Dissemination and Harvesting of Urban Data using Vehicular Sensing Platforms

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

Abstract— **Recent advances in vehicular communications make it possible to realize vehicular sensor networks, i.e., collaborative environments where mobile vehicles equipped with sensors of different nature (from toxic detectors to still/video cameras) inter-work 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 permit 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, different from 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 inapplicable already known solutions for sensor network data reporting. The paper describes MobEyes, an effective middleware specifically designed for proactive urban monitoring, that 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 have demonstrated their effectiveness in terms of indexing completeness, harvesting time, and overhead. In particular, the paper includes i) analytic models for MobEyes protocol performance and their consistency with simulation-based results, ii) evaluation of performance as a function of vehicle mobility, iii) effects of concurrent exploitation of multiple harvesting agents with single/multi-hop communications, iv) evaluation of network overhead and overall system stability, and v) MobEyes validation in a challenging urban tracking application where the police reconstructs the movements of a suspicious driver, say, by specifying the car license number.**

I. Introduction

Vehicular Ad Hoc Networks (VANETs) are acquiring commercial relevance because of recent advances in inter-vehicular communications and decreasing costs of related equipment. This is stimulating a brand new family of visionary services for vehicles, from entertainment applications to tourist/advertising information, from driver safety to opportunistic transient connectivity to the fixed Internet infrastructure [1], [2], [3], [4]. In particular, Vehicular Sensor Networks (VSN) are emerging as a new tool for effectively monitoring the physical world, especially in urban areas where a high concentration of vehicles equipped with on board sensors is expected [5]. Vehicles are typically not affected by strict energy constraints and can be easily equipped with powerful processing units, wireless transmitters, and sensing devices even of some complexity, cost, and weight (GPS, chemical spill detectors, still/video cameras, vibration sensors, acoustic detectors, etc.). VSN represent a significantly novel and challenging deployment scenario, considerably different from more traditional wireless sensor network environments, thus requiring innovative specific solutions. In fact, differently from 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 important, they can host sensors that may generate huge amounts of data, such as multimedia video streams, thus making impractical the instantaneous data reporting solutions of conventional wireless sensor networks.

VSN 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 permit 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 9/11 and the London bombings, VSN could have actually helped emergency recovery and forensic investigation/criminal apprehension. In the London bombings police agents were able to track some of the suspects in the subway using closed-circuit TV cameras, but they had a hard time finding helpful evidence from the double-decker bus; this has motivated the installation of more cameras in fixed locations along London streets. VSN 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, used for a bank robbery, in order to identify thieves, say. 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 recorded by fixed cameras. As for privacy, let us briefly note that people are willing to sacrifice privacy and to accept a reasonable level of surveillance when the data can be collected and processed only by recognized authorities (with a court order), for forensic purposes and/or for counteracting terrorism and common crimes.

As shown by the above examples, the reconstruction of a crime and, more generally, the forensic investigation of an event monitored by VSN require the collection, storage, and retrieval of massive amounts of sensed data. This 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 collected by video sensors to a police authority sink because of sheer volume. Moreover, input filtering is not possible because a priori nobody knows 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 caused by mobility and attacks.

To that purpose, we have designed and implemented MobEyes, a novel middleware that supports VSN-based proactive urban monitoring applications. MobEyes exploits wireless-enabled vehicles equipped with video cameras and a variety of sensors to perform event sensing, processing/classification of sensed data, and inter-vehicle ad hoc message routing. Since it is impossible to directly report the sheer amount of sensed data to the authority, MobEyes keeps sensed data in mobile node storage; on board 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; 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 exploit intrinsic vehicle mobility and simple single-hop inter-vehicle communications. In that way, MobEyes harvesting agents can create a low-cost opportunistic index to query the distributed sensed data storage, thus enabling us to answer questions such as: which vehicles were in a given place at a given time? which route did a certain vehicle take in a given time interval?, and 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 MobEyes vehicular sensing architecture. We synthesize the existing techniques to build a MobEyes system that satisfies the key design principles, namely disruption tolerance, scalability, and non-intrusiveness.
- We propose an analytic model that can accurately predict MobEyes performance, and formally show that our protocol is scalable and non-intrusive.
- We provide extensive simulation results as follows: i) evaluation of performance dependence on vehicle mobility models, ii) effects of concurrent exploitation of multiple harvesting agents with single/multi-hop communications, iii) evaluation of network overhead and overall system stability, and iv) MobEyes validation in a challenging urban tracking application where the police obtain the route followed by a car by simply specifying its plate number.
- We overview the primary security requirements stemming 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 permits to enable/disable them at deployment time depending on the required degree of security/privacy.

The rest of the paper is organized as follows. Section II describes background and related work, by positioning the original MobEyes contributions. Section III presents the overall MobEyes architecture, while 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. Finally, Section VIII concludes the paper.

II. Background and Related Work

The idea of embedding sensors in vehicles is very novel. To our knowledge, the only research project dealing with similar issues is MIT's CarTel [6], [7]. In CarTel users submit their queries about sensed data on a portal 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 in the proximity of open access points to the Internet. Differently from CarTel, MobEyes exploits mobile collector agents instead of relying on the wired Internet infrastructure, thus improving robustness. Related work has recently emerged in two other related fields: VANET and "opportunistic" sensor networks. Finally, to permit the 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 is envisioning a large number of applications specifically designed for VANET, including: (i) safe cooperative driving where emergency information is diffused to neighbor vehicles and real-time response is required to avoid accidents [3]; (ii) entertainment support, e.g., content sharing [1], advertisements [2], peer-to-peer marketing [8]; and (iii) distributed data collection, e.g., parking lot [9] or traffic congestion info [10]. 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 located close to the sender, with high delivery rate and short delay. Recent research addressed this issue by proposing original broadcast strategies [11], [3]. However, single-hop broadcast does not provide full support to advertising applications; effective multi-hop dissemination solutions should be also investigated [12].

Packet delivery issues in areas with sparse vehicles have stimulated several recent research contributions to investigate carry-and-forward strategies. In [13], authors simulate a straight highway scenario to compare two ideal strategies: pessimistic (i.e., synchronous), where sources send packets to destinations only as soon as a multi-hop path is available, and 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 i) unbounded message buffers and bandwidth, and ii) of easily predictable mobility patterns as for vehicles on a highway, the latter has demonstrated to achieve a lower delivery delay. However, in more realistic situations, carry-and-forward protocols call for careful design and tuning. MaxProp [14], part of the UMass DieselNet project [15], is a ranking strategy to determine packet delivery order when node encounters occasionally occur, as well as dropping priorities in the case of full buffers. Precedence is given to packets 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. VADD [16] rests on the assumption that most node encounters happen in intersection areas. Effective decision strategies are proposed, highly reducing packet delivery failures and delay.

Applications for distributed data collection in VANET call for geographic dissemination strategies that deliver packets to all nodes belonging to target remote areas, despite possibly interrupted paths [17], [10]. MDDV [17] 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, [10] proposes several strategies based on virtual potential fields 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 [18], [19], [20], [21]. Several of these networks are relevant to our research because they can easily implement opportunistic dissemination protocols [22], [23].

Dartmouth's MetroSense [24], [18] is closely related to MobEyes. [18] describes a three tier architecture for MetroSense: servers in the wired Internet are in charge of storing/processing sensed data; Internet-connected stationary Sensor Access Points (SAP) act as gateways between servers and mobile sensors (MS); MS move in the field opportunistically delegating tasks to each other, and "muling" [25], [26] 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 (DFT-MSN) for human-oriented pervasive information gathering [27]. The trade-off 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, distributed index collection, etc.

Application-level protocols for the resolution of queries to sensed data have been proposed in [19], [20]. 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; finally, Migratory Services are components that react to changing context, e.g., the target moving out of range by migrating to other nodes [19]. [20] 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,; nodes in the target area form a virtual ad hoc query server.

Among recent research projects about opportunistic sensing, we mention Intel IrisNet [28] and Microsoft SenseWeb [29]. Both projects investigate the integration of heterogeneous sensing platforms in the Internet via a common data publishing architecture. Finally, we point out CENS' Urban Sensing project [30], [21], a recently started multi-disciplinary project addressing "participatory" sensing, where urban monitoring applications receive data from mobile sensors operated by people.

Finally, regarding dissemination of sensed data through peers, we can mention two solutions from the naturalistic environment, namely ZebraNet [22] and SWIM [23]. ZebraNet addresses remote wildlife tracking, e.g., zebras in 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: a flooding-based approach where zebras exchange all the data within their buffers (either locally generated or received from other animals) with neighbors, and a history-based protocol where data is uploaded only to zebras with good track record of base station encounters. SWIM [23] addresses sparse mobile sensor networks with fixed Infostations as collecting points. Sensed data is epidemically disseminated via single-hop flooding to encountered nodes and offloaded when Infostations are in reach.

While all the above schemes implement peer-to-peer dissemination, none fits the performance requirements of the MobEyes target scenario. Flooding generates excessive overhead, while 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 [22]. In addition, let us note that MobEyes nodes do not transmit raw collected data but, thanks to abundance of on board computing resources, they locally process data and relay only short summaries. That is a relevant advantage since data collected in VSN, 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 [31] 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, $S = \{s_1, s_2, \cdots, s_\omega\}$, consists of m bits, which are initially set to 0. The filter uses ℓ independent random hash functions $h_1, \cdots, h_\ell$ 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 been widely used for applications where "space saving" outweighs the drawback of errors.

Bloom filters have gained momentum in networking fields by virtue of 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 [32]. Bloom filters are mostly used for: simple membership checking [33], [34], set difference (or reconciliation) [35], and set intersection [36]. Fan et al. proposed a Summary Cache where a set of distributed Web proxies use Bloom filters to disseminate the content of their cache [33]. Similarly, Ye et al. exploited them to store MAC addresses for simple membership checking in wireless sensor networks [34]. For set reconciliation, Byers et al. proposed a method that uses a comparison tree over a Bloom filter [35], 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 [36].

III. MobEyes Architecture

For the sake of clarity, we present the MobEyes solution using one of its possible practical application scenarios: collecting information from MobEyes-enabled vehicles about criminals that spread poisonous chemicals in a particular section of the city (say, 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* 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 in order to create a distributed metadata index, useful for forensic purposes such as crime scene reconstruction and criminal tracking.

