReminder* seconds to remind the samplers and sensors respectively on when the data can be forwarded. The beacons also function to inform any new nodes that are joining that there are existing watchers and samplers. Using this simple synchronization, samplers and sensors do not need to turn on their respective high-power and low-power radios unnecessarily. This phase lasts for *wCollectData*, until watchers and samplers have served for *wLifeTime*.

3.3 Crowd Discovery and Formation

In the absence of a crowd of people, the goal of CrowdWatch is to efficiently detect the existence of a formation of a crowd by monitoring the *density of nodes* in a given neighborhood. A node can approximate the density of people in its neighborhood by employing efficient neighbor discovery through available wireless channels.

Mobile phones have been used to estimate density by scanning the environment for discoverable Bluetooth devices [14, 15]. Ideally, all nodes monitor their neighborhood all of the time. However, not only is such an approach too expensive in terms of energy [16], it is also unlikely to be necessary. In a sparse environment, it is possible to sample the neighborhood at a much lower rate and look for increases in density or the rate of discovery. Therefore, during crowd detection, each CrowdWatch node runs *an asynchronous discovery protocol* (e.g., Searchlight [17],U-Connect [18]) over its Wi-Fi radio. Such protocols allow each node to perform *neighbor detection* without requiring the nodes to be synchronized or run at the same duty cycle. In a sparse network, discovery with high-duty cycle is wasteful. However, if all nodes operate at a very low-duty cycle (e.g., 1%) then it may take too long to detect the existence of a crowd. For example, if we use Searchlight [17], a slotted asynchronous deterministic protocol, and let all nodes operate at 1% duty cycle, in the worst case, successful discovery between a pair of neighbor nodes might takes as much 10000 seconds.

To tackle the problem of long *discovery latencies* due to low duty cycles, CrowdWatch integrates both cooperative and node-based mechanisms. Since CrowdWatch is designed around the concept of samplers and watchers, which are a subset of the nodes in an area, some watchers are configured to perform neighbor discovery with a higher duty cycle. Instead of having *every node* discover with lower duty cycle, a few nodes may become more pro-active and discover with a higher duty cycle. This can dramatically cut down the worst-case discovery latency between the samplers and watchers, allowing much faster detection of a crowd if one exists. Upon successful detection of a group/crowd, a watcher stops duty-cycling for a while and disseminate the information to other sampler nodes so that all nodes (samplers and watchers) have up-to-date crowd information.

4. EVALUATION

In this section, we evaluate the overhead of the CrowdWatch framework, the energy improvements that it enables as well as the accuracy of using CrowdWatch as opposed to traditional crowd-sourcing solutions, which have each of the nodes to upload their data to a central server via its high-power radio. In our evaluations, we focus on three metrics: radio on time, which captures energy efficiency, load, which captures the load imposed on the cellular or Wi-Fi connections to the servers or cloud, and accuracy, which captures the impact of CrowdWatch's probabilistic approach.

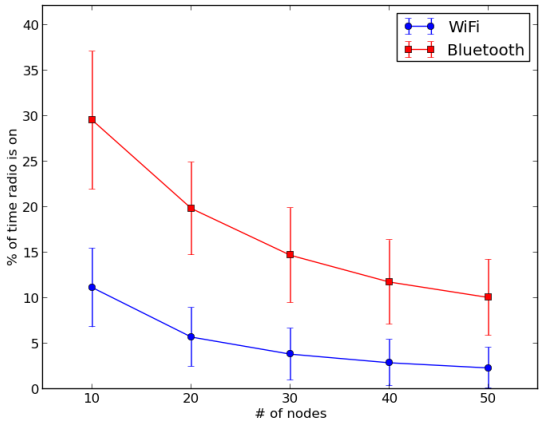

Figure 3: Average fraction of time a node keeps both its high-power and low-power radio on.

Parameter	Value (seconds)
wLifeTime	20
wContention	2
wReminder	1
wCollectData	5

Table 1: CrowdWatch parameters

4.1 Methodology

We implemented CrowdWatch framework in an extension of the OMNeT Network Simulation Framework that supports dual-radios for each node [19]. Since Bluetooth is not yet supported in OMNeT, we emulated Bluetooth using the Zigbee IEEE 802.15 WPAN protocol, with the communication range adjusted appropriately to 10m.

