Content Management in a Mobile Ad Hoc Network: Beyond Opportunistic Strategy

Suk-Bok Lee* Starsky H. Y. Wong† Kang-Won Lee† Songwu Lu*

*Computer Science Department
UCLA, CA 90095

†IBM T. J. Watson Research Center
Hawthorne, NY 10532

Abstract—**We study the challenging problem of strategic content placement in a dynamic MANET. Existing content placement techniques cannot cope with such network dynamics since they are designed for fixed networks. Opportunistic caching approaches are insufficient as they do not actively manage contents for certain goals. In this paper, we present a novel content management approach called LACMA, which leverages the location information available to mobile devices via GPS. The main idea of LACMA is to bind data to geographic location (as opposed to network nodes). This location-based strategy decouples the content placement problem from the changing network topology, and allows us to design an optimization framework even in a dynamic MANET environment. We present key components of LACMA used for strategic content placement and content-location binding (through proactive content push). We evaluate LACMA and compare its performance with existing caching schemes and show that LACMA considerably outperforms existing schemes over a wide range of scenarios.**

I. Introduction

Content-centric networking (CCN) [6] is a new networking paradigm based on *named data* instead of *named hosts*. This new way of thinking provides a nice abstraction for data access and content distribution. It is also expected to provide a more efficient means of communication: when a network user requests a content, any node with a copy can act as a server and respond to the request, thereby reducing the access latency, network congestion, and bandwidth consumption. For CCN to deliver performance benefits, however, it is important to carefully replicate popular contents and strategically place them in vicinity of requesting nodes. In this paper, we consider the content management problem in a mobile ad hoc network (MANET), whose network makes the problem very challenging, although at the same time, the effect of good content placement will be most pronounced in MANETs because of tenuous throughput when contents are accessed over multiple hops.

In the literature, content placement and replication strategies have been primarily studied in the context of fixed networks [1], [3], [7], [11]. Needless to say, they cannot constitute a good content management solution for our problem due to their inability to cope with network dynamics, i.e., content placement must be recomputed every time the network topology changes due to node mobility and wireless channel variations. The opportunistic caching strategies that have been proposed in the literature do not try to optimize content placement based on explicit objectives [4], [14]. Hence, they do not provide sufficient performance gain in our problem, which we demonstrate in this paper.

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions are those of the authors and do not represent the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding copyright notation.

In this paper we propose LACMA, a novel *L*ocation-*A*ided *C*ontent *M*anagement *A*rchitecture for a content-centric MANET. The design of LACMA is based on the observation that MANETs usually form a network in a close proximity where user/node population is relatively dense. To deal with the dynamic topology, LACMA binds data to a geographic location (as opposed to network nodes based on topology). In other words, LACMA tries to keep a content copy within a specified geographic boundary based on the location information made available by GPS and by proactively replicating the content as necessary. This location-based strategy decouples the content placement problem from changing network topology, and thus significantly simplifies the problem. As a result it allows us to design an optimization framework that will work even in a highly dynamic environment as MANETs.

The main contributions of this paper are as follows:

- We formulate the location-based placement problem that takes content popularity into account, and develop an efficient algorithm via dynamic programming. The main idea is to define multi-level virtual grids and allocate popular contents to fine-grained grids so that they are more likely available from nearby.
- We design a practical content management scheme that tries to preserve the location-binding placement property in the presence of node mobility without excessive content hand-off overhead. This is achieved by opportunistic information gathering and probabilistic replication.
- We report extensive performance evaluation results of LACMA in comparison with state-of-the-art opportunistic schemes. They show that LACMA significantly outperforms the existing schemes, e.g., by reducing hop counts by up to 45% and query traffic by 54%.

II. Preliminaries

We consider a *content-centric* MANET, where a set of mobile users are willing to share popular contents in a cooperative manner. The content retrieval in such a network will be take some steps similar to the following:

1) A user generates and broadcasts a query for a content c_i (where i is a unique identifier).
2) A node receiving the query sends back c_i to the requesting node if it has a copy in its cache. This is done by unicast reversing the path in the request. Otherwise, it rebroadcasts the query.
3) Upon receiving the response, the requester then caches a copy of c_i, and it can act as a server of c_i for future requests.

