# Caching Strategies Based on Information Density Estimation in Wireless Ad Hoc Networks

[1]D.Nagaraju, [2]L.Srinivasa rao, [3]K.Nageswara Rao
[123]Sathupally MIST Khammam, Andhra Pradesh, India

**Abstract:**
We address cooperative caching in wireless networks, where the nodes may be mobile and exchange information in a peer-to-peer fashion. We consider both cases of nodes with largeand small-sized caches. For large-sized caches, we devise a strategy where nodes, independent of each other, decide whether to cache some content and for how long. In the case of small-sized caches, we aim to design a content replacement strategy that allows nodes to successfully store newly received information while maintaining the good performance of the content distribution system. Under both conditions, each node takes decisions according to its perception of what nearby users may store in their caches and with the aim of differentiating its own cache content from the other nodes'. The result is the creation of content diversity within the nodes neighborhood so that a requesting user likely finds the desired information nearby. We simulate our caching algorithms in different ad hoc network scenarios and compare them with other caching schemes, showing that our solution succeeds in creating the desired content diversity, thus leading to a resource-efficient information access.

**Index Terms**— Data caching, mobile ad hoc networks.

## I. Introduction

**P**ROVIDING information to users on the move is one of the most promising directions of the infotainment business, which rapidly becomes a market reality, because infotainment modules are deployed on cars and handheld devices. The ubiquity and ease of access of third- and fourth-generation (3G or 4G) networks will encourage users to constantly look for content that matches their interests. However, by exclusively relying on downloading from the infrastructure, novel applications such as mobile multimedia are likely to overload the wireless network (as recently happened to AT&T following the introduction of the iPhone [1]). It is thus conceivable that a peer-to-peer system could come in handy, if used in conjunction with cellular networks, to promote content sharing using ad hoc networking among mobile users [2]. For highly popular content, peer-to-peer distribution can, indeed, remove

• ***Large-sized caches.***
In this case, nodes can potentially store a large portion (i.e., up to 50%) of the available information items. Reduced memory usage is a desirable (if not required) condition, because the same memory may be shared by different services and applications that run at nodes. In such a scenario, a caching decision consists of computing for how long a given content should be stored by a node that has previously requested it, with the goal of minimizing the memory usage without affecting the overall information retrieval performance;

• ***Small-sized caches.***
In this case, nodes have a dedicated but limited amount of memory where to store a small percentage (i.e., up to 10%) of the data that they retrieve. The caching decision translates into a cache replacement strategy that selects

The information items to be dropped among the information items just received and the information items that already fill up the dedicated memory. We evaluate the performance of Hamlet in different mobile network scenarios, where nodes communicate through ad hoc connectivity. The results show that our solution ensures a high query resolution ratio while maintaining the traffic load very low, even for scarcely popular content, and consistently along different network connectivity and mobility scenarios.

### *A. Cooperative Caching*

Distributed caching strategies for ad hoc networks are presented according to which nodes may cache highly popular content that passes by or record the data path and use it to redirect future requests. Among the schemes presented in [9], the approach called Hybrid Cache best matches the operation and system assumptions that we consider; we thus employ it as a benchmark for Hamlet in our comparative evaluation. In [10], a cooperative caching technique is presented and shown to provide better performance than Hybrid Cache. However, the solution that was proposed is based on the formation of an overlay network composed of "mediator" nodes, and it is only fitted to static connected networks with stable links among nodes. These assumptions, along with the significant communication overhead needed to elect "mediator" nodes, make this scheme unsuitable for the mobile environments that we address. The work in [11] proposes a complete framework for information retrieval and caching in mobile ad hoc networks, and it is built on an underlying routing protocol and requires the manual setting of a networkwide "cooperation zone" parameter. Note that assuming the presence of a routing protocol can prevent the adoption of the scheme in [11] in highly mobile networks, where maintaining network connectivity

