MobEyes: Smart Mobs for Urban Monitoring with a Vehicular Sensor Network

Uichin Lee[†], Eugenio Magistretti [*], Biao Zhou[†], Mario Gerla[†], Paolo Bellavista[*], Antonio Corradi[*]
[†]Department of Computer Science, University of California, Los Angeles, CA 90095
[*]Dipartimento di Elettronica, Informatica e Sistemistica, University of Bologna, Bologna, Italy 40136
{uclee,zhb,gerla}@cs.ucla.edu, {emagistretti,pbellavista,acorradi}@deis.unibo.it

***Abstract*— Vehicular sensor networks are emerging as a new network paradigm of primary relevance, especially for proactively gathering monitoring information in urban environments. Vehicles typically have no strict constraints on processing power and storage capabilities. They can sense events (e.g., imaging from streets), process sensed data (e.g., recognizing license plates), and route messages to other vehicles (e.g., diffusing relevant notification to drivers or police agents). In this novel and challenging mobile environment, sensors can generate a sheer amount of data, and traditional sensor network approaches for data reporting become unfeasible. This paper proposes MobEyes, an efficient lightweight support for proactive urban monitoring based on the primary idea of exploiting vehicle mobility to opportunistically diffuse summaries about sensed data. The reported experimental/analytic results show that MobEyes can harvest summaries and build a low-cost distributed index with reasonable completeness, good scalability and limited overhead.**

I. Introduction and Background

Mobile wireless devices such as cell phones, PDAs, and Wi-Fi laptops become ubiquitous in our daily lives and guide us into the era of pervasive computing. For instance, clothes and cars equipped with such devices are going to seamlessly give us helpful information when we are traveling a new city or shopping. Not only do such devices enrich our daily activities, but also create an environment such that epidemics of cooperation can thrive, e.g., among rescue workers or pedestrians. Futurist Howard Rheingold first named these kinds of cooperated activities as “smart mobs,” where people with shared interests/goals can pervasively and seamlessly collaborate using wireless mobile devices [1].

Reflecting on tragedies such as 9/11 and London Bombing, we envision that such smart mobs may actually help relieve losses or investigate accidents if they were properly organized beforehand. For example, in the London Bombing, the police were able to track some of the suspects in the subway using closed-circuit TV cameras, but they had a hard time finding helpful evidence from the double-decker bus, in spite of the abundance of phone pictures taken by shutterbugs. This incident convinced the British police to install more cameras to read license plates and track vehicles. Yet in such a scenario, a smart mob approach should be preferred because completely distributed opportunistic cooperation would make it very hard for potential attackers to disable surveillance.

The reconstruction of a crime and, more generally, the posterior investigation of an event potentially monitored by distributed mobile sensors, require the collection, storage, and retrieval of massive amounts of sensed data. This is a major departure from conventional sensor networks where data are usually collected, examined, and dispatched to a “sink” under predefined conditions, such as alarm thresholds. For instance, it is impossible to deliver all the data detected by video sensors on cars to a police authority sink because of streaming data size. In addition, sensing nodes usually cannot determine a priori whether their data will be of any use for future investigations. Then, this becomes the problem of searching in a massive, mobile, and completely decentralized storage of sensed data, by ensuring low intrusiveness, good scalability, and disruption tolerance against sensor mobility (and terrorist attacks) via completely decentralized cooperation.

A wide range of emerging urban monitoring applications are clear proof of growing interest in the field. Just to mention a couple of examples, Intel Research IrisNet [2] provides large-scale monitoring based on wired PCs equipped with off-the-shelf cameras and microphones. IrisNet agents can collect and process raw sensed data to answer application-relevant queries, e.g., to track available parking in a metropolitan area. Another example is the Gunshot Detection System (GDS) [3] that uses a wireless network of acoustic sensors, strategically and statically deployed to determine locations from where shots are fired. GDS sensors locally process acoustic data to identify shot number and gun type; connectivity is only used to notify police agents. In summary, most urban monitoring solutions, as the two briefly described above, rely on i) static sensor deployment, and ii) predetermined communication/software infrastructures. These aspects make them hardly scalable and vulnerable to attacks.