Traditional content placement problem. The traditional replica placement problem in a network $G(N, E)$ with m contents, $c_1, \cdots, c_m$, is to find a placement Ψ on the graph (while each node x can store replica up to its capacity w_x) to minimize the total access cost of client queries in the network:

$$\Gamma(G, \Psi) = \sum_{x \in N} \sum_{i=1}^{m} f_{x,c_i} \times d(x, c_i), \tag{1}$$

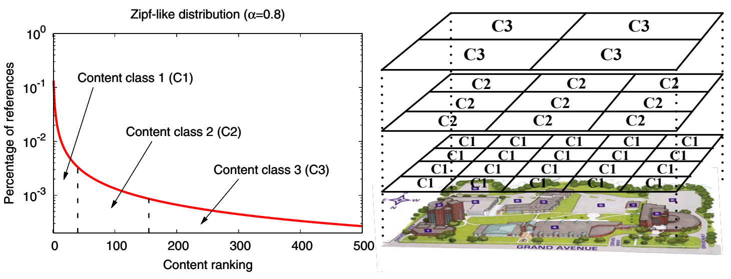
where f_{x,c_i} is access frequency of content c_i by node x, i.e., reflecting content popularity, and $d(x, c_i)$ is the distance (e.g., hop count along the shortest path) between x and the closest node who has a copy of c_i. This problem is known to be NP-hard even for a fixed network [1].

III. LACMA Overview

The main insight for our design is that, although content management on a time-varying graph is intractable, content management based on a fixed geographic location is feasible. LACMA decouples the node mobility and the corresponding topology dynamics, thus streamlining the content placement and its maintenance procedure.

Consider a physical area (e.g., university quad) where a MANET is deployed. For simplicity, we assume the shared data contents follow a Zipf-like popularity distribution (see Fig. 1(a)), which has been observed in various studies for library books, video rental, and Web objects [2]. To achieve good data access performance, intuitively, we need to *place* copies of popular data contents more densely in the physical area occupied by the MANET. To ensure this condition is met LACMA takes the following steps. First, LACMA constructs multiple layers of virtual grid with same grid size in a layer, but different grid sizes across layers (Fig. 1(b)). Then we *bind* the most popular contents to the bottom-most grid layer, which is the densest (content class C_1 in Fig. 1(a) and 1(b)), then *bind* the next most popular contents to the next grid layer (content class C_2 in Fig. 1(a) and 1(b)), and so on. In this way more popular contents will be more densely replicated and will likely be more close to potential requestors. We note that this concept of content-location binding is a *logical* one. In other words, when we say a content is bound to a grid cell, we mean at least one node in that cell contains the copy in its local storage. When we say a content x is bound to a grid layer C_l, we mean that x is bound to all cells in C_l.

In order to make this approach work properly, we need to address the following issues:



(a) Content popularity distribution. (b) Placement of content class 1, 2, 3

Fig. 1. Illustration of the location-binding content placement. In this example, there are three layers of grids and corresponding three content classes.

- How to optimally construct the layers of grids (e.g. determining the number of layers and their corresponding cell size) and bind which contents to which layers?
- How to maintain each cell's location-binding contents in the present of node mobility?

IV. LACMA Design

A. *Location-Binding Content Placement*

The placement goal of our location-based design is to minimize the expected access cost to reach contents in the network. Hence, we first translate the traditional replica placement objective of Eq. (1) in the location-based framework.

Location-binding placement objective. In order to eliminate the dependency of content placement on the actual network topology, we aim to limit $d(x, c_i)$, the distance from any node x to any content c_i that belongs to the same content class l. The basic idea is to place contents with similar popularity into a cell of the same layer l, rather than trying to optimize $d(x, c_i)$ directly. This will ensure:

$$\sqrt{2} \cdot S_l \geq d(x, c_i), \qquad \forall c_i \in C_l \tag{2}$$

where C_l represents a content class l, and S_l is the width of a square cell in layer l. Note that the l.h.s in Eq. (2) is the maximum possible distance within a cell (i.e., diagonal length). In this way, we approximate the algorithms that have been designed to directly optimize the total data access cost represented as Eq. (1).