is either impossible or more communication expensive than the querying/ caching process. Furthermore, the need of a manual calibration of the "cooperation zone" makes the scheme hard to configure, because different environments are considered. Conversely, Hamlet is self contained and is designed to self adapt to network environments with different mobility and connectivity features. One vehicular ad hoc network scenario is addressed in [12], where the authors propose both an information retrieval technique that aims at finding the most popular and relevant data matching a user query and a popularity-aware data replacement scheme. The latter approach ensures that the density of different content is proportional to the content's popularity at the system steady state, thus obeying the square-root rule proposed in [13] for wired networks. We point out that the square-root rule does not consider where copies of the data are located but only how many copies are created. It is thus insufficient in network environments whose dynamism makes the positioning of content of fundamental importance and renders steady-state conditions (as assumed in [13]) hard to be achieved.

***B. Content Diversity***

Similar to Hamlet, in [6], mobile nodes cache data items other than their neighbors to improve data accessibility. In particular, the solution in [6] aims at caching copies of the same content farther than a given number of hops. Such a scheme, however, requires the maintenance of a consistent state among nodes and is unsuitable for mobile network topologies. The concept of caching different content within a neighborhood is also exploited in [7], where nodes with similar interests and mobility patterns are grouped together to improve the cache hit rate, and in [8], where neighboring mobile nodes implement a cooperative cache replacement strategy. In both works, the caching management is based on instantaneous feedback from the neighboring nodes, which requires additional messages. The estimation of the content presence that we propose, instead, avoids such communication overhead.

***C. Caching With Limited Storage Capability***

In the presence of small-sized caches, a cache replacement technique needs to be implemented. Aside from the scheme in [8], centralized and distributed solutions to the cache placement problem, which aim at minimizing data access costs when network nodes have limited storage capacity, are presented in [14]. Although centralized solutions are not feasible in ad hoc environments, the distributed scheme in [14] makes use of cache tables, which, in mobile networks, need to be maintained similar to routing tables. Hamlet does not rely on cache tables, and thus, it does not incur the associate high communication penalty. In [15], a content replacement strategy that aims at minimizing energy consumption is proposed. To determine which content should be discarded, the solution exploits the knowledge of data access probabilities and distance from the closest provider—an information that is typically hard to obtain and is not required by Hamlet. A content replacement scheme that addresses storage limitations is also proposed in [6]. It employs a variant of the last recently used (LRU) technique, which favors the storage of the most popular items instead of the uniform content distribution targeted by Hamlet. In addition, it exploits the cached item IDs provided by the middleware to decide on whether to reply to passing-by queries at the network layer, as well as link-layer traffic monitoring to trigger prefetching and caching. In [17], the popularity of content is taken into account, along with its update rate, so that items that are more frequently updated are more likely to be discarded. Similarly, in [18], cache replacement is driven by several factors, including access probability, update frequency, and retrieval delay. These solutions thus jointly address cache replacement and consistency, whereas in this paper, we specifically target the former issue. However, as will be pointed out, Hamlet can easily be coupled with a dedicated cache consistency scheme, e.g., [9] and [12]. 2196 IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 60, NO. 5, JUNE 2011

***D. Data Replication***

Although addressing a different problem, some approaches to data replication are relevant to the data caching solution that we propose. One technique of eliminating information replicas among neighboring nodes is introduced in [` 11], which, unlike Hamlet, requires knowledge of the information access frequency and periodic transmission of control messages to coordinate the nodes' caching decisions. In [5], the authors propose a replication scheme that aims at having every node close to a copy of the information and analyze its convergence time. However, unlike Hamlet, the scheme implies a significant overhead and an exceedingly high convergence time, thus making it unsuitable for highly variable networks. Finally, the work in [22] adopts a cross-layer approach to data replication in mobile ad hoc networks, where network-layer information on
the node movement path helps to trigger the replication before network partitioning occurs.

