Dissemination and Harvesting of Urban Data Using Vehicular Sensing Platforms

Uichin Lee, Eugenio Magistretti, Mario Gerla, Paolo Bellavista, *Senior Member, IEEE*, and Antonio Corradi, *Member, IEEE*

***Abstract*—Recent advances in vehicular communications make it possible to realize vehicular sensor networks, i.e., collaborative environments where mobile vehicles that are equipped with sensors of different nature (from toxic detectors to still/video cameras) interwork to implement monitoring applications. In particular, there is an increasing interest in proactive urban monitoring, where vehicles continuously sense events from urban streets, autonomously process sensed data (e.g., recognizing license plates), and, possibly, route messages to vehicles in their vicinity to achieve a common goal (e.g., to allow police agents to track the movements of specified cars). This challenging environment requires novel solutions with respect to those of more-traditional wireless sensor nodes. In fact, unlike conventional sensor nodes, vehicles exhibit constrained mobility, have no strict limits on processing power and storage capabilities, and host sensors that may generate sheer amounts of data, thus making already-known solutions for sensor network data reporting inapplicable. This paper describes MobEyes, which is an effective middleware that was specifically designed for proactive urban monitoring and exploits node mobility to opportunistically diffuse sensed data summaries among neighbor vehicles and to create a low-cost index to query monitoring data. We have thoroughly validated the original MobEyes protocols and demonstrated their effectiveness in terms of indexing completeness, harvesting time, and overhead. In particular, this paper includes 1) analytic models for MobEyes protocol performance and their consistency with simulation-based results, 2) evaluation of performance as a function of vehicle mobility, 3) effects of concurrent exploitation of multiple harvesting agents with single/multihop communications, 4) evaluation of network overhead and overall system stability, and 5) performance validation of MobEyes in a challenging urban tracking application where the police reconstruct the movements of a suspicious driver, e.g., by specifying the license number of a car.**

***Index Terms*—Middleware, opportunistic communications, urban monitoring, vehicular communications, vehicular sensor networks.**

Manuscript received February 21, 2007; revised October 19, 2007, April 16, 2008, and June 27, 2008. First published July 22, 2008; current version published February 17, 2009. This work was supported in part by the International Technology Alliance through the U.S. Army Research Laboratory, the U.K. Ministry of Defense under Agreement W911NF-06-3-0001, and the U.S. Army Multidisciplinary University Research Initiative under Funding W911NF0510246. The review of this paper was coordinated by Prof. X. Shen.

U. Lee and M. Gerla are with the Department of Computer Science, University of California at Los Angeles, Los Angeles, CA 90095 USA (e-mail: uclee@cs.ucla.edu; gerla@cs.ucla.edu).

E. Magistretti is with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77251-1892 USA (e-mail: emagistretti@rice.edu).

P. Bellavista and A. Corradi are with the Department of Electronics, Computer Sciences and Systems, University of Bologna, 40126 Bologna, Italy (e-mail: pbellavista@deis.unibo.it; acorradi@deis.unibo.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2008.928899

I. Introduction

VEHICULAR ad hoc networks (VANETs) are acquiring commercial relevance because of recent advances in intervehicular communications and decreasing costs of related equipment. This situation stimulates a brand new family of visionary services for vehicles, i.e., from entertainment applications to tourist/advertising information, and from driver safety to opportunistic transient connectivity to the fixed Internet infrastructure [1]–[4]. In particular, vehicular sensor networks (VSNs) are emerging as a new tool for effectively monitoring the physical world, particularly in urban areas where a high concentration of vehicles that are equipped with onboard sensors is expected [5]. Vehicles are typically not affected by strict energy constraints and can easily be equipped with powerful processing units, wireless transmitters, and sensing devices, even of some complexity, cost, and weight [e.g., Global Positioning System (GPS), chemical spill detectors, still/video cameras, vibration sensors, and acoustic detectors]. VSNs represent a significantly novel and challenging deployment scenario, which is considerably different from more traditional wireless sensor network environments, thus requiring innovative specific solutions. In fact, unlike wireless sensor nodes, vehicles usually exhibit constrained mobility patterns due to street layouts, junctions, and speed limitations. In addition, they usually have no strict limits on processing power and storage capabilities. Most importantly, they can host sensors that may generate huge amounts of data (e.g., multimedia video streams), thus making the instantaneous data reporting solutions of conventional wireless sensor networks impractical.

VSNs offer a tremendous opportunity for different large-scale applications—from traffic routing and relief to environmental monitoring and distributed surveillance. In particular, there is an increasing interest in proactive urban monitoring services where vehicles continuously sense events from urban streets, maintain sensed data in their local storage, autonomously process them (e.g., recognizing license plates), and possibly route messages to vehicles in their vicinity to achieve a common goal (e.g., to allow police agents to track the movements of specified cars). For instance, proactive urban monitoring could usefully apply to *post facto* crime scene investigation. Reflecting on tragedies such as the 9/11 and London bombings, VSNs could have actually helped emergency recovery and forensic investigation/criminal apprehension. In the London bombings, police agents tracked some of the suspects in the subway by using closed-circuit TV cameras, but they had a hard time finding helpful evidence from the double-decker bus; this

scenario has motivated the installation of more cameras in fixed locations along London streets. VSNs could be an excellent complement to the deployment of fixed cameras/sensors. The completely distributed and opportunistic cooperation among sensor-equipped vehicles has the "deterrent" effect of making it harder for potential attackers to disable surveillance. Another less sensational but relevant example is the need to track the movements of a car that was used for a bank robbery to identify thieves, for example. It is highly probable that some vehicles have spotted the unusual behavior of thieves' car in the hours before the robbery and might be able to identify the threat by "opportunistic" correlation of their data with other vehicles in the neighborhood. It would be much more difficult for the police to extract that information from the massive number of multimedia streams that were recorded by fixed cameras. As for privacy, let us briefly note that people are willing to sacrifice privacy and accept a reasonable level of surveillance when the data can be collected and processed only by recognized authorities (i.e., with a court order) for forensic purposes and/or for counteracting terrorism and common crimes.

As shown by the aforementioned examples, the reconstruction of a crime and, more generally, the forensic investigation of an event that was monitored by VSNs require the collection, storage, and retrieval of massive amounts of sensed data. This condition is a major departure from conventional sensor network operations, where data is dispatched to "sinks" under predefined conditions such as alarm thresholds. Obviously, it is impossible to deliver all the streaming data that were collected by video sensors to a police authority sink because of sheer volume. Moreover, input filtering is not possible, because nobody knows a priori which data will be of use for future investigations. The problem becomes one of searching for sensed data in a massive mobile opportunistically collected and completely decentralized storage. The challenge is to find a completely decentralized VSN solution, with low impact on other services, good scalability (up to thousands of nodes), and tolerance of disruption that was caused by mobility and attacks.

For that purpose, we have designed and implemented MobEyes, a novel middleware that supports VSN-based proactive urban monitoring applications. MobEyes exploits wireless-enabled vehicles that are equipped with video cameras and a variety of sensors to perform event sensing, processing/classification of sensed data, and intervehicle ad hoc message routing. It is impossible to directly report the sheer amount of sensed data to the authority, so MobEyes keeps sensed data in mobile node storage, and onboard processing capabilities are used to extract features of interest (e.g., license plates). Mobile nodes periodically generate data summaries with extracted features and context information such as timestamps and positioning coordinates, whereas mobile agents (e.g., police patrolling cars) move and opportunistically harvest summaries as needed from neighbor vehicles. MobEyes adopts VSN custom-designed protocols for summary diffusion/harvesting that exploits intrinsic vehicle mobility and simple single-hop intervehicle communications. That way, MobEyes harvesting agents can create a low-cost opportunistic index to query the distributed sensed data storage, thus enabling us to answer several questions: Which vehicles were in a given place at a given time? Which route did a certain vehicle take in a given time interval? Which vehicle collected and stored the data of interest?

In this paper, we make the following contributions.

- We define a vehicular sensing platform and propose the MobEyes vehicular sensing architecture. We synthesize the existing techniques to build a MobEyes system that satisfies the key design principles: 1) disruption tolerance; 2) scalability; and 3) nonintrusiveness.
- We propose an analytic model that can accurately predict MobEyes' performance and formally show that our protocol is scalable and nonintrusive.
- We provide extensive simulation results as follows: 1) evaluation of performance dependence on vehicle mobility models; 2) effects of the concurrent exploitation of multiple harvesting agents with single/multihop communications; 3) evaluation of the network overhead and overall system stability; and 4) performance validation of MobEyes in a challenging urban tracking application where the police obtain the route, followed by a car, by simply specifying its plate number.
- We provide an overview of the primary security requirements that stem from VSN-based proactive urban monitoring applications and show how they can be addressed via state-of-the-art solutions in the literature. MobEyes integrates these solutions and allows for enabling/disabling them at deployment time, depending on the required degree of security/privacy.

The rest of this paper is organized as follows. Section II describes background and related paper, by positioning the original MobEyes contributions. Section III presents the overall MobEyes architecture, whereas Section IV details our original protocols for opportunistic summary diffusion/harvesting. Section V analytically models the performance of MobEyes protocols, which are extensively evaluated via simulations in Section VI. Section VII gives a rapid overview of security/privacy issues and related solutions. Section VIII concludes this paper.

II. Background and Related Work

The idea of embedding sensors in vehicles is very novel. To our knowledge, the only research project that deals with the MobEyes-addressed challenging issues of car-based sensing and distributed opportunistic search of sensed data is the Massachusetts Institute of Technology's CarTel [6], [7]. In CarTel, users submit their queries about sensed data on a portal that is hosted on the wired Internet. Then, an intermittently connected database is in charge of dispatching "queries" to vehicles and of receiving replies when vehicles move near open access points to the Internet. Other interesting research projects have focused on providing mobile Internet access to vehicles. For instance, InternetCar aims at providing vehicles with seamless Internet connectivity by envisioning various applications such as a traffic information dissemination service where "raw" sensed data from vehicles are collected in a central server, and traffic information is distributed to the drivers [8], [9]. Unlike CarTel and InternetCar, MobEyes exploits mobile collector agents instead of relying on the wired Internet infrastructure,

thus improving robustness. In addition, note that the FleetNet project, which aims at developing an intervehicle communication platform for vehicular applications, recognized the potential of services that distribute location-tagged information (e.g., traffic jam warning) by collecting and processing data from cars in a distributed fashion [10], [11]. In this paper, we originally propose how one can design and implement such applications using VSNs.

Related work has recently emerged in two other related fields: 1) VANET and 2) "opportunistic" sensor networks. For full understanding of the MobEyes proposal, we provide a brief illustration of how Bloom filters work and of their exploitation in networking fields.

A. VANET

Recent research has been envisioning a large number of applications that are specifically designed for VANET, including 1) safe cooperative driving where emergency information is diffused to neighbor vehicles, and real-time response is required to avoid accidents [3]; 2) entertainment support, e.g., content sharing [1], advertisements [2], and peer-to-peer (P2P) marketing [12]; and 3) distributed data collection, e.g., parking lot [13] and traffic congestion information [14]. So far, however, most VANET research has focused on routing issues. Several VANET applications, e.g., related to safety or traffic/commercial advertising, call for the delivery of messages to all nodes that are close to the sender, with high delivery rate and short delay. Recent research has addressed this issue by proposing original broadcast strategies [3], [15]. However, single-hop broadcast does not provide full support to advertising applications; effective multihop dissemination solutions should be also investigated [16].

Packet delivery issues in areas with sparse vehicles have stimulated several recent research contributions to investigate carry-and-forward strategies. In [17], the authors simulate a straight highway scenario to compare two ideal strategies: 1) pessimistic (i.e., synchronous), where sources send packets to destinations only as soon as a multihop path is available and 2) optimistic (i.e., carry-and-forward), where intermediate nodes hold packets until a neighbor closer to the destination is detected. Under the implicit assumptions of 1) unbounded message buffers and bandwidth and 2) easily predictable mobility patterns as for vehicles on a highway, the optimistic scenario has demonstrated to achieve a lower delivery delay. However, in more realistic situations, carry-and-forward protocols call for careful design and tuning. MaxProp [18], which is part of the University of Massachusetts DieselNet Project [19], is a ranking strategy for determining packet delivery order when node encounters occasionally occur, as well as dropping priorities in the case of full buffers. Precedence is given to packets that are destined to the other party, then to routing information, to acknowledgements, to packets with small hop-counts, and, finally, to packets with a high probability of being delivered through the other party. Vehicle-assisted data delivery [20] rests on the assumption that most node encounters happen in intersection areas. Effective decision strategies are proposed, which highly reduce packet delivery failures and delay.

Applications for distributed data collection in VANETs call for geographic dissemination strategies that deliver packets to all nodes that belong to target remote areas, despite possibly interrupted paths [14], [21]. MDDV [21] exploits geographic forwarding to the destination region, favoring paths where vehicle density is higher. In MDDV, messages are carried by head vehicles, i.e., best positioned toward the destination with respect to their neighbors. As an alternative, [14] proposes several strategies based on virtual potential fields that were generated by propagation functions: Any node estimates its position in the field and retransmits packets until nodes placed in locations with lower potential values are found. This procedure is repeated until minima target zones are detected.

B. Opportunistic Sensor Networking

Traditionally, sensor networks have been deployed in static environments, with application-specific monitoring tasks. Recently, opportunistic sensor networks have emerged, which exploit existing devices and sensors, such as cameras in mobile phones [22]–[25]. Several of these networks are relevant to our research, because they can easily implement opportunistic dissemination protocols [26], [27].

Dartmouth's MetroSense [22], [28] is closely related to MobEyes. [22] describes a three-tier architecture for MetroSense: 1) Servers in the wired Internet are in charge of storing/processing sensed data; 2) Internet-connected stationary sensor access points (SAPs) act as gateways between servers and mobile sensors (MSs); 3) MSs move in the field, opportunistically delegating tasks to each other and "muling" [29], [30] data to SAP. MetroSense requires infrastructure support, including Internet-connected servers and remotely deployed SAP. Similarly, Wang *et al.* proposed data delivery schemes in Delay/Fault-Tolerant Mobile Sensor Network for human-oriented pervasive information gathering [31]. The tradeoff between data delivery ratio/delay and replication overhead is mainly investigated in the case of buffer and energy resource constraints. In contrast, MobEyes does not require any fixed infrastructure by using mobile sinks (or agents) and addresses VANET-specific deployment scenarios (e.g., powerful sensing platforms and distributed index collection).

