Secure and Efficient Transformation by using Caching Stratergies: Wireless Networks

[1]B.Ganga Bhavani, [2]M.Vishnu Murthy
[1]Dept. of CSE, Bonam Venkata Chalamayya Engg. College, Odalarevu, AP, India

Abstract
Most researches in ad hoc networks focus on routing, and not much work has been done on data access. A common technique used to improve the performance of data access is caching. Cooperative caching, which allows the sharing and coordination of cached data among multiple nodes, can further explore the potential of the caching techniques. Due to mobility and resource constraints of ad hoc networks, cooperative caching techniques designed for wired network may not be applicable to ad hoc networks. 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 large-and 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.

Keywords
Wireless Networks, Data Caching, Mobile Ad-Hoc Networks

I. Introduction

Wireless ad hoc networks have received considerable attention due to the potential applications in battlefield, disaster recovery, and outdoor assemblies. Ad hoc networks are ideal in situations where installing an infrastructure is not possible because the infrastructure is too expensive or too vulnerable. Due to lack of infrastructure support, each node in the network acts as a router, forwarding data packets for other nodes. Most of the previous researches in ad hoc networks focus on the development of dynamic routing protocols that can efficiently find routes between two communicating nodes.

A Mobile Aad-Hoc Network (MANET) is a collection of autonomous wireless nodes that may move unpredictably, forming a temporary network without any fixed backbone infrastructure . In such a network, each node not only plays the role of an end system, but also acts as a router that forwards packets to desired destination nodes. These nodes are capable of both single and multi-hop communication. Mobility and the absence of any fixed infrastructure make MANETs very attractive for military and rescue operations, sensor networks and time-critical applications.

Providing 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). 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. For highly popular content, peer-to-peer distribution can, indeed, remove bottlenecks by pushing the distribution from the core to the edge of the network.

In such an environment, however, a cache-all-you-see approach is unfeasible, because it would swamp node storage capacity with needless data that were picked up on the go. Thus, several techniques of efficiently caching information in wireless ad hoc networks have been investigated in the literature; for example, see the surveys and the related work discussed in Section II.

The solution that we propose, called Hamlet, aims at creating content diversity within the node neighborhood so that users likely find a copy of the different information items nearby (regardless of the content popularity level) and avoid flooding the network with query messages. Although a similar concept has been put forward, the novelty in our proposal resides in the probabilistic estimate, run by each node, of the information presence (i.e., of the cached content) in the node proximity. The estimate is performed in a cross-layer fashion by overhearing content query and information reply messages due to the broadcast nature of the wireless channel. By lever-aging such a local estimate, nodes autonomously decide what information to keep and for how long, resulting in a distributed scheme that does not require additional control messages. The Hamlet approach applies to the following cases.

A. 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;

B. 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.

II. Basic Definitions

Several papers have addressed content caching and content replacement in wireless networks. In the following sections, we review the works that are most related to this paper, highlighting the differences with respect to the Hamlet framework that we propose.

A. Data Replication

Database replication is the frequent electronic copying data from a database in one computer or server to a database in another so that all users share the same level of information. The result is a distributed database in which users can access data relevant to their tasks without interfering with the work of others. The implementation of database replication for the purpose of eliminating data ambiguity or inconsistency among users is known as normalization. Database replication can be done in at least three different ways:

- Snapshot replication: Data on one server is simply copied to another server, or to another database on the same server.
- Merging replication: Data from two or more databases is combined into a single database.
- Transactional replication: Users receive full initial copies of the database and then receive periodic updates as data changes.

A Distributed Database Management System (DDBMS) ensures that changes, additions, and deletions performed on the data at any given location are automatically reflected in the data stored at all the other locations. Therefore, every user always sees data that is consistent with the data seen by all the other users

B. Content Diversity

Similar to Hamlet, in mobile nodes cache data items other than their neighbors to improve data accessibility. In particular, the solution in 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, where nodes with similar interests and mobility patterns are grouped together to improve the cache hit rate, 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

Cache management is more complex in cooperative caching because deciding what to cache can also depend on the node's neighbors. Cooperative caching presents two problems: cache replacement and cache admission control.