## Iii. System Outline and Assumptions

Hamlet is a fully distributed caching strategy for wireless ad hoc networks whose nodes exchange information items in a peer-to-peer fashion. In particular, we address a mobile ad hoc network whose nodes may be resource-constrained devices, pedestrian users, or vehicles on city roads. Each node runs an application to request and, possibly, cache desired information items. Nodes in the network retrieve information items from other users that temporarily cache (part of) the requested items or from one or more gateway nodes, which can store content or quickly fetch it from the Internet. We assume a content distribution system where the following assumptions hold:

1) A number $I$ of *information items* is available to the users, with each item divided into a number $C$ of *chunks*;
2) user nodes can overhear queries for content and relative responses within their radio proximity by exploiting the broadcast nature of the wireless medium; and
3) user nodes can estimate their distance in hops from the query source and the responding node due to a hop-count field in the messages.

Although Hamlet can work with *any* system that satisfies the aforementioned three generic assumptions, for concreteness, we detail the features of the specific content retrieval system that we will consider in the emainder of this paper. The reference system that we assume allows user applications to request an information item $i$ ($1 \leq i \leq I$) that is not in their cache. Upon a request generation, the node broadcasts a *query message* for the $C$ chunks of the information item. Queries for still missing chunks are periodically issued until either the information item is fully retrieved or a timeout expires. If a node receives a fresh query that contains a request for information $i$'s chunks and it caches a copy of one or more of the requested chunks, it sends them back to the requesting node through *information messages*. If the node does not cache (all of) the requested chunks, it can rebroadcast a query for the missing chunks, thus acting as a forwarder. The exact algorithm that is followed by a node upon the reception of a query message is detailed in the flowchart in Fig. 1

(a). Fig. 1. Flowcharts of the processing of (a) query and (b) information messages at user nodes.We denote the address of the node that generated the query as *asrc*, the query identifier as *id*, the address of the last node that forwarded the query message as *alast*, and the set of queried chunks as $\bar{c}$. The functional blocks that are the focus of this paper are highlighted in

(b). Once created, an information message is sent back to the query source. To avoid a proliferation of information copies along the path, the only node that is entitled to cache a new copy of the information is the node that issued the query. Information messages are transmitted back to the source of the request in a unicast fashion, along the same path from which the request came. To this end, backtracking information is carried and updated in query messages. Nodes along the way either act as relays for transit messages (if they belong to the backtracking node sequence) or simply overhear their transmission without relaying them. Fig. 1(b) depicts the flowchart of the operations at a node that receives a message that contains an information chunk. A node that receives the requested information has the option to cache the received content and thus become a provider for that content to the other nodes. Determining a strategy of taking such caching decisions is the main objective of this paper, and as such, the corresponding decision blocks are highlighted in Fig. 1(b).

We point out that Hamlet exploits the observation of query and information messages that are sent on the wireless channel as part of the operations of the content-sharing application, e.g., the previously outlined approach. As a consequence, Hamlet does not introduce any signaling overhead. Furthermore, several optimizations can be introduced to improve the aforementioned basic scheme for the discovery of content. Although our focus is not on query propagation, it is important to take the query process into account, because it directly determines the network load associated with the content retrieval operation. While deriving the results, we consider the following two approaches to query propagation. 1) *Mitigated flooding*. This approach limits the propagation range of a request by forcing a time to live (*TTL*) for the query messages. In addition, it avoids the forwarding of already-solved requests by making the nodes wait for a *query lag time* before rebroadcasting a query;

2) *Eureka* [13]. This approach extends mitigated flooding by steering queries toward areas of the network where the required information is estimated to be denser. Note that this paper focuses on cooperative caching and we do not tackle information consistency; thus, we do not take into account different versions of the content in the system model. We note, however, that the previous version of this paper [14] jointly evaluated Hamlet with a basic scheme for weak cache consistency based on an epidemic diffusion of an updated cache content and we showed that weak consistency can be reached, even with such a simple approach, with latencies on the order of minutes for large networks. If prompter solutions are sought, Hamlet lends itself to be easily integrated with one
of the existing consistency solutions found in the literature (e.g., [9], [12], [15], ). In particular, these works propose push, pull, or hybrid approaches to achieve different levels of cache consistency. In the case of Hamlet, a *push* technique can be implemented through the addition of invalidation messages broadcast by gateway nodes, whereas information providers can *pull* an updated content (or verify its freshness) before sending the information to querying nodes. In either case, no major modification of the Hamlet caching scheme is required: the only tweaking can consist of resetting the estimation of the infor-mation presence upon the notification/detection of an updated version to ease the diffusion of the new information.