Urban monitoring can greatly benefit from the exploitation of Vehicular Sensor Networks (VSN). Many car manufacturers are planning to install wireless connectivity in their vehicles to enable communications both with roadside base stations and between vehicles, for the purposes of safety, driving assistance, and entertainment [4]. These Vehicular Ad hoc Networks (VANET) have distinct features, such as high node speed (up to 30m/s) and mobility patterns relatively easy to predict, due to constraints imposed by roads, speed limits, and commuting habits. VSN can be built on top of VANET by equipping vehicles with onboard sensing devices. Unlike traditional sensor networks, VSN nodes are not subject to major memory, processing, storage, and energy limitations.

However, the typical scale of geographic-wide VSN (e.g., millions of nodes), the volume of generated data (e.g., also streaming data sensed by cameras), and vehicle mobility make it unfeasible to adopt traditional sensor network data-centric solutions such as Directed Diffusion. Further, the mobility of VSN nodes makes it less efficient to exploit known mobile collectors such as MULEs [5].

To tackle the challenging technical issues of VSN-based urban monitoring, we have developed the MobEyes system, which supports the formation of smart mobs of sensor-equipped vehicles. Since it is usually unfeasible to directly report the sheer amount of sensed data to a centralized collector, e.g., the police authority, MobEyes proposes that sensed data stay with mobile monitoring nodes. Vehicle-local processing is exploited to extract features of interest, e.g., license plates from traffic images. MobEyes VSN nodes generate data summaries with features and context information (timestamp, positioning coordinates, ...). Then, MobEyes collectors, e.g., police patrolling agents, move and opportunistically harvest summaries from neighbor vehicles. Collectors use summaries to identify, and then pump out, only the sensed data of interest from the carrying vehicles.

The original MobEyes protocols for summary diffusion/harvesting take advantage of vehicle mobility and only exploit single-hop communications. As thoroughly demonstrated by the reported experimental results, in common deployment scenarios, opportunistic summary diffusion and harvesting can index sensed data in feasible time, with good scalability, and with limited overhead, by maintaining a completely decentralized disruption-tolerant organization. Note that MobEyes can be applied to a wide spectrum of applications, not only to forensic data management: for instance, MobEyes-enabled VSN can measure pollution and collect traffic information, such as road conditions and congestion.

II. MobEyes: Guidelines and Architecture

Given the delay-tolerant nature of urban monitoring applications, the primary solution guideline in MobEyes is exploiting vehicle mobility to help inexpensive summary delivery. Let us start by presenting MobEyes while at work in one possible application scenario, i.e., the collection of data sensed by vehicles casually in the nearby of criminals who are in flight on a car after having spread poisonous chemicals. Assume that vehicles are equipped with cameras and/or chemical detection sensors. They continuously generate a huge amount of sensed data, store and process them locally, and produce short summaries periodically or in an event-driven way, e.g., when chemical readings overcome specified thresholds. Summaries are opportunistically disseminated to neighbor vehicles, thus making it easier for authorized police agents to build an a-posteriori index, e.g., for crime scene reconstruction, by harvesting such distributed metadata.

To support the above tasks, we have developed MobEyes by following the component-based architecture in Figure 1. The key component is the MobEyes Diffusion/Harvesting Processor (MDHP), detailed in the next section. MDHP works by opportunistically disseminating/harvesting summaries produced by MobEyes Data Processor (MDP). MDP accesses sensor data via a uniform MobEyes Sensor Interface (MSI).