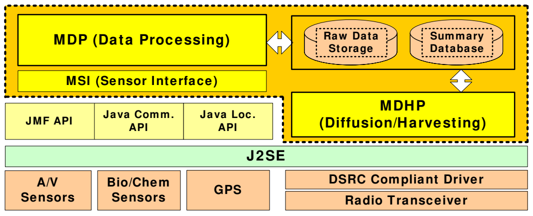


Fig. 1. MobEyes sensor node architecture.

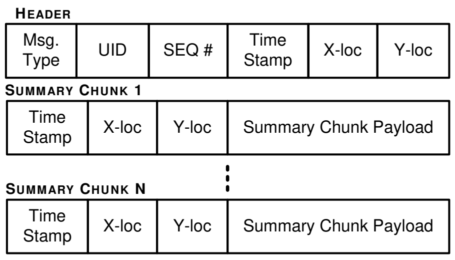


Fig. 2. Packet format: a single summary packet contains multiple *summary chunks*.

To support all the above tasks, we have developed MobEyes according to the component-based architecture depicted in Figure 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). Since vehicles are not strictly resource-constrained, our MobEyes prototype is built on top of the Java Standard Edition (J2SE) virtual machine. MDP is in charge of reading raw sensed data (via MSI), processing it, and generating chunks. Chunks include metadata (vehicle position, timestamp, vehicle ID number and possible additional context such as simultaneous sensor alerts) and features of interest extracted by local filters (See Figure 2). For instance, in the above application scenario, MDP includes a filter determining license plate numbers from multimedia flows taken by cameras [37]. Finally, MDP 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 the sake of effectiveness. Therefore, the generation rate and 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 [38].

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 6B, sensed data with 10B (e.g., concentrations of potential toxic agents), timestamp with 2B, and vehicle location with 5B. Then, in our scenario, MDP can pack 65 chunks in a single 1500B summary, even without exploiting any data aggregation or encoding technique. In usual deployment environments chunks are generated every [2-10] seconds and, thus, a single summary can include all the chunks about a [2-10] minute interval. MSI permits MDHP to access raw sensed data independently of actual sensor implementation, thus simplifying the integration with many different types of sensors. MSI currently implements methods to access 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) API, the Sun Communication API, and the JSR179 Location API.

IV. MobEyes Diffusion/Harvesting Processor

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

A. MDHP Protocol Design Principles

A vehicular sensing platform, built on top of a vehicular ad hoc network, has the following specific characteristics that differentiates it from more established and investigated deployment scenarios. First, it has unique mobility patterns. Vehicles move at relatively high speed (e.g., up to 80mph) 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, work place, etc.) by following their preferred paths. The dynamic behavior of mobile nodes, i.e., join/leave/failure, usually result 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, non-uniform 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 hundreds of thousands vehicles because sensing applications primarily target urban environments. Finally, 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 must be able to operate even with disruptions (caused by sparse network connectivity, obstacles, and non-uniform vehicle distribution) and with arbitrary delays. High churning of vehicles must be considered; for robustness purposes, data replication is a must.
- *Scalability*: MDHP protocols must be able to scale up to hundreds of thousands nodes (e.g., the number of vehicles potentially interworking in a large city).
- *Non-intrusiveness*. 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; limiting the use of bandwidth below a certain threshold is imperative.

Given the above 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., non-delay-tolerant) and cause the network traffic to scale with the number of nodes in the network (i.e., non-scalable, intrusive). For instance, in Epidemic Data Dissemination (EDD) where data is spread whenever connectivity is available (i.e., data is replicated without any restriction), the size of exchanged data scales with network size; thus, EDD is intrusive and non-scalable. EDD is more suitable for sparse and small-scale wireless networks. Note that the formal analysis corroborating 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 non-intrusive and scalable.

B. Summary Diffusion

By following the above guidelines, in MobEyes any regular node periodically advertises a new packet with generated summaries to its current neighbors in order to increase the opportunities for agents to harvest summaries. Clearly, excessive advertising will introduce too much overhead (as in EDD), while 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

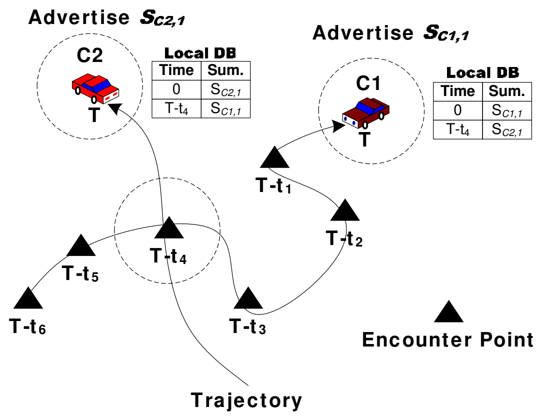


Fig. 3. MobEyes single-hop passive diffusion

in Figure 2, a packet header includes packet type, generator ID, locally unique sequence number, packet generation timestamp, and generator's current position. Each packet is uniquely identified by a <generator ID,sequence number> pair and contains a set of summaries locally generated during a fixed time interval.[1]

Neighbor nodes receiving 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 (*passive* diffusion). MobEyes can be configured to perform either single-hop passive diffusion (only the original source of the data advertises its packet to current single-hop neighbors) or k-hop *passive* diffusion (the packet travels up to k-hop as it is forwarded by j-hop neighbors with $j < k$). Other diffusion strategies could be easily included in MobEyes, for instance 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.

Figure 3 depicts the case of two sensor nodes, C1 and C2, that encounter 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 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 $\langle ID,\, Seq.\#\rangle$. 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 $S_{C1,1}$.

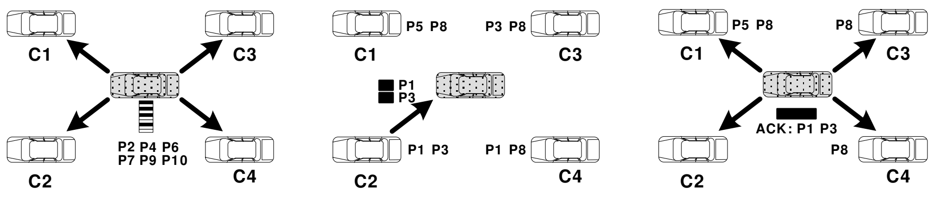
[1]The optimal interval can be determined by noting that the harvesting time distribution is characterized by average (μ) and standard deviation (ρ). Then, Chebyshev inequality, $P(|x - \mu| \geq k\rho) \leq \frac{1}{k^2}$ allows us to choose k such that we can guarantee the needed *harvesting latency*,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

Summary diffusion is time and location sensitive (spatial-temporal information diffusion). In fact, regular nodes keep track of freshness of summary packets by using a sliding window with a maximum window size (i.e., fixed expiration time). In addition, since a single summary packet may contain multiple summaries, 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 [39] and to maintain a geo-reference service that can be used to proactively access the data. That 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, namely *on-demand* mode and *proactive* (or background) mode. The on-demand mode is suitable for cases when the police agents react to an emergency call, for example, the earlier 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., time-space window). The agents can *flood* a query with such information (à la Route Request in on-demand routing). Each regular node resolves the query and returns its summary to the agents (à la Route Reply in on-demand routing). So, the *on-demand* strategy is more likely a traditional sensor network-based data harvesting protocol, e.g., Directed Diffusion [40] – i.e., the reply is "diffused" in the direction of the querier. The main difference in MobEyes would be that a query has a spatio-temporal range. The police agents as a team will collect as many summaries of interests as they can.

However, it is not very practical to exploit an on-demand strategy in MobEyes for the following reasons. First, agents should provide a query with the *range* of spatio-temporal 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 passing by the scene within the last several hours, resulting in tens of hundreds of vehicles. Second, the on-demand scheme is quite similar to conventional data harvesting and requires maintaining a "concast" tree from the query originator. But the number of



(a) Broadcast a harvest request (b) C2 first returns missing packets (c) Broadcast acknowledgement

Fig. 4. MobEyes proactive summary harvesting

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 non-trivial and not always possible due to intermittent connectivity and network partitions. To overcome intermittance, query/response can be opportunistically 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. Finally, in "covert" operations, agents may not want to broadcast a query (e.g., in order not to alert the criminals that are currently being pursued, say). 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. Namely, 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 collected by a "limited" number of vehicles. The vehicles can be contacted based on the originator's vehicle ID number stored in each summary. A message is sent to each vehicle requesting it to upload the file at the nearest police access point. Note that the request message can exploit georouting, either exploiting the Geo Location Service that maps vehicle ID to the current vehicle location or using "Last Encounter Routing" techniques [41], [39]. The latter is particularly convenient here because nodes memorize the time and place of encounters at the time 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 as it can speed up the search considerably. For instance, if the inspection of the information 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 to do another time consuming collection of summaries from vehicles. On the negative side, maintaining that index is costly, as agent resources must be dedicated to the task.

In the sequel we will assume proactive index construction. Thus, the agents collect all summaries indiscriminately. There is no loss of generality, however, since 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 being harvested. In the on-demand scheme, the target set is a specific space-time window. In the proactive scheme, the target set is the entire geographic area within agent responsibility; there is no limit on harvesting time, though 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 generated in the specified area. Obviously, a police node is interested in harvesting only summary packets 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 (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 and still valid summary packets. Since each summary has a <unique node ID, sequence number> pair, we use this as 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, neighbors update their lists of missing packets.
5) Steps 3 and 4 are repeated until all missing packets are sent.

[2]see also Section II

An example of summary harvesting is shown in Figure 4. The agent first broadcasts its Bloom filter related to the packets collected so far (P2, P4, P6, P7, P9 and P10) as in Figure 4(a). Each neighbor receives the filter and creates a list of missing packets. For example, C3 has P3 and P8 to return, while C4 has P1 and P8. In Figure 4(b), C2 is the first node to return missing packets (P1, P3) and the agent sends back an acknowledgement piggybacked with the list of received packets. Neighbor nodes overhear the message and update their lists: C3 and C4 both remove P1 from their lists, as depicted in Figure 4(c).

