AdTorrent: Delivering Location Cognizant Advertisements to Car Networks

Alok Nandan, Saurabh Tewari, Shirshanka Das, Mario Gerla, Leonard Kleinrock

Computer Science Department
University of California Los Angeles
Los Angeles, CA 90095-1596
{alok, stewari, shanky, gerla,lk}@cs.ucla.edu

***Abstract*— AdTorrent is an integrated system for search, ranking and content delivery in car networks. AdTorrent builds on the notion of Digital Billboards, a scalable "push" model architecture for ad content delivery. We present a detailed analysis of the performance impact of key design parameters such as scope of the query flooding on the query hit ratio. Our mobility model for the urban, vehicular scenario can be used in conjunction with the analytical model for estimating query hit ratio by a system designer to determine the scope of the query flooding as a function of the available storage per vehicle for their application.**

I. Introduction

One of the most important sources of revenue for big Internet-based companies are advertisements. With vehicular networks poised to become part of the Internet, this new "edge" of the Internet represents the next frontier that advertising companies will be striving to reach. As advertisers struggle to reach increasingly distracted and jaded American consumers, they have sought nontraditional media for their advertisements (Ads), from elevators to cell phone screens.

Content-targeted advertising paradigm has proved to be a resounding success in advertising on the conventional Internet. As the Internet expands to mobile devices, even vehicular nodes are becoming a part of the "edge" of the Internet. Several interesting challenges in application design arise, while designing a targeted ad delivery mechanism for cars.

Consider this scenario: you are driving on Interstate-5 from Los Angeles to San Francisco to visit relatives. On the way, you realize that you need to buy some gift for them. You initiate a search for "new DVD releases". The Ad software is not only keyword aware but also *location* aware. Hence the search results return not only the content or latest DVD releases but also the latest deals on those DVD releases in stores.

Imagine another similar scenario: you are traveling to Las Vegas and are 50 miles from the city. You want to search for all hotels in the vicinity that cost less than 200 dollars per night, preferably with virtual tours of the hotels.

AdTorrent seeks to provide to the user, relevant Ads guided by a particular keyword search. Ads potentially can be multimedia clips, for example, virtual tours of hotel rooms, trailers of movies in nearby theatres or a conventional television ad.

A. Our Contributions

Vehicular Ad Hoc Networks (VANETs) present interesting challenges to protocol design. One of the key differentiating characteristics is the time-varying nature of vehicle densities and the mobility model. Mobility has an important impact on application design.

The contributions of this paper are as follows: (1) firstly, we propose a novel push-model based location-aware ad service architecture, designed for vehicular environments (2) secondly, we present a group mobility model for urban vehicular traffic, (3) thirdly, we present a peer-peer protocol that enables efficient keyword-set based search and quickly delivers top ranked content to the end user using swarming, (4) finally, we present a model for hop limit selection of search query flooding in the AdTorrent network. Our results on the optimal hit rate and the cache probability distribution that maximizes the overall hit rate as a function of the hop limit of query flooding is applicable in any hop-limited query flooding application. Our analysis of hit rates for LRU-based cache management extends previous work in the area by including the effect of swarming on the steady-state cache probabilities.

B. Organization of the Paper

The rest of the paper is organized as follows. Section II describes the Vehicular communication architecture. Section III gives an overview of the operation of the ad service in a vehicular scenario. Section IV describes the novel mobility model we used for the purpose of evaluation and details our evaluation of the performance of our protocol using simulation. Section V gives a brief overview of AdTorrent, a push-model of content dissemination based on a popular swarming protocol. Section VI describes the model for hop limit selection where we derive the maximum hit rate achievable for a specific hop limit as a function of the cache size and describe our model for computing hit rates in a swarming-based content delivery scenario when the underlying cache management is based on LRU. We outline the related work in section VII. Finally, Section VIII concludes the paper.

II. Preliminaries

In this section we describe the vehicular environment and the assumptions about the environment we used to design our

protocol.