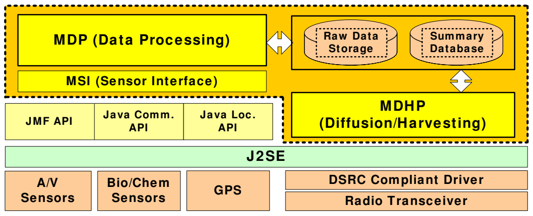


Fig. 1. A high level architecture of a sensor node

MDP is in charge of reading raw sensed data via MSI, processing them, and generating summaries. Summaries include context metadata (location, timestamp, ...) and features extracted by local filters. For instance, MDP includes a filter that can determine license plate numbers from multimedia flows taken by cameras [6]. Moreover, MDP commands the storage of both raw data and summaries in two local databases.

Summary generation rate and size are crucial for MobEyes performance. Developers of MobEyes-based applications can i) specify the desired generation rate as a function of vehicle speed and expected vehicle density (periodic summaries) or ii) indicate which results from selected filters have to trigger generation (event-driven summaries). Summary size depends on application-specific requirements. In the considered scenario, MobEyes generates a summary any time a license plate number is recognized; the summary includes the plate number (6 bytes), additional sensed data, e.g., toxic agent concentrations (10), timestamp (2), and vehicle location (8). In addition, MDHP disseminates/harvests summaries by putting together a set of them into a single packet. In the considered scenario, MDP can pack 58 summaries in a single 1500 bytes message, without exploiting any data aggregation or encoding technique; typically, summaries are generated every [2-10] seconds and, thus, a single message can include all the summaries of a [2-10] minutes interval.

To achieve high portability and openness, **MSI** permits MDHP to access raw data independently of sensor implementation, thus simplifying the integration with heterogeneous sensors of different types. MSI currently implements methods to access camera streaming outputs, serial port I/O streams, and GPS information. To interface with sensor implementations, MSI exploits well-known standard specifications: Java Media Framework (JMF), Java Communications, and JSR179 Location API.

III. MDHP Protocols

The section first details our original summary diffusion protocol where private vehicles (regular nodes) opportunistically and autonomously spread summaries while moving. Then, it describes our novel summary harvesting protocol used by police agents to proactively build a low-cost and possibly partial index of mobile VSN storage.

A. Summary Diffusion

Any regular node periodically advertises a packet with newly generated summaries to its current neighbors. Each packet is uniquely identified (generator ID + locally unique sequence number). This advertisement to neighbors provides more opportunities to the agents to harvest the summariesmake , and the duration of periodic advertisement should be fixed properly to fulfill the desired latency requirements because harvesting latency depends on it.

Neighbors receiving a packet store it in their local summary databases. Therefore, depending on node mobility and encounters, packets are opportunistically diffused into the network. MobEyes is usually configured to perform passive diffusion: only the packet source advertises its packets. Two different types of passive diffusion are implemented in MobEyes: single-hop passive diffusion (packet advertisements only to single-hop neighbors) and k-hop passive diffusion (advertisements travel up to k-hop as they are forwarded by j-hop neighbors with $j < k$). MobEyes can also adopt other diffusion strategies, for instance single-hop active diffusion, where any node periodically advertises all packets (generated and received) in its local database at the expense of a greater traffic overhead. As detailed in the following section, in a usual urban VANET, it is sufficient for MobEyes to exploit the lightweight k-hop passive diffusion strategy, with very small k values, to achieve needed diffusion.

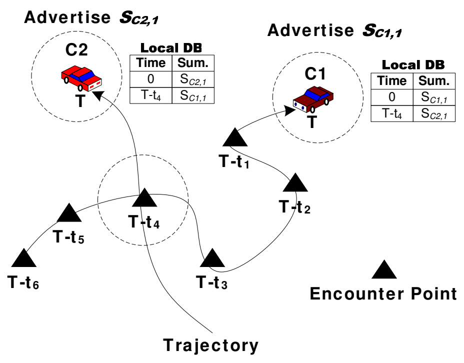


Fig. 2. MobEyes single-hop passive diffusion

Figure 2 depicts the case of a VSN node C1 encountering with other VSN nodes while moving (for the sake of readability, only C2 is explicitly represented). Encounters occur when two nodes exchange summaries, i.e., when they are within their radio ranges and have a new summary packet to advertise. In the figure dotted circles and timestamped triangles represent respectively radio ranges and C1 encounters. In particular, the figure shows that C1 (while advertising $S_{C1,1}$) encounters C2 (advertising $S_{C2,1}$) at time $T - t_4$. As a result, after $T - t_4$ C1 includes $S_{C2,1}$ in its storage, and C2 includes $S_{C1,1}$.