Note that membership checking in a Bloom filter is probabilistic and false positives are possible, even if rare (as rapidly shown in the following section). In Figure 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 pseudo random function (PRF) where a PRF takes two arguments, X_k is the key (k=$1, 2, \cdots, m$) and i is the input value, and produces an output value $o = F_{X_k}(i)$. All nodes are initially given the same set of keys X_k where k=$1, 2, \cdots, m$. For the purpose of periodic changes, the key for k-th hash function in n-th 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. Note also that in our application a set changes over time with summaries being inserted and deleted because summaries have spatio-temporal 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 [33]. 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: i.e., 1 if its value is greater than 0; 0 otherwise.

For the sake of simplicity, thus far we assumed that there is a single agent working to harvest summaries. Actually, MobEyes can handle concurrent harvesting by multiple agents (possibly several hops apart) that can cooperate by exchanging their Bloom filters among multi-hop routing paths; thus, this 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 the benefits in terms of latency and accuracy 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, instead, 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 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 passive harvesting was. Therefore, we model the progress of passive harvesting, from which we formulate the progress of active harvesting. Finally, we extend the model to analyze k-hop relay scope. We assume that there are N nodes moving within $L \times L$ m^2 area, and each node advertises a single summary packet (total N summary packets). For the sake of analysis, we use two different mobility models, both with uniform spatial node distribution, namely random direction and Manhattan mobility models. In the random direction mobility model, nodes move towards random directions (chosen out of [0, 2π]) at a speed of v on average. The Manhattan mobility model restricts node's mobility patterns along grids. Because a target direction is uniformly chosen in both cases, the steady state node distribution is uniform [42], [5] – thus, node density is *location independent*. The Manhattan mobility model exhibits uniform node distribution in the sense that nodes are uniformly distributed in the area where mobile nodes can go along. So, the node density could be simply expressed as $\rho = \delta N/L^2$, where δ is used to control the size of the area covered area by mobile nodes in the Manhattan mobility model and $\delta = 1$ in the random direction mobility model. The value δ for Manhattan mobility 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, let us say that given a $L \times L$ 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, $\delta = \frac{L \times L}{2R \times L} = \frac{L}{2R}$. Let v^* denote the average relative speed of nodes. As shown in [43], $v^* = \frac{v}{2\pi} \int_0^{2\pi} \sqrt{(1+\cos\theta)^2 + \sin^2\theta d\theta}$ $= 1.27v$. In general, we simply assume that the average relative speed can be modeled as proportional to the average speed: $v^* = cv$ where c is a constant.

By extending [44], 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 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 [25]. During the 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

$\alpha = \rho v \Delta t 2R$. Since each of these nodes will advertise its summaries, 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 collected by a regular node by time slot t. As described above, during time slot Δt a regular node will receive α summaries. Since the node has E_t summaries, 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 moving together along the same path in a Manhattan mobility model) will affect the *effective* number of neighbors. Since we are interested in the average behavior, 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 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 for a regular node to encounter any random node, thus receiving a summary from it. The cumulative distribution of random variable T can be derived by dividing Equation 2 by N.

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

From this, 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 $p = \frac{\alpha\eta}{N}$. The average is given as $\mathbb{E}[T] = \frac{1}{p} - 1 = \frac{N}{\alpha\eta} - 1$. 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 Equation 5, given a square area of L^2, the average time for a regular node to collect a summary is independent of node density. In fact, it is a function of average relative speed and communication range. Intuitively, as node density increases (N increases), a node can collect more summaries during a given time slot. However, higher density means that the total number of summaries to collect is higher. Thus, the two factors compensate each other.

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

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

Let H_t denote the expected number of distinct summaries harvested by the agent till time slot t. Since the agent has H_t summaries, the probability of acquiring a new summary is $1 - H_t/N$. Hence, multiplying this probability by the expected number of summaries harvested from neighbors (Equation 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 received from neighbors to consider the restricted mobility. Note that, as described before, restricted mobility such as in the Manhattan model reduces the rate of new encounters (adjusted by η) and also exerts baleful influence on the rate of active harvesting since neighbors tend to carry overlapping summaries (adjusted by γ).

From Equation 7, we see that H_t grows much faster than E_t. During a time slot, the number of collected summaries in Equation 2 is constant (α), whereas in Equation 7 it is a function of time. Moreover, Equation 6 is a function of the number of neighbors, i.e., related to node density. As N (or 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 above equations are based on the random motion model, but for restricted mobility models such as the Manhattan 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. As stated before, we are assuming a rectangular area, $\Delta t 2R$. Increasing by k-hop the relay scope is the same as multiplying the area by k times.

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

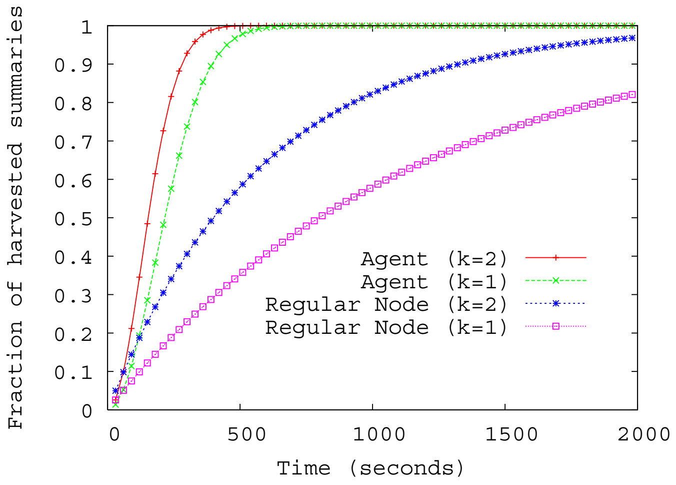


Fig. 5. Fraction of harvested summaries with $k = 1, 2$

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

B. Scalability

The feasibility of MobEyes strictly depends on its scalability over wide VSN, 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.

About 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 randomly sent within $[iT_a, (i+1)T_a)$ time slot for all i, where T_a is the advertisement period. So, the number of packets received by a node is bounded by the number of its neighbors while it is traveling for T_a, thus depending on node density but not on the overall number of nodes. In contrast, any "flooding"-based diffusion protocol is not scalable because a node could potentially receive a number of packets proportional to the network size.

To give a rough idea of the traffic generated by MobEyes diffusion, let us simply use $T_a = 2R/v^*$ (the time for a mobile node to traverse 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 be neither too short nor too long compared to 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 chances to send packets to encountered nodes, thus slowing down dissemination. In our target deployment environment, $v^* = 20m/s$, $R = 250m$, the advertisement period $T_a = 12.5s$, and the fixed packet size $S = 1500B$. Consequently, the transmission time for one packet is about $T_x = 1ms$. 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 (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} seconds. 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 $\frac{\vartheta T_{exp}}{T_x}$. As a result, the system can be modeled using an M/D/1 queue. The stability condition, $N\lambda' < \frac{\vartheta T_{exp}}{T_x}$, gives us the upper bound $N < \frac{\vartheta T_{exp}}{\lambda' T_x}$. 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 $T_{exp} = 250s$), that number is $N < \frac{0.01 \times 250}{2 \times 0.001} = 1,250$. As a consequence, in the case of node numbers equal or not far from 1,250, there is the need to deploy more than one harvesting agent to maintain the system stable, i.e., to be able to harvest summaries more rapidly than regular nodes generate them.

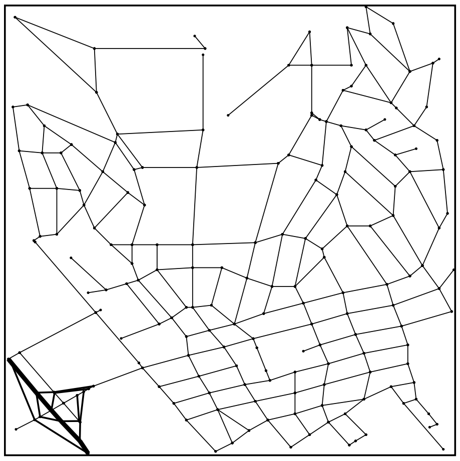


Fig. 6. Map of Westwood area in vicinity of UCLA campus

VI. MobEyes Performance Evaluation

We evaluated MobEyes protocols via extensive ns-2 [45] simulations. This section shows the most important results, with the goal of investigating MobEyes performance from the following perspectives:

1) *Analysis Validation*. We simulate MobEyes protocols for summary collection on regular nodes as well as 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 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 reported 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 Turn Over*. We show that MobEyes performance does not relevantly change even in the case of more complex 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 moving in a fixed region of size $2400m \times 2400m$. The default mobility model is Real-Track (RT), introduced by our colleagues in [46]. RT permits to model realistic vehicle motion in urban environments. In RT nodes move following virtual tracks, 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 UCLA campus, as obtained by the US Census Bureau data for street-level maps [47] (Figure 6). At any intersection, each node randomly selects the next track it will run through; speed is periodically allowed to change (increase or decrease) by a quantity uniformly distributed in the interval [0,$\pm\Delta s$]. To evaluate the impact of the adopted mobility model on MobEyes performance, thus quantitatively estimating the generality of the reported experimental results, we tested two additional well-known models, namely Manhattan (MAN) [44] and Random WayPoint (RWP) [48]. Similarly 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 deployment of mobile nodes. In our simulation, we adopted a 10×10 grid. RWP instead 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 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 [49]", generally producing limited distortion on protocol performance. Let us remark that MobEyes agents do not exploit any special trajectory or controlled mobility pattern, but move conforming with regular nodes.

Our simulations consider number of nodes $N = 100, 200, 300$. Vehicles move with average speed $v = 5, 15, 25$; 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 3s through all the simulations. We use a Bloom filter with 8192bits (1024B) and 10 hash functions. Since we have a large filter size (i.e., 8192bits) compared to the number of summaries, the false positive probability is negligible.[4] We note that if the value of this parameter is too large, MobEyes effectiveness is reduced since it is possible that two nodes do not exchange messages, even if they occasionally enter in each other transmission range; this effect is magnified, as node speed v increases. The chosen value has been experimentally determined to balance the effectiveness of our protocol and the message overhead, even in the worst case, i.e., $v = 25$. A deeper and more formal investigation of the optimal value of the advertisement period is object of future work.