## IV. Hamlet Framework

The Hamlet framework allows wireless users to take *caching decisions* on content that they have retrieved from the network. The process that we devise allows users to take such decisions by leveraging a node's local observation, i.e., the node's ability to overhear queries and information messages on the wireless channel. In particular, for each information item, a node records the distance (in hops) of the node that issues the query, i.e., where a copy of the content is likely to be stored, and the distance of the node that provides the information. Based

on such observations, the node computes an index of the *information presence* in its proximity for each of the $I$ items. Then, as the node retrieves content that it requested, it uses the presence index of such an information item to determine Fig. 2. whether a copy of the content should be cached, for how long, and possibly which content it should replace. By doing so, a node takes caching decisions that favor high content diversity in its surroundings, inherently easing the retrieval of data in the network. Note that our technique works on a per-item basis, and its results apply to all chunks that belong to the same content. In the following sections, we first detail how a node estimates the presence of information chunks in its proximity. Next, we separately describe the role of the information presence index in caching decisions for nodes with large- and small-sized caches. In the former case, the information presence index determines the cache content drop time, whereas in the latter case, it drives the cache content replacement.

### *A. Information Presence Estimation*

We define the *reach range* of a generic node $n$ as its distance from the farthest node that can receive a query generated by node $n$ itself. As an example, in an ideal setting in which all nodes have the same radio range, the reach range is given by the product of the TTL and the node radio range. Next, we denote by $f$ the frequency at which every node estimates the presence of each information item within its *reach range*, and we define as $1/f$ the duration of each estimation step (also called time step hereafter). A node $n$ uses the information that was captured within its reach range during time step $j$ to compute the following two quantities: 1) a provider counter by using application-layer data and 2) a transit counter by using data that were collected through channel overhearing in a cross-layer fashion. These counters are defined as follows.

• *Provider counter* $dic(n, j)$. This quantity accounts for the presence of new copies of information $i$'s chunk $c$, delivered by $n$ to querying nodes within its reach range, during step $j$. Node $n$ updates this quantity every time it acts as a provider node (e.g., node $P$ in the upper plot of Fig. 2).