The network consists of a set of N nodes with same computation and transmission capabilities, communicating through bidirectional wireless links between each other, this is the infrastructure-less ad-hoc mode of operation. There are wireless gateways at regular intervals providing access to the rest of the Internet using infrastructure support (either wired or multi-hop wireless). We assume a CSMA/CA MAC layer protocol (IEEE 802.11a) that provides RTS/CTS-Data/ACK handshake sequence for each transmission.

Our vehicular wireless architecture is composed of two kinds of communications, namely, vehicle-vehicle and vehicle-gateway. Dedicated Short-Range Communication (DSRC) [4] is a short to medium range communication technology operating in the 5.9 GHz range can be used for vehicle-vehicle communication. For a more detailed description of the DSRC characteristics, we refer the reader to [8].

- *Data is not strictly real-time:* There are no real-time constraints on the data, thus in some sense, the data is delay-tolerant.
- *Data is meta-tagged:* Meta-data can be the file-name, the format and/or key-words extracted from the data. For some types of data, such as text documents, metadata can be extracted manually or algorithmically. Some types of files have metadata built-in; for example, ID3 tags on MP3 music files.
- *Communication between vehicles is over a low data rate connection:* This constraint depends on the radio technology used. Currently, 802.11x devices will offer *goodput* of the order of a few hundred Kbps.
- *Push model:* Data is being continually "pushed" by the access points to the nodes in the transmission range.
- *Multi-hop delivery:* It is infeasible to transmit data to more than a few hops.

III. The Digital Billboard Architecture

The digital billboard architecture serves to deliver Ads to the vehicles that pass within the range of the Access Points (APs). This architecture is:

- *Safer:* Physical billboards can be distracting for drivers
- *Aesthetic:* The skyline is not marred by unsightly boards.
- *Efficient:* With the presence of a good application on the client (vehicle) side, users will see the Ad only if they actively search for it or are interested in it.
- *Localized:* The physical wireless medium automatically induces locality characteristics into the advertisements.

Every Access Point (AP) disseminates certain sets of Ads that are relevant to the proximity of the AP deployment. This is reasonable since it is the extension of the physical billboards that we very often see lined on the streets and freeways that advertise the best offers available at the next restaurant. Our AP acts as a "digital billboard". This model makes sense economically as well since business owners in the vicinity subscribe to this digital billboard service for a fee. The APs continually disseminate these advertisements to the vehicles that traverse the coverage area. The dissemination rate can be determined by different *levels of service* demanded and paid for by the billboard owner.

Leveraging this architecture, we want to design a location-aware distributed mechanism to search, rank and deliver content to the end-user (the vehicle). We focus not only on simple text-based Ads but also on larger multimedia Ads, for example, trailers of movies playing at the nearby theater, virtual tours of hotels in a 5 mile radius, or conventional television advertisements relevant to local businesses.

Every node that runs the application collects these advertisements and indexes the data based on certain meta-data which could be keywords, location and other information associated with the data. We assume that Ads are uniquely identifiable using a document identifier (DocId).

In the next section we describe a group mobility model for an urban vehicular network. The model guides us in the design of AdTorrent, a protocol for advertisement search and delivery on the vehicular network.

IV. Mobility Model

The mobility model is designed to describe the movement pattern of mobile users, and how their location and velocity change over time. Mobility patterns play a significant role in determining the protocol performance and thus are an important parameter to the protocol design phase. It is important for mobility models to emulate the movement pattern of targeted real life applications in a reasonable way. Otherwise, the observations made and the conclusions drawn from the simulation studies may be misleading. Thus, when evaluating our vehicular ad hoc network protocol, it is imperative to choose the proper underlying mobility model. Different application scenarios lend themselves to different mobility models. For example, a campus-wide wireless network deployment will see different mobility patterns (less constrained, more random) than an urban vehicular grid scenario (low entropy of vehicles, group mobility).