Each of the simulations were run for 1000 virtual minutes and were repeated 10 times. Additionally, since our target is a mobile crowd, each of the simulated nodes were programmed to move randomly 4m/s every minute. The target area was set to be 50m x 50m. The values of the parameters of CrowdWatch described in the previous section is shown in Table 1. Unless specified, *samplerReq* and *sensorReq* are both 50%.

4.2 Evaluation of CrowdWatch

To capture the energy saving capabilities of CrowdWatch, we look at the amount of time each of the radios is on and consuming energy. As shown in Figure 3, CrowdWatch nodes keep their low-power radio on for around 17% of the time and their high-power radio on for around 7% of the time. Additionally, note that the standard deviation of how long different nodes turn on either of its radio on is low ($\approx$ 5% for low-power and 3% for high-power radio). This shows

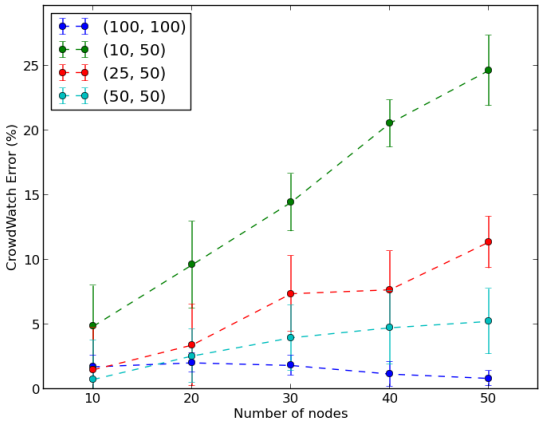

Figure 4: The accuracy of CrowdWatch. The pair indicates the *sensorReq* and *samplerReq*

.

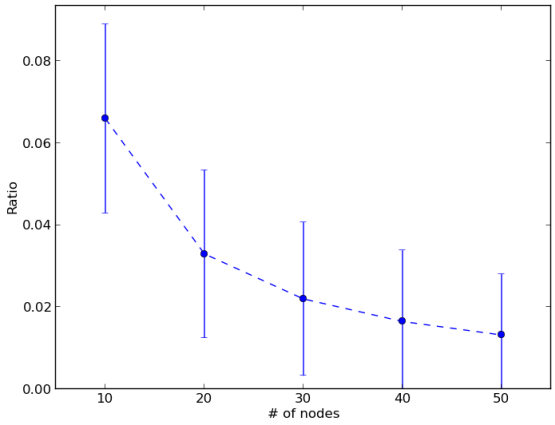

Figure 5: Overhead of CrowdWatch.

the number of times a node is elected as a sampler or watcher is more or less equivalent to that of the other nodes, which captures the fairness of our protocol. Finally, as the number of nodes in the area is increased, the percentage of time either of the radios is on decreases. This captures one of the key design features of CrowdWatch: it is not necessary to have perfect knowledge of the environment, just an accurate understanding of it. Since only a fraction of the nodes are needed as samplers and watchers, the other can sleep and conserve energy.

Next, we simulate an application that runs on top of CrowdWatch and computes the demographic distribution of the number of mobile nodes and we compare it with a traditional crowd-sourcing application in which each of the nodes is submitting the data to the server and so will yield a more accurate result. Note that this is the most difficult type of data to collect since all nodes need to participate to avoid errors. For an application that is sampling the air quality of a particular area, having some nodes in the Bluetooth neighborhood sleeping will not negatively impact accuracy.

With fewer nodes in the area, more nodes need to be awake to send claims and so the data error rate is smaller (see Figure 4). However, as the number of nodes increases, more nodes can sleep and so CrowdWatch's view of the environment will be less accurate. Additionally, Figure 4 also shows how different percentage of samplers and sensors affect the accuracy of the applications. When the percentage of sensors is increased, the accuracy of the application also increases correspondingly. At 50 nodes with 10% *sensorReq*, CrowdWatch has an error rate of 24%. However, when we increase the *sensorReq* to 50%, the error rate drops to 4%. At 100% for both, this is similar to traditional crowd-sourcing in which everybody is participating. The error in this case is $\tilde{2}$% since nodes are not initially synchronized.

Finally, Figure 5 shows the ratio of messages sent to the cloud of CrowdWatch and traditional crowd-sourcing which represents the stress that is put on the infrastructure. At 10 nodes, CrowdWatch sends 7% of the total messages of traditional crowd-sourcing while at 50 nodes, the ratio drops by a factor of 4 to slightly less than 2%. Of course, CrowdWatch performance will vary with different values of *wLifeTime*, *wContention* and *wCollectData* but the general trends will remain.