B. Summary Harvesting

In parallel with diffusion, summary harvesting can take place. A MobEyes police agent can request the collection of diffused summaries by proactively querying its neighbor regular nodes. The ultimate goal is to collect all the summaries generated in a given area. Obviously, a police agent is interested in harvesting summaries it has not collected so far: to focus only on missing packets, a MobEyes agent compares its already collected packets with the packet list at each neighbor (set difference problem), by exploiting a space-efficient data structure for membership checking, i.e., a Bloom filter. A Bloom filter for representing a set of n elements consists of m bits, initially set to 0. The filter applies k independent random hash functions $h_1, \cdots, h_k$ to MobEyes packet identifiers and records the presence of each element into the m bits by setting k corresponding bits. To check the membership of the element x, it is sufficient to verify whether all $h_i(x)$ are set.

Therefore, the MobEyes harvesting procedure consists of the following steps:

1) The police agent broadcasts a "harvest" request with its Bloom filter.
2) Each neighbor prepares a list of "missing" packets from the received Bloom filter.
3) One of the neighbors returns missing packets to the agent.
4) The agent sends back an acknowledgment with a piggybacked list of just received packets. Upon listening or overhearing this, neighbors update their missing packet lists for the agent.
5) Steps 3 and 4 are repeated until there is no remaining packet.

Note that Bloom filter membership checking is probabilistic. In particular, false positives may occur and induce MobEyes regular nodes not to send packets still missing at the agent. The probability of a false positive depends on m and n [7]. Nevertheless, in MobEyes, the agent can obtain a missing packet with high probability, because it is highly probable that other nodes have the packets as time passes, and the harvesting procedure is repeated as the agent moves. For example, in usual VSN deployment scenarios (e.g., with 10 neighbors on average), we can show that the probability of missing one packet due to false positives after repeating the procedure 5 times is extremely low, i.e., about 10^{-12} (see [8]).

For the sake of presentation simplicity, thus far we assumed that there is a single police agent harvesting summaries. Actually, MobEyes supports concurrent harvesting by multiple agents that can cooperate by exchanging their Bloom filters, with the benefits in terms of latency/accuracy shown in SectionIV-B. Note that strategically controlling the trajectory of police agents and properly scheduling the exchange of their Bloom filters are part of our future work.

Even if out of the scope of the paper, let us note that security issues in VSN-based urban monitoring are critical, especially when applying to crime reconstruction. For a detailed description of the security solutions integrated in MobEyes, please see [8]. Briefly, to counteract false summary injection attacks, MobEyes exploits different heuristics on spatial graphs of summaries, by removing false packets based on temporal correlations and mutual observations from different regular nodes. The expensive operations of building spatial graphs and identifying false packets are performed off-line at police agents once collected the summary index. In addition, by introducing public key infrastructure (PKI), MobEyes regular nodes can encrypt their packets with the police public key, and thus, concealed summary diffusion is feasible.

IV. MobEyes Performance Results

We have thoroughly validated and evaluated MobEyes protocols via analytic studies and extensive simulations using ns-2 [9]. For the sake of briefness, this paper presents the most relevant performance figures about diffusion/harvesting latencies and traffic overhead. A wider set of performance results is available in [8].