In the presence of small-sized caches, a cache replacement technique needs to be implemented. Aside from the scheme in, 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. Although centralized solutions are not feasible in ad hoc environments, the distributed scheme 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 , 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 . 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, 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, 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.

D. Cooperative Caching

In 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 the approach called HybridCache best matches the operation and system assumptions that we consider; we thus employ it as a benchmark for Hamlet in our comparative evaluation. In, a cooperative caching technique is presented and shown to provide better performance than HybridCache. However, the solution that was proposed is based on the formation of an over-lay 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 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 network wide "cooperation zone" parameter. Note that assuming the presence of a routing protocol can prevent the adoption of the scheme in 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.

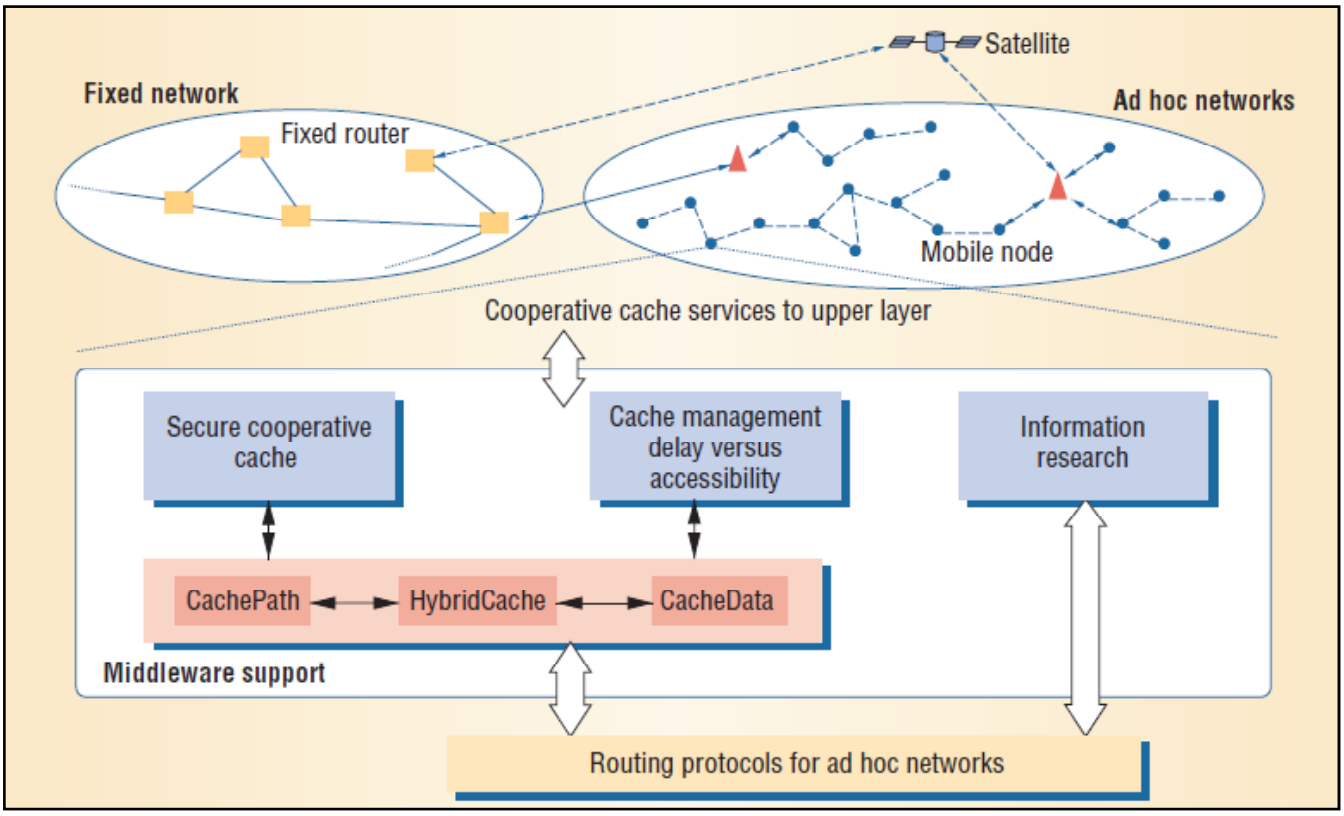

Fig. 1: Cooperative Caching in Ad-Hoc Networks. Middleware Supportmechanisms Provide Secure Cooperative Caching, Cache Management, and Information Search

