*This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2007 proceedings.*

# Efficient Retrieval of User Contents in MANETs

Marco Fiore, Claudio Casetti, Carla-Fabiana Chiasserini
Dipartimento di Elettronica, Politecnico di Torino, Italy
Email: firstname.lastname@polito.it

***Abstract*—We consider a cooperative environment in wireless mobile networks where information is exchanged among nodes in a peer-to-peer fashion. We apply a pure peer-to-peer approach (i.e., without the intervention of servers) and we seek to devise an efficient query/response propagation algorithm. Our approach, called *Eureka*, identifies the regions of the network where the required information is more likely to be stored and steers the queries toward those regions. To discriminate among regions, we introduce the concept of *information density* and a procedure that allows nodes its estimation. The effectiveness of our scheme is evaluated through simulation in a vehicular environment with realistic mobility models.**

## I. Introduction

A mobile ad hoc network (MANET) consists of wireless devices that communicate over bandwidth-constrained links. The wireless nodes are free to move, join or leave the network – factors that, along with the time-varying behavior of the wireless channel, lead to a highly-dynamic network system. Applications on top of MANETs often require a device to use resources/services located at a gateway node or at another user device. Thus, given the network characteristics, two fundamental problems arise: (i) how to discover services and resources available at other nodes in the MANET, and (ii) how to transfer information between two network nodes, possibly with the help of intermediate devices when no direct link exists.

In this work we look at the MANET as a peer-to-peer network, where user nodes not only require content delivery but also act as content providers. Our aim is to provide mobile users with data services in an effective manner, despite the scarcity of bandwidth and the intermittent connectivity due to the highly-dynamic nature of MANETs. We develop a solution, named *Eureka*, whose key idea is to exploit the *information density* concept and allow users to estimate *where* in the MANET the information they are looking for can be found. Our approach yields several advantages: (i) waste of bandwidth is avoided by sensibly (and selectively) forwarding content queries; (ii) not only query overhead is reduced, but also fewer duplicated reply messages are sent back to the requesting node; (iii) in a network where a contention-based MAC is used, the selective forwarding of queries and reduction of duplicates lowers the collision probability, hence the congestion level; (iv) the number of successful deliveries and the system responsiveness remain almost unchanged with respect to flooding-based strategies, and even improve as the number of content items grows; (v) the use of GPS is not required, making our solution suitable for various wireless environments.

A viable application of peer-to-peer MANETs is in the field of vehicular networks [1], [2], where the constraints imposed by the road topology limit the regions where the information is to be sought. In this case, it is of the utmost importance to minimize the transmission overhead toward sections of the network where the chances of successful information retrieval are slim. At the same time, the highly mobile environment typical of vehicular networks provides an interesting challenge to the performance of a peer-to-peer application. For these reasons, after setting the description of our proposed approach within the general framework of MANETs, we focus on the benefits that can be attained when it is applied to vehicular networks. A discussion of related works can be found at the end of the paper.

## II. System and Assumptions

We consider a MANET and one or more gateway nodes that may be either fixed or mobile. Each user node in the MANET is equipped with a data cache and may wish to access the information stored elsewhere in the network, e.g., at one or more gateways or at other nodes. Connectivity among users and between user and gateways is, however, spotty and cooperation among users is highly desirable. Targeting a solution that must be suitable for different network environments, we do not require nodes to be equipped with additional localization hardware, such as GPS.

The cooperative environment we are addressing falls within the category of peer-to-peer (P2P) networking, allowing users to share files on their own host computers. Unlike traditional client-server networking, peer nodes simultaneously act as both "clients" and "servers" to the other nodes in the network. Coordination among nodes is achieved in several ways, although, for the purpose of the present work, we just consider *pure* P2P: peers act as clients and servers, and there is no central coordination by one or more servers. Indeed, any attempt at providing coordination would suffer from the intermittent nature of connections in a MANET.

Our work specifically tackles the issue of *controlled broadcast of queries* by devising a solution that allows the propagation of queries only towards regions of the network where the information is more likely to be cached. To discriminate among such regions, we define a quantity called *information density*, and a procedure that lets nodes estimate it.

Before detailing our solution, we briefly introduce the network scenario and the basic functionalities required in the system for content request and delivery [3]. We assume that $N$ distinct information items, such as web documents or data files, are available at the gateway node(s) and may be requested by users. Each information item is further divided into single downloadable units, called *chunks*, each small enough to fit in a single MAC frame. When a user node seeks a specific information item, it advertises to the system which chunks it is missing. The missing chunks are then retrieved from (possibly) different sources, following a procedure that is general enough to apply to different network systems. Such procedure is based on a cross-layer approach, involving the application, network and MAC layers, and does not assume the use of any specific routing protocol. The basic features that are required in the system for information request and delivery are outlined below.

- Each user application requests an information item not in its cache, say information item $i$ ($1 \leq i \leq N$), with rate $\lambda_i$. Upon a request generation, the node broadcasts query messages, each for as many as $C$ chunks of the information item. We assume that all nodes know by default the number of chunks into which every information item is divided, as well as the chunks sequence number. Queries for missing chunks are periodically issued until the information item is fully received.
- A node receiving the query and caching one or more of the requested chunks sends them back to the requesting node, through an *information message*. If the node does not own any of the requested chunks, or it owns only a few of them, it can decide whether to rebroadcast a query for the missing chunks. If it decides to do so, it stores routing details of the query (among others, the query ID, its source address and the address of the node from which the query was received), and sets the query status for the missing chunks to *pending*.
- Once created, an information message is sent back to the query source along the same path the request came from. The information message is conveyed through a unicast transmission at the MAC layer, exploiting the query routing details that have been stored by the node application on the way there. No action is required at the network layer. Also, all nodes are able to promiscuously listen to the channel at the MAC layer: this gives the nodes the opportunity to know that a pending query has been satisfied elsewhere and set the status of the retrieved chunks to *solved*, thus avoiding the relay of duplicated information messages.
- Information chunks retrieved by the requesting node are locally cached and then dropped at a rate of $\mu$ chunks per second.