The simulation results reported here are obtained by considering hundreds of vehicle nodes with IEEE802.11 connectivity, 11Mbps bandwidth, nominal radio ranges from 100m to 300m, and two-ray ground reflection as radio propagation model. Vehicles move with an average speed of 10m/s in a real map of the $2,400m \times 2,400m\ Westwood$ area in the UCLA campus vicinity. The adopted mobility model is Real-Track (RT) [10], which can represent urban mobility more realistically than other simpler and widely used mobility models, such as Random Way Point (RWP) and Manhattan. For instance, RT can model vehicles that tend to move as a group because of traffic signals and switch directions only at road intersections. Anyway, [8] includes MobEyes evaluation results also for RWP and Manhattan, thus showing the applicability of the proposed solution while adopting different mobility models. Let us observe that the choice of reporting here the results for RT mobility in the Westwood area represents a notable worst case scenario because that deployment environment has a relatively less homogeneous spatial distribution of roads and intersections.

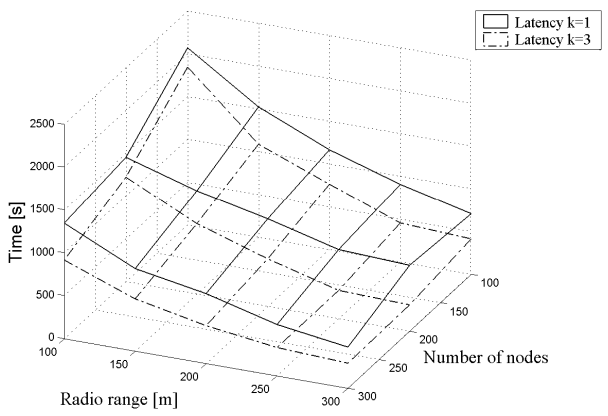


Fig. 3. Summary diffusion latency as a function of radio range and node density

A. Summary Diffusion Latency

Summary diffusion latency measures the time for a police agent to harvest a summary packet, from either the packet generator or a node that has received the packet via diffusion (infected node). We selected an agent and a generator and initially put them at opposite corners of the Westwood map. We ran simulations for passive diffusion by varying the number of nodes (N=100-300), their radio ranges (R=100m-300m), and k-hop relay scope (k=1-3). Reported results are average values over 10 scenarios, each of which was averaged over 30 runs. Figure 3 shows that diffusion latency decreases while increasing radio range or node density. That depends on the fact that range/density growth accelerates diffusion and, thus, the probability for the agent to encounter infected nodes. In addition, since range variations quadratically affect the area covered via single-hop communications, they have greater impact on latency than node density. Moreover, higher k decreases latency, at the expenses of a greater traffic overhead.

B. Summary Harvesting Latency

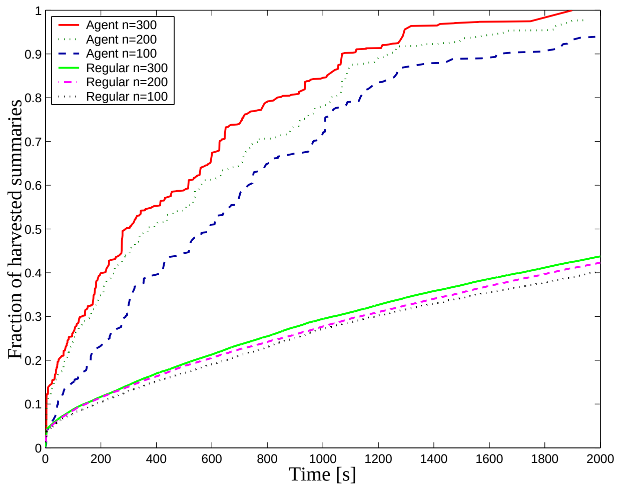
A crucial performance figure to evaluate MobEyes feasibility is summary harvesting latency, i.e., the time for an agent to harvest all summaries generated by regular nodes. In Figure 4, we compare the timeline of summary harvesting with the timeline of the average number of summaries passively diffused at a regular node. The figure reports results while varying node density, k-hop relay scope, and harvesting agent number. In particular, it shows that the harvesting latency decreases as node density increases. Density growth increases the number of infected nodes, thus expediting diffusion. As a result, the harvesting latency decreases since the police agent can pick up more summaries from regular nodes. On the other hand, the passive diffusion rate is independent of node density, but is a function of node speed, transmission range, and network size as shown in [8]. Intuitively, density growth increases the number of infected nodes, but in the case of passive harvesting, a node should collect more summaries, and thus, this offsets the benefit of the density growth.