In modeling and analyzing the mobility models in a VANET, we are more interested in the movement of individual nodes at the microscopic-level, including node location and velocity relative to other nodes, because these factors directly determine when the links are formed and broken, since the communication pattern is peer-to-peer.

We used the US Census bureau data for street level maps. As a starting point, using the methodology from [11], we generate the mercator projection of the data, in our case the local map of an area around UCLA in Fig. 1.

However, in [11], the actual mobility model is quite similar to the Random Waypoint model in the sense that the vehicle's arrival and direction and speed are similar to the Random Waypoint model. This results in the vehicular mobility model being very similar to the Random Waypoint model. In reality, a complex mobility behavior is observed. Some nodes move in groups; while others move individually and independently; a fraction of nodes are static. Moreover, the group affiliation

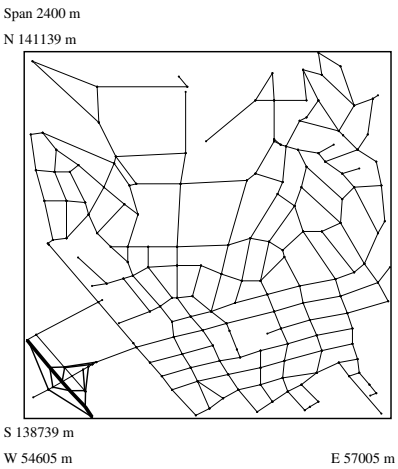

Fig. 1. Local Map of Westwood, an area around UCLA

is not permanent. The mobile groups can dynamically re-configure themselves triggering group split and mergence. All these different mobility behaviors coexist in vehicle or urban scenarios. We refer to the non-uniform, dynamic changing scenario described above as "heterogeneous" group mobility scenario [15]. A good realistic mobility model must capture all these mobility dynamics in order to yield realistic performance evaluation results, which, unfortunately, is not satisfactorily captured in any of the existing models.

We propose a "real track" based group mobility model (RT model) that closely approximates the above "heterogeneous" mobility patterns happening in the scenarios of vehicle ad hoc networks. It models various types of node mobility such as group moving nodes, individually moving nodes as well as static nodes. Moreover, the RT model not only models the group mobility, it also models the dynamics of group mobility such as group merge and split.

The key idea of our proposed model is to use some "real tracks" to model the dynamics of group mobility. In our simulation scenarios, these "real tracks" are derived from the streets from actual maps. The grouped nodes must move following the constraint of the tracks. At the switch stations, which are the intersections of tracks/streets, a group can then be split into multiple smaller groups; some groups may be even merged into a bigger group. Such group dynamics happen randomly under the control of configured split and merge probabilities.

Nodes in the same group move along the same track. They also share the same group movement towards the next switch station. In addition, each group member will also have an internal random mobility within the scope of a group. The mobility speeds of these groups are randomly selected between the configured minimum and maximum mobility speeds. One can also define multiple classes of mobile nodes, such as pedestrians, and cars, etc. Each class of nodes has different requirements: such as moving speed etc. In such cases, only nodes belonging to the same class can merge into a group.

Groups split and merge happen at the switch stations. Each group is defined with a group stability threshold value. When at the switch stations, each node in the group will check whether its stability value is beyond its group stability threshold value. If it is true, this node will choose a different track from its group. A group split happens. When several groups arrive at the same station and select the same track for the next movement, naturally, they will be merged into one bigger group.

The proposed RT model is also capable of modeling randomly and individually moving nodes as well as static nodes(such as sensors). Such non-grouped nodes are not restricted by the switch stations and real tracks. Instead, their movements are modeled as random moves in the whole field.