Finally, we modeled communications as follows: IEEE 802.11 MAC protocol, 2.4 GHz transmission band, 11Mbps bandwidth, 250m nominal radio range, and Two-ray Ground propagation model [50]. The values of these parameters have been chosen similar to other work in the literature [16] [13]. Where not differently stated, reported results are average

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

values out of 35 repetitions. Other MobEyes configuration parameters will be introduced in the following sub-sections, when discussing the related aspects of MobEyes performance.

B. Analysis Validation

Our first goal is to validate the results obtained in Section V. In particular, we investigate the regular node collection and agent harvesting processes, as described respectively by Equations 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 it generated in the epoch $t = 0$ for the rest of the simulation run. Since Equations 2 and 7 characterize the spreading processes of all summaries generated in the same epoch, 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.

Figures 7 and Figure 8 show results collected for number of nodes $N = 100/300$, average speed $v = 5/25$, and all the three RWP, MAN, and RT mobility models. In particular, Figure 7 plots the cumulative distribution of summaries 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 node density as we have shown in Equation 4. Our analytic model (Equation 2) accurately fits the simulation results for RWP and MAN. The curves for the RT model exhibit slightly worst fitting; they start deviating from analytical results after certain thresholds (analytical results are not reported in Figure 7(c) and Figure 8(c) for the sake of figure readability). RWP shows slightly better accuracy mainly due to unconstrained motion of nodes and to their tendency to gather at the center of the field as time passes [42]. Although both MAN and RT show the restricted mobility patterns, their node distributions are different: MAN has almost uniform node distribution over the grids, whereas RT exhibits non-uniform node distribution over the map as shown in [51]. Since our analytic model is based on uniform node distribution, as expected, it fits slightly better the MAN simulation results. In addition, experimental results helped us to tune the constant compensation factor η we introduced in Section V to take into account the non-uniform movement patterns. In detail, the values we found are 0.97 and 0.9 respectively for $v = 15m/s$ and $v = 25m/s$ 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.

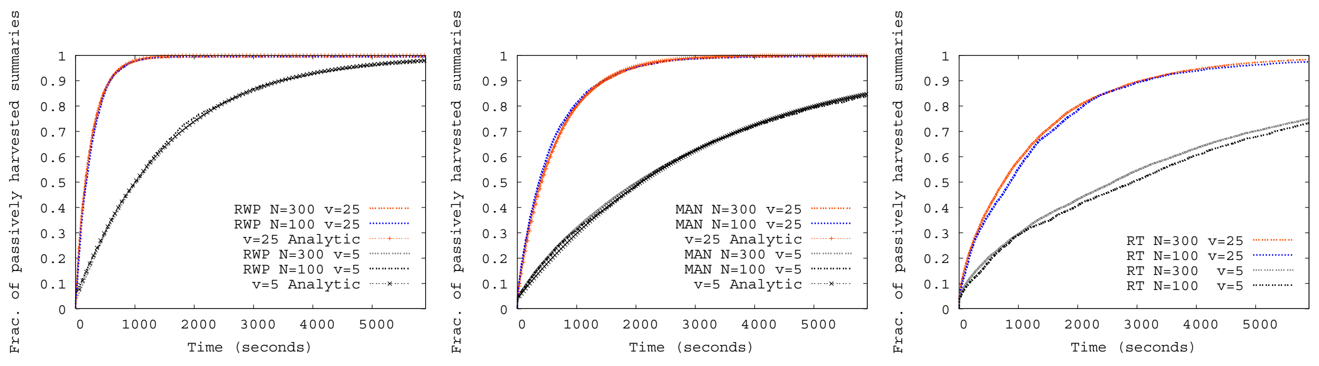
Figure 8 plots the cumulative distribution of summaries harvested by a police agent as a function of time. The figure shows that the results are mainly dependent on the speed. Unlike passive harvesting, the density plays an important role in active harvesting. In the analysis section, we show that the higher the density, the faster is the harvesting progress (Equation 7). Intuitively, if there are more neighbors, the agent has a higher chance of getting a random summary. Our analytic model fits well the simulations, especially when we have large N and v. This set of results allowed us to tune the parameter γ accounting for the effect of overlapping of summaries contributed from regular agents (see Section V). In detail, the values we found are 0.21 and 0.21 respectively for $v = 5m/s$ and $v = 25m/s$ 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 harvested by the agent(s) in 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 [52] we only investigated the RT mobility model; here, we extend the results also 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 re-broadcasts a newly received packet with probability $p = 0.5$.[5] The summary harvesting latency is a crucial figure to determine the feasibility of the MobEyes approach, since it allows us to estimate the fraction of harvested summaries by the agent within a certain time t. This estimation is useful to decide the tuning of the parameters (k-hop relay scope and number of agents) to address the application-specific requirements of services built on top of MobEyes. Figure 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. Figure 9 plots the cumulative distribution of the summaries harvested for $N = 300$, $v = 15m/s$. In the case of multiple agents, the harvesting process considers the union of the summary sets harvested by agents. The figure clearly shows that k-hop relay scope and multiple agents highly impact harvesting latency.

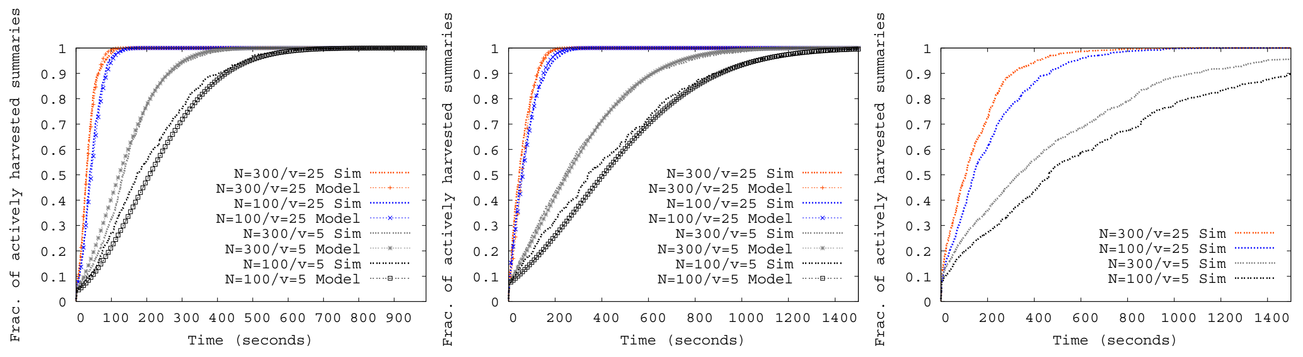
By carefully inspecting the results in Figure 9, it is possible to obtain some guidelines for the choice of MobEyes parameters. For example, given as a baseline a network with $N = 300$ nodes moving with an average speed $v = 15m/s$, fixed $k = 1$, a single agent employs $530s$, $236s$, and $116s$ to harvest 95% of the summaries generated respectively in RT, MAN, and RWP mobility models. By increasing k to 3, times respectively reduce to $420s$, $176s$, $86s$, showing an improvement of about $20-30\%$ in all cases. Instead, increasing the number of agents to three, times become respectively $280s$, $123s$, and $68s$; in this case, the improvement is in the $40-50\%$ range. Finally, if we set $v = 25m/s$, times become $211s$, $67s$, and $43s$; the improvement is around $60-70\%$. Interestingly, the relative impact of the three parameters (harvesting team size, multihop forwarding, and speed) shows a limited dependence on the mobility model. This holds also for the results we collected

[5]In the simulation, we use this 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 exploiting solutions for efficient broadcast, such as the ones proposed in [53].



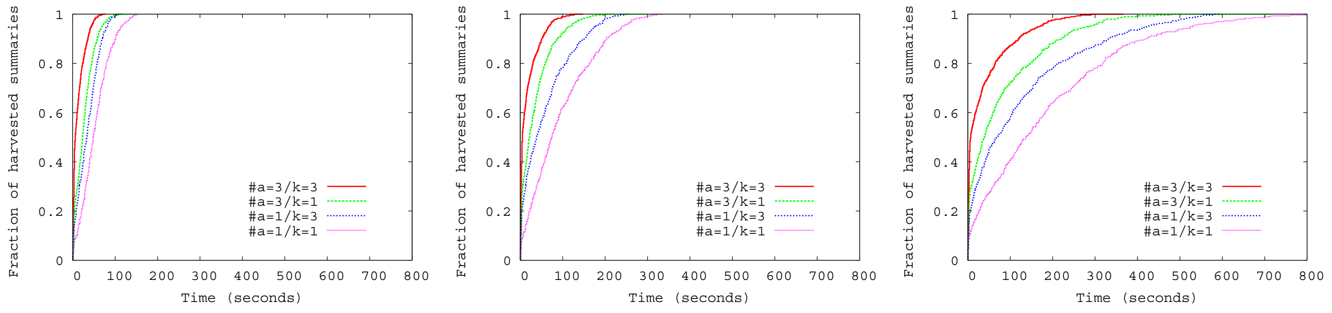
(a) RWP (b) MAN (c) RT

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

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

for different cases (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 imposed on the underlying vehicular communication technology and to determine if MobEyes can coexist with other applications. For example, the parameter k shows the largest impact on the performance; the effect due to a small number of agents is negligible, since they are only responsible for local single-hop traffic. Figure 10 shows the average received packets per node per second, obtained during a simulation time of $1000s$. In this set of simulations, we fixed $k = 1$, and changed all the other parameters, i.e., mobility model (RWP, MAN, RT), N (100, 200, 300), and v (5, 15, 25). As expected, the number of received packets linearly increases as the number of nodes increases. Therefore, for the sake of clarity, the figure only reports the case with $N = 300$. In addition, the number of received packets exhibits no dependence on v. In all considered cases, the overhead is limited, on the order of few (two to five) packets per second, proving the low impact of MobEyes on the available bandwidth.