## III. Information Density-driven Retrieval: Motivation and Description

A critical aspect of the information sharing mechanism described above is the propagation of query messages in the network. On the one hand, queries for information chunks must be forwarded by relays until they reach nodes holding such chunks, and some redundancy in forwarding is necessary to compensate for the unreliable nature of broadcast transmission of queries (i.e., no acknowledgments). On the other hand, congestion deriving from excessive spreading and chunk duplication must be limited.

The simplest solution for query propagation is plain flooding of requests, but this is hardly viable in tightly-meshed, bandwidth-hungry networks where congestion is more than likely. To counter the drawbacks of flooding we design our approach along the guidelines summarized below. The first two are well-known techniques that we include in our approach, while the third solution is part of our contribution.

1) Introduction of a *query time to live* ($TTL$) to shorten the reach of broadcast queries. A balance should be stricken between small values of $TTL$, which limits the success probability of a query, and query load. Such balance is highly dependent on the network scenario [4], [5].
2) Introduction of a *query lag time* at each relay, to delay the propagation of a request in the hope that a node in the neighborhood returns a response (thus making any further query propagation useless). A similar mechanism is used, for instance, in DSR to prevent route reply storms. Although the query lag time and the $TTL$ concur in establishing a *mitigated flooding*, further improvements are necessary: if the requested chunks are not found in the neighborhood of the query source, queries propagate as far as the $TTL$ allows them in a fashion resembling that of plain flooding.
3) Targeting *areas* of the network where the requested information is more likely to be cached. To this end, it is crucial that requests be forwarded only towards those nodes in the network that might cache the requested information, hardly an easy task in a MANET where the topology changes rapidly, and limited-memory nodes cache only a few information chunks. Relying on GPS-aided routing (such as [6]) to identify and reach a peer holding the requested information proves ineffective due to connectivity and information volatility (beside being subject to coverage black-outs of GPS signals). Our idea, instead, is to introduce the concept of *information density*, i.e., the amount of information cached by nodes in a specific area, and to exploit it to decide where queries must be forwarded to.

In order to implement the scheme described above, each query header must include a HOP_COUNT field, initialized to $TTL$ and decremented by 1 at each relay. Also, each node computes an *information density estimate* in a distributed manner and independently for each information item, as detailed in Sec. IV. A node generating a query adds to the query header its own density estimate for the requested information (ESTIMATED_DENSITY field). Nodes receiving the query have the option to act as relays or not, by checking the information density estimate they computed against the one stored in the query message. Since we want requests to be propagated towards information-denser areas, we let nodes forward a query only if their own estimate is higher than that carried by the query [1]. Such forwarding process is run independently for each query message, which makes the scheme robust to very fast changes in network topology and/or information distribution. In Sec. VI we show that this scheme is capable of reducing the number of query messages as well as the number of duplicated replies, while preserving, if not improving, the number of solved requests and the query response time.

## IV. Information density estimation

We define the *information density function*, $\delta_i(x, y)$, as the spatial density of information chunks cached at nodes participating in the network, around a point whose spatial coordinates are $(x, y)$. The subscript $i$ refers to the information item $i$, with $1 \leq i \leq N$. We measure the information density in $copies/m^2$ (in case of uni-dimensional topologies such as a highway scenario, we consider $\delta_i(x)$ measured in $copies/m$).

Our aim is to provide each node in the network with an estimate of the information density in its *proximity*, so that it can choose whether to forward queries or not according to such estimate. Instrumental to the definition of "node proximity" in our case is the definition of *reach range* of a generic node $n$, as its distance from the farthest node that can receive a query generated by node $n$ itself. The reach range obviously depends on the query $TTL$ and is bounded by the product of $TTL$ and the nodes radio range. Notice that Eureka uses the *difference* between estimates of the information density computed at two nodes (i.e., locations), rather than their absolute values. Therefore it is not important to us that the estimates match the absolute values of the actual density, but that the density estimates over the network area closely reflect the actual information density as far as the trends are concerned.

The process we devise to estimate the information density is fully distributed and is run by all nodes participating in the network. The process amounts to merging estimates observed by each node on its own and estimates received by neighboring nodes. At each sampling step $j$, a node $n$ computes an information density sample $s_{i,j}(n)$ for each information item $i$ it is aware of [2], by using information captured within its reach range. It then filters the computed samples through a Moving Average (MA) algorithm.

More specifically, the sample $s_{i,j}(n)$ represents the estimated number of new copies of chunks belonging to an information item $i$ which were created within reach range of node $n$, during sampling step $j$. New copies are weighted by their distance from node $n$, so that new, close-by copies have a greater impact on the sample than those cached far from the tagged node. Each sample includes two contributions. The first one is a sample locally computed by node $n$ considering all generated, overheard and received information messages, and it is referred to as $s^l_{i,j}(n)$. The second one is a distributed sample, computed by node $n$ from samples advertised by its neighboring nodes, and it is referred to as $s^d_{i,j}(n)$. These two contributions are detailed in Sec. IV-A and Sec. IV-B, while the overall sample computation is described in Sec. IV-C.

In the MA filter we use, the most recent $W$ samples have a greater impact on the filter output, while the contribution of older samples is exponentially decreased at each sampling step. The value returned by the filter at step $j$ is the information density estimate as seen by node $n$ for the $i$-th information item, $\hat{\delta}_{i,j}(n)$. The sampling frequency $f_c$ is set as a function of the cached chunks drop rate $\mu$ and the filter parameter $W$. A detailed description of the MA filter is given in Sec. IV-D.