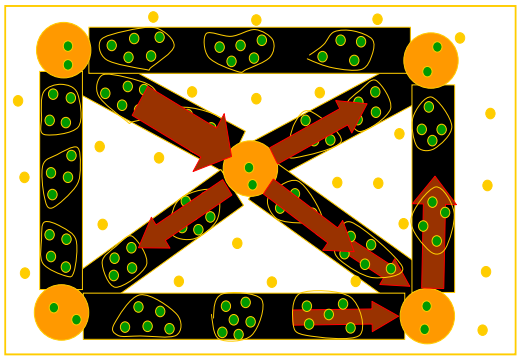

Fig. 2. Overview of Real Track Based Group Mobility Model.

Fig. 2 illustrates a main idea of the proposed real track based group mobility model. In this example, group moving nodes are moving towards switch stations along the tracks. They split and merge at switch stations as shown in the figure. The black nodes in Fig. 2 represent the individually moving nodes and static nodes. They are placed and move independent of tracks and switch stations.

We evaluate the scenarios along the following metrics as defined in [11]. For brevity we present only the average connectivity duration metric, which is the most essential for protocol design in our scenario.

Average connectivity duration: This is the duration of the time two nodes have a path between them. We further quantify this metric based on the maximum allowable hops for any path between the two nodes. This metric is relevant to our application as it justifies the usage of a swarming content delivery model in the presence of limited connectivity between the nodes.

We used a 500m transmission range for the radios. In our case we adjusted the number of nodes, to 30, 50 and 60, spread over an area of $2400m \times 2400m$. The average number of nodes in the transmission range were 4.1, 6.9 and 8.1 respectively. Each run of simulation were 900s long. Also we evaluate the scenarios at regular intervals of 10s.

We observe from Fig. 3 that for a 4-hop limit path the connectivity duration has an almost 100% increase as opposed

to a 3-hop limited path. Longer connectivity durations lead to robust protocol performance (since the initiated downloads have a higher chance of being completed). For urban vehicular scenarios, the results in Fig. 3 suggest that the incremental gain from increasing the hop limit up to 4 might be useful for increasing the robustness of protocol performance.

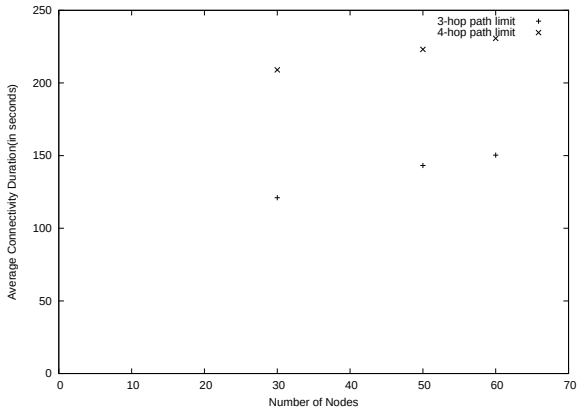

Fig. 3. Average Connectivity duration

V. AdTorrent: Design

We outline the primary design goals of *AdTorrent* as follows:

1) location aware torrent[1] ranking algorithm;
2) search should be simple and robust, in presence of node failures and departures;
3) leverage churn
4) minimal overhead of communication;

There are three main tasks performed by our application. Namely, *search* for relevant ad-content, *query dissemination* and *content delivery*. We address each of these functions in the following sections.

A. Search

Search involves associating keywords with document identifiers and later retrieving document identifiers that match combinations of keywords. Each file is associated with a set of metadata: the file name, its format, genre (e.g. in advertisements). For some types of data, such as text documents, metadata can be extracted manually or algorithmically. Some types of file have metadata built-in; for example, ID tags on MP3 files.

Distributed Hash Tables (DHTs) have been proposed for distributed lookups. We do not use a DHT for distributed lookup, since it is well-known that DHTs are not very stable under high churn [10]. Our query dissemination mechanism aims to achieve robustness rather than communication efficiency.

However, we introduce certain optimizations to the index information dissemination that will reduce the amount of search query communication overhead.