The latter result could mislead to conclude that speed increments would not impact the harvesting latency, since the number of received packets would not change. This only apparently invalidates our previous results (see Figure 7) for

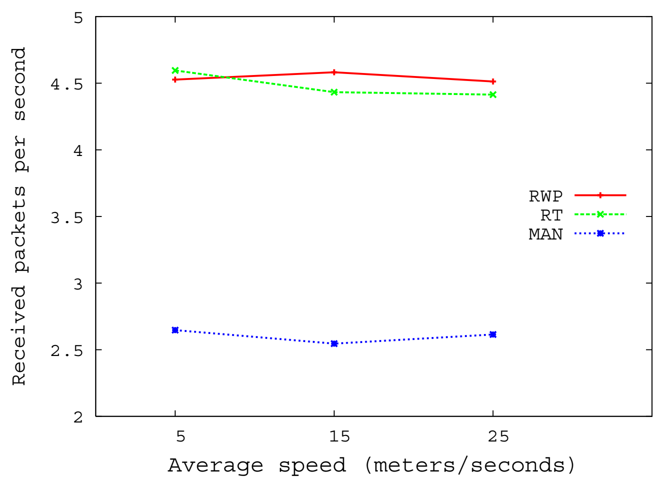


Fig. 10. Total number of received packets *(k=1)*

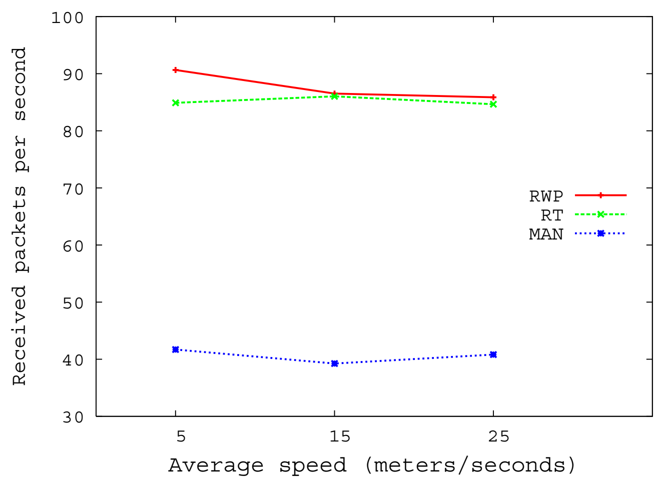


Fig. 11. Total number of received packets *(k=3)*

the reasons illustrated in the following. For a fixed advertisement interval, as average speed increases, the probability of useful meetings (i.e., of receiving a non-redundant summary) increases because there is more mixing among mobile nodes. For example, given an average speed v, let us assume that the average period during which any two nodes are within their communication ranges simply be $2R/2v$. Then with v set to 5 and 25 m/s, and $R = 250m$, the periods can be estimated as $50s$ and $10s$ respectively. This implies that the cases of $5m/s$ has roughly 5 times higher chances of receiving redundant advertisements than the case of $25m/s$. It is interesting to note that, fixed the average speed, there exists an optimal advertisement period allowing to maximize non-redundant summary diffusion, while minimizing the overhead. It will be part of our future work to analytically determine this value.

Figure 11 shows the magnifying effect produced by an increase of the parameter k. k-hop relaying produces an enlargement of the area where summary packets are diffused intuitively proportional to k^2. Consequently, also the number of nodes affected by a single summary diffusion will be about k^2 larger than the single-hop case. Moreover, while in the single-hop case nodes receive any summary packet only once, 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 larger than k^2 but lower than k^2 times the average number of neighbors (please note that k-hop distant nodes do not relay packets, thus reducing the latter factor for k-hop as well as $(k-1)$-hop distant nodes). The combination of these results with those in Figure 9 lead us to conclude that parameter k permits to decrease harvesting latency (about $20-30\%$ for $k=3$) at the price of relevant overhead increase (around $15-20$ times). The proper balance of latency/k tradeoff can be only decided depending on specific characteristics and requirements of the supported urban monitoring application.

E. Stability and Scalability Check

In the following, 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 harvested on longer periods remains acceptable and that the harvesting latency does not grow as time passes. With regard to the results presented so far, here we remove the assumption about the single summary generation epoch at $t = 0$. Nodes generate new summaries with period $T = 120s$ and advertise the last generated summary: let us observe that according to the discussion presented in Section III this rate represents a practical worst case. For the sake of clarity of presented results, 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 (especially related to results in Figure 13) are far more jumbled; anyway, they are available at the already referred MobEyes Web site. The following results are reported for the case of a single harvesting agent, $k = 1$, $N = 100$, $v = 15m/s$, and nodes moving according to the RT model.

Figure 12 plots the cumulative distribution of the number of summaries generated and harvested as a function of time (we ran simulations for $6000s$). 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. Figure 13 provides further evidence of the stability of the system; curves show the harvesting latency for summaries generated during some generation epochs. For the sake of figure clarity, the graph does not exhaustively represent every generation epoch, but only samples one generation epoch every $T * 7 = 840s$ till the end of the simulation time. The different curves show similar trends, without any performance degradation caused by the increase of the number of summaries in the network. The harvesting related to the

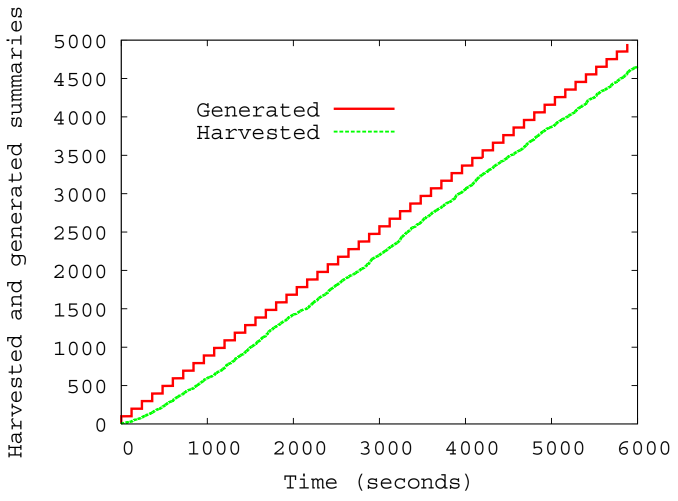


Fig. 12. Cumulative distribution of generated and harvested summaries *over all epochs*

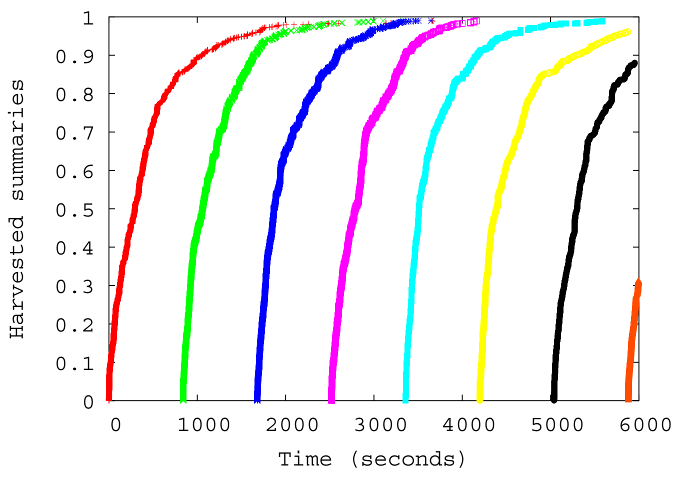


Fig. 13. Cumulative distribution of harvested summaries *per epoch*

last summary generation epoch is evidently incomplete (25% of the summaries are harvested within the timeline), since the epoch starts $120s$ before the end of the simulation.

In addition, we evaluate the most challenging scenario with N=1000 to quantitatively evaluate MobEyes scalability. Figure 14 shows that the harvesting latency is reduced considerably compared to the case with $N = 100$, thus confirming that the proposed protocol scales well. The result matches with the observation from our analytic model in Section V-A where we found that the harvesting delay decreases as the number of nodes increases.

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

F. Tracking Application

In the Introduction we sketched some application cases for MobEyes. For the sake of proving its effectiveness in

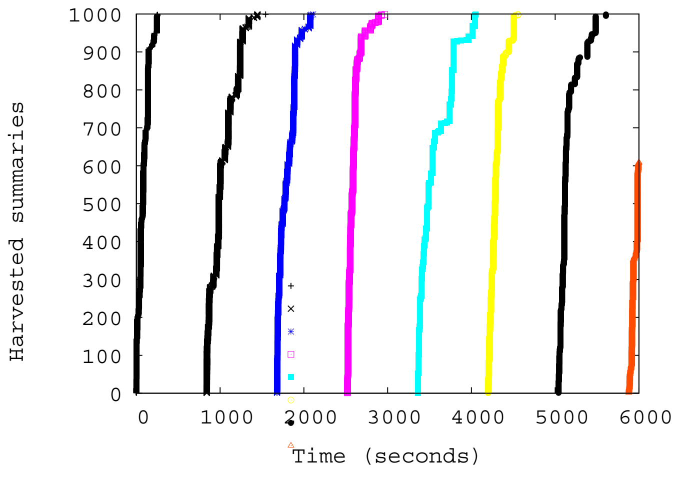


Fig. 14. Cumulative distribution of harvested summaries *per epoch* with 1000 nodes

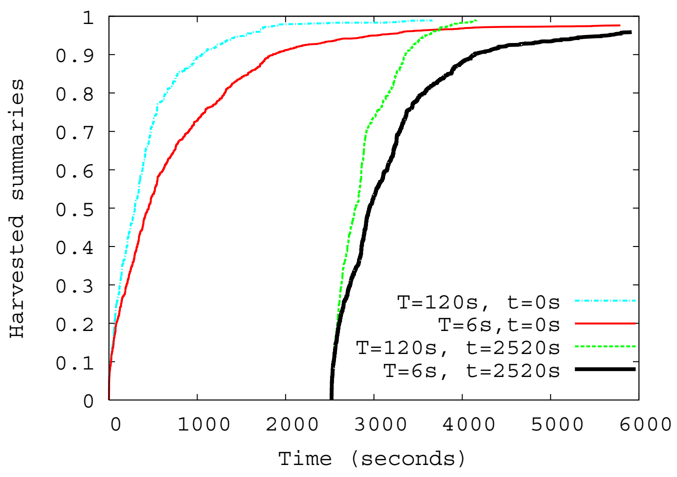


Fig. 15. Cumulative distribution of harvested summaries *per epoch*

supporting urban monitoring, we also simulated a vehicle tracking application where the agent reconstructs node trajectories exploiting the collected summaries. This is a challenging urban monitoring application, since it requires our system (1) to monitor a large number of targets, i.e., all participant vehicles, (2) to periodically generate fresh information on these targets, since they are highly mobile, (3) to deliver to the agent a high share of the generated information to well reconstruct the targeted node trajectory. Moreover, since 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.