### A. Local information density sample

Consider the $j$-th sampling interval and information item $i$, node $n$ computes the local part of the information density sample, $s^l_{i,j}(n)$, as follows.

*a)* If node $n$ generates a *reply* information message containing chunks of information item $i$, as an answer to a query for some chunks it owns, then a new copy of such chunks is going to be cached at the node that generated the relative query. Node $n$ must therefore account for the presence of such new copy at a distance $h_Q$, which is equal to the number of hops covered by the query, as shown in the upper plot in Figure 1. This distance can be retrieved by looking at the query HOP_COUNT field. A contribution

$$1 - \frac{h_Q - 1}{TTL}$$

is then added to the current density sample $s^l_{i,j}(n)$. This value ranges between $\frac{1}{TTL}$ if the new copy is cached $TTL$ hops away, while it increases up to 1 if the node that generated the query is within transmission range of $n$.

*b)* If node $n$ receives a *new* transiting information message, i.e., a message containing a chunk considered as pending (as illustrated by the lower plot of Figure 1), it must account for: i) the presence of a new copy of the chunk that will be cached by a node at a distance $h_Q$ hops and ii) the presence of an existing copy that is cached at a distance $h_I$ hops. The $h_Q$ value can be retrieved by looking at the query list entry (HOP_COUNT field), while the $h_I$ value can be found by looking at the HOP_COUNT field of the information message header that records the number of traversed hops. Thus, in this case the following contribution is added to the current density sample $s^l_{i,j}(n)$:

$$\left(1 - \frac{h_Q - 1}{TTL}\right) + \left(1 - \frac{h_I - 1}{TTL}\right)$$

[1]To account for inaccuracies in the estimation process, a node rebroadcasts the query if its own estimate is at least 75% as great as the estimate of the query source

[2]When a node has to process a query for an information item it is not aware of, it considers the corresponding density estimate to be equal to zero

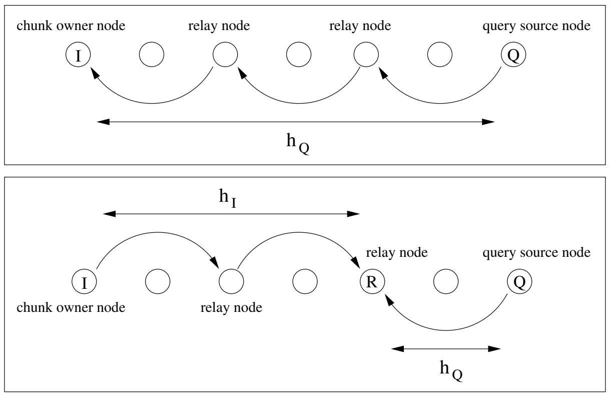

Fig. 1. Representation of case *a)* (upper plot) and case *b)* (lower plot). Upper plot: $h_Q$ value for node $I$ sending an information message back to the requesting node $Q$. Lower plot: $h_Q$ and $h_I$ values for relay node $R$ with respect to the query source $Q$ and node $I$ caching the information chunks

As in case *a)*, the above expression weighs both the new and the existing chunk copies by their distance from node $n$. Note that this behavior is the same no matter whether the receiving node is the final recipient of the information chunk, a relay node in the return path, or a node just overhearing the passing message.

*c)* The last case accounts for the reception of an information message whose contribution must not (or cannot) be related to a corresponding query. This may happen for two reasons: either the corresponding query list entry status is set to solved (which means that the message is considered as *duplicated* information), or no query list entry is found (which means that the node moved within transmission range of nodes in the return path after the query was generated and propagated by these nodes). In either case, only the contribution due to the presence of an existing copy is considered and is computed as:

$$1 - \frac{h_I - 1}{TTL}$$

The local information density sample $s^l_{i,j}(n)$ is calculated by summing the three contributions from cases *a)*, *b)*, and *c)*, weighted by the number of their occurrences.

## B. Distributed information density sample

The second contribution to the information density estimate, i.e., the distributed sample $s^d_{i,j}(n)$, is calculated by a node $n$ using the local information density samples advertised by its neighbor nodes. Indeed, every time a node $m$ generates or relays a query for some chunks of information item $i$, it advertises its local information density sample for this item, $s^l_{i,j}(m)$, by setting a query header field (named SAMPLE_DENSITY) to such value. A node $n$ receiving the query can then compute its distributed sample for the $j$-th step and information item $i$, $s^d_{i,j}(n)$, by averaging all received values:

$$s^d_{i,j}(n) = \frac{\sum_{m \in \mathcal{M}_{i,j}(n)} s^l_{i,j}(m)}{|\mathcal{M}_{i,j}(n)|}$$

where $\mathcal{M}_{i,j}(n)$ is the set of neighbor nodes which advertised their local sample to node $n$, for information item $i$ and sampling step $j$, and $|\mathcal{M}_{i,j}(n)|$ is the set cardinality.

## C. Overall information density sample

The overall density sample for information item $i$ and sampling time $j$, $s_{i,j}(n)$, is computed by node $n$ by simply averaging the local and distributed contributes. The expression of the overall information density sample is then:

$$s_{i,j}(n) = \frac{s^l_{i,j}(n) + s^d_{i,j}(n)}{2}$$

Observe that nodes can advertise the local information density sample they computed during sampling step $j$ only during the following step $j+1$. As a consequence, at step $j+1$, a node $n$ holds its own local samples up to the $(j+1)$-th sampling step, and distributed samples up to the $j$-th step. Since mixing contributions corresponding to different steps would be incoherent, $s_{i,j}(n)$ is only computed at the $(j+1)$-th step. This amounts to a delay of one sampling step in the computation of the information density sample $s_{i,j}(n)$. However, if the employed sampling frequency is sufficiently high with respect to the information density dynamics, as it should be, such a delay does not affect the validity of the estimate.