• *Transit counter* $ric(n, j)$. This quantity accounts for the presence of new copies of information $i$'s chunk $c$, transferred between two nodes within $n$'s reach range and FIORE *et al.*: CACHING STRATEGIES BASED ON INFORMATION DENSITY ESTIMATION IN AD HOC NETWORKS 2201 replacement schemes in [9] and [14]. It is composed of 300 wireless nodes deployed over a square area of a side equal to 200 m. Nodes can be static, positioned according to a uniform random distribution, or mobile, wandering according to a random-direction mobility model with reflections. The node speed is uniformly distributed in the range $[0.5vm, 1.5vm]$, where $vm$ is the average node speed—a varying parameter in our simulations. The node radio range is set to 20 m, resulting, for static nodes, in a fully connected network. In all the scenarios, we deploy two fixed gateway nodes at opposite ends of the topology. Each gateway permanently stores 1/2 of the information items, whereas the other half is provided by the other gateway. Initially, nodes have an empty cache; they randomly request any among the $I$ items that are not in their cache according to a Poisson process with parameter $\lambda i = \Lambda qi$ $(1 \leq i \leq I)$. $\Lambda$ is the query generation rate per node, whereas $qi$ represents the content popularity level (i.e., the probability that, among all possible content, a node requests item $i$). The *TTL* value for query messages is set to ten and five hops for the case of large- and small-sized caches, respectively, and the query lag time is 50 ms. Note that the impact of all the aforementioned query propagation parameters on the information-sharing behavior has been studied in [23]; here, we only consider what has been identified as a good parameter setting. With regard to the Hamlet parameters, the estimation frequency is such that $1/f = 0.2MC$; indeed, through extensive simulations, we observed that the impact of $f$ is negligible, as long as $1/f$ is not greater than 20% of the maximum caching time. As we fix $\tau = fMC$, this setting of $f$ leads to a value of $\tau$ as small as 5. Then, we have $\alpha = 0.9$ and $W = 0.5$; indeed, we have verified that this combination yields a smoother behavior of the presence index $pi(n, j)$. The values of the remaining parameters are separately specified for large- and small-sized caches. The information-sharing application lies on top of a User Datagram Protocol (UDP)-like transport protocol, whereas, at the media access control (MAC) layer, the IEEE 802.11 standard in the promiscuous mode is employed. No routing algorithm is implemented: queries use a MAC-layer broadcast transmission, and information messages find their way back to the requesting node following a unicast path. Information messages exploit the request to send/clear to send (RTS/CTS) mechanism and MAC-level retransmissions, whereas query messages of broadcast nature do not use RTS/CTS and are never retransmitted. The channel operates at 11 Mb/s, and signal propagation is reproduced by a two-ray ground model. Simulations were run for 10 000 s. In the aforementioned scenarios, our performance evaluation hinges upon the following quite-comprehensive set of metrics that are aimed at highlighting the benefits of using Hamlet in a distributed scenario:

1) the ratio between solved and generated queries, called solved-queries ratio;
2) the communication overhead;
3) the time needed to solve a query;
4) the cache occupancy.

We have further recorded the spatiotemporal distribution of information and the statistics of information survival, because they help in quantifying the effectiveness of Hamlet in preserving access to volatile information. As aforementioned, we did not explore the problem of cache consistency, because such an
issue is orthogonal to this paper.

## Vi. Evaluation With Large-Sized Caches

Here, we evaluate the performance of Hamlet in a network of nodes with large storage capabilities, i.e., with caches that can store up to 50% of all information items. Because such characteristics are most likely found in vehicular communication devices, tablets, or smartphones, the network environments under study are the City and Mall scenarios. As discussed in Section IV, in this case, the Hamlet framework is employed to compute the caching time for information chunks retrieved by nodes, with the goal of improving the content distribution in the network while keeping the resource consumption low.

We first compare Hamlet's performance to the results obtained with a deterministic caching strategy, called *DetCache*, which simply drops cached chunks after a fixed amount of time. Then, we demonstrate the effectiveness of Hamlet in the specific task of information survival. In all tests, we assume $I$ = 10 items, each comprising $C$ = 30 chunks. All items have identical popularity, i.e., all items are requested with the same rate $\lambda = \Lambda/I$ by all network nodes. The choice of equal request rates derives from the observation that, in the presence of nodes with a large-sized memory, caching an information item does not imply discarding another information item; thus, the caching dynamics of the different items are independent of each other and only depend on the absolute value of the query rate. It follows that considering a larger set of items would not change the results but only lead to more time-consuming simulations. Each query includes 20 B plus 1 B for each chunk request, whereas information messages include a 20-B header and carry a 1024-B information chunk. The maximum caching time $MC$ is set to 100 s, unless otherwise specified. Queries for chunks that are still missing are periodically issued every 5 s until either the information is fully retrieved or a timeout that is set to 25 expires.