More in details, as regular cars move in the field, they generate new summaries every $T = 120s$ and continuously advertise the last generated summary. Every summary contains 60 *summary chunks*, which are created every $ChunkPeriod = 2s$ and include the license plate and position of the vehicle nearest to the summary sender at the generating time, 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 scouting 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 is possible by aggregating data related to the same license plate, 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 and longest periods during which the agent does not have any record for that vehicle. The longest period typically represents situations in which a node moves in a zone where 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 Figure 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 $6000s$ simulation. The plot shows that in most cases the average uncovered interval floats between $[2.7s - 3.5s]$; the maximum uncovered interval shows that even in the worst case the agent has at least one sample every $200s$ for more than 90% of the participants. A more immediate visualization of the tracking accuracy is given in Figure 17. This figure shows, for the case of a node with a maximum uncovered interval equal to $200s$ (i.e., locating this node in the lowest $10th$ percentile), its real trajectory (the unbroken line), and the sample points the agent collected.

G. Border Effects and Turnover

Usual mobility models [54], such as RWP, MAN, and RT, assume that nodes remain within the simulated area during the entire simulation (in the following, we indicate them as *closed*

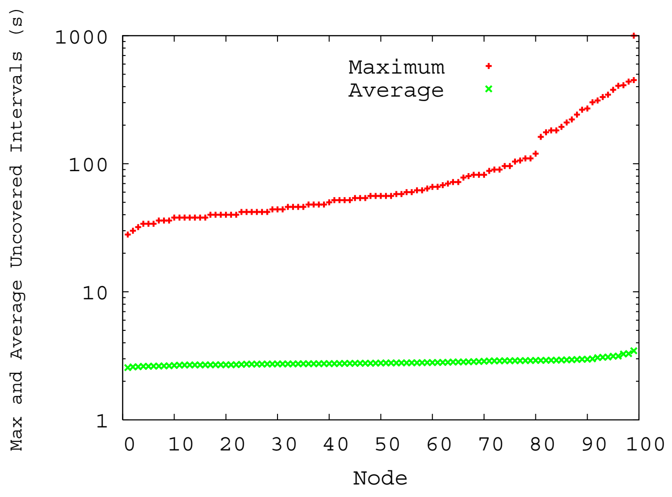


Fig. 16. Maximum uncovered intervals per node

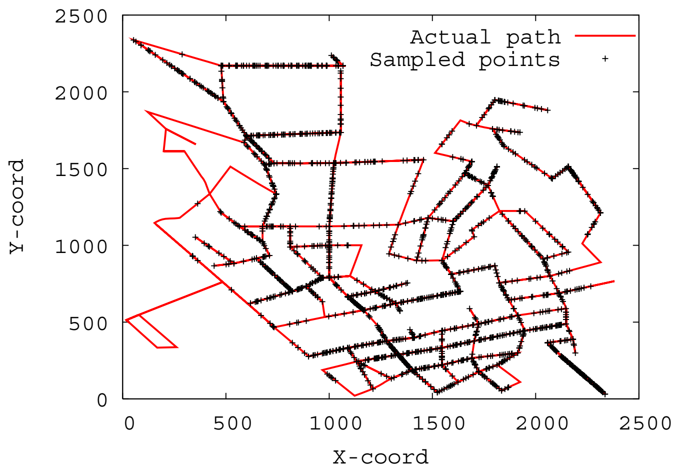


Fig. 17. Actual node trajectory vs harvested sampled points

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, on the order of several hundreds square Kms, the amount of time that nodes continuously reside within the area is likely very long, namely, for most nodes we can assume a closed mobility model. Second, the worst effect of dynamic scenarios takes place when nodes leave a specific area carrying several summaries (locally generated or collected) not harvested by the local agent yet. Nonetheless, we remark that carried information does not vanish as nodes leave, but can be harvested later by remote agents, responsible for the adjacent area the leaving nodes are moving into.

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

This dynamic effect is better evaluated for long simulation periods and periodic summary generation epochs. Thus, we confirm the settings used in Sections VI-E and VI-F; in addition, we consider a single harvesting agent, $k = 1$, $N = 100$, $v = 15m/s$. Nodes generate new summaries synchronously, and only as long as they remain in the area. To avoid that nodes stay within the area only for very short periods, we introduce a constraint on their minimum residing time equal to 10% of the whole simulation. Even with this assumption, more than 550 nodes need to take turns on the simulation area, to maintain 100 nodes always present. The agent does not follow open-RT model, but traditional RT, i.e., it always remains within the area.

Figures 18 and 19 present the experimental results corresponding to those in Sections VI-E and VI-F, but obtained with the open-RT model. Significant conclusions can be drawn, especially from Figure 18: also under these unfavorable assumptions, the agent is able to 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 by 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. Let us remark once more that those summaries are not irreparably lost, but will be probably harvested by agents in charge of the adjacent areas. Figure 19 shows 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 below $4s$ for more than 75% of the nodes (and below $10s$ for 90%) and that the $85th$ percentile of the vehicles can be tracked with a worst-case inaccuracy of $200s$.

H. Discussion

Mobile vs. 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 the purpose of 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 performs better than the static agents since for a given period of time, mobile agents will encounter more 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 higher than that between mobile and static nodes, and thus, α is greater

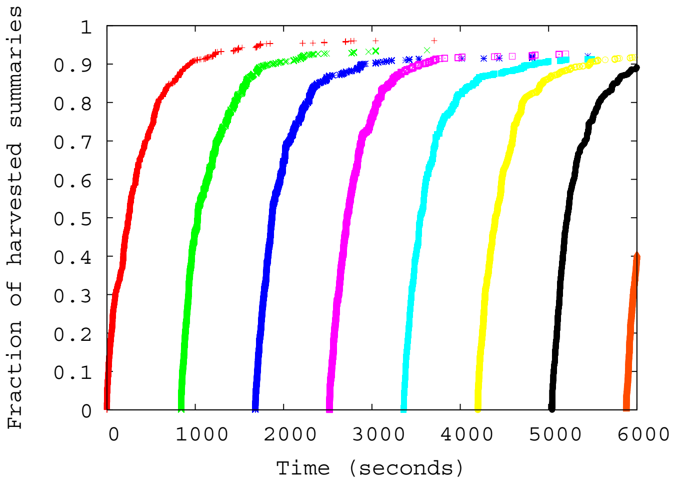


Fig. 18. Cumulative distribution of harvested summaries *per epoch (open-RT)*

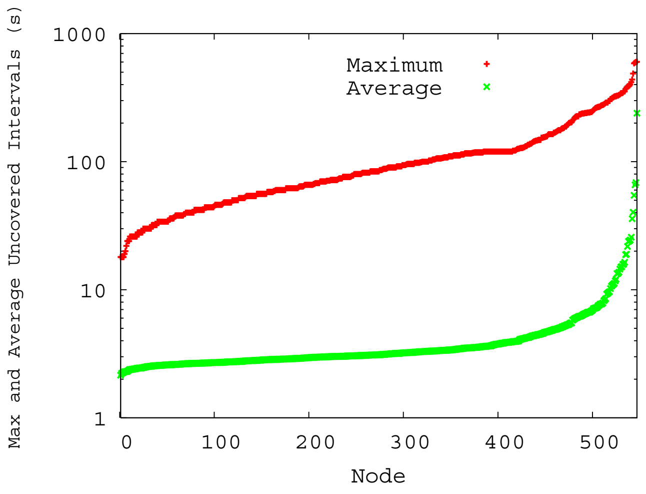


Fig. 19. Maximum uncovered intervals per node *(open-RT)*

if the agents are mobile (i.e., lower latency). In addition, especially in urban grids, it is usually the case that the spatial distribution of vehicles in steady state is non-uniform. If an agent is misplaced (i.e., where there are few vehicles on average), the harvesting latency will be large. In order to clearly understand the performance difference and how the position of a static node affects the latency, we use a scenario with 100 nodes moving at the 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 (N=100, V=25m/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.

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 be analytically determined using our models in Section V. The large communication range reduces the dissemination/harvesting latency, but this will limit scalability due to increased wireless channel contention.

Comparison with other schemes: Let us compare the performance with other two naïve schemes, namely direct contact and 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 their summaries. Upon receiving a message, each node re-broadcast the message with probability p. The performance of direct contact is the same as that of "passive harvesting." Recall that a regular node performs passive harvesting since it only stores summaries received from its 1-hop neighbors. Our analytic model clearly shows that the agents harvest summaries much faster than the regular nodes. Thus, it makes sense to focus only on the performance of the flooding scheme. Given $p = 1$, we measure the fraction of collected summaries generated for a period of 2000s. Each message is generated every 20s (total 100 messages per node). We simulate 50 nodes moving at the average speed of 15m/s according to the RT mobility model. We measure the completeness of summary harvesting, i.e., the fraction of harvested messages out of their total number. Our results show that MobEyes can collect 100% of summaries whereas the probabilistic scheme collects only 32% of summaries. Note that, of course, the completeness gain achieved by MobEyes comes at the cost of increased latency.

VII. MobEyes Privacy and Security

MobEyes nodes continually generate and diffuse summaries containing private information, e.g., license plate numbers. Thus, privacy is of critical importance. On the one hand, non authorized nodes must not be allowed access to private information. On the other hand, the harvesting process should not reveal the information that is being sought, since this may tip the attackers or may cause unnecessary panic in the public. In general, we can summarize the security requirements of MobEyes as follows:

- *Authentication*. Harvesting agents (authority nodes) must authenticate summary senders and vice versa.
- *Non-repudiation*. A summary originator cannot deny the transmission of a summary (liability issue); in that way, upon request from the agent, the summary sender must submit the full file with related sensed data.
- *Privacy*. Only legitimate users (authority nodes) can access summaries. Moreover, summaries must be privately 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 be able to filter out false summary data injected by attackers.
- *Query Confidentiality*. In some cases, e.g., bio-attacks and search for crime suspects, even the nature of the query should not be disclosed, not to create unnecessary panic in the population or to avoid tipping the criminals.