## D. Filtering and information density estimate

A Moving Average (MA) filter is used to define the behavior in time of the computed samples, in a way to replicate the real behavior of cached information chunks.

The filter is built so that the value of each new sample is kept almost constant to its original value for $W$ sampling steps since it was computed, after which it is exponentially decreased. To correctly describe the behavior of cached chunks in time, each filtered sample should ideally contribute to the information density estimate for a period of time equal to the average cache time of chunks at nodes, which is $1/\mu$. From the above considerations, we must have: $WT_c = \frac{1}{\mu}$, where $T_c$ denotes the duration of a sampling interval. It follows that the sampling frequency is given by: $f_c = \frac{1}{T_c} = W\mu$.

The analytical expression of the filter, returning the final value of the estimate for node $n$, information item $i$ and sampling step $j$ ($j \geq W$), is the following:

$$\hat{\delta}_{i,j}(n) = \sum_{k=1}^{W-1} \left(1 + \alpha^W - \alpha^{W-k}\right) s_{i,j-k}(n) + \left(1 + \alpha^W - \alpha\right) \sum_{k=W}^{j} \alpha^{k-W+1} s_{i,j-k}(n) \quad (1)$$

with $0 \leq \alpha \leq 1$. The first term of the right hand side of (1) represents the contribution of the $W$ most recent samples. The second term causes the exponential decrease of sample values that are older than $W$ sampling steps.

In our implementation, we set: $W = 100$ and $\alpha = 0.5$; the values of $W$ and $\alpha$ determine the filter impulse response as depicted in Figure 2. Note that, the larger the $W$, the higher the sampling frequency and the more often the information density estimate is computed; thus a trade-off exists between computation load and accuracy in following the dynamics of the information density. The parameter $\alpha$ instead determines how the "mid-age" samples are weighted: the smaller the $\alpha$, the greater the impact of the "mid-age" samples (as shown in Figure 2); we highlight that too small values of $\alpha$ may lead to significant discontinuities in the density estimate.

## V. Simulation scenario

Simulations were performed using the network simulator *ns2*, where we implemented the information sharing application and the information density estimation mechanism. We tested the performance of Eureka in a vehicular environment, using two realistic mobility models so as to simulate different traffic conditions. The parameter setting and the two mobility models are detailed next.

*a) Settings:* A vehicle enters the simulated scenario with an empty cache. It requests an information item not in its cache according to an i.i.d. Poisson process with parameter $\lambda$. We point out that different request rates for different information items could be considered as well, since our density estimate process is performed independently for each item. The $TTL$ value for query messages is set to 10 hops; indeed, in [3] we observed that a smaller value may reduce the probability to find the information in the MANET, while a larger value does not lead to any significant improvement. Cached chunks are deterministically discarded $1/\mu$ seconds after they have been received.

Every information item comprises 30 chunks. Each query contains up to $C$ requests for chunks; its size is equal to 21 bytes plus 1 byte for each chunk request. When the information density estimation process is employed, queries include the SAMPLE_DENSITY field (2-byte long) used by nodes to advertise their local density sample, as well as the ESTIMATED_DENSITY field (2-byte long) reporting the estimated density value for the requested item used for the query forwarding decision. Each information message includes a 21-byte header and carries one information chunk, whose size is equal to 1024 bytes.

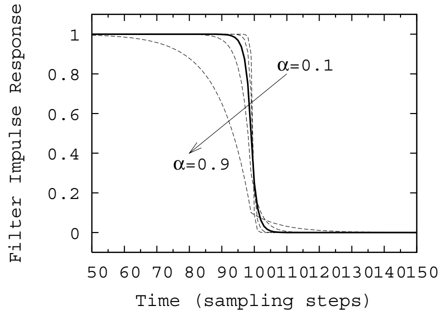

Fig. 2. MA filter impulse response, with $W = 100$ and varying $\alpha$

The information sharing application lies on top of a UDP-like transport protocol. As already mentioned, the way the information sharing application works makes it independent of the network layer protocol. Comparison tests with DSDV and AODV proved such a statement, as they returned almost identical results. The results shown in the next section were obtained considering AODV. At the MAC layer, 802.11 in promiscuous mode is employed. Information messages exploit the RTS/CTS mechanism and MAC-level retransmissions, while query messages, of broadcast nature, do not use RTS/CTS and are never retransmitted. The channel bandwidth is set to 11 Mbps, and nodes have a radio range of 100 m.

*b) Highway scenario:* We consider a 5-km-long, straight, unidirectional road, composed of three parallel lanes. A gateway node is located halfway along the road. On each lane vehicle interarrival times are exponentially distributed with parameter that depends on the lane, namely 4, 3 and 2 vehicles/s, respectively. As vehicles enter the highway, they are associated to a desired speed, uniformly selected over a range of values that again depends on the lane the vehicles start on, namely [15 m/s,25 m/s], [25 m/s,35 m/s], and [35 m/s,45 m/s], respectively. Vehicles drive all the way down the road until the end of the 5-km segment, where they are removed. Along the road they interact with each other, according to two realistic vehicular traffic micromobility models: the Intelligent Driver Model (IDM), which regulates the vehicle acceleration and deceleration, and the MOBIL model, which determines lane changes and passing [7].

*c) Urban scenario:* We extended the vehicular mobility model provided by the CanuMobiSim tool [8], introducing traffic lights and stop signs at road intersections, as well as a realistic behavior of vehicles approaching the intersections [9]. As a consequence, queues of vehicles decelerating and/or coming to a full stop near crowded intersections are also modeled. We simulated the road topology for the city section shown in Figure 3, which includes several road intersections regulated by traffic lights or stop signs. Vehicles enter the city section from one of the border entry/exit points, randomly choose another border entry/exit point as their destination, compute the shortest path to it and then cross the city section accordingly, following the available road paths. A vehicle entering the topology is assigned a desired speed, uniformly chosen in the interval [10 m/s,20 m/s]. When a vehicle reaches its destination, it stops for a random amount of time, uniformly distributed between 0 and 60 s, then it re-enters the city section. Vehicles stopped at the border of the topology are temporarily inactive. Employing the micromobility models mentioned above, on the selected road topology we observe, on average, 70 vehicles traveling at a mean speed of 21 km/h.