### *A. Benchmarking Hamlet*

We set the deterministic caching time in DetCache to 40 s, and we couple DetCache and Hamlet with both the mitigated flooding and Eureka techniques for query propagation. We are interested in the following two fundamental metrics: 1) the ratio of queries that were successfully solved by the system and 2) the amount of query traffic that was generated. The latter metric, in particular, provides an indication of the system effectiveness in preserving locally rich information content: if queries hit upon the sought information in one or two hops, then the query traffic is obviously low. However, whether such a wealth of information is the result of a resource-inefficient cache-all-you-see strategy or a sensible cooperative strategy, e.g., the approach fostered by Hamlet, remains to be seen. Thus, additional metrics that are related to cache occupancy

TABLE I AVERAGE OCCUPANCY OF THE NODE CACHES, EXPRESSED AS A PERCENTAGE OF THE CHUNKS TOTAL NUMBER FOR $\lambda$ = 0.003 and information cache drop time must be coupled with the aforementioned metrics. Fig. 5 shows the solved-queries ratio (top plot) and the amount of query traffic (bottom plot) as $\lambda$ varies in the City scenario. When DetCache is used, the higher the query rate, the larger the number of nodes that cache an information item. This case implies that content can be retrieved with higher probability and also that it is likely to be found in the proximity of the requesting node, thus reducing the query traffic per issued request. Note that, due to its efficient query propagation mechanism, Eureka reduces the propagation of useless queries (and,hence, collisions), yielding a higher solved-queries ratio than mitigated flooding. However, it is evident that deterministic caching does not pay off as much as cooperative caching does in Hamlet. Table I shows that the average occupancy of node caches in Hamlet is comparable to the values observed with DetCache. Thus, it is the quality, not the quantity, of the information cached by Hamlet that allows it to top a sophisticated propagation scheme such as Eureka as far as the solved-queries ratio is concerned. The positive effect of the caching decisions can also be observed in Fig. 5 in terms of the reduced overhead and latency

TABLE II

AVERAGE QUERY SOLVING TIME (IN SECONDS), WITH $\lambda$ = 0.003 in solving queries. Indeed, Hamlet reduces the overhead by shortening the distance between requesting nodes and desired information content. Similarly, Table II shows how sensible caching choices can significantly reduce the time required to solve queries, again due to the homogeneous availability of information that they generate in the network. Further proof of such virtuous behavior by Hamlet is provided in Fig. 6, where mitigated flooding is used for query propagation. The figure depicts the time evolution of content presence over the road topology for one information item; in particular, the $z$-axis of each plot shows the fraction of different chunks that comprise an information item that are present in a squared area of 600 m2. On the one hand, it can be observed that mitigated flooding with DetCache creates a sharp separation between the area where the content source resides, characterized by high item availability, and the region where, due to vehicular traffic dynamics, information-carrying nodes rarely venture. On the other hand, Hamlet favors the diffusion of content over the entire scenario so that nodes in areas away from the information source can also be served. Fig. 7 refers to the Mall scenario. The poor performance of Eureka in this case is due to the lack of information items over large areas of the Mall scenario, resulting in queries not being forwarded and, thus, remaining unsolved [13]. Interestingly, Hamlet greatly reduces the query traffic for any $\lambda$, although providing a much higher solved-queries ratio. With regard to the caching occupancy, because Hamlet leads to results that are comparable with the results obtained with DetCache (see Table I, Mall scenario), it can be asserted that the performance gain achieved through Hamlet is due to the