5. CONCLUSION

Today's crowd monitoring systems have not been able to achieve their goals without losing crucial information in the process or putting too much stress on the phones and the infrastructures. In this paper, we present CrowdWatch, a mobile crowd-sourcing framework in which a small functional service resides on each users' off-the-shelf smartphones and coordinates with the other phones in its direct vicinity. The result is a scalable and energy-efficient crowd-sourcing mechanism that enables the tracking of crucial crowd monitoring information.

In the future, we plan to utilize feedback-loops to dynamically adjust CrowdWatch parameters (e.g. *wLifeTime*, etc) depending on the density of the crowd. We also plan to implement CrowdWatch in smartphones to further determine its feasibility and performance. Finally, the CrowdWatch architecture is only a small part of a crowd monitoring solution. We are currently in the process of designing real-world applications that will use the CrowdWatch architecture to let us go beyond safety applications and study the dynamics of interpersonal interactions and conversations at an unprecedented scale.

6. REFERENCES

[1] I. Saleemi, K. Shafique, and M. Shah, "Probabilistic modeling of scene dynamics for applications in visual surveillance," in *Proc. of IEEE Transactions on Patterns Recognition and Machine Intelligence*, 2009.

[2] "Crowd-monitoring makes the olympics safer - smartphone-apps support the security forces in london," http://www.dfki.de/web/presse/pressemitteilungen_intern/2012/crowd monitoring.

[3] F. J., J. Whiteaker, A. Cuellar Amrod, and J. Woodhams, "Wireless lan design guide for high density client environments in higher education," *Cisco Design Guide*, 2011.

[4] "Cisco connected stadium wi-fi for sports and entertainment venues," *Cisco White Paper*, 2011.

[5] I. Saleemi, L. Hartung, and M. Shah, "Scene understanding by statistical modeling of motion patterns," in *Proc. of IEEE CVPR*, 2010.

[6] K. Starbird, "Digital volunteerism during disaster: Crowdsourcing information processing," in *ACM CHI'2011 Workshop*, 2011.

[7] U. Crowdsourcing, "Proceedings of 2nd workshop on ubiquitous crowdsourcing, 2011," http://www.personal.psu.edu/u1o/crowdsourcing/program.html.

[8] S. Bisker and F. Casalegno, "Crowd computing for some: weaving threads of privacy through public spaces," http://metamanda.com/crowdcomputing/subs/bisker.pdf.

[9] D. G. Murray, E. Yoneki, J. Crowcroft, and S. Hand, "The case for crowd computing," in *ACM MobiHeld'10*, 2010.

[10] "funf open sensing framework," http://funf.org/.

[11] S. V. A. Davies, J. Yin, "Crowd monitoring using image processing," in *IEEE Electronic and Communications Engineering Journal, Vol. 7, No. 1*, 1995.

[12] M. Bakht, J. Carlson, A. Loeb, and R. Kravets, "United we find: Enabling mobile devices to cooperate for efficient neighbor discovery," in *The Thirteenth International Workshop on Mobile Computing Systems and Applications (HotMobile)*. ACM, 2012.

[13] R. Crepaldi, M. Bakht, and R. Kravets, "Quicksilver: Application-driven inter- and intra-cluster communication in vanets," in *Third ACM Workshop on Mobile Opportunistic Networking (MobiOpp)*. ACM, 2012.

[14] J. Weppner and P. Lukowicz, "Collaborative crowd density estimation with mobile phones," 2011.

[15] T. Nicolai and H. Kenn, "About the relationship between people and discoverable bluetooth devices in urban environments," in *The 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*. ACM, 2007.

[16] A. K. Pietiläinen, E. Oliver, J. Lebrun, G. Varghese, and C. Diot, "MobiClique: middleware for mobile social networking," in *WOSN '09: the 2nd ACM workshop on Online social networks*. ACM, August 2009.

[17] M. Bakht, M. Trower, and R. H. Kravets, "Searchlight: Helping mobile devices find their neighbors," in *The 3rd ACM SOSP Workshop on Networking, Systems, and Applications on Mobile Handhelds*, October 2011.

[18] A. Kandhalu, K. Lakshmanan, and R. R. Rajkumar, "U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol," in *International Conference on Information Processing in Sensor Networks*, 2010.

[19] Ó. Helgason and S. Kouyoumdjieva, "Enabling multiple controllable radios in omnet++ nodes," in *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011, pp. 398–401.