## VI. Performance Study

We first outline the impact of the parameters on the system performance. Then, we assess the accuracy of our information density estimation on both the highway and the urban scenario.

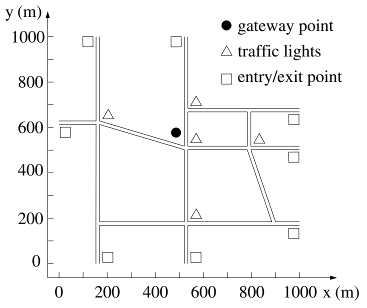

Fig. 3. Urban scenario road topology. Non-marked intersections are regulated by stop signs

Finally, we investigate the effectiveness of Eureka in terms of traffic reduction, query success and query solving time.

## A. Impact of parameters

The impact of the system parameters on the overall performance is as follows.

- The query generation rate, $\lambda$, obviously affects the query traffic and, in turn, the information traffic.
- The cache dropping rate, $\mu$, is inversely proportional to the amount of information in node caches, and thus, to the query success rate. Also, with high values of $\mu$, queries have to travel farther to find the information and, if not properly directed, they generate duplicated responses.
- The number $C$ of chunk requests per query affects the time needed for a response and the amount of information traffic if more than one node replies; again, targeting only information-dense areas limits the congestion deriving from an excessive number of duplicates. Furthermore, a large $C$ may lessen the effectiveness of the query lag time, further increasing congestion.
- The information set cardinality, $N$, has an impact on the traffic load, hence on the system performance, but it does not affect the accuracy of the density estimate, since the estimate is performed on single information items.

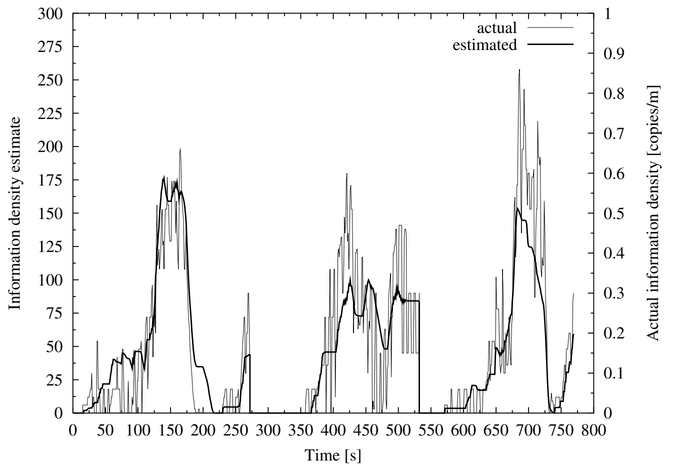

Fig. 4. Actual and estimated density of a generic information item as a function of time, as seen by a network node travelling on the highway

## B. Information density estimation

Our first goal is to ascertain the accuracy of the procedure for information density estimation on which Eureka is based. Note that the estimates and actual values reported in the following plots have different numerical ranges since estimating the *exact* value of information density would require that vehicles know the geographical distance corresponding to each hop distance. Furthermore, since Eureka just uses the *difference* between estimates of the information density computed at two points, it matters that estimates match the trend of the actual density, not the absolute value.

In all simulation results reported here, we take the cardinality $N$ of the information set to be equal to 15 and the number $C$ of requested chunks per query equal to 5. For each vehicle we set the query generation rate $\lambda$ and the cache dropping rate $\mu$ at 0.006 queries/s and 0.025 drops/s, respectively. We underline that several tests were performed with different values of $\lambda$ and $\mu$, obtaining similar results. A comparison of density estimate and actual density in a highway scenario is provided by Figure 4. We followed 3 trips of a sample vehicle as it travels on the highway (each trip is delimited by a gap in the information density estimate, corresponding to an inactivity period). Each trip duration is different since, after being repositioned at the beginning of the highway, a new speed is chosen and different traffic conditions are encountered. The y-axis reports, on the left, the information density estimated by the moving vehicle and, on the right, the actual information density in copies/m that was computed in a range of 100 m around the vehicle at each time instant. The comparison in Figure 4 shows that the density estimation behavior is a very good match of the actual density behavior.

We repeated a similar collection of estimates and density values in the urban scenario; the results are presented in Figure 5. To better understand the behavior of the information density, the figure is added a strip marking the times when the vehicle crosses an intersection (crosses), or reaches the gateway node (filled circles). Intuitively, these spots exhibit higher densities because of the vehicle queues at the intersections, and because of the presence of the gateway node. It can be seen that each density peak is matched by an estimate peak, confirming the accuracy of the estimation procedure.

Finally, we mention that we also compared the actual and the estimated information density at fixed time instants, in snapshots of the highway and urban scenarios (plots are omitted here for the lack of space). Again, we observed an excellent match of estimates and actual values trends in all the considered cases.