One Ad-Hoc network scenario is addressed in, 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 for wired networks. It is thus insufficient in network environments whose dynamism makes the positioning of content of fundamental importance and renders steady-state conditions hard to be achieved

1. When a Data Item di Arrives

if di is the requested data by the current node
then cache di
else /* Data passing by */
if there is a copy of di in the cache
then update the cached copy if necessary
else if si < Ts or TTLi < TTTL then
cache data item di and the path;
else if there is a cached path for di, then
cache data item di;
update the path for di;
else
cache the path of di;

2. When a Request for Data Item di Arrives

if there is a valid copy in the cache
then send di to the requester;
else if there is a valid path for di in the cache then
forward the request to the caching node;
else
forward the request to the data center;

III. Current System

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.

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.

A. Cooperative caching Schema

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

ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

in the flowchart.

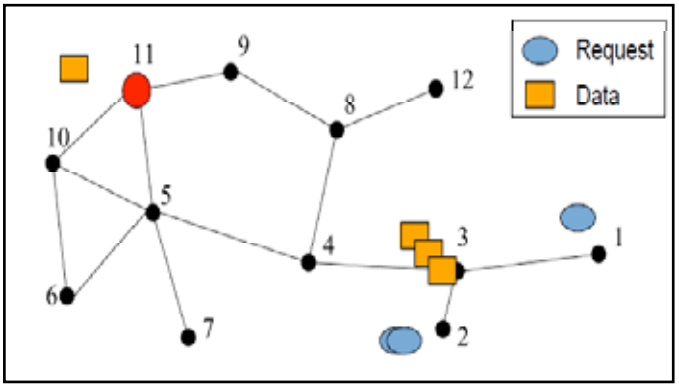

Fig. 2: Passing-by Data are Cached to Serve Future Requests

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.

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 ßooding

This approach limits the propagation range of a request by forcing a time to live (T T L) 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

This approach extends mitigated flooding by steering queries toward areas of the network where the required information is estimated to be denser.