more uniform content distribution across node caches. Finally, Table II confirms that such an improved availability of information shortens the waiting time to receive requested items. When comparing results obtained from the Mall and City scenarios, we note that the solved-queries ratio is consistently lower. We recall that vehicular mobility in the City environment is characterized by scattered connectivity but high node speed, whereas the Mall environment provides a better network connectivity level but reduced node mobility. The low node mobility in the Mall keeps items away from the sources of unpopular items for long periods of time. Thus, the probability of solving requests for such rare content is low, unless an efficient caching scheme allows nodes to preserve at least a few copies of every item in every neighborhood, as Hamlet does. It is also worth pointing out that, with respect to the City environment, the Mall includes a smaller number of nodes; thus, fewer queries are issued, and a much smaller amount of query traffic is generated. Finally, we may wonder how well Hamlet performs with respect to DetCache when the cache time employed by the latter approach is set to a value other than 40 s. Through extensive 2206 IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 60, NO. 5, JUNE 2011 Fig. 13. Memory-constrained mobile nodes: Query-solving ratio for each information item when using HybridCache and Hamlet, with $I$ = 300. The plots refer to *vm* that is equal to 1 m/s (left) and 15 m/s (right). Furthermore, it is not only the sheer quantity of data that makes a difference but its spatial distribution also plays a major role. If several nodes cache a rare item but they are all very close to each other, queries that were generated in other areas of the network take more hops to be satisfied. This case happens with HybridCache, as proven by the spatial distribution of the 100th, 200th, and 300th items, as shown in Fig. 12(a). Conversely, the spatial distribution achieved by Hamlet, as shown in Fig. 12(b), is more uniform, leading to a faster more likely resolution of queries. We now compare the performance of HybridCache and Hamlet in the scenario with memory-constrained mobile nodes. We test the two schemes when $I$ = 300 and for an average node speed *vm* equal to 1 and 15 m/s. The solved-queries ratio recorded with HybridCache and Hamlet on a per-item basis are shown in Fig. 13. Comparing the left and right plots, we note that the node mobility, even at high speed, does not seem to significantly affect the results due to the high network connectivity level. The spatial redistribution of content induced by node movements negatively affects the accuracy of Hamlet's estimation process, which explains the slight reduction in the solved query ratio at 15 m/s. That same movement favors HybridCache, at least at low speed, because it allows unpopular information to reach areas that are far from the gateway. However, the difference between the two schemes is evident, with Hamlet solving an average of 20% requests more than HybridCache, when nodes move at 15 m/s. Note that, for the query resolution delay and the average cache utilization at the network nodes, we obtained qualitatively similar results as in the static case, with Hamlet achieving more homogeneous solving times and fairer distribution of content in the network than HybridCache. *B. Impact of the Zipf Distribution Skewness* Finally, we study the impact of the Zipf distribution exponent on the performance of the cache replacement strategies. We recall that an exponent that is equal to zero implies perfect homogeneity, i.e., Zipf distribution that degenerates into a uniform distribution, whereas the difference in popularity among content becomes much more unbalanced as the exponent grows. We focus on a network where ten items are available and each node can cache at most one complete item. The choice of this setting is mandated by the fact that, in the presence of Fig. 14. Memory-constrained static (top) and mobile (bottom) nodes: Solvedqueries ratio and query traffic as the Zipf distribution exponent varies when using HybridCache and Hamlet, with $I$ = 10. hundreds of different items, unbalanced popularity distributions (i.e., exponents higher than 0.5) lead to very low $\lambda i$ for the 100 or so least popular items, thus making requests for such content extremely rare. Fig. 14 depicts the evolution of the solved-queries ratio and the query traffic as the Zipf exponent ranges vary. By comparing the two plots, we note that the presence of mobility (*vm* = 1 m/s) leads to a higher number of unsolved requests and in a larger amount of traffic generated within the network under HybridCache, because queries propagate far from the source without finding the desired item. However, what is most interesting is how the network load tends to decrease as the Zipf exponent grows, both in the absence and presence of node movements. On the one hand, higher values of the exponent lead to more unbalanced query rates, with very few items that are extremely popular and a long tail of seldom-accessed data.

Being requested so often, popular items become commonly found in nodes caches, and the relative queries are solved faster, generating small traffic. On the other, when the Zipf exponent is small, the distribution of queries is more balanced, with information more evenly distributed in the network. This case implies that items can usually be found but are hardly cached very close to the requesting node. Thus, the different items are all requested at a fairly high rate but are not immediately found, generating larger query traffic.

## VIII. Conclusion

We have introduced Hamlet, which is a caching strategy for ad hoc networks whose nodes exchange information items in a peer-to-peer fashion. Hamlet is a fully distributed scheme FIORE *et al.*: CACHING STRATEGIES BASED ON INFORMATION DENSITY ESTIMATION IN AD HOC NETWORKS 2207 where each node, upon receiving a requested information, determines the cache drop time of the information or which content to replace to make room for the newly arrived information. These decisions are made depending on the perceived "presence" of the content in the node's proximity, whose estimation does not cause any additional overhead to the information sharing system.