## C. Curbing the traffic load

We now look at the benefits that can be derived by exploiting the density estimate to improve the system performance. Here we compare the performance of flooding, mitigated flooding (i.e., flooding with $TTL$ and query lag) and Eureka in terms of:

TABLE I
PERFORMANCE UNDER THE HIGHWAY SCENARIO, FOR $N$=15, $C$=15, $\lambda$=0.003 AND VARYING $\mu$

| | Query traffic [Mbps] | | Information traffic [Mbps] | | Solved queries (requests/s) | | Query solving time [s] | |
|---|---|---|---|---|---|---|---|---|
| | $\mu$=0.005 | $\mu$=0.025 | $\mu$=0.005 | $\mu$=0.025 | $\mu$=0.005 | $\mu$=0.025 | $\mu$=0.005 | $\mu$=0.025 |
| Flooding | 0.8 | 1.0 | 11.1 | 10.4 | 5.1 | 4.4 | 46.6 | 51.2 |
| Mitigated flooding | 0.7 | 0.9 | 9.4 | 9.0 | 5.2 | 4.7 | 44.1 | 47.6 |
| Eureka | 0.06 | 0.1 | 4.6 | 5.2 | 5.3 | 5.3 | 35.5 | 40.3 |

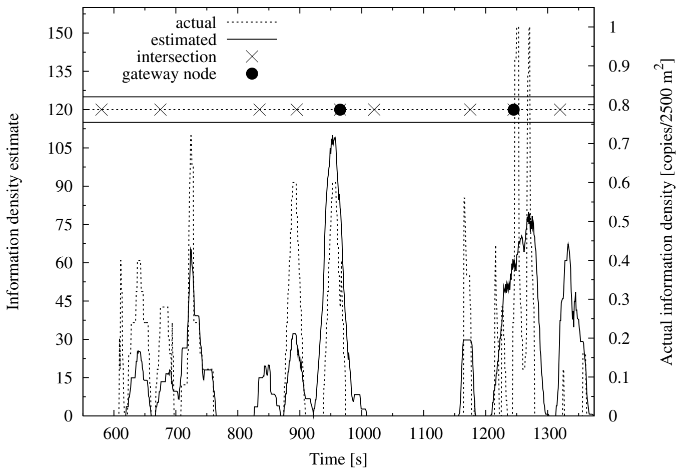

Fig. 5. Actual and estimated density of a generic information item as a function of time, as seen by a network node travelling in the urban scenario; a strip at the top of the plot highlights the times at which the node approaches intersections and gateways

- *query traffic*: the total amount of traffic ascribed to the transmission (and replication) of queries in the whole network; results take into account the additional density-related fields in the query header for the case of Eureka;
- *information traffic*: the total amount of traffic ascribed to the transmission (and replication) of reply messages carrying the requested information back to query source; results take into account also duplicated reply messages;
- *solved queries*: the number of queries per second that receive at least a successful reply;
- *query solving time*: the average time period elapsed since the generation of a query until the reception of the last missing chunk at the requesting node; clearly only solved queries contribute to these statistics.

We first consider the highway scenario, and evaluate the above performance metrics for $N$=15, $C$=15, and different values of query generation rate and information dropping rate. We highlight that similar trends were observed by setting $N$ and $C$ at different values.

Let us consider the results in Table I, obtained for $\lambda$=0.003 and $\mu$=0.005, 0.025. Note that $\lambda$=0.003 and $\mu$=0.005 correspond, respectively, to vehicles generating a query per information item about every 6 minutes and to a chunk caching time of about 3 minutes. Also, given $\lambda$, an increase in $\mu$ leads to a reduction in the amount of information stored at the network nodes and, thus, to a more difficult content retrieval. As a consequence, larger $\mu$'s imply a growth in the query traffic and

TABLE II
PERFORMANCE OF EUREKA UNDER THE HIGHWAY SCENARIO, FOR $N$=15, $\lambda$=0.003, $\mu$=0.005 AND VARYING $C$

| | $C$=5 | $C$=15 | $C$=30 |
|---|---|---|---|
| Information traffic [Mbps] | 4.0 | 4.6 | 4.7 |
| Query solving time [s] | 52.8 | 35.5 | 33.4 |

query solving time for all the schemes under study. However, comparing the performance of the three strategies, we observe that, for both values of $\mu$, flooding and mitigated flooding give a query traffic that is one order of magnitude greater than with Eureka. It follows that the information traffic is almost doubled without Eureka: such increase is mainly due to a score of duplicated reply messages that were elicited by as many unnecessary queries that reached faraway nodes in areas where the information was less dense. However, were those queries really unnecessary? They were, according to the rate of solved information queries and to the query solving times reported in Table I. Indeed, for $\mu$=0.005 Eureka provides a rate of return of information items at a node requesting them that is slightly higher than in the case of flooding-based strategies. With a faster cache drop rate ($\mu$=0.025), the improvement becomes even more evident, due to the lesser query traffic, hence the lower congestion level, that it causes. This is an interesting behavior, which highlights how the performance of flooding-based techniques in systems using a contention-based channel access scheme, may severely degrade due to the collisions that affect the query propagation. As for the query solving time, we observe that Eureka allows significantly faster query responses, thanks again to the lower congestion level that it causes.

Overall, the results obtained in the highway scenario show that a better outcome in terms of user satisfaction (i.e., higher number of solved queries and lower query response time) is achieved through Eureka, despite the great reduction in the number of queries and, in turn, of reply messages.