As Figure 4(b) shows, multi-hop relaying in passive diffusion and parallel deployment of multiple police agents can significantly reduce harvesting latency. Initial positions of agents are randomly selected within the map and there is no control on their trajectories. From the figure, it is also possible to estimate which is the fraction of harvested summaries in the case of strict time constraints. For instance, for maximum harvesting time allowed=1,000s, a single agent can collect 80% summaries with 1-hop relay scope. Given the application requirements determining generation rate, MobEyes can be configured to achieve the most suitable tradeoff between latency/completeness and traffic overhead by properly choosing k-hop relay scope and number of harvesting agents.

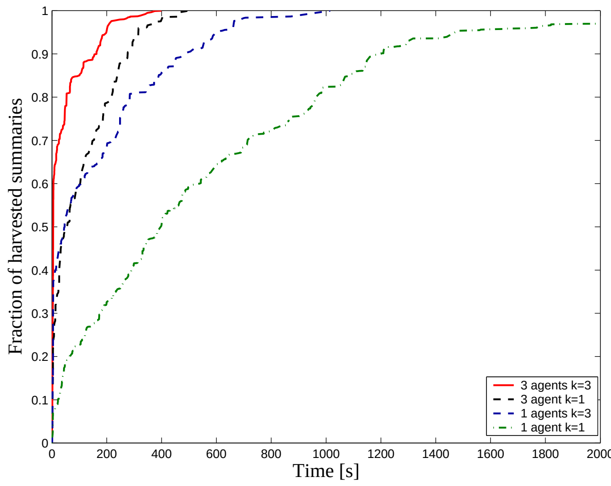
C. MobEyes Scalability

Two main performance figures are suitable indicators to estimate MobEyes scalability over wide VSN: network traffic due to passive diffusion and number of regular nodes that a single police agent can handle.

About passive diffusion, it is simple to analytically evaluate MobEyes radio channel utilization (see [8]). By considering periodic summary advertisement, the diffusion process can be modeled as a packet randomly sent within a time slot of $[kT_a, (k+1)T_a)$ for all k, where T_a is the advertisement period. So, the number of packets received by a node is bounded by the number of its neighbors while moving during T_a, which depends on node density (in contrast, any "flooding"-based diffusion protocol is not scalable because a node could receive a number of packets proportional to network size). It is possible to analytically demonstrate that,



(a) Effect of node density



(b) Effect of k-hop relay scope and multiple agents

Fig. 4. Summary harvesting latency

in common deployment scenarios (T_a=25s, packet size=1500 bytes, 2,000 regular nodes), the worst case link utilization is less than 2% [8].

Similarly, about the maximum number of nodes manageable by a single police agent, it is possible to analytically determine that number via a queuing model with Poisson arrival rate for summary packets (see [8]). To guarantee stability, i.e., to avoid that packets are generated faster than harvested, one agent can manage up to one thousand regular nodes simultaneously moving in the above considered scenario. Let us notice that in the case of more populated areas, stability maintenance is guaranteed by deploying additional MobEyes harvesting agents.

V. Related Work

Few recent VANET research activities have proposed solutions partially similar to MobEyes in different application scenarios and with different goals. [11] presents protocols to reduce message delivery delay to known destinations by exploiting predictable vehicle mobility. [12] aims at disseminating a single message to all nodes within a target region. Both solutions are based on time-limited message broadcasting. In [13], vehicles are interested in the whole data generated within a locality: that work investigates broadcast rate influence on diffusion and suggests rate adaptation based on traffic conditions. [14] proposes VANET resource discovery based on opportunistic diffusion: differently from MobEyes, [14] does not convey information to data sinks and nodes advertise their whole information, either locally generated or not. The accent is on minimizing local memory occupation via spatio-temporal spreading control, and not on preserving communication channels via lightweight diffusion protocols.