Application-level protocols for the resolution of queries to sensed data have been proposed in [23] and [24]. Contory abstracts the network as a database and can resolve declarative queries. Spatial programming hides remote resources such as nodes under local variables, thus enabling transparent access. Migratory Services are components that react to changing context, e.g., the target that moves out of range by migrating to other nodes [23]. [24] presents VITP, a query-response protocol to obtain traffic-related information from remote areas: The primary idea is that the source specifies the target area when injecting a query in the environment, and nodes in the target area form a virtual ad hoc query server.

Among recent research projects about opportunistic sensing, we mention Intel IrisNet [32] and Microsoft SenseWeb [33]. Both projects investigate the integration of heterogeneous sensing platforms in the Internet via a common data publishing architecture. We also point out the Center for Embedded Network

Sensing's Urban Sensing Project [25], [34], which is a recently started multidisciplinary project that addresses "participatory" sensing, where urban monitoring applications receive data from MSs operated by people.

With regard to dissemination of sensed data through peers, we can mention two solutions for monitoring wildlife habitats: 1) ZebraNet [26] and 2) SWIM [27]. ZebraNet addresses remote wildlife tracking, e.g., zebras in the Mpala Research Center in Kenya, by equipping animals with collars that embed wireless communication devices, GPS, and biometric sensors. As GPS-equipped animals drift within the park, their collars opportunistically exchange sensed data, which must make its way to the base station (the ranger's truck). ZebraNet proposes two dissemination protocols: 1) a flooding-based approach where zebras exchange all the data within their buffers (either locally generated or received from other animals) with neighbors and 2) a history-based protocol where data is uploaded only to zebras with a good track record of base station encounters. SWIM [27] addresses sparse MS networks with fixed Infostations as collecting points. Sensed data is epidemically disseminated via single-hop flooding to encountered nodes and is offloaded when Infostations are within reach.

Although all the aforementioned schemes implement P2P dissemination, none fits the performance requirements of the MobEyes target scenario. Flooding generates excessive overhead, whereas history-based is ineffective in the very dynamic VANET where the base station (the agent's vehicle) rapidly moves without following a specific mobility pattern [26]. In addition, MobEyes nodes do not transmit raw collected data but, due the abundance of on board computing resources, they locally process data and relay only short summaries. That advantage is relevant, because data that were collected in VSNs, e.g., from video cameras, may be order-of-magnitude larger than data sensed in naturalistic scenarios.

C. Bloom Filter and Its Applications

Bloom filter [35] is a space-efficient randomized data structure for representing a set and is mainly used for membership checking. A Bloom filter for representing a set of ω elements, $S = \{s_1, s_2, \ldots, s_\omega\}$, consists of m bits, which are initially set to 0. The filter uses ℓ independent random hash functions $h_1, \ldots, h_\ell$ within m bits. By applying these hash functions, the filter records the presence of each element into the m bits by setting ℓ corresponding bits. To check the membership of the element x, it is sufficient to verify whether all $h_i(x)$ are set to 1. Although membership checking in a Bloom filter is probabilistic, and false positives are possible, it has widely been used for applications where "space saving" outweighs the drawback of errors.

Bloom filters have gained momentum in networking fields due to their space efficiency, thus reducing the amount of transmitted data and, consequently, saving energy. Readers can find a survey of Bloom filter-based networking applications in [36]. Bloom filters are mostly used for simple membership checking [37], [38], set difference (or reconciliation) [39], and set intersection [40]. Fan *et al.* proposed a summary cache where a set of distributed Web proxies use Bloom filters to disseminate the content of their cache [37]. Similarly, Ye *et al.* exploited them to store MAC addresses for simple membership checking in wireless sensor networks [38]. For set reconciliation, Byers *et al.* proposed a method that uses a comparison tree over a Bloom filter [39], which allows for faster search of elements in the difference set (when difference sets are small). To find set intersection without transmission entire sets, Reynolds *et al.* used Bloom filters for keyword search in a P2P overlay network [40].

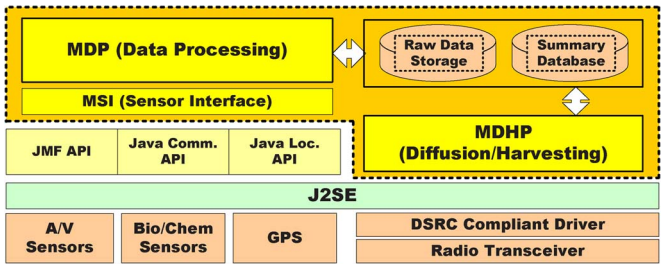


Fig. 1. MobEyes sensor node architecture.

III. MobEyes Architecture

For clarity, we present the MobEyes solution using one of its possible practical application scenarios, i.e., collecting information from MobEyes-enabled vehicles about criminals that spread poisonous chemicals in a particular section of the city (e.g., a subway station). We suspect that the criminals have used vehicles for the attack. Thus, MobEyes will help detect the vehicles and permit tracking and capture. Here, we assume that vehicles are equipped with cameras and chemical detection sensors. Vehicles continuously generate a huge amount of sensed data, store it locally, and periodically produce short *summary chunks* that were obtained by processing sensed data, e.g., license plate numbers or aggregated chemical readings. Summary chunks are aggregated in *summaries* that are opportunistically disseminated to neighbor vehicles, thus enabling metadata harvesting by the police to create a distributed metadata index, which is useful for forensic purposes, e.g., crime scene reconstruction and criminal tracking.

To support all the aforementioned tasks, we have developed MobEyes according to the component-based architecture, as depicted in Fig. 1. The key component is the MobEyes Diffusion/Harvesting Processor (MDHP), which will be discussed in detail in the next section. MDHP works by opportunistically disseminating/harvesting summaries produced by the MobEyes Data Processor (MDP), which accesses sensor data via the MobEyes Sensor Interface (MSI). Vehicles are not strictly resource constrained, so our MobEyes prototype is built on top of the Java Standard Edition virtual machine. MDP is in charge of reading raw sensed data (via MSI), processing it, and generating chunks. Chunks include metadata (e.g., vehicle position, timestamp, vehicle ID number, and possible additional context such as simultaneous sensor alerts) and features of interest that were extracted by local filters (see Fig. 2). For instance, in the aforementioned application scenario, MDP includes a filter determining license plate numbers from multimedia flows taken by cameras [41]. Finally, MDP

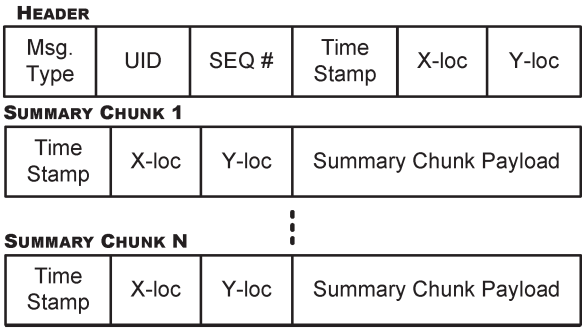


Fig. 2. Packet format. A single summary packet contains multiple *summary chunks*.

commands the storage of both raw data and chunks in two local databases. MDHP disseminates/harvests summaries by packing a set of chunks into a single packet for effectiveness. Therefore, the generation rate and the size of chunks and summaries are relevant to MobEyes' performance. Additional details about the design and implementation of the MobEyes prototype are out of the scope of this paper and can be found in [42].

Developers of MobEyes-based applications can specify the desired generation rate as a function of vehicle speed and expected vehicle density. The chunk size mainly depends on application-specific requirements: in the scenario under consideration, each recognized license plate is represented with 6 B, sensed data with 10 B (e.g., concentrations of potential toxic agents), timestamp with 2 B, and vehicle location with 5 B. Then, in our scenario, MDP can pack 65 chunks in a single 1500-B summary, even without exploiting any data aggregation or encoding technique. In usual deployment environments, chunks are generated every [2, 10] s; thus, a single summary can include all the chunks about a [2, 10]-minute interval. MSI permits MDHP to access raw sensed data, independent of actual sensor implementation, thus simplifying the integration with many different types of sensors. MSI currently implements methods for accessing camera streaming outputs, serial port I/O streams, and GPS information by only specifying a high-level name for the target sensor. To interface with sensor implementations, MSI exploits well-known standard specifications to achieve high portability and openness: the Java Media Framework (JMF) application programming interface (API), the Sun Communication API, and the JSR179 Location API.

IV. MobEyes Diffusion/Harvesting Processor

In this section, we review the design principles of MDHP protocols, i.e., disruption tolerance, scalability, and nonintrusiveness. Private vehicles (i.e., regular nodes) opportunistically and autonomously spread summaries of sensed data by exploiting their mobility and occasional encounters. Police agents (i.e., authority nodes) proactively build a low-cost distributed index of the mobile storage of sensed data. The main goal of the MDHP process is to create a highly distributed and scalable index that allows police agents to place queries into the huge urban monitoring database without even trying to combine this index in a centralized location.

A. MDHP Protocol Design Principles

A vehicular sensing platform, which is built on top of a VANET, has the following specific characteristics that differentiate it from more established and investigated deployment scenarios. First, it has unique mobility patterns. Vehicles move at a relatively high speed (e.g., up to 80 mi/h) on roads that may have multiple lanes and different speed limits. Instead of random motion patterns, drivers navigate a set of interest points (e.g., home and work place) by following their preferred paths. The dynamic behavior of mobile nodes, i.e., join/leave/failure, usually results in modifications of the set of participating nodes (called churning). Moreover, there are time-of-the-day effects such that the overall volume of vehicles changes over time, e.g., high density during rush hours or some special event. Thus, the spatial distribution of vehicles is variable and nonuniform, and in some cases, the network can be partitioned. As a result, vehicles may experience disruptions and intermittent connectivity. Second, the network scales up to hundred thousands of vehicles, because sensing applications primarily target urban environments. Third, unlike conventional sensor networks where the communication channel is dedicated to sensing nodes, the primary purpose of vehicular communications is for safety navigation, and sensing platforms cannot fully utilize the overall available bandwidth.

Under these circumstances, we have decided to consider the following design principles for MDHP protocols.

- *Disruption tolerance*. It is crucial that MDHP protocols can operate, even with disruptions (caused by sparse network connectivity, obstacles, and nonuniform vehicle distribution) and with arbitrary delays. High churning of vehicles must be considered; for robustness, data replication is a must.
- *Scalability*. MDHP protocols must scale up to hundred thousands of nodes (e.g., the number of vehicles that potentially interwork in a large city).
- *Nonintrusiveness*. Intrusive protocols may cause severe contention with safety applications and could deter reliable propagation of important messages in a timely fashion. MDHP protocols should not disturb other safety applications, and limiting the use of bandwidth below a certain threshold is imperative.

Given the aforementioned motivations and the deriving design principles, simple flooding and probabilistic gossiping cannot be used for MDHP. In fact, they require the network to be fully connected (i.e., nondelay tolerant) and cause the network traffic to scale with the number of nodes in the network (i.e., nonscalable and intrusive). For instance, in Epidemic Data Dissemination (EDD), where data is spread whenever connectivity is available (i.e., the data is replicated without any restriction), the size of exchanged data scales with the network size; thus, EDD is intrusive and nonscalable. EDD is more suitable for sparse small-scale wireless networks. Note that the formal analysis that corroborates this sketched observation can be found in Section V-B. Unlike these approaches, we propose to use "mobility-assist" information dissemination and harvesting in MobEyes. Data are replicated via periodic "single-hop" broadcasting (i.e., only the data originator can

broadcast its data) for a given period of time. Through the mobility of carriers, the data will be delivered to a set of harvesting agents. Mobility-assist dissemination and harvesting per se are delay and disruption tolerant, and, as extensively detailed in the following, single-hop broadcasting-based localized information exchange makes our protocols nonintrusive and scalable.

B. Summary Diffusion

By following the aforementioned guidelines, in MobEyes, any regular node periodically advertises a new packet with generated summaries to its current neighbors to increase the opportunities for agents to harvest summaries. Clearly, excessive advertising will introduce too much overhead (as in EDD), whereas no advertising at all (i.e., direct contact) will introduce unacceptable delays, as agents will need to directly contact each individual source of monitoring information to complete the harvesting process. Thus, MobEyes tries to trade off delivery latency with advertisement overhead. As depicted in Fig. 2, a packet header includes the packet type, generator ID, locally unique sequence number, packet generation timestamp, and generator's current position. Each packet is uniquely identified by a $\langle$generator ID, sequence number$\rangle$ pair and contains a set of summaries that are locally generated during a fixed time interval.[1]

Neighbor nodes that receive a packet store it in their local summary databases. Therefore, depending on the mobility and the encounters of regular nodes, packets are opportunistically diffused into the network (i.e., *passive* diffusion). MobEyes can be configured to perform either single-hop passive diffusion (i.e., only the original source of the data advertises its packet to current single-hop neighbors) or k-hop *passive* diffusion (i.e., the packet travels up to k-hop as it is forwarded by j-hop neighbors with $j < k$). Other diffusion strategies could easily be included in MobEyes. One strategy is single-hop active diffusion, where any node periodically advertises all packets (generated by itself and received from others) in its local summary databases, at the expense of a greater traffic overhead. As detailed in the experimental evaluation section, in a usual urban VANET (node mobility restricted by roads), it is sufficient for MobEyes to exploit the lightweight k-hop passive diffusion strategy with very small k values to achieve the desired diffusion levels.

Fig. 3 depicts the case of two sensor nodes C1 and C2 that interact with other sensor nodes while moving (the radio range is represented as a dotted circle). For ease of explanation, we assume that there is only a single encounter, but in reality, any nodes within dotted circle are considered encounters. In the figure, a black triangle with a timestamp represents an encounter. According to the MobEyes summary diffusion protocol, C1 and C2 periodically advertise a new summary packet $S_{C1,1}$ and $S_{C2,1}$, respectively, where the subscript denotes $\langle ID, Seq.\# \rangle$.

[1]The optimal interval can be determined by noting that the harvesting time distribution is characterized by the verage (μ) and standard deviation (ρ). Then, the Chebyshev inequality $P(|x - \mu| \geq k\rho) \leq 1/k^2$ allows us to choose k such that we can guarantee the needed *harvesting latency*, thus fixing the period as $\mu + k\rho$. See related details in Section V.

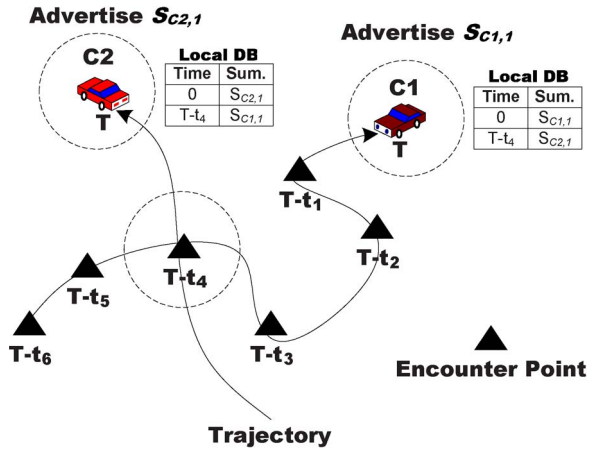


Fig. 3. MobEyes single-hop passive diffusion.