[1]we use the terms : document and torrent interchangeably

Indexing: Vertical partitioning divides an index across keywords. Horizontal partitioning divides an index across documents, so all the nodes have an entry for *each* keyword. Vertical partitioning minimizes the cost of searches. However, horizontal partitioned index reduces the cost of update of document. In our scenario, the number of queries will far outnumber the number of updates, since we assume the documents typically searched for, are not changing frequently. Our index is partitioned based on set of keywords. This was first introduced in KSS [5]. The motivation of using a keyword set based indexing is the reduction of overhead in terms of query data information. The downside of this approach is higher cost of insert and storage. In VANETs, we believe, storage will not be a limitation. In the AdTorrent application, one of the key characteristics will be the infrequent updates of ads, maybe once in one day or less.

The index we maintain is distributed. However, every node tries to maintain information of the all the two-hop neighborhood of itself. The documents are indexed on keys. Keys consists of SHA-1 hashes of the keywords sets. Along with the keys and the URL of the data, we also store additional meta-data associated with the data. The metadata is stored in an index corresponding to each subset of at most K metadata items. KSS uses a distributed inverted index to answer queries efficiently with minimal communication overhead. Each entry of the index contains: (1) the hash of the searchable sets of keywords as the index key, (2) a pointer to the data such as the URL of the data and, (3) meta-data associated with the data.

Placement: In a wireless scenario, it makes sense to co-locate the index and the data corresponding to the index entry. This is to reduce the overhead of data discovery latency once the index for that data has been located.

B. Query Data Dissemination Optimization

Each node disseminates the content availability information in the form of a bloom filter. Bloom filter [1] is an efficient method to test for set membership. In our case, the bloom filter is constructed to test the keyword membership for a particular node. A bloom filter is computed by each node based on the keywords related to the data, the node has stored. Since the data downloaded is only once every AP encounter or if the node explicitly downloads some swarming torrent, hence the updates of bloom filter and dissemination is not very frequent.

We now enumerate the basic steps of the algorithm.

The indexing scheme described above does not have a document ranking algorithm. The order of query results propagation and display is equally important for successful and timely dissemination in a VANET. This assumes further importance in VANET since the mobility of nodes might render some query results obsolete or irrelevant in short period of time. We incorporate a location metric in the document ranking scheme. One way to support the document ranking would be to score a document based on the following categories.

1) location
2) max # of pieces

Algorithm 1 AdTorrent: Query Processing, Ranking and Content Delivery

user_input = **search("A B C")**

num_local_entries = **lookup_local_index**(hash(AB), hash(BC), hash(CA))
if (num_local_entries $> k_1$)
    goto *LookupDone*
else
    /* Found $< k_1$ local entries */
    /* not in the 2-hop neighborhood */
    num_remote_entries = **scoped_flood**(hash(XY), m)
    /* $\forall$ XY $\in$ AB,BC,CA */
    After T_1 seconds, if NO response, return NO
    If k_1 entries are found then

LookupDone:
    /* now have k_1 entries (local or remote in 1-4 hops) */
    send_udp_ctrl(Hash(XY))→METADATA(e.g. TorrentID)

    /* Collect meta_data after T_2 */
    torrent_ranking(meta_data, params)

Step Final:
    swarm(TorrentID)
    /* returns a list of Peers & HopCounts*/
    /* (this may be beyond the the scope of the search) */
    decentralized_tracker()
    /* By allowing the list of Peers beyond the k-hop scope of the search, we add some randomization */

3) stability of neighbors
4) relevance of the DocID to the Meta-Data queried

C. Content Delivery

Once an accurate document ranking has been performed, the actual delivery of content can be done by swarming. One of the factors that determined the ranking of a document in the query results was the number of sub-pieces of the document that were available and the location of the pieces. Thus the torrent ranking guides the system to choose documents which are more amenable to swarming downloads. The vehicle now joins the existing BitTorrent-like stream to start getting pieces of the document from neighboring nodes. We propose to do this using our earlier work in [8]. Swarming allows us to be robust to node failures (cars going out of range or powering down) and efficient in terms of delivery (the cars form a sort of end-system-multicast tree). However, the success of swarming especially in wireless ad-hoc networks depends hugely upon cooperation among vehicles at both the routing layer (forwarding packets for others) and at the application layer (sharing the advertisements that have been downloaded). In the next section we address the concerns with respect to selfish behavior on the AdTorrent network and discuss ways to mitigate it.