Content popularity. We assume that the content access rate follows Zipf-like distribution, generally considered as representative of content popularity. In a Zipf-like distribution, the access probability of the $i^{th}(1 \leq i \leq m)$ content item is represented as follows:

$$P(i; \alpha, m) = \frac{1/i^{\alpha}}{\sum_{z=1}^{m}(1/z^{\alpha})} \tag{3}$$

where α $(0 \leq \alpha \leq 1)$ is the value of the exponent characterizing the distribution. When $\alpha = 1$, it follows the strict Zipf distribution; when $\alpha = 0$, it follows the uniform distribution.

Problem formulation. We now formulate the location-binding content placement problem as follows:

- Given the (i) content popularity distribution $P(i; \alpha, m)$, (ii) node count, (iii) cache space per node, and (iv) the area size, minimize the expected data access cost *using as* *few layers of grids as possible*, and determine the content classes and their corresponding grid cell size.

We model it as a dynamic programming problem. Without loss of generality, we assume that content c_i is the i^{th} popular item in a set of contents C. We use δ_l to denote the proportion of the storage space that a node can use up for content class l, such that $\sum_l \delta_l = 1$. We let δ^* be the smallest assignable proportion, and we set $\delta^* = 0.01$ in our evaluation. Given a content class C_l represented by indices (a, b) where c_a and c_b denote the most popular and the least popular contents in that class respectively, we can estimate $\psi(a, b, \delta_l)$, the average access cost to reach the contents that belong to class C_l:

$$\psi(a, b, \delta_l) = \tilde{d} \cdot S_{(a,b,\delta_l)} \times \int_a^b P(x; \alpha, m)\, dx \tag{4}$$

where $\tilde{d}$ is the mean distance between two random points from the interior of a unit square, and $S_{(a,b,\delta_l)}$ is the (smallest possible) width of a square cell (by which the whole area is divisible) that can accommodate all contents of class l such that the following condition holds:

$$\frac{n}{B^2} \cdot [S_{(a,b,\delta_l)}]^2 \cdot w \cdot \delta_l \geq \sum_{i=a}^{b} size(c_i) \tag{5}$$

where n is the total number of nodes, B is the width of the whole square area, and w represents the average storage capacity of each node. The above condition prevents the over-assignment of contents beyond the aggregate storage capacity assigned for content class C_l within a cell.

We note that the number of layers reflects a tradeoff between content class granularity and maintenance cost. To balance this trade-off, we introduce a parameter K that represents the penalty for adding new layer. By tuning K, we can control the use of additional layers to a greater or lesser extent.

The total cost of the content placement is now defined as a sum of the following terms: (i) the number of layers of grids (i.e., number of content classes) times the multiplier $K > 0$; (ii) for each layer of a grid, the average access cost to reach contents of the class (according to Eq. (4)).

Dynamic programming recurrence. We let $\Phi(j, \delta)$ denote the optimal cost by our location-based placement for contents $c_1, ..., c_j$ with using δ available storage proportion ($\Phi(0, 0) = 0$ as a boundary case). Then it is possible to solve the placement problem by using the following dynamic programming to compute $\Phi(j, \delta)$ resursively:

$$\Phi(1, \delta) = \tilde{d} \cdot S_{(1,1,\delta)} \times P(1; \alpha, m) \tag{6}$$

$$\Phi(j, \delta^*) = \tilde{d} \cdot S_{(1,j,\delta^*)} \times \int_1^j P(x; \alpha, m)\, dx \tag{7}$$

$$\Phi(j, \delta) = \min_{\substack{1 \leq i \leq j \\ 0 \leq \hat{\delta} \leq \delta}} (\psi(i, j, \hat{\delta}) + K + \Phi(i-1, \delta - \hat{\delta})) \tag{8}$$

Eq. (6) states that if there is only one content, it results in a single layer with its cell size mainly determined by δ. Eq. (7) denotes that when $\delta = \delta^*$, all the unassigned contents go to a single layer, whose cell size depends on the aggregate of those contents. The recursive Eq. (8) signifies that an optimal solution $\Phi(j, \delta)$ can be obtained by identifying the last content class $C_{(i,j)}$ with $\hat{\delta}$ storage proportion – plus an additive penalty K for this layer – together with optimal solution $\Phi(i-1, \delta-\hat{\delta})$ for the remaining contents and storage space.