At time $T - t_4$, C2 encounters C1, and thus, they exchange those packets. As a result, C1 carries $S_{C2,1}$, and C2 carries $S_{C1,1}$.

Summary diffusion is time and location sensitive (spatial-temporal information diffusion). In fact, regular nodes keep track of the freshness of summary packets by using a sliding window with a maximum window size (i.e., fixed expiration time). In addition, a single summary packet may contain multiple summaries, so we define "aggregate" packet sensing location as the average of the sensing locations of all summaries in the packet. When a packet expires or the packet originator moves away more than a threshold distance from the aggregate packet sensing location, the packet is automatically disposed. The expiration time and the maximum distance are system parameters that should be configured, depending on urban monitoring application requirements. Let us also briefly note that summaries always include, of course, the time and location where the sample was taken. Upon receiving an advertisement, neighbor nodes keep the encounter information (the advertiser's current position and current timestamp). This also allows MobEyes nodes, when the type of urban monitoring applications makes it applicable, to exploit spatial-temporal routing techniques such as last encounter routing [43] and to maintain a georeference service for proactively accessing the data, which is obtained as a simple byproduct of summary dissemination, without additional costs.

C. Summary Harvesting

In parallel with diffusion, MobEyes summary harvesting takes place. There are two possible modes of harvesting "diffused" information: 1) the *on-demand* mode and 2) the *proactive* (or background) mode. The on-demand mode is suitable for cases when the police agents react to an emergency call, e.g., the previously mentioned poisonous gas incident. Police agents will converge to the outskirts of the area (keeping a safe distance of course) and will query vehicles for summaries that correspond to a given time interval and area (i.e., a time–space window). The agents can *flood* a query with such information (i.e., like a route request in on-demand routing). Each regular node resolves the query and returns its summary

to the agents (i.e., like a route reply in on-demand routing). Therefore, the *on-demand* strategy is more likely a traditional sensor network-based data harvesting protocol, e.g., Directed Diffusion [44]—the reply is "diffused" in the direction of the querier. The main difference in MobEyes would be that a query has a spatiotemporal range. The police agents, as a team, will collect as many summaries of interest that they can.

However, it is not very practical to exploit an on-demand strategy in MobEyes due to the following reasons. First, agents should provide a query with the *range* of spatiotemporal information, even in the usual cases when they have no precise prior information. An improperly chosen query range may require accessing a large number of vehicles (e.g., for a given chemical attack that happened in a busy street, the police may want to find out *all* the vehicles that pass by the scene within the last several hours, which results in thousands of vehicles). Second, the on-demand scheme is quite similar to conventional data harvesting and requires maintaining a "concast" tree from the query originator. However, the number of vehicles is expected to be very large in MobEyes, and vehicles are mobile; thus, route management would have relevant implementation costs in terms of overhead. Third, collecting a complete set of summaries would be nontrivial and not always possible due to intermittent connectivity and network partitions. To overcome intermittence, a query/response can opportunistically be disseminated in a delay-tolerant network style. However, in such a case, the delay will be comparable to "proactive" harvesting, with the additional cost of a separate dissemination process. Fourth, in "covert" operations, agents may not want to broadcast a query (e.g., so that they will not alert the criminals that are currently being pursued). Then, the police must consider physically dispatching agents to the location of interest and collect summaries via physical contacts. In summary, this on-demand "mechanical" search will be extremely costly and potentially very time consuming.

To overcome such issues, we are proposing a "proactive" version of the search based on distributed index construction. That is, in each area, there are agent vehicles that collect all the summaries as a background process and create a distributed index. In this case, there is no time–space window concern during collection. The only requirement is to collect all the summaries in a particular area. Now, for specific information (e.g., the poisonous gas level monitoring), the query is directed to the target regular vehicles by exploiting the agents' distributed index. The time–space window concept can be applied to the "index" to find the vehicles in a particular place and time and then pursue the hot leads. For example, upon receiving a specific query, the agents collectively examine the "index," find a match, and decide to inspect, in more detail, the video files that were collected by a "limited" number of vehicles. The vehicles can be contacted based on the originator's vehicle ID number that is stored in each summary. A message is sent to each vehicle, which requests it to upload the file at the nearest police access point. Note that the request message can exploit georouting by either exploiting the Geo Location Service that maps the vehicle ID to the current vehicle location or using "Last Encounter Routing" techniques [43], [45]. The latter technique is particularly convenient here, because nodes memorize the time and place of encounters at the time that summary exchanges take place.

After the desired summaries have been found, both on-demand and proactive processes require contacting the cars under consideration. However, the proactive approach is much more powerful, because it can considerably speed up the search. For instance, if the inspection of the information that was collected in the crime area indicates a possible escape direction by the terrorists, one can immediately search again the proactively created index for a new time–space window without having another time-consuming collection of summaries from vehicles. However, maintaining that index is costly, because agent resources must be dedicated to the task.

In the sequel, we will assume proactive index construction. Thus, the agents indiscriminately collect all summaries. There is no loss of generality, however, because the procedure will also allow on-demand index construction for a specific time–space request. In fact, the only difference between the two harvesting schemes is the size of the set that is being harvested. In the on-demand scheme, the target set is a specific time–space window. In the proactive scheme, the target set is the entire geographic area within agent responsibility. There is no limit on the harvesting time, although old records are timed out.

By considering the proactive (or background) harvesting model, the MobEyes police agent collects summaries from regular nodes by periodically querying its neighbors. The goal is to collect all the summaries that were generated in the specified area. Obviously, a police node is interested in harvesting only summary packets that it has not collected so far. To focus only on missing packets, a MobEyes authority node compares its list of summary packets with that of each neighbor (i.e., a set difference problem) by exploiting a space-efficient data structure for membership checking (i.e., a Bloom filter).[2] A MobEyes police agent uses a Bloom filter to represent its set of already-harvested still-valid summary packets. Each summary has a $\langle$unique node ID, sequence number$\rangle$ pair, so we use it as an input for the hash functions. The MobEyes harvesting procedure consists of the following steps.

1) The agent broadcasts a "harvest" request message with its Bloom filter.
2) Each neighbor prepares a list of "missing" packets based on the received Bloom filter.
3) One of the neighbors returns missing packets to the agent.
4) The agent sends back an acknowledgment with a piggy-backed list of returned packets and, upon listening to or overhearing this message, neighbors update their lists of missing packets.
5) Steps 3 and 4 are repeated until all missing packets are sent.

An example of summary harvesting is shown in Fig. 4. The agent first broadcasts its Bloom filter that is related to the packets that have been collected so far (i.e., P2, P4, P6, P7, P9, and P10), as shown in Fig. 4(a). Each neighbor receives the filter and creates a list of missing packets. For example, C3 has P3 and P8 to be returned, whereas C4 has P1 and P8.

[2]See also Section II.

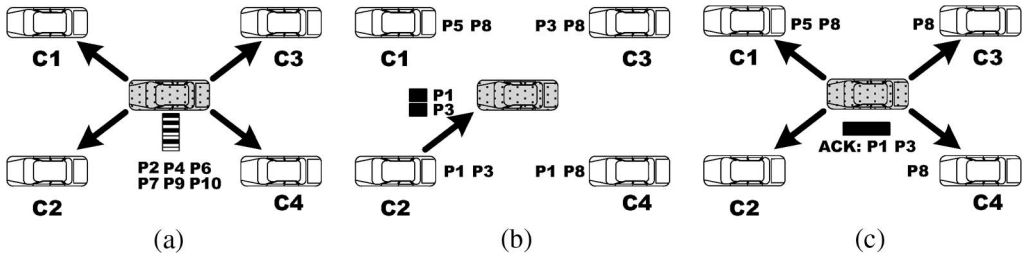


Fig. 4. MobEyes proactive summary harvesting. (a) Broadcast a harvest request. (b) C2 first returns missing packets. (c) Broadcast acknowledgement.

In Fig. 4(b), C2 is the first node to return missing packets (P1, P3), and the agent sends back an acknowledgement that is piggybacked with the list of received packets. Neighbor nodes overhear the message and update their lists: C3 and C4 both remove P1 and P3 from their lists, as depicted in Fig. 4(c).

Note that membership checking in a Bloom filter is probabilistic, and false positives are possible, even if they are rare (as rapidly shown in the following section). In Fig. 4(b), for example, a false positive on P1 makes C2 return only P3. None of the neighbors can send P1. To deal with this problem, the agent periodically changes the set of hash functions. Suppose that we use m hash functions. Each hash function is a pseudorandom function (PRF), where a PRF takes two arguments, X_k is the key $(\mathrm{k} = 1, 2, \ldots, m)$, and i is the input value, which produces an output value $o = F_{X_k}(i)$. All nodes are initially given the same set of keys X_k, where $\mathrm{k} = 1, 2, \ldots, m$. For periodic changes, the key for the kth hash function in the nth epoch X_k^n can be calculated by hashing the initial value X_k^n n times. The Bloom filter contains the epoch number, which allows the neighbors to find the set of keys for m hash functions. Even with failure, by periodically incrementing the epoch number, the agent can gather the missing packets. In addition, note that, in our application, a set changes over time, with summaries being inserted and deleted, because summaries have spatiotemporal properties. For deletion operations, Fan *et al.* introduced the idea of counting Bloom filters, where each entry in the Bloom filter is not a single bit but rather a small counter [37]. When an item is inserted, the corresponding counters are incremented. When an item is deleted, the corresponding counters are decremented. For actual filter transfer, instead of sending the full counting Bloom filter, each counter is represented as a single bit. That is, the counter is 1 if its value is greater than 0; otherwise, it is 0.

For simplicity, so far, we assumed that there is a single agent that works to harvest summaries. In fact, MobEyes can handle concurrent harvesting by multiple agents (possibly several hops apart) that can cooperate by exchanging their Bloom filters among multihop routing paths; thus, this step creates a distributed and partially replicated index of the sensed data storage. In particular, whenever an agent harvests a set of j new summary packets, it broadcasts its Bloom filter to other agents, with benefits in terms of latency and accuracy, as shown in the following sections. Note that strategically controlling the trajectory of police agents, properly scheduling Bloom filter updates, and efficiently accessing the partitioned and partially replicated index are part of our future work. In the following section, we focus on the primary goal of identifying the tradeoffs between dissemination and harvesting in a single geographic area and the dependence of MobEyes' performance on various parameters. We also analyze the traffic overhead that was created by diffusion/harvesting and show that it can scale well to very large node numbers.

V. Delay and Scalability Analysis

In this section, to evaluate and validate the effectiveness of the MobEyes protocols, we present analytic results about summary diffusion/harvesting and scalability.

A. Summary Harvesting Delay

In MobEyes, regular nodes receive summaries from their neighbors (passive harvesting), and these summaries are harvested by police agents (active harvesting). Obviously, the effectiveness of active harvesting also depends on how extensive the passive harvesting was. Therefore, we model the progress of passive harvesting, from which we formulate the progress of active harvesting. We extend the model to analyze k-hop relay scope. We assume that there are N nodes that move within an $L \times L$ m^2 area, and each node advertises a single summary packet (i.e., a total of N summary packets). For analysis, we use two different mobility models, both with uniform spatial node distribution: 1) the random-direction model and 2) the Manhattan mobility model. In the random-direction mobility model, nodes move toward random directions (chosen out of $[0, 2\pi]$) at a speed of v on the average. The Manhattan mobility model restricts node mobility patterns along grids. Because a target direction is uniformly chosen in both cases, the steady-state node distribution is uniform [5], [46]. Thus, the node density is *location independent*. The Manhattan mobility model exhibits uniform node distribution, because nodes are uniformly distributed in the area where mobile nodes can go along. Therefore, the node density can simply be expressed as $\rho = \delta N/L^2$, where δ is used to control the size of the area that is covered by mobile nodes in the Manhattan mobility model, and $\delta = 1$ in the random-direction mobility model. The value δ for the Manhattan mobility model is a function of grid layout and communication range R, because node movement patterns are restricted to the exploited grids (i.e., the covered area size C is smaller than L^2). Thus, we have $\delta = L^2/C$. For instance, given an $L \times L$ area, the mobility pattern is restricted to a single

strip. The covered area size is $2R \times L$ (where $2R$ is the radio range of mobile nodes), instead of L^2; thus, $\delta = L \times L/2R \times L = L/2R$. Let v^* denote the average relative speed of nodes. As shown in [47], $v^* = v/2\pi \int_0^{2\pi} \sqrt{(1+\cos\theta)^2 + \sin^2\theta} d\theta = 1.27$ v. In general, we simply assume that the average relative speed can be modeled to be proportional to the average speed, i.e., $v^* = cv$, where c is a constant.