One important aspect that sets apart MobEyes "forensic sensed data" security from conventional "safe navigation" VANET security is the "real time" and "criticality" of the safe navigation application. Consider for example 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, else, it tells her to slow down. An adversary can "anticipate" the message from the mirror and tell her that the way is clear, while instead 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 a driver from alerting the vehicles behind her that there is a boulder on the road.

MobEyes has strongly different security requirements. A false report cannot create much damage since it is not acted upon immediately (for example, 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 [55]. For the sake of brevity, in this section we will simply outline several MobEyes security approaches, reserving the detailed, rigorous discussion of MobEyes security to future publications.

A. PKI Model

In MobEyes, we assume the existence of Public Key Infrastructure (PKI). Every node has a private/public key pair, which is issued through the Certificate Authority (CA) such as the police. Let PK_A and SK_A denote the node A's public and secret key pair. Let $\mathcal{H}(\cdot)$ denote a one-way hash function (e.g., SHA-1). For the sake of illustration, we assume that each node reads license plate numbers of nearby vehicles and prepares a summary which may contain a set of license plate numbers annotated with time and location. The summary should be encrypted by using the police public key (PK_{C_a}) so no one but the police can read it. Only the police need to authenticate the signature of the summary generator, while neighboring vehicles have no such need. Therefore, MobEyes does not need continuous access to the distributed PKI infrastructure. The digital signature and the certificate of the originator ($Cert_{C_x}$) are also encrypted in the same way. The verifier (i.e., the Police), upon decryption, uses the certificate $Cert_{C_x}$ signed by the authority where C_x is the node ID. Thus, for a given summary S_{C_x} generated at time T and position P, node C_x sends the following advertisement to its one hop neighbors (denoted as $*$).

$$C_x \rightarrow * : M_x, T, P$$

where $M_x = \{S_{C_x}, T, P, \{\mathcal{H}(S_{C_x})\}_{SK_{C_x}}, Cert_{C_x}\}_{PK_{C_a}}$.

The parameters T = time and P = location, corresponding to summary collection, are in the clear as they are the indexes to the summary database kept by each private vehicle. They are necessary to process on demand queries by the police. Every time the originator reissues the same summary, it introduces "jitter" in T and P (say several seconds and several meters) so that the police agent can still retrieve the record (as it falls in its space time window of interest), but the eavesdropper cannot infer the presence of the same user along the path as detailed in Section VII-C. Then, neighbor nodes will store the message in their local database, indexed by T and P. Through Bloom filter set reconciliation, the agent will harvest the encrypted summary if it falls within the space-time window of interest. After decrypting the summary (and eliminating aliases), the agent stores it in its local database.

B. Attack Model

As just shown, standard PKI mechanisms provide authentication and non-repudiation. In this section we focus on the rest of the MobEyes requirements, namely privacy, service availability, and data consistency, by addressing the following MobEyes-specific attack models:

- *Location Tracking*. Periodic broadcasting of identical summaries could facilitate attackers in tracking the route of a vehicle.
- *Denial of Service (DoS)*. Attackers could inject a large number of bogus summaries in order to slow down correct summary harvesting by agents.
- *False Data Injection*. Attackers could inject fabricated summaries in order to mislead investigations or make the data inconsistent.
- *Query Confidentiality*. Attackers could infer "important" information from the content of police queries.

Let us finally note that in MobEyes we can exclude Sybil attacks, where a node illegitimately claims multiple identities. In fact, the certification authority issues a public/private key pair for a given vehicle (unique identifier per node). We will assume that the signature/certificate cannot be forged.

C. Location Tracking Attack

Let us consider a typical tracking scenario where the attackers can infiltrate base stations on the roadsides and listen to all the advertised summaries. In MobEyes, a node periodically advertises its encrypted summary. If the message remains the same, the attacker can infer the trajectory of the sender node. We have solved this problem in a very simple way. Each repeated summary is slightly altered by changing T and P (i.e., it is a slightly modified clone). Assuming that the traffic was moderately dense when the sample was collected (say a few cars within a 100m street segment), the attacker cannot recognize the presence of the same vehicle from the sequence of summaries. The overhead introduced by this solution is minimal. Since it is less likely that a node meets the same vehicles many times, a few vehicles will store two or more "clones." The police agent detects and discards clones upon collecting and decrypting summary messages.

D. Denial of Service

Apart from channel jamming (which can be easily detected at the physical layer and immediately stopped and punished by authorities), a serious DoS attack to MobEyes may be caused by the injection of a large number of summaries into the network, e.g., caused by sensor data interface malfunctioning. If the summaries are playback messages, they are immediately detected and dropped. If the summaries look like legitimate summaries, the attack can be handled by using *rate limited summary diffusion* which shares the same idea of RREQ rate limit in a secure routing protocol [57]. Each node keeps track of the incoming rate of summary for each MAC address. Recall that a node will periodically rotate its MAC address for privacy. However, the rate monitoring period (say, a few seconds) is much smaller than the rotation period. If the rate is above a certain threshold, intermediate nodes simply discard incoming summaries. The rate is a system parameter and is determined based on the types of sensed data. Suspicious activity is reported to authorities for further analysis. For instance, the authorities can map pseudorandom MAC addresses to vehicle numbers, thus requesting the owner to fix the device or (in case of malicious attack) by revoking key pairs and prosecuting the abuser.

E. False Data Injection

False data injection is a very serious attack in conventional sensor networks [58], [34], [59]. Fortunately, in a VSN designed for forensic investigation, this type of attack is easy to detect and neutralize thanks to the observations of other nodes. There are several possible attacks of this type. First, the attacker could aim to mislead the search for kidnapping criminals, say, and it reports that it was at the right place/time of crime and saw a set of "fabricated" license plates. The attacker (or colluding attackers) will be quickly uncovered when the police investigate the phony license plates. A second false report attack is for the criminal to claim it was at a different place at the time of crime. Video taped records from crime witnesses will also permit to uncover the false report. A third type of attack could be the reporting of false sensed values. If the attackers collude, they may indeed create enough false reports to offset the scale. However, fluctuations in values would prompt the authorities to investigate, thus leading to the vehicle IDs of the cars injecting false reports, with their possible prosecution.

F. Query Confidentiality

Agents must retrieve information confidentially. Specifically, the agent tries to avoid a situation where criminals may take an evasive action if they realized the police are on their heels. Another possible concern is the investigation of chemicals that might be connected to a bio-attack. Since this investigation may be launched after a tip, and in most cases will turn out to be a false alarm, the public should not be told of the specific target of this investigation. Otherwise phenomenal traffic congestion may follow, with potentially very serious damage to vehicles and drivers. This requires a secure query such that the vehicles cannot tell what the agent is searching for. To this end, MobEyes exploits *private keyword searching*, proposed by Ostrovsky et al. [61].

Let us assume that the harvesting agent aims to retrieve images of some target vehicles. The agent has already harvested the summaries in the suspect area and has determined which vehicles were in the right place/time and might store the original images/licenses she is interested in. The agent cannot bluntly ask these vehicles if they have the target data. Rather, it prepares a dictionary of the license plate numbers in the nearby area (acquired from the harvested summaries). Each item is tagged $E(1)$ if interested; otherwise $E(0)$. $E(x)$ is a homomorphic public-key encryption function which has two important properties: $i)$ $E(x)$ is probabilistic, in particular it will encrypt a single bit in many different ways, such that any instance of $E(0)$ and any instance of $E(1)$ cannot be distinguished; and $ii)$ $E(x)$ is homomorphic such that $E(x) \times E(y) = E(x + y)$.

The agent will broadcast this tagged dictionary as a query in the vicinity of each of the vehicles that hold the information. After receiving a query, a vehicle start resolving the query by processing each document in its local storage as follows. For each document D with license plate number x, compute the encrypted value g ($g = E(1)$ or $g = E(0)$) and then calculate g^D. If the agent is interested in D, g^D will result in $g^D = E(D)$; otherwise, $g^D = E(0)$. Given output g^D, the agent can find D exactly. Each vehicle has an output buffer initialized with $E(0)$. Hashing is used to find a slot to store the output. The output will be multiplied with the value in the slot, leading to the result that only interested documents will be stored in the buffer. The output buffers of all the local vehicles (neighbors of the vehicles with useful data) are later read by the agent. As a result, the police agent discovers the vehicles with useful information and instructs them to upload all of their data for a proper time-space window (not too narrow to raise suspicions, nor too large to bring in too much junk data) to the next police access point.

VIII. Conclusions

In this paper, we proposed the decentralized and opportunistic MobEyes solution for proactive urban monitoring in VSN. The 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 showed that MDHP protocols are disruption tolerant, scalable, and non-intrusive 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 are extending the MobEyes prototype to determine the best trajectory of mobile agents when collaborating in summary harvesting. In addition, we are formally investigating how to determine the optimal value for the summary advertisement period depending on node speed/population and on urban monitoring requirements about traffic/latency. Finally, we will explore hybrid strategies that combine broadcast with epidemic dissemination, even dynamically adapting to urban density conditions and application needs.

Acknowledgement

We would like to thank Xiaoyan Hong and Jiejun Kong for sparing their valuable time to review the security section of the paper. We are also thankful to Giulio Galante for reviewing an earlier version of this manuscript.

References

[1] Alok Nandan, Shirshanka Das, Giovanni Pau, Mario Gerla, and Medy Y. Sanadidi. Co-operative Downloading in Vehicular Ad-Hoc Wireless Networks. In *IEEE WONS*, St. Moritz, Swiss, Jan. 2005.

[2] Alok Nandan, Saurabh Tewari, Shirshanka Das, Giovanni Pau, Mario Gerla, and Leonard Kleinrock. AdTorrent: Delivering Location Cognizant Advertisements to Car Networks. In *IFIP WONS*, Les Menuires, France, Jan. 2006.

[3] Qing Xu, Tony Mak, Jeff Ko, and Raja Sengupta. Vehicle-to-vehicle safety messaging in DSRC. In *ACM VANET*, Philadelphia, PA, USA, Oct. 2004.

[4] Jorg Ott and Dirk Kutscher. A Disconnection-Tolerant Transport for Drive-thru Internet Environments. In *IEEE Infocom*, Miami, FL, USA, Apr. 2005.

[5] Uichin Lee, Eugenio Magistretti, Biao Zhou, Mario Gerla, Paolo Bellavista, and Antonio Corradi. Efficient Data Harvesting in Mobile Sensor Platforms. In *IEEE PerSeNS*, Pisa, Italy, Mar. 2006.