Finally, even if not specifically addressing VANET deployment, it is worth considering the 7DS system, which proposes periodic advertisements to randomly mobile neighbor peers [15]. 7DS advertisements are similar to MobEyes summaries, but are diffused based on local assumptions about data relevance that cannot apply to VSN-based applications.

VI. Conclusion

MobEyes demonstrates the feasibility of autonomous VSN-based smart mobs for proactive urban monitoring, if coupled with lightweight mobility-assisted opportunistic protocols for summary diffusion/harvesting. The reported evaluation shows that i) MDHP is scalable up to thousands of nodes with limited overhead and reasonable latency, and ii) MobEyes can achieve suitable tradeoffs between harvesting latency/completeness and overhead via proper configurations of k-hop relay scope and police agent number.

References

[1] H. Rheingold, *Smart Mobs: The Next Social Revolution*. Basic Books, 2003.

[2] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan, “IrisNet: An Architecture for a Worldwide Sensor Web,” *IEEE Pervasive Computing*, vol. 2, no. 4, pp. 22–33, Oct.-Dec. 2003.

[3] “GDS: Technology being tested to detect and report gunshots,” http://www.computerworld.com/mobiletopics/mobile/technology/story/0,10801,77554,00.html.

[4] “Standard Specification for Telecommunications and Information Exchange Between Roadside and Vehicle Systems - 5 GHz Band Dedicated Short Range Communications (DSRC) Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” Sept. 2003.

[5] R. C. Shah, S. Roy, S. Jain, and W. Brunette, “Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks,” *Elsevier Ad Hoc Networks Journal*, vol. 1, no. 2-3, pp. 215–233, Sept. 2003.

[6] L. Dlagnekov and S. Belongie, “Recognizing Cars,” UCSD CSE, Tech. Rep. CS2005-0833, 2005.

[7] L. Fan, P. Cao, and J. Almeida, “Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol,” in *ACM SIGCOMM*, Vancouver, Canada, Aug.-Sept. 1998.

[8] U. Lee, E. Magistretti, B. Zhou, M. Gerla, P. Bellavista, and A. Corradi, “MobEyes: Smart Mobs for Urban Monitoring with a Vehicular Sensor Network,” UCLA CSD, Tech. Rep., 2006.

[9] “ns-2 (The Network Simulator),” http://www.isi.edu/nsnam/ns/.

[10] B. Zhou, K. Xu, and M. Gerla, “Group and Swarm Mobility Models for Ad Hoc Network Scenarios Using Virtual Tracks,” in *IEEE MILCOM*, Monterey, CA, USA, Oct.-Nov. 2004.

[11] J. Zhao and G. Cao, “VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks,” in *IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.

[12] H. Wu, R. Fujimoto, R. Guensler, and M. Hunter, “MDDV: a Mobility-entric Data Dissemination Algorithm for Vehicular Networks,” in *VANET’04*, Philadelphia, PA, USA, Oct. 2004.

[13] L. Wischhof, A. Ebner, and H. Rohling, “Information Dissemination in Self-organizing Intervehicle Networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, pp. 90–101, Mar. 2005.

[14] B. Xu, A. Ouksel, and O. Wolfson, “Opportunistic Resource Exchange in Inter-vehicle Ad-hoc Networks,” in *IEEE/ACM MDM’04*, Berkeley, CA, USA, Jan. 2004.

[15] M. Papadopouli and H. Schulzrinne, “Effects of Power Conservation, Wireless Coverage and Cooperation on Data Dissemination Among Mobile Devices,” in *ACM MobiHoc’01*, Long Beach, CA, USA, Oct. 2001.