By extending [48], we now develop a deterministic discrete-time model. Let us first reason on how many summaries a node can receive for a given time slot. For ease of exposition, we assume that all nodes are static, except for one regular node. This node randomly moves and collects summaries by passively listening to advertisements from encountered nodes. In this case, the node (or the passive harvester) behaves just as a data mule in traditional sensor networks [29]. During time slot Δt, a regular node travels a distance $r = v\Delta t$, and the covered area size is $v\Delta t 2R$, where R is the radio range. The expected number of encountered nodes in this area is simply $\alpha = \rho v \Delta t 2R$. Each of these nodes will advertise its summaries, so the regular node will receive α summaries. The dual scenario is when all nodes are mobile, but the passive harvesting node is static. Without loss of generality, if all nodes are mobile, we can simply replace the average speed with the average relative speed: $\alpha = \rho v^* \Delta t 2R$ where v^* is the average relative speed.[3]

Given α, we can estimate the progress of passive harvesting as follows. Let E_t denote the number of distinct summaries that were collected by a regular node by time slot t. As we have previously described, during time slot Δt, a regular node will receive α summaries. The node has E_t summaries, so the probability that the received summary is new is simply $1 - E_t/N$. Thus, the expected number of new summaries out of α is given as $\alpha(1 - E_t/N)$. It is obvious that restricted movement patterns (e.g., two nodes that move together along the same path in the Manhattan mobility model) will affect the *effective* number of neighbors. We are interested in the average behavior, and we can model this by simply multiplying α with a constant compensation factor η. Therefore, we have the following relationship:

$$E_t - E_{t-1} = \alpha\eta \left(1 - \frac{E_{t-1}}{N}\right). \tag{1}$$

Equation (1) is a standard difference equation with the following solution:

$$E_t = N - (N - \alpha\eta)\left(1 - \frac{\alpha\eta}{N}\right)^t. \tag{2}$$

Equation (2) tells us that the distinct number of collected summaries is geometrically increasing. As time tends to infinity, $E_t = N$. Let us define a random variable T to denote the time that a regular node encounters any random node, thus receiving a summary from it. The cumulative distribution of random variable T can be derived by dividing (2) by N, i.e.,

$$F_T(t) = 1 - \left(1 - \frac{\alpha\eta}{N}\right)^{t+1}. \tag{3}$$

Then, we can derive the probability mass function $f_T(t)$ as follows:

$$f_T(t) = \frac{\alpha\eta}{N}\left(1 - \frac{\alpha\eta}{N}\right)^t. \tag{4}$$

Equation (4) is a *modified* geometric distribution with success probability $p = \alpha\eta/N$. The average is given as $\mathbb{E}[T] = 1/p - 1 = N/\alpha\eta - 1$. Since $\alpha = \rho v^* \Delta t 2R$, by replacing $\rho = \delta N/L^2$, we have $\alpha = N/L^2 v^* \Delta t 2R$. Thus, we have

$$\mathbb{E}[T] = \frac{N}{\alpha\eta} - 1 = \frac{L^2}{\delta v^* \Delta t 2R\eta} - 1. \tag{5}$$

As shown in (5), given a square area of L^2, the average time for a regular node to collect a summary is independent of the node density. In fact, it is a function of the average relative speed and communication range. Intuitively, as the node density increases (i.e., N increases), a node can collect more summaries during a given time slot. However, higher density means that the total number of summaries to be collected is higher. Thus, the two factors compensate for each other.

Unlike regular nodes, the agent actively harvests summaries from its neighbors. Every node moves randomly and starts passive harvesting at time 0, so it is expected that every node has the same number of summaries that were collected by time t (E_t). Therefore, the probability that a neighbor node does not have a random summary is given as $1 - E_t/N$. The probability that none of the $\alpha\eta$ neighbors has a summary is simply $(1 - E_t/N)^{\alpha\eta}$. The probability that at least one of the neighbors has a random summary is $1 - (1 - E_t/N)^{\alpha\eta}$. The expected number of distinct summaries that the agent receives from its neighbors at time slot t can be expressed by simply multiplying that probability by N, i.e.,

$$N\left(1 - \left(1 - \frac{E_{t-1}}{N}\right)^{\alpha\eta}\right). \tag{6}$$

Let H_t denote the expected number of distinct summaries that were harvested by the agent up to time slot t. The agent has H_t summaries, so the probability of acquiring a new summary is $1 - H_t/N$. Hence, multiplying this probability by the expected number of summaries that were harvested from the neighbors [see (6)] gives us the number of new summaries harvested during time slot t as follows:

$$H_t - H_{t-1} = \gamma N\left(1 - \left(1 - \frac{E_{t-1}}{N}\right)^{\alpha\eta}\right)\left(1 - \frac{H_{t-1}}{N}\right) \tag{7}$$

where the constant compensation factor γ adjusts the expected number of summaries that were received from the neighbors to consider the restricted mobility. Note that, as we have previously described, restricted mobility such as in the Manhattan mobility model reduces the rate of new encounters (adjusted by η) and exerts baleful influence on the rate of active

[3]We can think of this as follows. For example, in front of a freeway (where everybody is driving in one direction at a constant speed v), we count the number of vehicles that pass by. During Δt, it will be $\rho v \Delta t$. Now, let us assume that an observer is also moving. If it moves in the same direction, i.e., the relative speed is 0, it always observes the same vehicles. On the contrary, if it moves in the opposite direction, the relative speed is $2v$, and it will see $\rho 2v\Delta t$ vehicles.

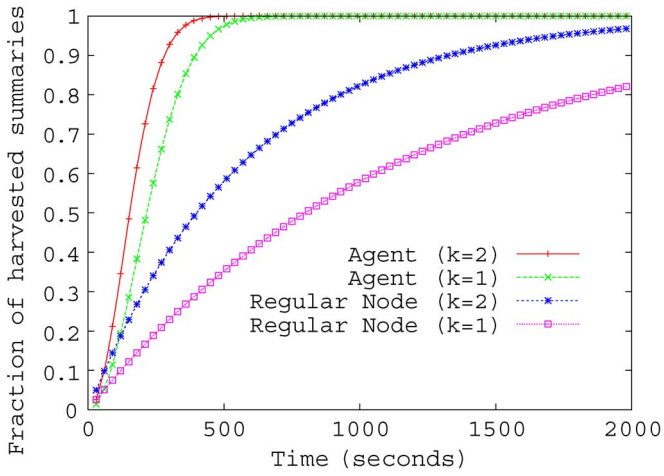


Fig. 5. Fraction of harvested summaries with $k = 1$ and 2.

harvesting, because neighbors tend to carry overlapping summaries (adjusted by γ).

Based on (7), we see that H_t grows much faster than E_t. During a time slot, the number of collected summaries in (2) is constant (α), whereas in (7), it is a function of time. Moreover, (6) is a function of the number of neighbors, i.e., related to the node density. As N (or the node density) increases, we can see that the harvesting delay also decreases.

The growing rates of E_t and H_t depend on mobility models. The aforementioned equations are based on the random-direction mobility model, but for restricted-mobility models, such as the Manhattan mobility model, the rate will be smaller than for the others (as shown in Section VI). In this case, MobEyes decides to use k-hop limited scope flooding, where a summary is forwarded up to k hop neighbors, as long as there is connectivity. Again, we are assuming a rectangular area $\Delta t 2R$. Increasing the relay scope by k hops is the same as multiplying the area by k times.

Let E^k denote the number of summaries that were collected by time slot t with k-hop relay scope. Thus, we have

$$E_t^k - E_{t-1}^k = k\alpha\eta\left(1 - \frac{E_{t-1}^k}{N}\right). \tag{8}$$

This equation tells us that, even though E_t grows rather slowly due to the mobility model, by increasing the hop count, we can increase the E_t rate (from α to $k * \alpha$). Let H^k denote the number of summaries that the agent harvested by time slot t with k-hop relay scope. Then, we have

$$H_t^k - H_{t-1}^k = \gamma N\left(1 - \left(1 - \frac{E_{t-1}^k}{N}\right)^{k\alpha\eta}\right)\left(1 - \frac{H_{t-1}^k}{N}\right). \tag{9}$$

For illustration, we assume that we have a total of $N = 200$ nodes within an area of 2400 m $\times$ 2400 m. The transmission range is $R = 250$ m, and the node relative speed is 10 m/s on the average. For system parameters, we used $\eta = 1, \gamma = 0.2$, and $\Delta t = 1$ s. The iterative solutions of both E_t and H_t are presented in Fig. 5, which shows that the agent can harvest summaries much faster than a regular node, and a k-hop relay relevantly decreases the overall delay.

B. Scalability

The feasibility of MobEyes strictly depends on its scalability over wide VSNs in terms of the network traffic due to both passive diffusion when the number of regular nodes grows and the number of regular nodes that a single harvesting agent can handle with a reasonable latency.

With regard to passive diffusion network traffic, it is possible to analytically estimate the MobEyes radio channel utilization. In the diffusion process, nodes periodically advertise their packets, without any synchronization among them. Therefore, we can model the process by considering a packet that was randomly sent within $[iT_a, (i+1)T_a)$ time slot for all i, where T_a is the advertisement period. Therefore, the number of packets that a node has received is bounded by the number of its neighbors while it is traveling for T_a, thus depending on the node density but not on the overall number of nodes. In contrast, any "flooding"-based diffusion protocol is not scalable, because a node can potentially receive a number of packets proportional to the network size.

To give a rough idea of the traffic that the MobEyes diffusion generated, let us simply use $T_a = 2R/v^*$ (i.e., the time that a mobile node traverses the diameter of its coverage area), where R is the transmission range, and v^* denotes the relative speed of two nodes. In fact, for a given speed, the T_a interval should neither be too short nor too long compared with the average connection duration among nodes. If it is too short, then we are unnecessarily sending out more packets to the same set of nodes, thus increasing link bandwidth utilization; on the contrary, if it is too long, a node misses its chances to send packets to encountered nodes, thus slowing down the dissemination. In our target deployment environment, $v^* = 20$ m/s, $R = 250$ m, the advertisement period $T_a = 12.5$ s, and the fixed packet size $S = 1500$ B. Consequently, the transmission time for one packet is about $T_x = 1$ ms. While traveling for T_a, a node moves $2R$, and the covered rectangular area size is $4R^2$. In addition, the covered area includes two half circles at the beginning and ending of the rectangle (due to the wireless communication range). Thus, a regular node will be exposed to advertisements from an area of $\mathcal{A} = \pi R^2 + 4R^2$. In the worst case, all nodes within this area are distinct and potentially send their generated packets to the considered node (i.e., potential senders $n = \mathcal{A}\rho$). Therefore, the worst case link utilization could be estimated as nT_x/T_a, where T_x is the transmission time of a packet. For instance, given a relatively high populated area with $N = 2,000$, the number of potential senders is $n \simeq 179$, and the MobEyes protocol has a very low worst case link utilization of 0.014, thus showing high scalability in terms of link bandwidth exploitation.

Similarly, we can give an approximated idea of the scalability of the harvesting process via a simple queuing model. Consider the usual situation of a police agent that harvests only fresh summaries, i.e., generated in the last T_{exp} s. Let us assume that the summary arrival rate is Poisson with rate $\lambda = N\lambda'$, and the harvesting rate is deterministic with rate μ. Given that the harvesting rate is limited by the channel utilization ϑ, the maximum μ is simply $\vartheta T_{exp}/T_x$. As a result, the system can be modeled using an M/D/1 queue. The stability condition

$N\lambda' < \vartheta T_{exp}/T_x$ gives us the upper bound $N < \vartheta T_{exp}/\lambda' T_x$. Therefore, it is possible to conclude that, for a given T_{exp} and arrival rate, there is a limit in the number of regular nodes that a single harvesting agent can handle. For instance, in the considered scenario ($\vartheta = 0.01$, $\lambda' = 2$, and $T_{exp} = 250$ s), that number is $N < 0.01 \times 250/(2 \times 0.001) = 1,250$. As a consequence, in the case of node numbers being equal to or not far from 1,250, there is a need to deploy more than one harvesting agent to maintain the system stable (i.e., to more rapidly harvest summaries than regular nodes generate them).

VI. MobEyes Performance Evaluation

We evaluated the MobEyes protocols via extensive ns-2 [49] simulations. This section shows the most important results, with the goal of investigating MobEyes' performance based on the following perspectives.

1) *Analysis validation*. We simulate the MobEyes protocols for summary collection on regular nodes and for agent harvesting and show that they confirm our main analytic results.
2) *Effect of k-hop relay and multiple agents*. We examine how MobEyes' effectiveness can be increased by leveraging k-hop passive diffusion and the deployment of multiple agents.
3) *Summary diffusion overhead*. We investigate the tradeoff between harvesting delay and the load that was imposed on the communication channel.
4) *Stability and scalability check*. We verify that the system is stable/scalable, even in the worst case of a single harvesting agent and of the highest summary generation rate in Section V.
5) *Tracking application*. We prove MobEyes' effectiveness in supporting a challenging tracking application, where trajectories of regular nodes are locally reconstructed by a police agent based on harvested summaries.
6) *Border effects and turnover*. We show that MobEyes' performance does not dramatically change in the case with more dynamic mobility models, where nodes are allowed to enter/exit from the simulated area.

Additional experimental results and MobEyes implementation details are available at http://www.lia.deis.unibo.it/Research/MobEyes/.

A. Simulation Setup

We consider vehicles that move in a fixed region of size 2400×2400 m. The default mobility model is Real-Track (RT), which was introduced by our colleagues in [50]. RT permits us to model realistic vehicle motion in urban environments. In RT, nodes move following virtual tracks, thus representing real accessible streets on an arbitrary loaded roadmap. For this set of experiments, we used a map of the *Westwood* area in the vicinity of the University of California at Los Angeles (UCLA) campus, as obtained by the U.S. Census Bureau data for street-level maps [51] (see Fig. 6). At any intersection, each node randomly selects the next track that it will run through. The speed is periodically allowed to change (i.e., increase or decrease) by a quantity that is uniformly distributed in the interval $[0, \pm\Delta s]$. To evaluate the impact of the mobility model on MobEyes' performance, we tested two additional well-known models: 1) Manhattan (MAN) [48] and 2) random waypoint (RWP) [52]. Similar to RT, MAN builds node trajectories following urban roads; however, in MAN roads are deployed according to a regular grid, thus allowing a more uniform node deployment. In our simulation, we adopted a 10×10 grid. RWP does not constrain node positions to follow actual road tracks but moves nodes toward randomly selected destinations with random speeds. When a node reaches its destination, it pauses for a fixed period (which we set to be equal to 0 by homogeneity with the other models) and then selects a new destination. Surprisingly, RWP is considered "a good approximation for simulating the motion of vehicles on a road [53]", generally producing limited distortion on protocol performance. Note that MobEyes agents do not exploit any special trajectory or controlled mobility pattern but move by conforming to regular nodes.

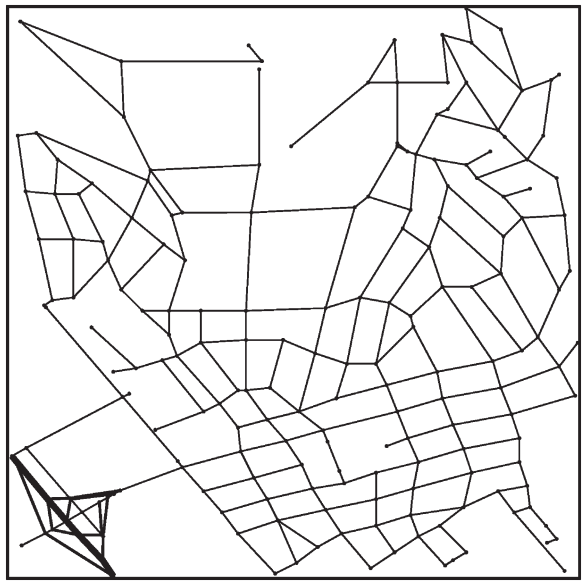


Fig. 6. Map of the Westwood area in the UCLA campus.