Placement algorithm. Using the above dynamic programming, we first build up the solutions $\Phi(j, \delta)$ for all j and δ by filling them in the array entry $M[j, \delta]$. We then use the values stored in the array M for the optimal location-binding placement. The pseudo-code of the resulting placement algorithm, the complexity analysis of the algorithm ($O(m^2/\delta^*)$), and the detailed behavior of the algorithm are presented in the full version of the paper [8].

B. Location-Based Content Maintenance

To maintain the location-binding placement in the face of mobility, nodes need to determine whether to *push the content or not* when leaving the current cell. The goal is to keep each cell containing at least one copy of its binding contents with high probability, while minimizing the cost of push operations.

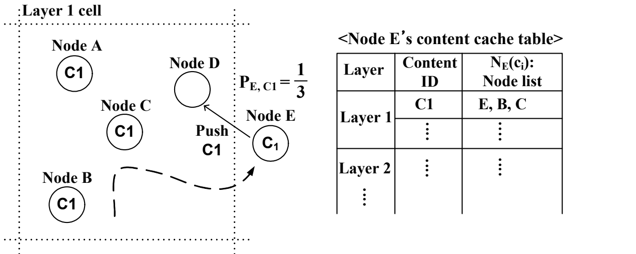
Probabilistic push operation. Probabilistic push exploits the local information collected by each node to contribute to the joint maintenance effort while reducing the unnecessary push operations as much as possible. The idea is that if node x estimates the current number of copies of content c_i in the cell, it only needs to push c_i with probability of inverse proportion to the estimated number, thus collectively achieving the cell maintenance efficiency and sharing the "push responsibility" with others who also hold c_i in the cell.

We denote with p_{x,c_i} the *push responsibility* of node x for content c_i such that x uses it ($0 < p_{x,c_i} \leq 1$) as the probability to push c_i when leaving the cell. As it is infeasible to achieve perfect synchronization about replica information among all members in the cell, we pursue the best-effort coordination from each individual node x's local point of view:

$$\sum_{x \in N_x(c_i)} p_{x,c_i} = 1 \quad \Rightarrow \quad p_{x,c_i} = 1/|N_x(c_i)| \tag{9}$$

where $N(c_i)$ is a set of nodes who store c_i in the cell, and $N_x(c_i)$ is a subset of $N(c_i)$ that x is aware of.

To make the local view $N_x(c_i)$ as close to the real $N(c_i)$ as possible, we employ two techniques: (1) query listening and (2) cache-list exchange with nodes encountered in the cell. Such information is processed/discarded by first checking the cell ID that is included in the message header.



Layer	Content ID	$N_E(c_i)$: Node list
Layer 1	C1	E, B, C
	⋮	⋮
Layer 2	⋮	⋮
⋮		

Fig. 2. Example of probabilistic push operation and its cache table. Only a single content c_1 is shown for ease of exposition.

The mean distance inside a square can be obtained by multiplying its width and the mean interior distance of a unit square $\tilde{d} = 0.5214$ [12].

1) Receiving a query request for c_i generated by node y in the same cell, each node x who holds c_i updates its view, i.e., $N_x(c_i) = \{y\} \cup N_x(c_i)$, since a new copy of c_i is going to be cached at y – responded by x or others. Note that every node is able to listen on any query messages originated inside the cell.
2) Every node y broadcasts the list of content IDs in the cache when encountering a new node in the cell or periodically, e.g., along with the HELLO neighbor discovery message [10]. Receiving node x adds y into $N_x(c_i)$ if y has c_i. Maintaining $N_x(c_i)$ prevents double counting of nodes encountered multiple times.