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

[7] Bret Hull, Vladimir Bychkovsky, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Yang Zhang, Hari Balakrishnan, and Samuel Madden. CarTel: A Distributed Mobile Sensor Computing System. In *ACM SenSys*, Boulder, CO, USA, Oct.-Nov. 2006.

[8] Uichin Lee, Joon-Sang Park, Eyal Amir, and Mario Gerla. FleaNet: A Virtual Market Place on Vehicular Networks. In *IEEE V2VCOM*, San Francisco, CA, USA, Jul. 2006.

[9] Murat Caliskan, Daniel Graupner, and Martin Mauve. Decentralized discovery of free parking places . In *ACM VANET*, Los Angeles, CA, USA, Sept. 2006.

[10] Davide Sormani, Gabriele Turconi, Paolo Costa, Davide Frey, Matteo Migliavacca, and Luca Mottola. Towards lightweight information dissemination in inter-vehicular networks . In *ACM VANET*, Los Angeles, CA, USA, Sept. 2006.

[11] Marc Torrent-Moreno, Daniel Jiang, and Hannes Hartenstein. Broadcast reception rates and effects of priority access in 802.11-based vehicular ad-hoc networks. In *ACM VANET*, Philadelphia, PA, USA, Oct. 2004.

[12] Gokhan Korkmaz, Eylem Ekici, Fusun Ozguner, and Umit Ozguner. Urban Multi-Hop Broadcast Protocols for Inter-Vehicle Communication Systems. In *ACM VANET*, Philadelphia, PA, USA, Oct. 2004.

[13] Zong Da Chen, H.T. Kung, and Dario Vlah. Ad Hoc Relay Wireless Networks over Moving Vehicles on Highways. In *ACM MOBIHOC*, Long Beach, CA, USA, Oct. 2001.

[14] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In *IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.

[15] UMass' DieselNet. `http://prisms.cs.umass.edu/dome/`.

[16] Jing Zhao and Guohong Cao. VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks. In *IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.

[17] Hao Wu, Richard Fujimoto, Randall Guensler, and Michael Hunter. MDDV: a Mobility-entric Data Dissemination Algorithm for Vehicular Networks. In *ACM VANET*, Philadelphia, PA, USA, Oct. 2004.

[18] Shane B. Eisenman, Gahng-Seop Ahn, Nicholas D. Lane, Emiliano Miluzzo, Ronald A. Peterson, and Andrew T. Campbell. MetroSense Project: People-Centric Sensing at Scale. In *ACM WSW*, Boulder, CO, USA, Oct.-Nov. 2006.

[19] Oriana Riva and Cristian Borcea. The urbanet revolution: Sensor power to the people! *IEEE Pervasive Computing*, 6(2), 2007.

[20] Marios D. Dikaiakos, Saif Iqbal, Tamer Nadeem, and Liviu Iftode. VITP: an information transfer protocol for vehicular computing . In *ACM VANET*, Cologne, Germany, Sept. 2005.

[21] Jeff Burke, Deborah Estrin, Mark Hansen, Andrew Parker, Nithya Ramanathan, Sasank Reddy, and Mani B. Srivastava. Participatory Sensing. In *ACM WSW*, Boulder, CO, USA, Oct.-Nov. 2006.

[22] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li-Shiuan Peh, and Daniel Rubenstein. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *ACM ASPLOS-X*, San Jose, CA, USA, Oct. 2002.

[23] Tara Small and Zygmunt J. Haas. The Shared Wireless Infostation Model - A New Ad Hoc Networking Paradigm (or Where There is a Whale, There is a Way). In *ACM MOBIHOC*, Annapolis, Maryland, USA, June 2003.

[24] University of Dartmouth MetroSense. `http://metrosense.cs.dartmouth.edu/`.

[25] Rahul C. Shah, Sumit Roy, Sushant Jain, and Waylon Brunette. Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks. *Elsevier Ad Hoc Networks Journal*, 1(2-3):215–233, Sept. 2003.

[26] Qun Li and Daniela Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *ACM MOBICOM*, Boston, MA, USA, Aug. 2000.

[27] Yu Wang and Hongyi Wu. DFT-MSN: The Delay/Fault-Tolerant Mobile Sensor Network for Pervasive Information Gathering. In *INFOCOM'06*, Barcelona, Spain, Apr. 2006.

[28] Phillip B. Gibbons, Brad Karp, Yan Ke, Suman Nath, and Srinivasan Seshan. IrisNet: An Architecture for a Worldwide Sensor Web. *IEEE Pervasive Computing*, 2(4):22–33, Oct.-Dec. 2003.

[29] Suman Nath, Jie Liu, and Feng Zhao. Challenges in Building a Portal for Sensors World-Wide. In *ACM WSW*, Boulder, CO, USA, Oct.-Nov. 2006.

[30] CENS' Urban Sensing. `http://research.cens.ucla.edu/projects/2006/Systems/Urban_Sensing/`.

[31] Burton H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *CACM*, 13(7):422–426, Jul. 1970.

[32] Andrei Broder and Michael Mitzenmacher. Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, 1(4):422–426, 2003.

[33] Li Fan, Pei Cao, and Jussara Almeida. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. In *ACM SIGCOMM*, Vancouver, Canada, Aug.-Sept. 1998.

[34] Fan Ye, Haiyun Luo, Songwu Lu, and Lixia Zhang. Statistical En-route Filtering of Injected False Data in Sensor Networks. In *INFOCOM*, Hong Kong, Mar. 2004.

[35] John W. Byers, Jeffrey Considine, Michael Mitzenmacher, and Stanislav Rost. Informed Content Delivery Across Adaptive Overlay Networks. In *ACM SIGCOMM*, Pittsburgh, PA, USA, Aug. 2002.

[36] Patrick Reynolds and Amin Vahdat. Efficient Peer-to-Peer Keyword Searching. In *Middleware'03*, Rio de Janeiro, Brazil, Jun. 2003.

[37] Louka Dlagnekov and Serge Belongie. Recognizing Cars. Technical Report CS2005-0833, UCSD CSE, 2005.

[38] Paolo Bellavista, Eugenio Magistretti, Uichin Lee, and Mario Gerla. Standard Integration of Sensing and Opportunistic Diffusion for Urban Monitoring in Vehicular Sensor Networks: the MobEyes Architecture. In *IEEE ISIE*, Vigo, Spain, Jun. 2007.

[39] Matthias Grossglauser and Martin Vetterli. Locating Nodes with EASE: Mobility Diffusion of Last Encounters in Ad Hoc Networks. In *IEEE INFOCOM*, San Francisco, CA, USA, Mar.-Apr. 2003.

[40] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed Diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks. In *ACM MOBICOM'00*, Boston, MA, USA, 2000.

[41] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A Scalable Location Service for Geographic Ad Hoc Routing. In *ACM MOBICOM*, Boston, MA, USA, 2000.

[42] Christian Bettstetter, Giovanni Resta, and Paolo Santi. The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 2(3):257–269, Jul.-Sept. 2003.

[43] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi Raghavendra. Performance Analysis of Mobility-Assisted Routing). In *ACM MOBIHOC*, Florence, Italy, May 2006.

[44] Fan Bai and Ahmed Helmy. Impact of Mobility on Mobility-Assisted Information Diffusion (MAID) Protocols. Technical report, USC, July 2005.

[45] ns-2 (The Network Simulator). `http://www.isi.edu/nsnam/ns/`.

[46] Biao Zhou, Kaixin Xu, and Mario Gerla. Group and Swarm Mobility Models for Ad Hoc Network Scenarios Using Virtual Tracks. In *IEEE MILCOM*, Monterey, CA, USA, Oct.-Nov. 2004.

[47] U.S. Census Bureau. TIGER, TIGER/Line and TIGER-Related Products. Available at. `http://www.census.gov/geo/www/tiger/`.

[48] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta G. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *ACM MOBICOM*, Dallas, TX, USA, Oct. 1998.

[49] Amit Kumar Saha and David B. Johnson. Modeling Mobility for Vehicular Ad Hoc Networks. In *ACM VANET'04*, Philadelphia, PA, USA, October 2004.

[50] TS Rappaport. *Wireless Communications: Principles and Practice*. IEEE Press Piscataway, NJ, USA, 1996.

[51] Uichin Lee, Joon-Sang Park, Eyal Amir, and Mario Gerla. FleaNet: A Virtual Market Place on Vehicular Networks. In *V2VCOM'06*, San Jose, CA, July 2006.

[52] Uichin Lee, Eugenio Magistretti, Biao Zhou, Mario Gerla, Paolo Bellavista, and Antonio Corradi. MobEyes: Smart Mobs for Urban Monitoring with Vehicular Sensor Networks. *IEEE Wireless Communications*, 13(5):51–57, Sept.-Oct. 2006.

[53] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. In *MobiCom'99*, Seattle, WA, USA, Aug. 1999.

[54] Tracy Camp, Jeff Boleng, and Vanessa Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC)*, 2(5):483–502, 2002.

[55] Maxim Raya and Jean-Pierre Hubaux. The Security of Vehicular Ad Hoc Networks. In *SASN*, Alexandria, VA, Nov. 2005.

[56] Uichin Lee, Eugenio Magistretti, Biao Zhou, Mario Gerla, Paolo Bellavista, and Antonio Corradi. Dissemination and Harvesting of Urban Data using Vehicular Sensing Platforms. Technical report, UCLA CSD, 2007.

[57] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In *MOBICOM*, Atlanta, GA, Sept. 2002.

[58] Sapon Tanachaiwiwat and Ahmed Helmy. Correlation Analysis for Alleviating Effects of Inserted Data in Wireless Sensor Networks. In *MobiQuitous*, San Diego, PA, Jul. 2005.

[59] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, and Peng Ning. An Interleaved Hop-by-Hop Authentication Scheme for Filtering False Data Injection in Sensor Networks. In *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2004.

[60] Philippe Golle, Dan Greene, and Jessica Staddon. Detecting and Correcting Malicious Data in VANETs. In *VANET'04*, Philadelphia, PA, October 2004.

[61] Rafail Ostrovsky and William Skeith. Private Searching on Streaming Data. In *CRYPTO*, Santa Barbara, CA, Aug. 2005.