Our simulations consider the number of nodes $N = 100, 200, 300$. Vehicles move with an average speed of $v = 5, 15, 25$. To obtain these values, we tuned the minimum speed to $v_m = 1$ and the maximum speed to $v_M = 11, 31, 51$, respectively. The summary advertisement period of regular nodes and the harvesting request period are kept constant and equal to 3 s through all the simulations. We use a Bloom filter with 8192 bits (1024 B) and 10 hash functions. We have a large filter size (i.e., 8192 bits) compared with the number of summaries, so the false-positive probability is negligible.[4] We note that if the value of this parameter is too large, MobEyes' effectiveness is reduced, because it is possible that two nodes do not exchange messages, even if they occasionally enter each other's transmission ranges. This effect is magnified, as the

[4]The false-positive probability is $p_f = (1 - (1 - 1/m)^{\ell\omega})^\ell \simeq (1 - e^{-\ell\omega/m})^\ell$, where m is total number of bits, ω is the number of elements, and ℓ is the number of hash functions.

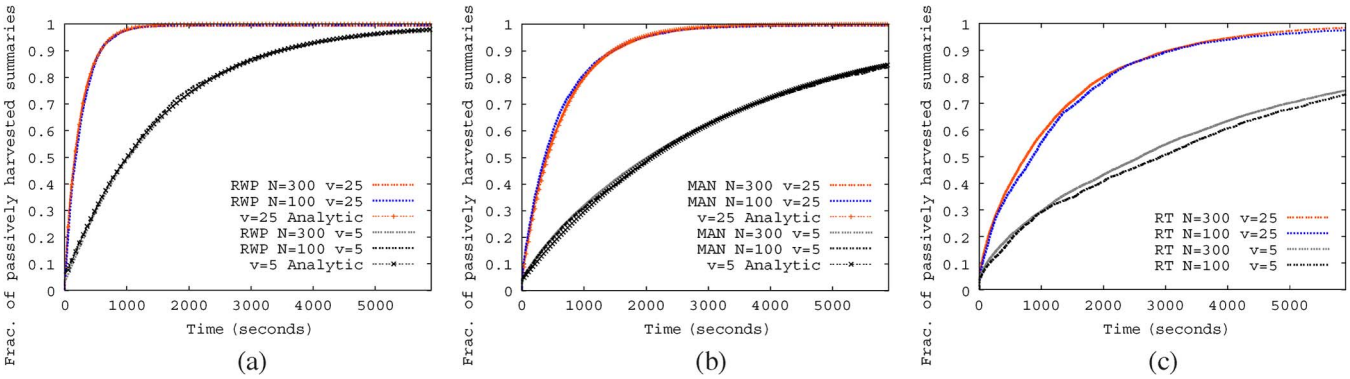


Fig. 7. Fraction of *passively* harvested summaries by a regular node. (a) RWP. (b) MAN. (c) RT.

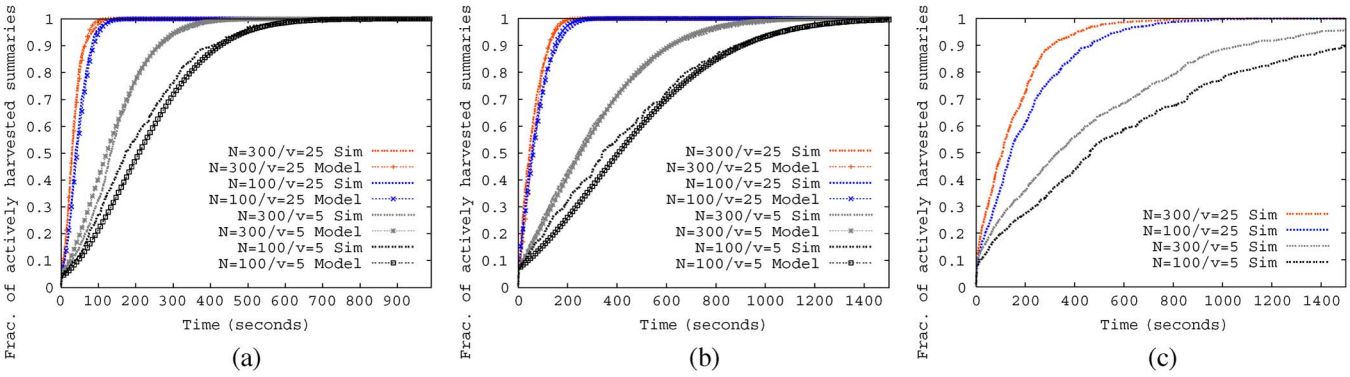


Fig. 8. Fraction of *actively* harvested summaries by an agent. (a) RWP. (b) MAN. (c) RT.

node speed v increases. The chosen value has experimentally been determined to balance the effectiveness of our protocol and the message overhead, even in the worst case, i.e., $v = 25$. A deeper more formal investigation of the optimal value of the advertisement period is an object of future work.

Finally, we modeled communications as follows: the MAC protocol is IEEE 802.11, the transmission band is 2.4 GHz, the bandwidth is 11 Mb/s, the nominal radio range is equal to 250 m, and two-ray ground is the propagation model [54]. The values of these parameters have been chosen similar to other work in the field [17], [20]. Unless stated otherwise, reported results are average values out of 35 repetitions. Other MobEyes configuration parameters will be introduced in the following sections when discussing the related aspects of MobEyes' performance.

B. Analysis Validation

Our first goal is to validate the results that were obtained in Section V. In particular, we investigate the regular node collection and agent harvesting processes, as described, respectively, by (2) and (7). Without loss of generality (see Section VI-E), let us assume that new summaries are synchronously generated by all regular nodes. A *generation epoch* is the time interval between two successive summary generations. In this set of experiments, every regular node continuously advertises the single summary that it generated in the epoch $t = 0$ for the rest of the simulation run. Equations (2) and (7) characterize the spreading processes of all summaries that were generated in the same epoch, so it is not necessary that regular nodes generate additional summaries. We remark that this assumption does not undermine the relevance of our results, because the process is stationary, as described in Section VI-E.

Figs. 7 and 8 show the results that were collected for a number of nodes $N = 100/300$, average speed $v = 5/25$, and the RWP, MAN, and RT mobility models. In particular, Fig. 7 plots the cumulative distribution of summaries that were collected by regular nodes as a function of time. The figure shows that the process highly depends on the average node speed. In fact, the speed determines, to a large extent, how quickly nodes "infect" other participants with their own summaries. The results do not depend on the node density, as shown in (4). Our analytic model (2) accurately fits the simulation results for RWP and MAN. The curves for the RT model exhibit worst fitting: They start deviating from that of analytical results after certain thresholds [i.e., the analytic results that were not reported in Figs. 7(c) and 8(c)]. RWP shows slightly better accuracy mainly due to node unconstrained motion and to the tendency to gather at the center of the field as time passes [46]. Although both MAN and RT show restricted mobility patterns, their node distributions are different: MAN has almost-uniform node distribution over the grids, whereas RT exhibits nonuniform node distribution over the map, as shown in [55]. Our model is based on uniform distribution, so the curve fitting works well with MAN. The experimental results helped us tune the constant compensation factor η that we have introduced in Section V to take into account the nonuniform movement patterns. In detail, the values that we have found are 0.97 and 0.9, respectively, for

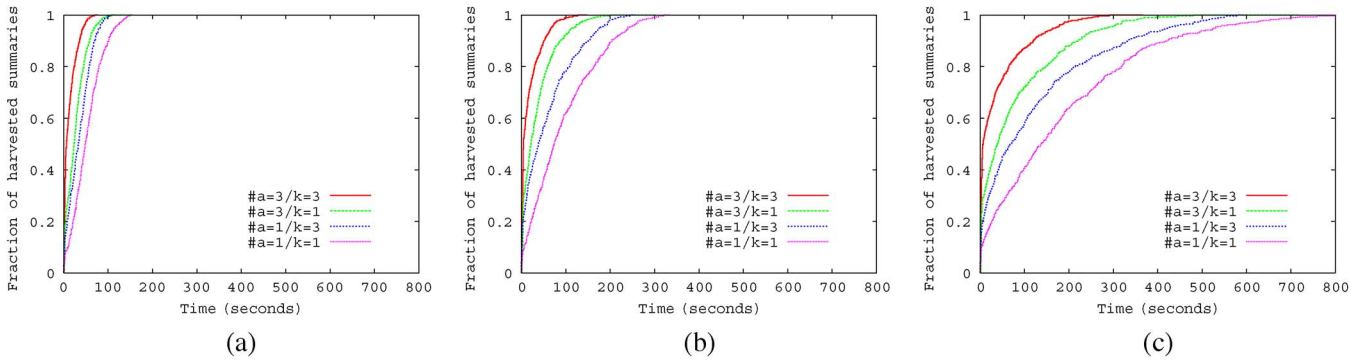


Fig. 9. Fraction of *actively* harvested summaries by multiple agents with k-hop relay ($N = 300$ and $v = 15$). (a) RWP. (b) MAN. (c) RT.

$v = 15$ m/s and $v = 25$ m/s in RWP and 0.40 and 0.36 in MAN. As restrictions on node mobility grow due to the mobility model, the values of η decreases, thus representing a slower process.

Fig. 8 plots the cumulative distribution of summaries that were harvested by a police agent as a function of time. The figure shows that the results are mainly dependent on the speed. Unlike in passive harvesting, density plays an important role in active harvesting. In the analysis section, we show that the higher the density, the faster the harvesting progress [see (7)]. Intuitively, if there are more neighbors, the agent has a higher chance of getting a random summary. Our analytic model well fits the simulations, particularly when we have large N and v. This set of results allowed us to tune the parameter γ, accounting for the effect of the overlapping of summaries that were contributed from regular agents (see Section V). In detail, the values that we have found are 0.21 and 0.21, respectively, for $v = 5$ m/s and $v = 25$ m/s in RWP and 0.15 and 0.20 in MAN.

C. Effect of k-Hop Relay and Multiple Agents

The effectiveness of MobEyes harvesting can be measured in terms of the fraction of summaries that were harvested by the agent(s) as a function of time. To enhance the validity of our conclusions, it is important to determine the dependence of the performance indicators on different mobility models. In [56], we only investigated the RT mobility model. In this paper, we extend the results to RWP and MAN. For every mobility model, we show plots for 1, 3 agents $(a\#)$ and for 1, 3 relay hops (k). For k-hop relaying, we use a probabilistic flooding, i.e., a node rebroadcasts a newly received packet with probability $p = 0.5$.[5] The summary harvesting latency is a crucial figure for determining the feasibility of the MobEyes approach, because it allows us to estimate the fraction of harvested summaries by the agent within a certain time t. This estimation is useful in deciding the tuning of the parameters (the k-hop relay scope and the number of agents) to address application requirements. Fig. 9 shows how the number of agents, the choice of the number of relaying hops k, and the average speed v of the nodes influence the process. Fig. 9 plots the cumulative distribution of the summaries that were harvested for $N = 300$ and $v = 15$ m/s. In the case of multiple agents, the harvesting process considers the union of the summary sets that were harvested by agents. The figure clearly shows that the k-hop relay scope and multiple agents highly impact harvesting latency.

By carefully inspecting the results in Fig. 9, it is possible to obtain some guidelines on the choice of MobEyes parameters. For example, given, as a baseline, a network with $N = 300$ nodes that move with an average speed of $v = 15$ m/s, fixed $k = 1$, a single agent employs 530 s, 236 s, and 116 s to harvest 95% of the summaries that were generated, respectively, in the RT, MAN, and RWP mobility models. By increasing k to 3, the times, respectively, reduce to 420 s, 176 s, and 86 s, which shows an improvement of about 20%–30% in all cases. On the other hand, by increasing the number of agents to three, the times become, respectively, 280 s, 123 s, and 68 s. In this case, the improvement is in the range of 40%–50%. If we set $v = 25$ m/s, the times become 211 s, 67 s, and 43 s, respectively, and the improvement is around 60%–70%. Interestingly, the relative impact of the three parameters (i.e., the harvesting team size, multihop forwarding, and speed) shows a limited dependence on the mobility model. This fact also holds for the results that we have collected for different cases (i.e., different values of N and v). In particular, speed has a larger impact than the number of agents, and k is the less-decisive factor.

D. Summary Diffusion Overhead

The study of the diffusion overhead helps us understand the requirements that were imposed on the underlying vehicular communication technology and to determine if MobEyes can coexist with other applications. For example, parameter k shows the largest impact on the performance. The effect due to a small number of agents is negligible, because the agents are only responsible for local single-hop traffic. Fig. 10 shows the average received packets per node per second, which was obtained during a simulation time of 1000 s. In this set of simulations, we fixed $k = 1$ and changed all the other parameters, i.e., the mobility model (RWP, MAN, and RT), N (100, 200, and 300), and v (5, 15, and 25). As expected, the number of received packets linearly increases as the number of nodes increases. Therefore, for clarity, Fig. 10 only reports the case with $N = 300$. In addition, the number of received packets

[5] In the simulation, we use the arbitrary value, but for a given scenario, we can pick the probability that can minimize the overhead (i.e., redundant broadcasts). Note that we can further reduce the overhead by using the efficient broadcast schemes in [57].

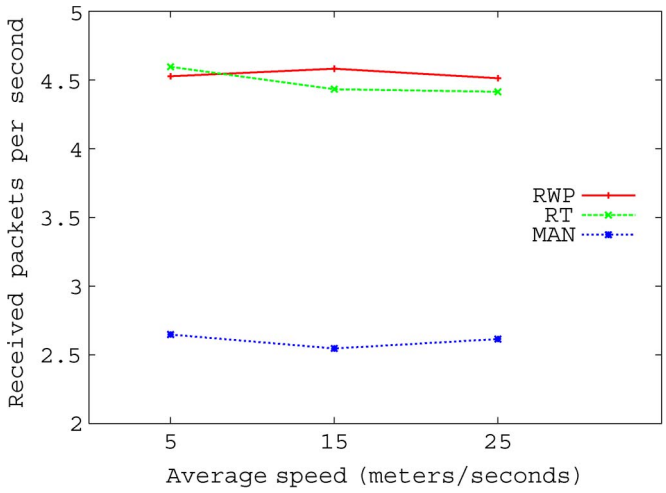


Fig. 10. Total number of received packets ($k = 1$).