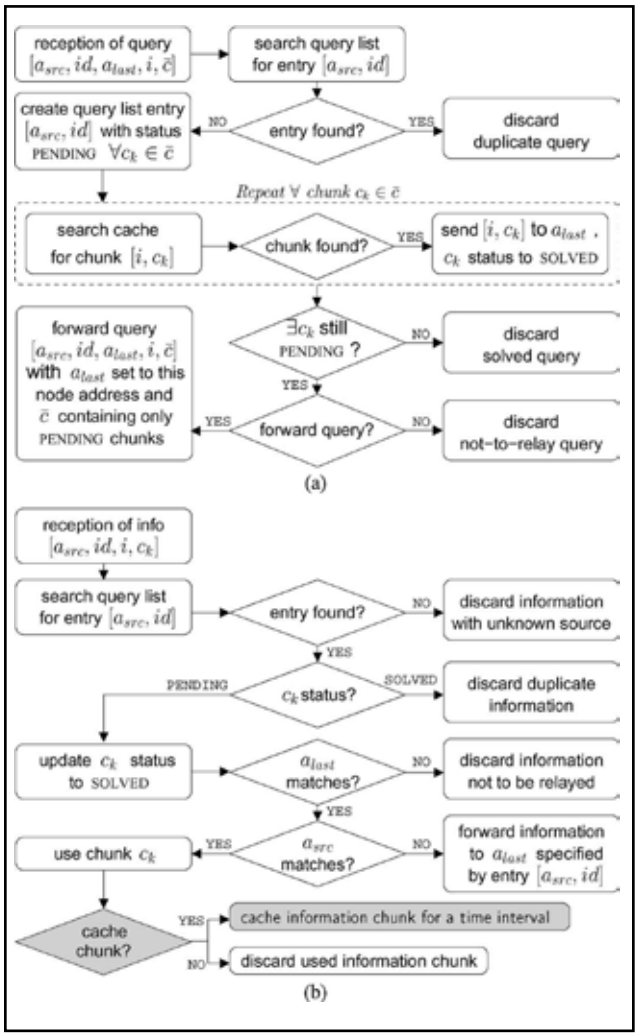

Fig. 3: Flowcharts of the Processing of (a) Query and (b) Information Messages at User Nodes. We denote the address of the node that generated the query as asrc, the query identifier as id, the address of the last node that forwarded the query message as alast, and the set of queried chunks as c. The functional blocks that are the focus of this paper are highlighted in (b).

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 [24] 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. 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 information presence upon the notification/detection of an updated version to ease the diffusion of the new information.

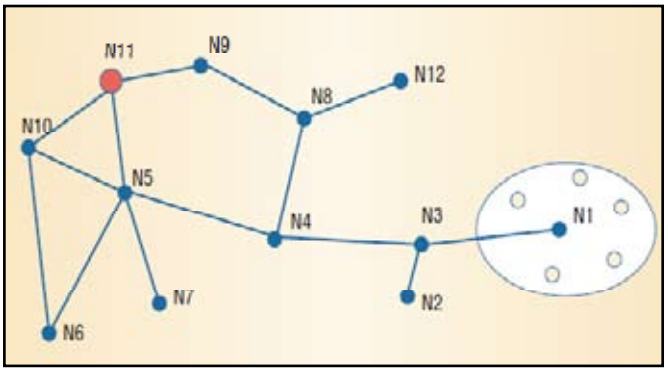

Fig. 4: Ad-Hoc Network. Node N11 is a Data Source and the Blue Nodes are Router Nodes. Node N1 is a Cluster Head Surrounded by Mobile Nodes

IV. Our 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 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.

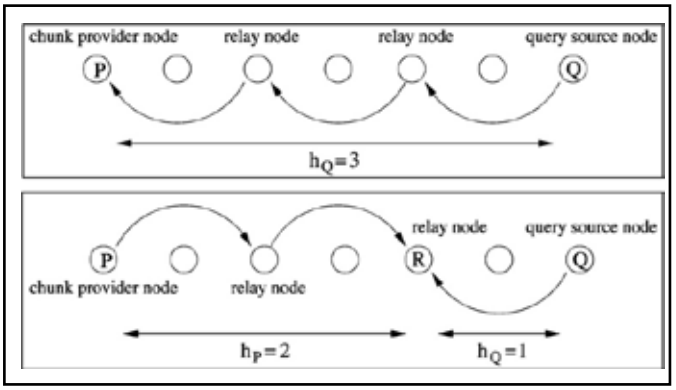

Fig. 5. Q and P Denote, Respectively, a Node that Issues a Query and a Node that Provides the Requested Content. Node R in the Lower Plot is a Relay Node, Overhearing the Exchanged Messages. The Upper and Lower Plots, Respectively, Represent the Case 1 h_Q Value for the Provider Node P and the Case 2 h_Q and h_P Values for the Relay Node R with Respect to the Query Source Q and the Provider P.\

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 n 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 received (or overheard) by n, during step j. Node n thus updates this quantity if it receives (or overhears) an information message, e.g., node R in the lower plot of fig. 2; thus, the transit counter is the only data structure that needs cross-layer access, i.e., the number of information copies whose transit the node has overheard at lower layers (and subsequently inspected).

The provider and transit counters are updated through the hop count information that is included in the query and information message header. The exact procedure is as detailed follows.

If node n generates a reply information message that contains chunks of information item i, as an answer to a query for some chunk c that it owns, then a new copy of such chunks is possibly cached at the node that generated the query. Node n must therefore account for the presence of such a new copy at a distance h_Q, which is equal to the number of hops that were covered by the query (see the upper plot of fig. 2). The provider counter is updated by a quantity that is inversely proportional to the distance h_Q as follows:

$$d_{ic}(n,j) = d_{ic}(n,j) + \frac{1}{h_Q}. \quad (1)$$

Note that the larger the h_Q, i.e., the farthest the new chunk copy, the lesser the added contribution.