For both cases, node x also records the time that y is added/updated in $N_x(c_i)$, which is used to clear y from $N_x(c_i)$ after a validity duration T_v, thus invalidating the potentially outdated information. The choice of T_v offers a design tradeoff; a larger T_v would lead to a smaller number of push operations at the risk of using obsolete information, and the opposite would be true for small T_v. The issue of selecting proper T_v with respect the cell size and node mobility is presented in the full version of the paper [8].

When leaving the cell, node x pushes c_i with probability $p_{x,c_i} = 1/|N_x(c_i)|$ to the remaining neighbors if none of them has c_i (see Figure 2). Once completing the push operation and entering a new cell of layer l, node x clears all $N_x(c_i)$ associated with layer l. Note that $N_x(c_i)$ always includes node x itself, thus making $p_{x,c_i} = 1$ when x does not observe any nodes with c_i in the cell.

V. Performance Evaluation

We evaluate the performance of LACMA through the network simulator *ns-2*. Our focus is to measure the benefits of LACMA and its associated cost, by comparing with other representative approaches over a wide range of scenarios.

Simulation setting. We simulate various network scenarios based on the following parameters: (i) the number of nodes in the network, (ii) mobile node speed, (iii) memory capacity on each node, (iv) content popularity distribution, (v) mean query generation time. For each test set we vary one of the key parameters while keeping the others at their default values. Table I summarizes a list of the default simulation parameters and their ranges. The results are averaged over 10 runs, each with 1-hour simulation time.

TABLE I
Simulation parameters

Parameter	Default value	Range
Area size	2.5×2.5 km^2	
Radio range	250 m	
Num. of contents m	1000 items	
Number of nodes n	300	200 – 400
Node speed v_{max}	2 m/s	2 – 20
Pause time (sec)	180	
Cache size	30 slots	5 – 30
Mean query interval	30 sec	10 – 60
Zipf parameter α	0.8	0 – 1
TTL (hops)	10	

Benchmark algorithms. We compare LACMA with three other reference algorithms, HybridCache [14], Hamlet [4], and CGA [13]. HybridCache is a state-of-the-art opportunistic caching technique, that allows nodes on the data path to cache the relaying item if its size is small, otherwise nodes just cache the data path. Hamlet is another advanced caching technique, whose goal is to save storage space while achieving content diversity via estimating the cached items in the neighborhood. CGA is a 2-approximation centralized offline replica placement algorithm (the current best result) for non-mobile networks. Due to the high computation complexity of CGA, we compare with CGA's performance in a relatively small setting, and present the results separately.