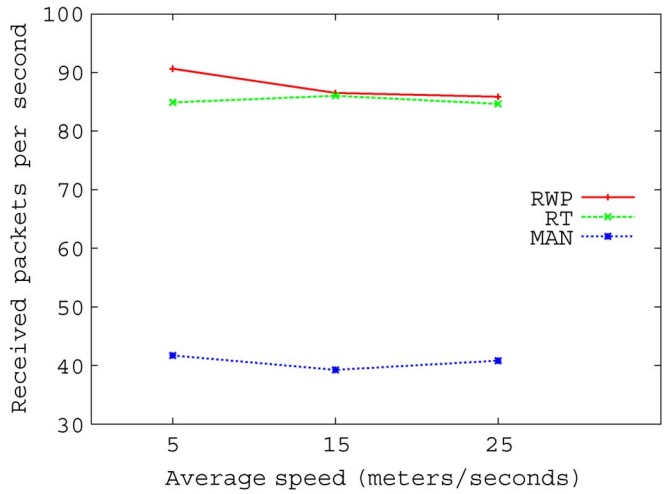


Fig. 11. Total number of received packets ($k = 3$).

exhibits no dependence on v. In all the considered cases, the overhead is limited, i.e., on the order of few (e.g., two to five) packets per second, which proves the low impact of MobEyes on the available bandwidth.

The latter result could mislead us to conclude that speed increments would not impact the harvesting latency, because the number of received packets would not change. This apparently invalidates our previous results (see Fig. 7) and has the following motivations. For a fixed advertisement interval, as the average speed increases, the probability of useful meetings (i.e., of receiving a nonredundant summary) increases, because there is more mixing among mobile nodes. For example, given an average speed v, let us assume that the average period that any two nodes are within their communication ranges simply be $2R/2v$. Then, with v being set to 5 and 25 m/s, and $R = 250$ m, the periods can be estimated as 50 and 10 s, respectively. This step implies that the case with 5 m/s has roughly five times higher chances of receiving redundant advertisements than the case with 25 m/s. It is interesting to note that, for a given average speed, there exists an optimal advertisement period that allows for maximizing nonredundant summary diffusion while minimizing the overhead. Analytically determining this value will be part of our future work.

Fig. 11 shows the magnifying effect that was produced by an increase in parameter k. k-hop relaying produces an enlargement of the area where summary packets are diffused intuitively proportional to k^2. Consequently, the number of nodes that were affected by a single summary diffusion will also be about k^2 larger than the single-hop case. Moreover, nodes receive any summary packet only once in the single-hop case, whereas with k-hop relaying, any node within k hops from the originator receives it a number of times proportional to the number of its neighbors. Thus, the total overhead is expected to increase by a factor that is larger than k^2 but is lower than k^2 times the average number of neighbors (note that k-hop distant nodes do not relay packets, thus reducing the latter factor for k-hop and $k-1$-hop distant nodes). The combination of these results with those in Fig. 9 lead us to conclude that parameter k allows for decreasing the harvesting latency (i.e., about 20%–30% for $k = 3$) but at the price of a relevant overhead increase (i.e., around 15–20 times). The proper balance of latency/k tradeoff can only be decided based on specific characteristics and requirements of the supported urban monitoring application.

E. Stability and Scalability Check

In what follows, we investigate the stability of MobEyes by verifying that continuous summary injections do not influence its performance to a large extent. In particular, we show that the ratio of summaries that were harvested on longer periods remains acceptable and that the harvesting latency does not grow as time passes. With regard to the results that we have presented so far, here, we remove the assumption about the single summary generation epoch at $t = 0$. Nodes generate new summaries with period $T = 120$ s and advertise the last generated summary. Note that in Section III, this rate represents a practical worst-case scenario. For clarity, we hold the synchronicity assumption: all nodes simultaneously generate new summaries at intervals multiple of T. We obtained similar performance with differently distributed generation intervals, i.e., Poisson with average value T, but plots (particularly related to results in Fig. 13) are far more jumbled. The following results are reported for the case of a single harvesting agent, $k = 1$, $N = 100$, $v = 15$ m/s, and with nodes moving according to the RT model.

Fig. 12 plots the cumulative distribution of the number of summaries that were generated and harvested as a function of time (i.e., we ran simulations for 6000 s). The graph shows that the harvesting curve tracks the generation curve with a certain delay, which can be traced to the harvesting latency in Section VI-B. This also motivates the difference of the endpoints of the two plots. Fig. 13 provides further evidence of the stability of the system, and the curves show the harvesting latency for summaries that were generated during some generation epochs. For clarity, the graph does not exhaustively represent every generation epoch but only samples one generation epoch every $T * 7 = 840$ s until the end of the simulation time. The different curves show similar trends, without any performance degradation that was caused by the increase in the number of summaries in the network. The harvesting that

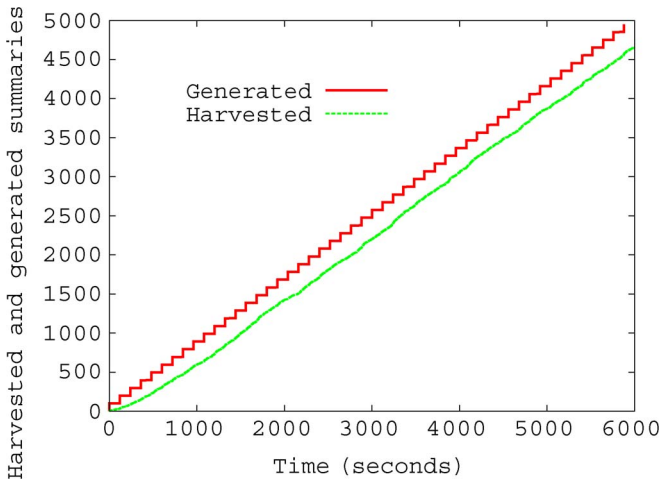


Fig. 12. Stability of summary harvesting with continuous summary injection. Cumulative distributions of generated and harvested summaries *over all generation epochs*.

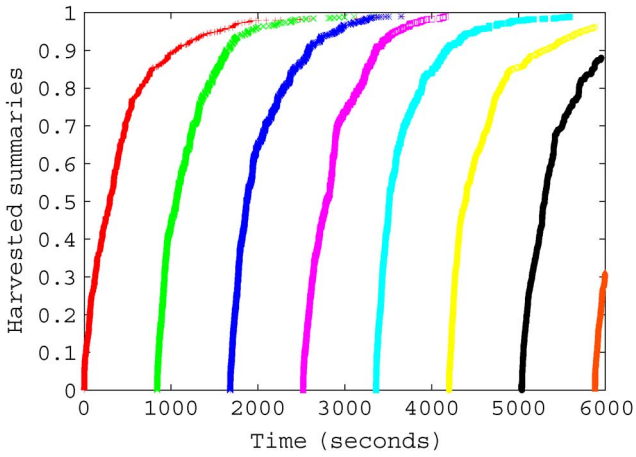


Fig. 13. Stability of summary harvesting with continuous summary injection. Cumulative distributions of harvested summaries in every seven-generation epoch ($120 * 7 = 840$ s interval).

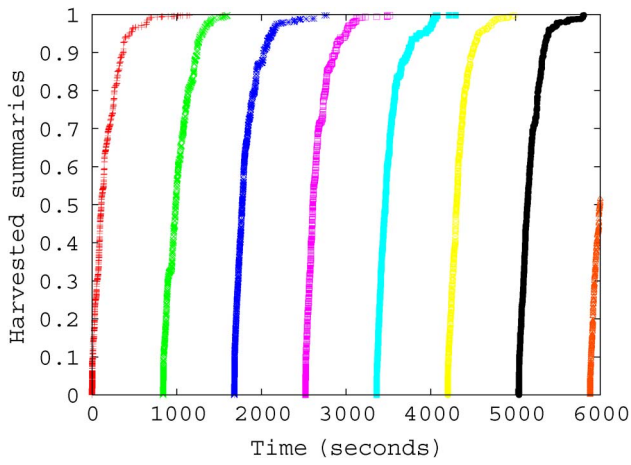


Fig. 14. Scalability of summary harvesting with $N = 1000$. Cumulative distributions of harvested summaries in every seven-generation epoch ($120 * 7 = 840$ s interval).

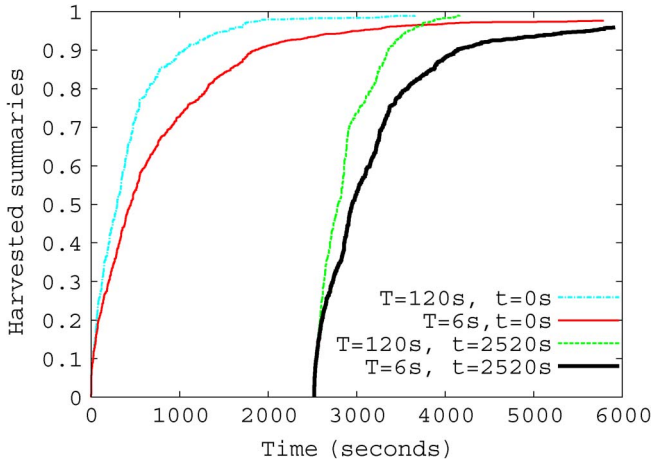


Fig. 15. Impact of different summary generation rates ($T = 6, 120$ S). Cumulative distributions of two epochs ($t = 0, 2520$ S).

is related to the last summary generation epoch is evidently incomplete (i.e., 25% of the summaries are harvested within the timeline), because the epoch starts 120 s before the end of the simulation. We also evaluate the scenario with N $= 1000$ to show the scalability. Fig. 14 shows that the harvesting latency is considerably reduced compared to the case with $N = 100$. Thus, our protocol scales well. The result matches with the observation from our analytic model in Section V-A, where we find that the harvesting delay decreases as the number of nodes increases.

We also investigated if higher summary generation rates afflict MobEyes' performance. We shortened T from 120 s to 6 s (with $T = 6$ s, the chunk generation rate is 100 ms). Such a generation rate is largely greater than the rate that is required for the set of applications that MobEyes addressed. Simulation results prove that MobEyes' performance starts degrading only when $T < 30$ s. Fig. 15 shows the harvesting process for two epochs (i.e., 0 and 2520 s) and compares $T = 120$ s with $T = 6$ s. The second case shows that MobEyes' performance gracefully degrades as the generation epoch shortens, thus demonstrating the high stability of the system when operating in usual summary rate conditions. We expect that the performance degradation happens in an earlier stage as the number of nodes increases. However, the advertisement period can be shortened, because the harvesting delay is also decreased. This reduces the overall overhead, and thus, the performance degradation happens rather gracefully with the number of nodes. For instance, our results with N $= 1000$ show that the performance starts degrading when $T < 50$ s. Note that if a single agent cannot sustain the configuration, multiple agents can be used for better scalability, as discussed in Section V-B.

F. Tracking Application

In the Introduction, we have sketched some application cases for MobEyes. To prove its effectiveness in supporting urban monitoring, we also simulated a vehicle tracking application where the agent reconstructs node trajectories that exploit the collected summaries. This application is challenging, because it requires our system to observe the following: 1) Monitor a large number of targets, i.e., all participant vehicles; 2) periodically generate fresh information on these targets, because they are highly mobile; and 3) deliver to the agent a high share of the

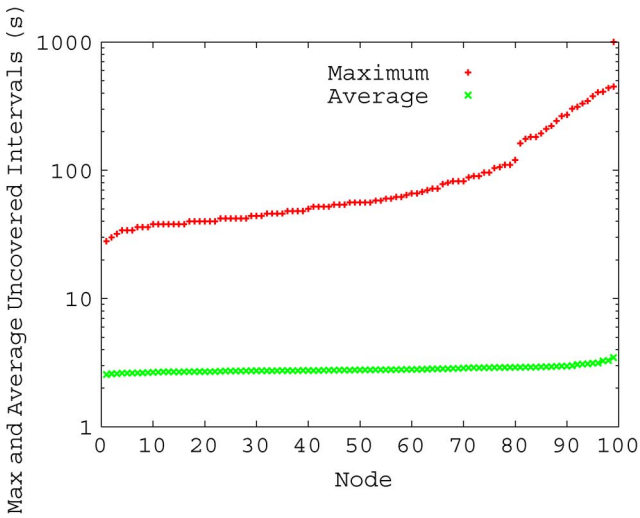


Fig. 16. Maximum uncovered intervals per node.

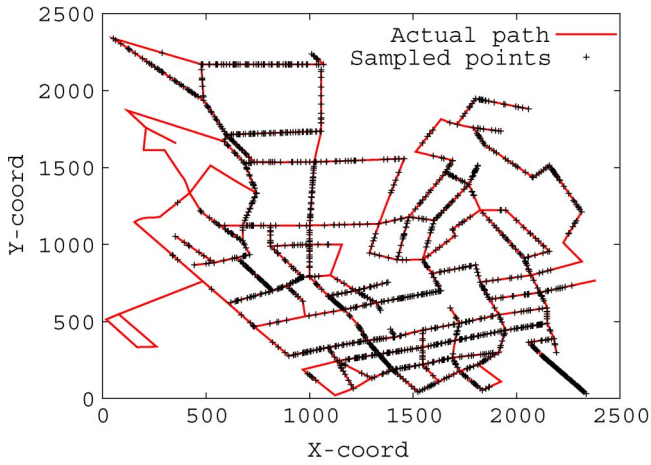


Fig. 17. Actual node trajectory versus harvested sampled points.

generated information. Moreover, because nodes are generally spread all over the area, this application shows that a single agent can maintain a consistent view of a large zone of responsibility. In more detail, as regular cars move in the field, they generate new summaries every $T = 120$ s and continuously advertise the last generated summary. Every summary contains 60 *summary chunks*, which are created every $ChunkPeriod = 2$ s and include the license plate and position of the vehicle nearest the summary sender at the generating time, which is tagged with a timestamp. The application exploits the MobEyes diffusion protocol with $k = 1$ to spread the summaries and deliver as much information as possible to a single agent that scouts the ground. As the agent receives the summaries, it extracts the information about node plates and positions and tries to reconstruct node trajectories within the area. This step is possible by aggregating data that are related to the same license plate, as reported from different summaries.

To determine the effectiveness of MobEyes, we decided to evaluate the *average uncovered interval* and *maximum uncovered interval* for each node in the field. Given a set of summary chunks related to the same vehicle and ordered on a time basis, these parameters measure, respectively, the average period for which the agent does not have any record for that vehicle and the longest period. The latter interval typically represents situations in which a node moves in a zone where the vehicle density is low; thus, it cannot be traced by any other participant. We associated the average and maximum uncovered intervals to each simulated node and present the results in Fig. 16 (note the logarithmic scale on the y-axis). Every point in the figure represents the value of the parameter for a different node. We sorted nodes on the x-axis so that they are reported with increasing values of *uncovered interval*. Results are collected along a 6000-s simulation. The plot shows that, in most cases, the average uncovered interval floats between [2.7 s–3.5 s]. The maximum uncovered interval shows that, even in the worst cases, the agent has at least one sample every 200 s for more than 90% of the participants. A more immediate visualization of the inaccuracy is given in Fig. 17, which shows, for the case of a node with a maximum uncovered interval that is equal to 200 s (i.e., locating this node in the lowest tenth percentile), its real trajectory (i.e., the unbroken line) and the sample points that the agent collected.