If node n receives or overhears a new transit information i message, which contains a chunk c whose query status was pending, it must then account for the presence of the

following copies: 1) a new copy of the chunk that will be cached by a node at a distance of h_Q hops and 2) an existing copy that is cached at a distance of hP hops (see fig. 2, lower plot). Thus, following the approach in (1), the transit counter is updated as follows:

$$r_{ic}(n,j) = r_{ic}(n,j) + \frac{1}{h_P} + \frac{1}{h_Q}. \quad (2)$$

Again, the larger h_P is, i.e., the farthest the information provider is, the lesser the contribution of the existing copy becomes.

The last case accounts for the reception or overhearing of an information message whose contribution must not (or cannot) be

related to a corresponding query. This condition may happen for the following two reasons:
The corresponding query was already solved (hence, the message is considered duplicated information), or node n had not received the corresponding query (e.g., node n just moved within the radio range of nodes in the return path of the information message and missed the query). In either case, only the contribution due to the presence of the copy at the provider is considered, hence

$$r_{ic}(n,j) = r_{ic}(n,j) + \frac{1}{h_P}. \quad (3)$$

Based on the aforementioned quantities, node n can compute a presence index of chunk c of information i, as observed during step j within node n's reach range. We refer to such a value as pic (n, j) and define it as

$$p_{ic}(n,j) = \min\{1, d_{ic}(n,j) + r_{ic}(n,j)\}. \quad (4)$$

According to (4), pic (n, j) comprises the range [0, 1]. A zero-value means that the presence of chunk c of information i was not sensed by n during time step j. Instead, if the chunk is cached one hop away from n, pic (n, j) is equal to one; this case is the "best," where the chunk would directly be available to n if needed. Intermediate values between 0 and 1 are recorded when n observes chunks that are cached more than one hop away.

B. Large-Sized Caches Computation of the Content Drop Time

We first consider the case in which nodes have a large-sized cache, enough to potentially store a large portion (i.e., 50% or more) of the content that they request. As aforementioned, consuming all the storage resources of a node to cache the retrieved data is not desirable, because the same memory may be shared by different services and applications that run at nodes. Thus, here, we exploit the aforementioned information presence estimate to determine a cache drop time after which the retrieved information items are removed from the memory: the goal is to reduce the cache utilization without affecting the performance of the content distribution system.
We denote by $\chi_i(n, j)$ the cache drop time that node n computes at the end of time step j for information item i. Such a drop time applies to all chunks, belonging to information item i, that will be received during time step (j + 1). To compute χ_i (n, j), node n estimates an overall probability of information presence, by composing the presence indices pic (n, j) of all chunks of information i, as follows.

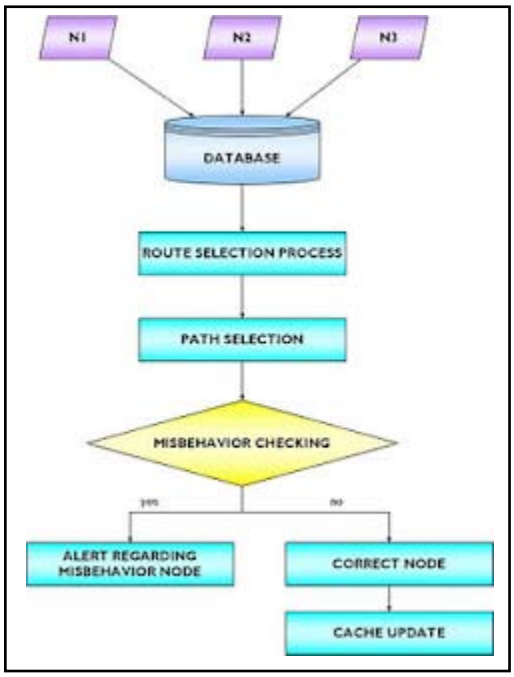

Fig. 6. System Architecture

$$p_i(n,j) = \min\left\{1, \sum_{k=j-\tau}^{j} \phi_i(n,k,j)\right\}. \quad (7)$$