As a last observation, we show the system behavior as the number $C$ of requested chunks per query varies. We set $\lambda$=0.003 and $\mu$=0.005, and report the results obtained with Eureka in Table II. Only the information traffic and the query solving time are presented, since, as expected, the values of the other metrics do not change significantly with $C$. Interestingly, the larger the $C$, the faster the query response but the higher the information traffic. Indeed, when a large number of chunks are requested with the same query message, two behaviors can be observed. On the one hand, a node receiving the query

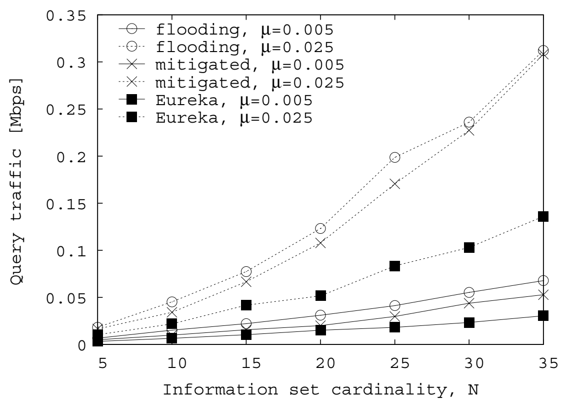

Fig. 6. Query traffic as a function of the number of information items available in the network, for different values of the dropping rate $\mu$ (urban scenario)

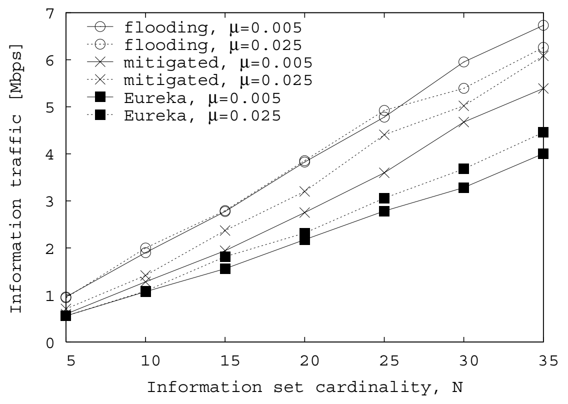

Fig. 7. Information message traffic versus the number of information items available in the network, for different values of the dropping rate $\mu$ (urban scenario)

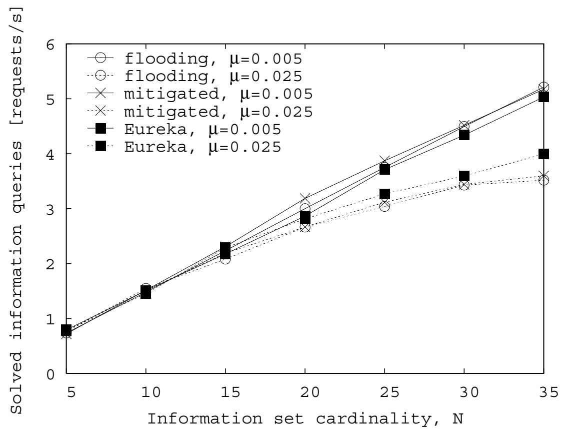

Fig. 8. Number of information queries solved per second versus the number of information items available in the network, for different values of the dropping rate $\mu$ (urban scenario)

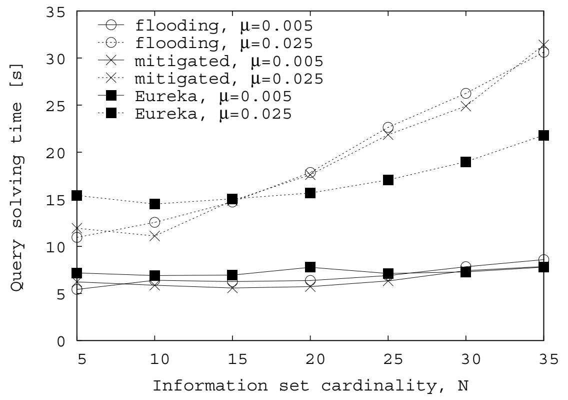

Fig. 9. Query response time versus the number of information items available in the network, for different values of the dropping rate $\mu$ (urban scenario)

replies by immediately sending all the requested chunks it has in its cache. On the other hand, if multiple nodes owning the information receive the query, all of them will reply by returning a large number of chunks, thus causing an increase in the information traffic. Based on these observations, in the following we consider $C$=15, as a good value to balance between information traffic and system responsiveness.

Next, we study the same performance metrics in the urban scenario. We set $C$=15, $\lambda$=0.003 and $\mu$=0.005, 0.025, while the number $N$ of information items varies. Figure 6 shows that, when medium-large information sets are considered (i.e., 10 items and more), the difference between the query traffic recorded in the case of mitigated flooding and for Eureka is around 100%, for both values of $\mu$. The gain of Eureka with respect to flooding is even higher. We also note that flooding and mitigated flooding give very close results. Recall that, under mitigated flooding, nodes receiving a query, but not owning the information, further propagate requests for the chunks that are not transmitted within the query lag time. For values of $C$ greater than 5, we observed quite bursty transmissions of reply messages, with a duration potentially longer than the query lag time. This implies that the effectiveness of the query lag time is greatly reduced, unless large delays are introduced.

Figure 7 collects information traffic values and confirms that a significant amount of extra information is generated as a consequence of the additional queries. The smaller gap (with respect to query traffic) between mitigated flooding and Eureka is justified by the fact that mitigated flooding propagates queries toward regions where information is scarce: about twice the query traffic is generated by mitigated flooding (compared to Eureka), while the reply messages are, on average, a mere 30% more. For each scheme, as the dropping rate $\mu$ increases, the information traffic increases as well, since the information is retrieved farther from the requesting node and several hops are necessary to deliver it. The only exception is in the case of flooding and information set cardinality greater than 25: this is because for $\mu = 0.025$ there are fewer replies, as a consequence of the high congestion caused by the query traffic. A confirmation is provided by Figure 8 showing the solving rate. For $\mu = 0.005$, the solving rates of the three schemes are very close, although this result is achieved by Eureka through a sizable reduction of query and information traffic. For large information sets and $\mu = 0.025$, even lower solving rates are achieved in the case of flooding and mitigated flooding, than with Eureka. Figure 9 presents the query response time. We observe that in general Eureka gives results that are comparable to the performance of flooding-based schemes, but it significantly outperforms flooding and mitigated flooding for $\mu = 0.025$ and an information set larger than 20.