G. Border Effects and Turnover

Usual mobility models [58], e.g., RWP, MAN, and RT, assume that nodes remain within the simulated area during the entire simulation (in the following discussion, we indicate them as *closed* mobility models). Even if this does not necessary hold for MobEyes applications, we observe that this assumption does not invalidate our findings. First, if we consider a sufficiently large area, which is on the order of several hundred kilometer squares, the amount of time that nodes continuously reside within the area is likely very long, i.e., for most nodes, a closed mobility model. Second, the worst effect of dynamic scenarios takes place when nodes leave a specific area and carry several summaries (locally generated or collected) that are not yet harvested by the local agent. Nonetheless, we remark that carried information does not vanish as nodes leave but can be harvested later by remote agents, who are responsible for the adjacent area where the leaving nodes are moving.

However, to estimate how node entrances/exits impact the presented results, we tested MobEyes with a novel mobility model, i.e., *open-RT*, which takes these effects into account. In open-RT, nodes follow the same patterns of RT but with one exception: as soon as a node reaches the endpoint of a track, which is close to the boundary of the area, it suddenly disappears. To keep the number of nodes within the area unchanged and obtain results comparable to the ones presented in the previous sections, we assume that the net vehicle flow in/out the area is null. Thus, any node that exits from the area is immediately replaced with one node that enters, and the latter node is placed at the endpoint of a random road, which is close to the boundary of the area.

This dynamic effect is better evaluated for long simulation periods and periodic summary generation epochs. Thus, we confirm the settings that were used in Sections VI-E and F. In addition, we consider a single harvesting agent, $k = 1$, $N = 100$, and $v = 15$ m/s. Nodes synchronously generate new summaries and only as long as they remain in the area.

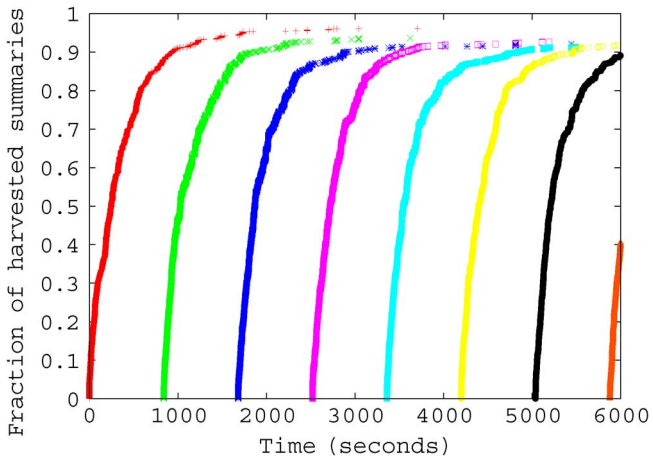


Fig. 18. Cumulative distribution of harvested summaries *per epoch (open-RT)*.

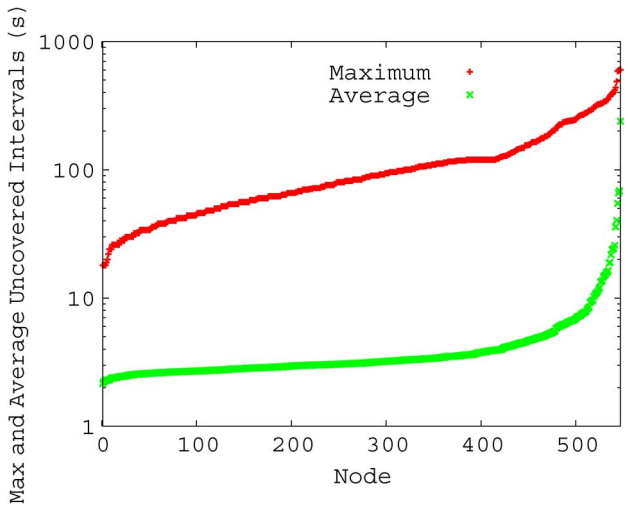


Fig. 19. Maximum uncovered intervals per node (*open-RT*).

To keep nodes from staying within the area only for very short periods, we introduce a constraint on their minimum residing time, which is equal to 10% of the whole simulation. Even with this assumption, more than 550 nodes need to take turns on the simulation area to keep 100 nodes always present. The agent does not follow the open-RT model but the traditional RT model, i.e., it always remains within the area.

Figs. 18 and 19 present results that correspond to those in Sections VI-E and -F but were obtained with the open-RT model. Significant conclusions can be drawn, particularly based on Fig. 18. Under these unfavorable assumptions, the agent can collect more than 85% of any generated summary, and in most cases, it reaches 90%. By inspecting simulation traces, we could find that missing summaries generally originate with vehicles leaving the area within a short interval from any epoch. In that case, the last generated summary is only advertised for that short interval and cannot spread enough to reach the agent. Note that those summaries are not irreparably lost but will probably be harvested by agents who are in charge of the adjacent areas. Fig. 19 shows the average and maximum uncovered intervals, as obtained with the open-RT model. The quality of the reconstructed trajectories is only slightly degraded, given that the average uncovered interval is shorter than 4 s for more than 75% of the nodes (and shorter than 10 s for 90%) and that the 85th percentile of the vehicles can be tracked with a worst-case inaccuracy of 200 s.

H. Discussion

1) Mobile Versus Static Agents: We assume that police agents are also mobile. It would be also possible that stationary agents can be placed on the roadside for summary harvesting. Then, the question is whether the performance of static agents is comparable with that of mobile agents. The answer is that mobile agents perform better than static agents, because, for a given period of time, mobile agents will encounter a larger number of nodes than static nodes. To be precise, in Section V, we show that the number of mobile nodes encountered during Δt is given as $\alpha = \rho v^* \Delta t 2R$, where v^* is the average relative speed. If the agent is static, the relative speed is simply replaced with the average speed. The relative speed between mobile nodes is typically faster than that between mobile and static nodes. Thus, α is greater if the agents are mobile (i.e., lower latency). In addition, in the urban grids, the spatial node distribution of vehicles in the steady state is usually nonuniform. If an agent is misplaced (i.e., where there are few vehicles on the average), the harvesting latency will be large. To clearly understand the performance difference and how the position of a static node affects the latency, we use a scenario with 100 nodes that move at an average speed of 25 m/s. We intentionally fix the location of a random node (as an agent) to its initial position, which is uniformly distributed in the simulated area, i.e., for a given scenario file $(\mathrm{N} = 100, \mathrm{V} = 25\ \mathrm{m/s})$, we generate 100 scenario files by fixing a node one by one to its initial position. Our simulation results show that the average latency widely varies, depending on the location. We find that the area with high spatial node distribution minimizes the latency, but the average latency of mobile agents is still lower than that of static agents.

2) Impact of Radio Range: The radio range is an important performance parameter. Recall that the radio range controls the number of encountered nodes per unit time $(\alpha = \rho v^* \Delta t 2R)$. The performance with various radio ranges can analytically be determined using our models in Section V. The large communication range reduces the dissemination/harvesting latency but will limit the scalability of network due to increased channel contention.

3) Comparison With Other Schemes: Let us compare the performance with other two naive schemes: 1) direct contact and 2) probabilistic flooding. In direct contact, regular vehicles do not advertise summaries, and the agent harvests the summaries only from the summary originator. CarTel [6] uses direct contact for data uploading. In probabilistic flooding, regular vehicles use probabilistic flooding to broadcast its summary. Upon receiving a message, each node rebroadcasts the message with a certain probability p. The performance of direct contact is the same as that of "passive harvesting." Recall that a regular node performs passive harvesting, because it only stores summaries that were received from its one-hop neighbors. Our analytic model clearly shows that the agents harvest summaries much faster than the regular nodes. Thus, we only consider the

performance of the flooding scheme. Given $p = 1$, we measure the fraction of collected summaries that were generated for a period of 2000 s. Each message is generated every 20 s (with a total of 100 messages per node). To generate intermittent connectivity, we simulate 50 nodes that move at the average speed of 15 m/s. The RT model is used. We measure the completeness of summary harvesting, i.e., the fraction of messages that were harvested out of a total of 49 000 $(49 * 100)$ messages. Our results show that MobEyes can collect 100% of summaries, whereas the probabilistic scheme collects 32% of summaries. Note that the completeness gain of MobEyes comes at the cost of increased latency.

VII. MobEyes Privacy and Security

MobEyes nodes continually generate and diffuse summaries that contain private information, e.g., license plate numbers. Thus, privacy is of critical importance. On one hand, unauthorized nodes must not be given access to private information, including vehicle location. On the other hand, the harvesting process should not reveal the information that is being sought, because this step may tip off the attackers and/or cause unnecessary panic to the public. In general, we can summarize the security requirements of MobEyes as follows.

- *Authentication*. Harvesting agents (i.e., authority nodes) must authenticate summary senders and *vice versa*.
- *Nonrepudiation*. A summary originator cannot deny the transmission of a summary (liability issue). This way, upon request from the agent, the summary source must submit the full file with related sensed data.
- *Privacy*. Only legitimate users (i.e., authority nodes) can access summaries. Moreover, summaries must privately be advertised such that the attackers cannot track users.
- *Service availability*. MobEyes summary diffusion/harvesting should be protected from denial-of-service (DoS) attacks.
- *Data integrity*. MobEyes should filter out false summary data that were injected by attackers.
- *Query confidentiality*. In some cases, e.g., biological attacks and search for crime suspects, even the nature of the query that was injected by harvesting agents should not be disclosed, should not create unnecessary panic in the population, or should avoid tipping off the criminals.

One important aspect that sets apart MobEyes "forensic sensed data" security from conventional VANET security for "safe navigation" is the "real time" and "criticality" of the safe navigation application. For instance, consider a dangerous curve on the road monitored by an "e-mirror." If no car is coming, the e-mirror tells the driver to proceed at normal speed; otherwise, it tells the driver to slow down. An adversary can "intercept and replay" a message from the mirror and tell the driver that the way is clear, whereas a truck is coming at high speed behind the curve. In this "safe drive" application, it is mandatory to authenticate alert messages. Thus, in safe navigation applications, message authentication is far more important than privacy. For instance, privacy concerns should not prevent drivers from alerting the vehicles behind them that there is a boulder on the road.

MobEyes has strongly different security requirements. A false report cannot create much damage, because it is not immediately acted upon (e.g., a wrong set of license plates at the crime scene). There is plenty of time to detect and, if necessary, punish the "impostors." On the other hand, drivers that propagate summaries want to be assured that their privacy will not be violated. This major difference in security concerns leads to MobEyes security approaches that are quite different (and, in fact, much simpler and generally more efficient) than conventional VANET security solutions. Readers can find general security issues for VANET in [59]. For brevity, in this section, we will simply outline several MobEyes security approaches, reserving the detailed rigorous discussion of MobEyes security to future publications.

In MobEyes, we assume the existence of a public key infrastructure (PKI). Standard PKI mechanisms provide authentication and nonrepudiation, so we focus on the rest of the MobEyes requirements (i.e., privacy, service availability, and data consistency) by addressing the following MobEyes-specific attack models and a brief description of possible solutions. Readers can find the details for each solution in the extended version of this paper [60].

- *Location tracking*. Periodic broadcasting of identical summaries could facilitate attackers in tracking the route of a vehicle. To change the encrypted summary, one can introduce perturbation of time and position information.
- *DoS*. Attackers could inject a large number of bogus summaries to slow down correct summary harvesting by agents. MobEyes can check summary validity using the PKI, and the rate of generating valid summaries can be limited using *rate-limited summary diffusion*, which shares the same idea of the router requirements rate limit in a secure routing protocol [61].
- *False data injection*. Attackers could inject fabricated summaries to mislead investigations or make the data inconsistent. Statistical methods in conventional sensor networks [38], [62], [63] can be used. As noted in [64], the mobility of vehicles makes it hard in reality.
- *Query confidentiality*. Attackers could infer "important" information from the content of police queries. *Private keyword searching*, as proposed by Ostrovsky *et al.* [65], can be used. Secure filters can be distributed to the regular vehicles, and agents can harvest the resulting encrypted data.

VIII. Conclusion

In this paper, we have proposed the decentralized opportunistic MobEyes solution for proactive urban monitoring in VSNs. MobEyes' key component is MDHP, which works by disseminating/harvesting summaries of sensed data and uses original opportunistic protocols that exploit intrinsic mobility of regular and authority nodes. One of the reasons for using original dissemination protocols is, for instance, to overcome the intermittent connectivity of urban grids in off-peak hours, which precludes the exploitation of conventional search/propagation techniques based on ad hoc multicast and broadcast. We have shown that MDHP protocols are disruption

tolerant, scalable, and nonintrusive via both analytic models and extensive simulations. MobEyes can be configured to achieve the most suitable tradeoff between latency/completeness and overhead by properly choosing primarily its k-hop relay scope and the number of harvesting agents. These encouraging results are stimulating further research activities. In particular, we have been extending the MobEyes prototype to determine the best trajectory of mobile agents when collaborating with summary harvesting. In addition, we have formally been investigating on how we can determine the optimal value for the summary advertisement period, depending on node speed/population, and on urban monitoring requirements about traffic/latency. We will explore hybrid strategies that combine broadcast with epidemic dissemination, even by dynamically adapting to urban density conditions and application needs.

ACKNOWLEDGMENT

The authors would like to thank X. Hong and J. Kong for giving their valuable time to review the security section of this paper and G. Galante for reviewing an earlier version of this paper.

REFERENCES

[1] A. Nandan, S. Das, G. Pau, M. Gerla, and M. Y. Sanadidi, "Co-operative downloading in vehicular ad-hoc wireless networks," in *Proc. IEEE WONS*. St. Moritz, Switzerland, Jan. 2005, pp. 32–41.

[2] A. Nandan, S. Tewari, S. Das, G. Pau, M. Gerla, and L. Kleinrock, "AdTorrent: Delivering location cognizant advertisements to car networks," in *Proc. IFIP WONS*, Les Menuires, France, Jan. 2006.

[3] Q. Xu, T. Mak, J. Ko, and R. Sengupta, "Vehicle-to-vehicle safety messaging in DSRC," in *Proc. ACM VANET*, Philadelphia, PA, Oct. 2004, pp. 19–28.