VI. Model

As discussed in the previous section, AdTorrent searches for relevant ad-content using a hop-limited query broadcast. Since setting a large hop-limit queries more nodes, a larger hop-limit improves the probability of finding the desired content and will likely increase the number of sources from which the content may be downloaded from. However, the gains in the quality of search results comes at the cost of significant increase in the messages sent per query in the network. Since only limited bandwidth is available in wireless medium, careful analysis of this trade-off between the quality of search results and the communication costs of the search is required. We develop a simplified model of our system to explore this trade-off in this section.

Notice that if sufficient storage space is available at each node, over time as nodes make requests and download content, nearly all content will become available within a few hops. Limited storage space necessitates deletion of previously obtained content preventing accumulation of a sufficiently large corpus of the offered content. Thus, the size of the per-node local storage space is a key factor in determining the hop-limit required to achieve an overall target *hit rate* (the probability of finding the desired content).

Even when the storage space is limited, if it was possible to have a replica of all the content within a few hops from each node we could still achieve a very overall hit rate with a short hop-limit. We formalize these ideas by deriving the allocation of a given per-node storage space that maximizes the overall hit rate for a given hop limit.

While it may be possible to devise distributed mechanisms to achieve the derived optimal allocation, we note that a very high query hit rate by itself does not imply a superlative system performance. For instance, it is conceivable that after accessing a particular content, a user may wish to access a content again in near future (e.g. to compare a hotel room to a previously viewed hotel room). Typically such request patterns imply access-based cache replacement schemes such as LRU (delete the **L**east **R**ecently **U**sed file). Since, LRU is the most commonly implemented cache replacement policy, we analyze the hit rates achieved under LRU cache replacements.

The AdTorrent protocol allows a node to download from multiple sources in parallel. This parallel downloading affects access-based cache management policies like LRU as one content request by a node now has the potential of affecting the LRU ranking of the requested content at many nodes (up to the maximum number of parallel download sessions allowed in the protocol). Therefore, our analysis must evaluate the effect of downloading from multiple sources on the overall hit rates.

We assume that there are N unique files in the system (the term file represents any ad that would be downloaded), each with an associated request rate λ_i for file i per node (the

request rates are *uniform* across nodes[2]). We assume that each file is of equal size[3]. Nodes have finite local storage space to store content files. We assume that the storage space at each node is equal[4] and has the capacity to store B files. Throughout our analysis in this section, we assume that a query is for a particular file[5]. We assume that a query flood of hop limit k reaches $M(k)$ peers.

We use the following notation for the system parameters in our model:

- k = Number of hops in the search query dissemination
- M = Number of nodes in the search range
- N = Number of unique files in the system
- B = Per-node storage size in number of files
- i = File Id
- λ_i = Request rate of file i per node
- $\lambda = \sum_{i=1}^{N} \lambda_i$
- S = Swarming parameter (the maximum number of peers a peer can download from in parallel)
- j = Location in the local cache

Content list aggregation: Note that the aggregation of the content list of neighboring peers is not explicitly included in our model as the relevant hit rates can be trivially obtained from our model that assumes no aggregation. The aggregation of the content list of 1-hop neighbors implies that, with a flood of hop-limit k, we obtain the information about nodes within the hop distance $k+1$. In other words, if we find a hop-limit of $k+1$ to provide acceptable hit rates in the model described herein, AdTorrent will have the same performance with a hop-limit of k. We would like to add that while content list aggregation is a good way to improve the search performance, we believe that aggregating content lists beyond 1-hop neighbors is unwise. In a mobile wireless scenario, neighbors can change frequently and, while updating for the content list of 1-hop neighbors in case of a change is easy, keeping the content list accurate for neighbors more than 1-hop away will require costly change propagation.