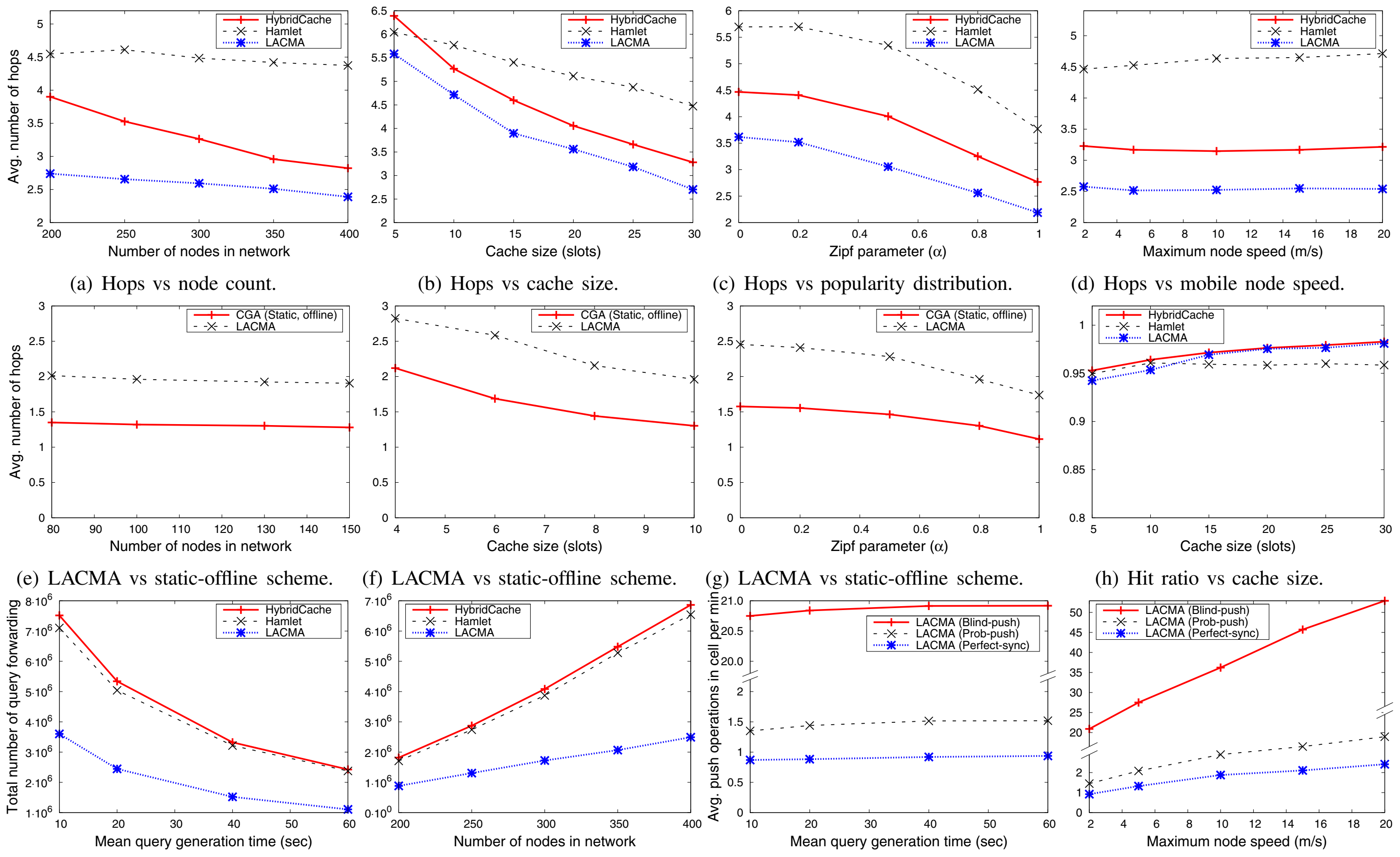
Simulation results. Fig. 3(a) plots the average hop count resulting from the three different schemes. We see that LACMA offers nearly 20–35% decrease in hop count compared to HybridCache; 40–45% over Hamlet. We stress that such hop count reduction (e.g., $4 \rightarrow 3$ hops) is, in fact, quite significant in MANETs because many studies [5], [9] have shown that the achievable throughput drops *rapidly* with the hop count of the transmission in multi-hop wireless networks. We also observe that LACMA performs around one hop worse than CGA (see Fig. 3(e), 3(f), and 3(g)). Our intention here is to give an insight of how close LACMA is to the absolute lower bound that is obtained via an offline algorithm.

In Fig. 3(b), we vary the cache size of each node from 5 to 30 slots that correspond to 0.5% and 3% of the total number of contents, respectively. We see that LACMA consistently outperforms the other two schemes, even when the cache size is small. This result suggests that the storage space partition in LACMA efficiently uses the cache memory.

Fig. 3(c) shows more interesting results, with varying the Zip parameter α from 0 to 1. We see that LACMA performs the best not only when α is large, but also when $\alpha = 0$ (uniform popularity). This can be explained by the fact that LACMA provides bounds on the distance to contents while the other two schemes solely operate on the query pattern.

Fig. 3(d) presents the results under different node speed. The reasons for these schemes to have mobility-insensitive performance are quite different. HybridCache and Hamlet are query driven so that the distance (from a node to any content) is not much affected by who are current neighbor nodes, in the sense that each node issues its requests independently of one another. On the other hand, LACMA provides the distance bound (i.e., cell-level placement) against node mobility.