[4] J. Ott and D. Kutscher, "A disconnection-tolerant transport for drive-thru Internet environments," in *Proc. IEEE INFOCOM*, Miami, FL, Apr. 2005, pp. 1849–1862.

[5] U. Lee, E. Magistretti, B. Zhou, M. Gerla, P. Bellavista, and A. Corradi, "Efficient data harvesting in mobile sensor platforms," in *Proc. IEEE PerSeNS*, Pisa, Italy, Mar. 2006, p. 352.

[6] *MIT's CarTel Central*. [Online]. Available: http://cartel.csail.mit.edu/

[7] B. Hull, V. Bychkovsky, K. Chen, M. Goraczko, A. Miu, E. Shih, Y. Zhang, H. Balakrishnan, and S. Madden, "CarTel: A distributed mobile sensor computing system," in *Proc. ACM SenSys*, Boulder, CO, Oct./Nov. 2006, pp. 125–138.

[8] *InternetCAR Project*. [Online]. Available: http://www.icar.wide.ad.jp

[9] T. Ernst, K. Mitsuya, and K. Uehara, "Network mobility from the InternetCAR perspective," *J. Interconnection Netw.*, vol. 4, no. 3, pp. 329–343, Sep. 2003.

[10] *FleetNet*. [Online]. Available: http://www.et2.tu-harburg.de

[11] W. Enkelmann, "FleetNet—Applications for intervehicle communication," in *Proc. IEEE IV*, Columbus, OH, Jun. 2003, pp. 162–167.

[12] U. Lee, J.-S. Park, E. Amir, and M. Gerla, "FleaNet: A virtual market place on vehicular networks," in *Proc. IEEE V2VCOM*, San Francisco, CA, Jul. 2006, pp. 1–8.

[13] M. Caliskan, D. Graupner, and M. Mauve, "Decentralized discovery of free parking places," in *Proc. ACM VANET*, Los Angeles, CA, Sep. 2006, pp. 30–39.

[14] D. Sormani, G. Turconi, P. Costa, D. Frey, M. Migliavacca, and L. Mottola, "Towards lightweight information dissemination in intervehicular networks," in *Proc. ACM VANET*, Los Angeles, CA, Sep. 2006, pp. 20–29.

[15] M. Torrent-Moreno, D. Jiang, and H. Hartenstein, "Broadcast reception rates and effects of priority access in 802.11-based vehicular ad-hoc networks," in *Proc. ACM VANET*, Philadelphia, PA, Oct. 2004, pp. 10–18.

[16] G. Korkmaz, E. Ekici, F. Ozguner, and U. Ozguner, "Urban multihop broadcast protocol for intervehicle communication systems," in *Proc. ACM VANET*, Philadelphia, PA, Oct. 2004, pp. 76–85.

[17] Z. Da Chen, H. T. Kung, and D. Vlah, "Ad hoc relay wireless networks over moving vehicles on highways," in *Proc. ACM MOBIHOC*, Long Beach, CA, Oct. 2001, pp. 247–250.

[18] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for vehicle-based disruption-tolerant networks," in *Proc. IEEE INFOCOM*, Barcelona, Spain, Apr. 2006, pp. 1–11.

[19] *UMass DieselNet*. [Online]. Available: http://prisms.cs.umass.edu/dome/

[20] J. Zhao and G. Cao, "VADD: Vehicle-assisted data delivery in vehicular ad hoc networks," in *Proc. IEEE INFOCOM*, Barcelona, Spain, Apr. 2006, pp. 1–12.

[21] H. Wu, R. Fujimoto, R. Guensler, and M. Hunter, "MDDV: A mobility-centric data dissemination algorithm for vehicular networks," in *Proc. ACM VANET*, Philadelphia, PA, Oct. 2004.

[22] S. B. Eisenman, G.-S. Ahn, N. D. Lane, E. Miluzzo, R. A. Peterson, and A. T. Campbell, "MetroSense project: People-centric sensing at scale," in *Proc. ACM WSW*, Boulder, CO, Oct./Nov. 2006.

[23] O. Riva and C. Borcea, "The urbanet revolution: Sensor power to the people!" *IEEE Pervasive Comput.*, vol. 6, no. 2, pp. 41–49, Apr.–Jan. 2007.

[24] M. D. Dikaiakos, S. Iqbal, T. Nadeem, and L. Iftode, "VITP: An information transfer protocol for vehicular computing," in *Proc. ACM VANET*, Cologne, Germany, Sep. 2005, pp. 30–39.

[25] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, "Participatory sensing," in *Proc. ACM WSW*, Boulder, CO, Oct./Nov. 2006, pp. 1–5.

[26] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet," in *Proc. ACM ASPLOS-X*, San Jose, CA, Oct. 2002, pp. 96–107.

[27] T. Small and Z. J. Haas, "The shared wireless infostation model: A new ad hoc networking paradigm (or where there is a whale, there is a way)," in *Proc. ACM MOBIHOC*, Annapolis, MD, Jun. 2003, pp. 233–244.

[28] University of Dartmouth MetroSense. [Online]. Available: http://metrosense.cs.dartmouth.edu/

[29] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: Modeling and analysis of a three-tier architecture for sparse sensor networks," *Elsevier Ad Hoc Netw. J.*, vol. 1, no. 2/3, pp. 215–233, Sep. 2003.

[30] Q. Li and D. Rus, "Sending messages to mobile users in disconnected ad-hoc wireless networks," in *Proc. ACM MOBICOM*, Boston, MA, Aug. 2000, pp. 44–55.

[31] Y. Wang and H. Wu, "DFT-MSN: The delay/fault-tolerant mobile sensor network for pervasive information gathering," in *Proc. INFOCOM*, Barcelona, Spain, Apr. 2006, pp. 1–12.

[32] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan, "IrisNet: An architecture for a worldwide sensor web," *Pervasive Comput.*, vol. 2, no. 4, pp. 22–33, Oct.–Dec. 2003.

[33] S. Nath, J. Liu, and F. Zhao, "Challenges in building a portal for sensors world-wide," in *Proc. ACM WSW*, Boulder, CO, Oct./Nov. 2006.

[34] *CENS' Urban Sensing*. [Online]. Available: http://research.cens.ucla.edu/projects/2006/Systems/Urban Sensing/

[35] B. H. Bloom, "Space/Time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[36] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Math.*, vol. 1, no. 4, pp. 422–426, 2003.

[37] L. Fan, P. Cao, and J. Almeida, "Summary cache: A scalable wide-area web cache sharing protocol," in *Proc. ACM SIGCOMM*, Vancouver, BC, Canada, Aug./Sep. 1998, pp. 254–265.

[38] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," in *Proc. INFOCOM*, Hong Kong, Mar. 2004, pp. 2446–2457.

[39] J. W. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed content delivery across adaptive overlay networks," in *Proc. ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002, pp. 47–60.

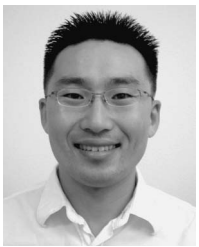
[40] P. Reynolds and A. Vahdat, "Efficient peer-to-peer keyword searching," in *Proc. Middleware*. Rio de Janeiro, Brazil, Jun. 2003, p. 997.

[41] L. Dlagnekov and S. Belongie, "Recognizing cars," Dept. Comput. Sci. Eng., Univ. California, San Diego, San Diego, CA, Tech. Rep. CS2005-0833, 2005.

[42] P. Bellavista, E. Magistretti, U. Lee, and M. Gerla, "Standard integration of sensing and opportunistic diffusion for urban monitoring in vehicular sensor networks: The MobEyes architecture," in *Proc. IEEE ISIE*, Vigo, Spain, Jun. 2007, pp. 2582–2588.

[43] M. Grossglauser and M. Vetterli, "Locating nodes with EASE: Last encounter routing in ad hoc networks through mobility diffusion," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar./Apr. 2003, pp. 1954–1964.

[44] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. ACM MOBICOM*, Boston, MA, 2000, pp. 56–67.
[45] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris, "A scalable location service for geographic ad hoc routing," in *Proc. ACM MOBICOM*, Boston, MA, 2000, pp. 120–130.
[46] C. Bettstetter, G. Resta, and P. Santi, "The node distribution of the random waypoint mobility model for wireless ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 2, no. 3, pp. 257–269, Jul.–Sep. 2003.
[47] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Performance analysis of mobility-assisted routing," in *Proc. ACM MOBIHOC*, Florence, Italy, May 2006, pp. 49–60.
[48] F. Bai and A. Helmy, "Impact of mobility on mobility-assisted information diffusion (MAID) protocols," Univ. Southern Calif., Los Angeles, CA, Jul. 2005. Tech. Rep.
[49] *ns-2 (The Network Simulator)*. [Online]. Available: http://www.isi.edu/nsnam/ns/
[50] B. Zhou, K. Xu, and M. Gerla, "Group and swarm mobility models for ad hoc network scenarios using virtual tracks," in *Proc. IEEE MILCOM*, Monterey, CA, Oct./Nov. 2004, pp. 289–294.
[51] U. S. Census Bureau, *TIGER, TIGER/Line and TIGER-Related Products*. [Online]. Available: http://www.census.gov/geo/www/tiger/
[52] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. G. Jetcheva, "A performance comparison of multihop wireless ad hoc network routing protocols," in *Proc. ACM MOBICOM*, Dallas, TX, Oct. 1998, pp. 85–97.
[53] A. Kumar Saha and D. B. Johnson, "Modeling mobility for vehicular ad-hoc networks," in *Proc. ACM VANET*, Philadelphia, PA, Oct. 2004, pp. 91–92.
[54] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Piscataway, NJ: IEEE Press, 1996.
[55] U. Lee, J.-S. Park, E. Amir, and M. Gerla, "FleaNet: A virtual market place on vehicular networks," in *Proc. V2VCOM*, San Jose, CA, Jul. 2006.
[56] U. Lee, E. Magistretti, B. Zhou, M. Gerla, P. Bellavista, and A. Corradi, "MobEyes: Smart mobs for urban monitoring with vehicular sensor networks," *Wireless Commun.*, vol. 13, no. 5, pp. 51–57, Sep./Oct. 2006.
[57] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proc. MobiCom*, Seattle, WA, Aug. 1999, pp. 151–162.
[58] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wirel. Commun. Mobile Comput.*, vol. 2, no. 5, pp. 483–502, Aug. 2002.
[59] M. Raya and J.-P. Hubaux, "The security of vehicular ad hoc networks," in *Proc. SASN*, Alexandria, VA, Nov. 2005, pp. 11–21.
[60] U. Lee, E. Magistretti, B. Zhou, M. Gerla, P. Bellavista, and A. Corradi, "Dissemination and harvesting of urban data using vehicular sensing platforms," Dept. Comput. Sci., Univ. Calif., Los Angeles, Los Angeles, CA, 2007. Tech. Rep.
[61] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in *Proc. MOBICOM*, Atlanta, GA, Sep. 2002, pp. 12–23.
[62] S. Tanachaiwiwat and A. Helmy, "Correlation analysis for alleviating effects of inserted data in wireless sensor networks," in *Proc. MobiQuitous*, San Diego, PA, Jul. 2005, pp. 97–108.
[63] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks," in *Proc. IEEE Symp. Security Privacy*, Oakland, CA, May 2004, pp. 259–271.
[64] P. Golle, D. Greene, and J. Staddon, "Detecting and correcting malicious data in VANETs," in *Proc. VANET*, Philadelphia, PA, Oct. 2004, pp. 29–37.
[65] R. Ostrovsky and W. Skeith, "Private searching on streaming data," in *Proc. CRYPTO*, Santa Barbara, CA, Aug. 2005, pp. 223–240.

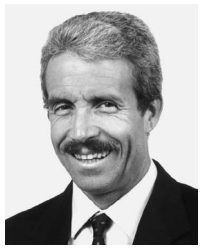


Uichin Lee is currently working toward the Ph.D. degree with the Department of Computer Science, University of California at Los Angeles (UCLA).

His research interests include mobile wireless sensor networks (e.g., vehicular/underwater sensors), delay-tolerant networks, and wireless vehicular applications.

Eugenio Magistretti received the M.S. and Ph.D. degrees in computer engineering from the University of Bologna, Bologna, Italy, in 2003 and 2007, respectively. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, Rice University, Houston, TX.

His research interests include protocols and algorithms for wireless networks, mobile ad hoc networks, and sensor networks.



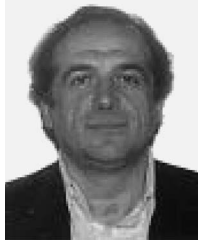
Mario Gerla received the M.S. degree in engineering from the Politecnico di Milano, Milano, Italy, in 1966 and the M.S. and Ph.D. degrees in engineering from the University of California at Los Angeles (UCLA) in 1970 and 1973, respectively.

From 1973 to 1976, he was with the Network Analysis Corporation, New York. He is currently a Professor with the Department of Computer Science, UCLA. His research interests include distributed computer communication systems and wireless networks. He has designed and implemented various network protocols (e.g., channel access, clustering, routing, and transport) under grants from the Defense Advanced Research Projects Agency and the National Science Foundation. He also leads the Office of Naval Research-supported MINUTEMAN Project at UCLA, which focuses on robust scalable network architectures for unmanned intelligent agents in defense and homeland security scenarios. He currently conducts research on scalable TCP transport for the Next-Generation Internet (see www.cs.ucla.edu/NRL) for recent publications.

Paolo Bellavista (SM'06) received the Ph.D. degree in computer science engineering from the University of Bologna, Bologna, Italy, in 2001.

He is currently an Associate Professor of computer engineering at the Department of Electronics, Computer Sciences and Systems, University of Bologna. He serves on the Editorial Board of the *Springer Journal of Network and Systems Management*. His research interests include middleware for mobile computing, location/context-aware services, adaptive multimedia and vehicular sensor networks, and mobile agent technologies.

Dr. Bellavista is a member of the Association for Computing Machinery and the Institute for Computer Sciences and Technology. He serves on the Editorial Board of *IEEE Communications Magazine* and the IEEE TRANSACTIONS ON SERVICES COMPUTING.



Antonio Corradi (M'80) received the *Laurea* degree in electronic engineering from the University of Bologna, Bologna, Italy, and the M.S. degree in electrical engineering from Cornell University, Ithaca, NY.

He is currently a Full Professor of computer engineering with the Department of Electronics, Computer Sciences, and Systems, University of Bologna. His research interests include distributed and parallel systems and solutions, middleware for pervasive and heterogeneous computing, infrastructure support for context-aware multimodal services, network management, and mobile agent platforms. He is a member of the Association for Computing Machinery and the Associazione Italiana per l'Informatica ed il Calcolo Automatico.