Note that, in (7), τ is the filter memory and contributions that are older than τ time steps are ignored. Thus, to compute (7), a node has to store at most 2τ + 1 values for each information item, i.e., the sum in (6) and χ_i(n, k) for τ + 1 and τ different time steps, respectively. As specified in the performance evaluation, τ can be set to a very small value so that the presence index computation requires minimal memory usage.
Finally, by denoting the maximum cache drop time by MC, the caching time for the chunks that belong to information i is obtained as

$$w_i(n,k,j) = \begin{cases} 1, & \text{if } j-k \le \Delta(n,k) \\ \alpha^{j-k-\Delta(n,k)}, & \text{otherwise} \end{cases}$$
$$\text{with} \quad \Delta(n,k) = \lfloor f\chi_i(n,k-1) - \log_\alpha W \rfloor. \quad (5)$$

According to (8), in the extreme situation where the entire information i is estimated to be cached within node n's reach range, i.e., $p_i(n, j) = 1$, the retrieved content will not be stored by the node; on the contrary, when node n observes a complete lack of content i within its reach range, i.e., $p_i(n, j) = 0$, the caching time will be equal to MC .

C. Small-Sized Caches: Content Replacement

When equipped with a small-sized cache, nodes cannot store all content that they request but are forced to choose which items to keep and which items to discard every time newly retrieved data fill up their memory. In this case, computing cache drop times is clearly not a solution, because the lingering of items in cache is primarily determined by the rate of reception of new content. Therefore, in the presence of limited dedicated storage resources, we exploit the information presence estimate to define a content replacement policy that favors a balanced distribution of data over the network so that all content is as "close" as possible to a requesting node.
The rationale of our content replacement strategy is very similar to the approach employed for the cache drop time computation. Again, we start by identifying the amount of time for which the

index p_{ic} (n, j) must be considered valid and define a new smoothing factor w^i (n, k, j) to that end as

$$\hat{w}_i(n,k,j) = \begin{cases} 1, & \text{if } j-k \le \Gamma(n,k) \\ 0, & \text{otherwise} \end{cases}$$

$$\text{with} \quad \Gamma(n,k) = \lfloor f\hat{\chi}_i(n,k-1) \rfloor . \qquad (9)$$

This case means that, at time step j, contribution p_{ic} (n, k) maintains its value unchanged if no more than Δ(n, k) time steps have passed, because the presence index was recorded. Otherwise, the filter forces an exponential decrease of the index value. In (5), the exponential decrease factor is denoted by α, where as Δ(n, k) is such that, after a time χ_i (n, k − 1), the smoothed presence index retains a fraction W of its original value. 2 In addition, w_i (n, k, j) depends only on the information identifier i, i.e., it has the same value for all chunks c that belong to information i, because the caching time is determined on a per-item basis. For clarity, examples of filter impulse responses for different values of χ_i(n, k − 1) are shown in fig. 3.

Next, let us consider time step j and the tagged node n. Node n estimates how complete a single information item i is in its surroundings, due to the contributions measured during a given step k, by summing up presence indices pic (n, k) that refer to all chunks c of i and smoothing them by the factor wi (n, k, j) to account for the likelihood that some content has been dropped. We have

$$\phi_i(n,k,j) = w_i(n,k,j)\frac{1}{C}\sum_{c=1}^{C} p_{ic}(n,k). \qquad (6)$$

In other words, (6) reflects the degree of completeness of information i that node n can expect at time step j, only considering contributions that were collected during time step k. We stress that 0 ≤ φi(n, k, j) ≤ 1.

The overall presence index for information item i, as shown by node n at time step j, can be computed by summing up the whose details will be discussed at the end of this section, for clarity.