## VII. Related Work

The technical challenges of delivering information to vehicles forming an ad hoc network are outlined in [2], while content delivery and sharing are studied in [1]. The cooperative downloading strategy in [1], named SPAWN, addresses peer discovery, content selection, and content discovery. Peer discovery, like our solution, leverages the broadcast nature of the wireless medium thus allowing nodes to overhear information about the content availability at neighbors. In [10] an in-vehicle entertainment system for downloading of audio and video traffic is studied; however the focus there is on predicting the availability of the information, rather than where the information can be found.

Relevant to our work are also the studies on service discovery. In [11], [12], service discovery protocols for large-scale MANETs are presented, which are based on the deployment of a virtual backbone of directories within the network. Each directory performs service discovery in its proximity, while global service discovery is provided by the cooperative action of the directories composing the backbone. Similar in scope to our work is the solution in [13], where a service discovery protocol aiming at an efficient usage of the network bandwidth is presented. More specifically, the protocol involves the transmission of service advertisements by each node that hosts a service or knows that one of its neighbors is hosting it; also, nodes cache the received advertisements for a given time interval. Based on the cached advertisements, a node can know at which hop distance a service may be found, or the nodes to which the service request can be selectively sent.

With regards to routing, several solutions have been proposed to reduce the routing overhead of on-demand protocols. For instance, Location Aided Routing [6] and Query Localization [14] limit the query flood by decreasing the number of nodes receiving route queries. The mechanism in [6] restricts the flooding of queries using GPS, while in [14] route requests are forwarded only in those areas where old paths existed. In the context of sensor networks, the study in [15] exploits the natural information gradient exhibited by physical phenomena to efficiently route queries toward the event source. Note that our work significantly differs from [15] since one of the main contributions is the definition of the *information density* concept in MANETs and of the procedure to estimate it.

Finally, an early version of the basic functionalities required by our system (the ones described in Sec. II) are described in detail in our previous work [3]. As for the concept of *information density*, we highlight that it is used also in some studies on sensor networks, such as [16]. There, however, an information density function $\rho(x, y)$, is introduced to model the position of source (positive values of $\rho(x, y)$) and sink nodes (negative values of $\rho(x, y)$). In our work, instead, the information density at a given location represents the distribution of contents in the MANET.

## VIII. Conclusions

We looked at a MANET as a peer-to-peer network where mobile users may request information contents as well as provide them to other nodes. We developed an efficient information retrieval mechanism, named Eureka, that specifically addresses the challenges posed by a wireless mobile environment. Eureka users estimate the information density in their proximity and use it to direct queries toward areas where the requested information is denser. Simulating a vehicular environment with realistic mobility models we showed that: (i) our information density estimate closely follows the behavior of the actual information density; (ii) the traffic due to query and duplicated information messages is greatly reduced, while the number of solved requests and the system responsiveness are preserved, if not even improved.

## Acknowledgment

This work has been supported by the Meadow project.

## References

[1] A. Nandan *et al.*, "Co-operative downloading in vehicular ad-hoc wireless networks," in *Proc. WONS'05*, St. Moritz, Switzerland, Jan. 2005, pp. 32–41.

[2] S. Ghandeharizadeh and B. Krishnamachari, "C2P2: Peer-to-peer network for on-demand automobile information services," in *Proc. GLOBE'04*, Zaragoza, Spain, Sep. 2004.

[3] M. Fiore, C. Casetti, and C.-F. Chiasserini, "On-demand content delivery in vehicular wireless networks," in *Proc. IEEE/ACM MSWiM'05*, Montreal, Canada, Oct. 2005.

[4] Z. Cheng and W. Heinzelman, "Flooding strategy for target discovery in wireless networks," in *Proc. ACM MSWiM'03*, San Diego, CA, Sep. 2003, pp. 33–41.

[5] N. Chang and M. Liu, "Optimal controlled flooding search in a large wireless network," in *Proc. WiOpt'05*, Riva del Garda, Italy, Apr. 2005.

[6] Y.-B. Ko and N. H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," *Wireless Networks*, vol. 6, no. 4, pp. 307–21, July 2000.

[7] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev.*, vol. 62, no. 2, Aug. 2000.

[8] Canu project. [Online]. Available: http://canu.informatik.uni-stuttgart.de

[9] Vanetmobisim project. [Online]. Available: http://vanet.eurecom.fr

[10] S. Ghandeharizadeh, S. Kapadia, and B. Krishnamachari, "PAVAN: A policy framework for content availability in vehicular ad-hoc networks," in *Proc. ACM VANET'04*, Philadelphia, PA, Oct. 2004.

[11] U. C. Kozat and L. Tassiulas, "Network layer support for service discovery in mobile ad hoc networks," in *Proc. IEEE INFOCOM'03*, San Francisco, CA, march 2003, pp. 1965–1975.

[12] F. Sailhan and V. Issarny, "Scalable service discovery for MANET," in *Proc. IEEE PerComm'05*, Kauai Island, Hawaii, Mar. 2005.

[13] D. Chakraborty and A. Joshi, "GSD: A novel group-based service discovery protocol for MANETS," in *Proc. IEEE MWCN'02*, Stockholm, Sweden, Sep. 2002.

[14] R. Castaneda, S. R. Das, and M. K. Marina, "Query localization techniques for on-demand routing protocols in ad hoc networks," in *Proc. ACM MobiCom'99*, Seattle, WA, Aug. 1999, pp. 186–194.

[15] J. Faruque and A. Helmy, "Gradient-based routing in sensor networks," in *Proc. ACM MobiCom'03*, San Diego, CA, Sep. 2003, poster.

[16] S. Toumpis and L. Tassiulas, "Packetostatics: Deployment of massively dense sensor networks as an electrostatics problem," in *Proc. IEEE INFOCOM'05*, Miami, FL, Mar. 2005, pp. 2290–2301.