We next evaluate the amount of query traffic generated from all schemes. Fig. 3(i) and 3(j) plot the simulation results of the total number of query messages propagated through the network, with varying the query generation time and the number of nodes respectively. We see that LACMA significantly reduces the query search traffic by 50–55% over the other two. The reason is that nodes can reach the requested contents within their cells most cases instead of further searching beyond the cell. Therefore, the query messages travel less number of hops compared to the other two schemes that blindly searches throughout the network. We note that all three schemes have the similar query hit-ratio (see Fig. 3(h)) that is



(a) Hops vs node count. (b) Hops vs cache size. (c) Hops vs popularity distribution. (d) Hops vs mobile node speed.

(e) LACMA vs static-offline scheme. (f) LACMA vs static-offline scheme. (g) LACMA vs static-offline scheme. (h) Hit ratio vs cache size.

(i) Query traffic vs query interval. (j) Query traffic vs node count. (k) Push cost vs query interval. (l) Push cost vs mobile node speed.

Fig. 3. Simulation results for different scenarios. The default simualtion parameters are listed in Table I.

defined as the ratio of successful search within TTL range.

We now turn our attention to the maintenance cost that LACMA pays for the above benefits. Fig. 3(k) and 3(l) plot the results of the average number of push operations performed in a cell per unit time, varying the mean query generation time and the node speed respectively. We make two observations. First, Prob-push significantly reduces the unnecessary push operations compared to Blind-push (nearly a factor of 15), and its performance is quite close to Perfect-sync. Second, for all three cases the push cost increases with the node speed, but the slopes of Prob-push and Perfect-sync are much lower than that of Blind-push. Prob-push still shows the close performance to Perfect-sync as the node speed changes.

VI. Conclusion

Sophisticated content management in a highly dynamic MANET environment has been considered unattainable so far. In this paper, we provided a new insight based on location-based content binding, proposed a novel content management mechanism, called LACMA, and demonstrated that careful content management is not only feasible but also beneficial in a MANET. We presented a detailed design for key components of LACMA. Through extensive simulation study, we have shown that LACMA outperforms the current best opportunistic caching approaches in all aspects by reducing number of hop counts and query traffic, without compromising the hit ratio or incurring excessive overhead.

References

[1] I. Baev and R. Rajaraman. Approximation algorithms for data placement in arbitrary networks. *SODA*, 2001.
[2] L. Breslau and P. Cao et al. Web caching and Zipf-like distributions: evidence and implications. *IEEE INFOCOM*, 1999.
[3] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. *ACM SIGCOMM*, 2002.
[4] M. Fiore, F. Mininni, C. Casetti, and C. -F. Chiasserini. To cache or not to cache?. *IEEE INFOCOM*, 2009.
[5] Z. Fu and P. Zerfos et al. The impact of multihop wireless channel on TCP throughput and loss. *IEEE INFOCOM*, 2003.
[6] V. Jacobson and D. K. Smetters et al. Networking named content. *ACM CoNEXT*, 2009.
[7] B. -J. Ko and D. Rubenstein. Distributed, self-stablizing placement of replicated resources in emerging networks. *IEEE ICNP*, 2003.
[8] S. -B. Lee, S. H. Y. Wong, K. -W. Lee and S. Lu. Content management in a mobile ad hoc network: beyond opportunistic strategy. *UCLA TR-110001*, 2011.
[9] J. Li, C. Blake, D. Couto, H. Lee, and R. Morris. Capacity of ad hoc wireless networks. *ACM MOBICOM*, 2001.
[10] C. E. Perkins, E. M. Belding-Royer, and S. Das. Ad hoc on demand distance vector (AODV) routing protocol. IETF Internet Draft, Jan 2002.
[11] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. *IEEE INFOCOM*, 2001.
[12] L. A. Santalo. Integral geometry and geometric probability. Addison-Wesley, 1976.
[13] B. Tang, H. Gupta, and S. Das. Benefit-based data caching in ad hoc networks. *IEEE INFOCOM*, 2006.
[14] L. Yin and G. Cao. Supporting cooperative caching in ad hoc networks. *IEEE INFOCOM*, 2004.