We can thereafter define the completeness of item i, estimated by node n from samples observed at time step k, as

$$\tilde{\phi}_i(n,k,j) = \tilde{w}_i(n,k,j)\frac{1}{C}\sum_{c=1}^{C} p_{ic}(n,k) \qquad (10)$$

And the overall presence index as

$$\hat{p}_i(n,j) = \sum_{k=j-\tau}^{j} \tilde{\phi}_i(n,k,j). \qquad (11)$$

As aforementioned, we define a maximum cache permanence time M_c, after which, a chunk is discarded to avoid stored information from becoming stale and node movement from leading to inconsistencies with respect to previous information presence ratings. The estimated caching time is then computed as

$$\hat{\chi}_i(n,k) = \left(1 - \frac{\hat{p}_i(n,k)}{\max_i\{\hat{p}_i(n,k)\}}\right) M_C. \qquad (12)$$

To provide an interpretation for (12), consider the case of chunks of the most common information item in the area. Because, as aforementioned, a node discards a chunk of in-formation i associated with the highest p^ (n, j), the estimated caching time for such a chunk is set to 0 in (12), and caching times of much less popular chunks are, instead, estimated to be much longer (up to M_c).

As the estimated presence decreases, the chance that chunks find space in the cache of requesting nodes grows, reaching the maximum estimated caching time M_c if the information is completely absent from the area, i.e., when p^ (n, j) = 0. Note that (12) does not depend on the content query rate as a precise design choice. Indeed, it is not likely that a node

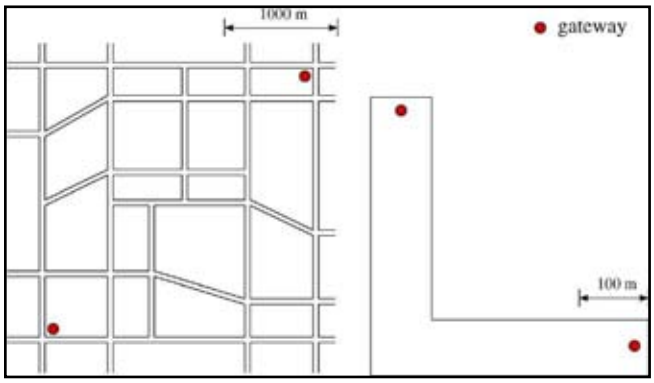

Fig. 7: Simulations: City (left) and Mall (Right)

V. Conclusions

Designed and implemented cooperative cache in wireless P2P networks. Designed asymmetric caching approach to reduce the cache layer overhead. 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 where each node, upon receiving a requested information, determines the cache drop time of the information or which con-tent 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 memory-constrained 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),"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, 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, M. Atiquzzaman,"A sur-vey of data replication techniques for mobile ad hoc network databases", VLDB J., Vol. 17, No. 5, pp. 1143–1164, Aug. 2008.

[4] A. Derhab, 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, 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, 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, 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, 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, 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, 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, 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, S. Shenker,"Replication strategies in unstructured peer-to-peer networks", in Proc. ACM SIGCOMM, Aug. 2002, pp. 177–190.
[14] B. Tang, H. Gupta, 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, 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.
[16] M. K. Denko, J. Tian,"Cross-layer design for cooperative caching in mobile ad hoc networks", in Proc. IEEE CCNC, Las Vegas, NV, Jan. 2008, pp. 375–380.
[17] H. Chen, Y. Xiao, X. Shen,"Update-based cache replacement policies in wireless data access", in Proc. BroadNets, Boston, MA, Oct. 2005, pp. 797–804.
[18] J. Xu, Q. Hu, W.-C. Lee, D. L. Lee,"Performance evaluation of an optimal cache replacement policy for wireless data dissemination", IEEE Trans. Knowl. Data Eng., Vol. 16, No. 1, pp. 125–139, Jan. 2004.

B.Ganga Bhavani received her B.Tech from J.N.T.U and M.TECH in CSE from JNTU.Her areas of interests include Network programming,Information Security,Computer Networks, Neural networks.. She has 5 years of experience in teaching to engineering students. He is the member of C.S.I, (India) and ISTE. She is now the Assistant Professor, Department of C.S.E at BVC Engineering College,Odalarevu. She is heading Special Interest Research Groups in Networking. She has published many technical papers both in international and national Conferences.

M Vishnu Murthy received her B.Tech degree from Bonam Venkata Chalamayya engineering college, JNTUK, Andhrapradesh, India and the pursuing M.Tech degree from Bonam Venkata Chalamayya engineering college, JNTUK, Andhra Pradesh, India.