We showed that, due to Hamlet's caching of information that is not held by nearby nodes, the solving probability of information queries is increased, the overhead traffic is reduced with respect to benchmark caching strategies, and this result is consistent in vehicular, pedestrian, and memoryconstrained scenarios. Conceivably, this paper can be extended in the future by addressing content replication and consistency. The procedure for information presence estimation that was developed in Hamlet can be used to select which content should be replicated and at which node (even if such a node did not request the content in the first place). In addition, Hamlet can be coupled with solutions that can maintain consistency among copies of the same information item cached at different network nodes, as well as with the versions stored at gateway nodes.

## References


[1] J. Wortham (2009, Sep.). Customers Angered as iPhones Overload AT&T. *The New York Times*. [Online]. Available: http://www.nytimes.com/2009/09/03/technology/companies/03att.html

[2] A. Lindgren and P. Hui, "The quest for a killer app for opportunistic and delay-tolerant networks," in *Proc. ACM CHANTS*, 2009, pp. 59–66.

[3] P. Padmanabhan, L. Gruenwald, A. Vallur, and M. Atiquzzaman, "A survey of data replication techniques for mobile ad hoc network databases," *VLDB J.*, vol. 17, no. 5, pp. 1143–1164, Aug. 2008.

[4] A. Derhab and N. Badache, "Data replication protocols for mobile ad hoc networks: A survey and taxonomy," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 2, pp. 33–51, Second Quarter, 2009.

[5] B.-J. Ko and D. Rubenstein, "Distributed self-stabilizing placement of replicated resources in emerging networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 3, pp. 476–487, Jun. 2005.

[6] G. Cao, L. Yin, and C. R. Das, "Cooperative cache-based data access in ad hoc networks," *Computer*, vol. 37, no. 2, pp. 32–39, Feb. 2004.

[7] C.-Y. Chow, H. V. Leong, and A. T. S. Chan, "GroCoca: Group-based peer-to-peer cooperative caching in mobile environment," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 1, pp. 179–191, Jan. 2007.

[8] T. Hara, "Cooperative caching by mobile clients in push-based information systems," in *Proc. CIKM*, 2002, pp. 186–193.

[9] L. Yin and G. Cao, "Supporting cooperative caching in ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 5, no. 1, pp. 77–89, Jan. 2006.

[10] N. Dimokas, D. Katsaros, and Y. Manolopoulos, "Cooperative caching in wireless multimedia sensor networks," *ACM Mobile Netw. Appl.*, vol. 13, no. 3/4, pp. 337–356, Aug. 2008.

[11] Y. Du, S. K. S. Gupta, and G. Varsamopoulos, "Improving on-demand data access efficiency in MANETs with cooperative caching," *Ad Hoc Netw.*, vol. 7, no. 3, pp. 579–598, May 2009.

[12] Y. Zhang, J. Zhao, and G. Cao, "Roadcast: A popularity-aware content sharing scheme in VANETs," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, Los Alamitos, CA, 2009, pp. 223–230.

[13] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-topeer networks," in *Proc. ACM SIGCOMM*, Aug. 2002, pp. 177–190.

[14] B. Tang, H. Gupta, and S. Das, "Benefit-based data caching in ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 7, no. 3, pp. 289–304, Mar. 2008.

[15] W. Li, E. Chan, and D. Chen, "Energy-efficient cache replacement policies for cooperative caching in mobile ad hoc network," in *Proc. IEEE WCNC*, Kowloon, Hong Kong, Mar. 2007, pp. 3347–3352.


**D.Nagaraju pursuing** M.Tech in the department of computer Science & Engineering. His interested areas are Network Security and dataminining.

**L.Srinivasa Rao** working as assistant professor in the department of computer Science & Engineering. His interested areas are Network Security and dataminining

**K.Nageswara Rao** working as associate professor in the department of computer Science & Engineering. His interested areas are Network Security and dataminining