For a VANET scenario, our real-track mobility model [15] is an ideal choice. The model can be run with the expected user density and an empirical expression of $M(k)$ can be obtained from the collected statistics. In our investigations, we use two different scenarios: an *dense urban scenario* and a sparser *highway scenario*. In the dense urban scenario, the growth model is grid-like and we obtain $M(k) = \alpha k^2$ with $\alpha = 4$ (our mobility model simulations for the 30 nodes in 2400m x 2400m area case in Section IV gave $\alpha = 4$). For the sparse highway scenario, we obtained $M(k) = \alpha k^{1.4}$ with $\alpha = 4$. In Table 1, we show the growth in number of nodes queried (which is directly proportional to the communication cost).

TABLE I

GROWTH RATE IN DENSE-URBAN AND SPARSE-HIGHWAY MODELS

Hop Limit	1	2	3	4	5	6	7
$M(k) \approx 4k^{1.4}$	4	11	19	28	38	49	61
$M(k) \approx 4k^2$	4	16	36	64	100	144	196

In most web and multimedia applications, different objects have been found to have very different popularity and we expect the same in our Ad-content distribution scenario. Skewed file popularity distribution is typically modeled by a Zipf-distribution and we will use the same model in our investigations.

A. Hit Rate Optimization

When the query flood has a hop-limit of k, the hit rate for file i, $H_i(k)$, is

$$H_i = 1 - (1 - p_i)^{M(k)}$$

The overall hit rate H can be written as

$$H = \sum_{i=1}^{N} \frac{\lambda_i}{\lambda} H_i = \sum_{i=1}^{N} \frac{\lambda_i}{\lambda} [1 - (1 - p_i)^{M(k)}]$$

$$H = 1 - \sum_{i=1}^{N} \frac{\lambda_i}{\lambda} (1 - p_i)^{M(k)}$$

As the hit rates can always be increased if more storage space was available, our optimization is under the the constraint of the available storage space. Since each peer is identical in our model, the file replicas of a file will be uniformly distributed in the network and, for our purposes, it is sufficient to model the probabilities of each file being in the cache. Therefore, for our optimization, we can write the following constraint

$$\sum_{i=1}^{N} p_i = B$$

The Lagrangian for our problem is

$$G = 1 - \sum_{i=1}^{N} \frac{\lambda_i}{\lambda} (1 - p_i)^{M(k)} + \gamma \big(\sum_{i=1}^{N} p_i - B \big)$$

where H is given in the equation above. Optimizing the hit rate w.r.t p_i gives the optimal p_i to be

$$p_i = 1 - \frac{(N - B) \lambda_i^{-\frac{1}{M(k)-1}}}{\sum_{i=1}^{N} \lambda_i^{-\frac{1}{M(k)-1}}} \ \forall \ i$$

[2]We note that the search is localized over a small geographic area so node interests are not necessarily very different. We believe that the uniform request rates assumption provides an adequate average case analysis (i.e. a more accurate model allowing for variations in request rates of files across nodes where the average per-node request rate is λ_i would not give results that are qualitatively any different).

[3]While file sizes can be different, cache replacement is implemented for fixed size data blocks. Thus, any inaccuracy in the analytical model is on account of correlated requests for the disk blocks and not the equal file size assumption. We do not believe correlated requests make a qualitative difference in the results of the model.

[4]The storage capacity in question is the capacity allocated for the push-model data storage and we expect that very few users will allocate more storage than the minimum required by the AdTorrent application.

[5]Even though a query has a set of keywords, a typical user is looking for a particular item when they make the query.

Therefore, the optimal value of H, H^{opt}, is

$$H^{opt} = 1 - (1 - \frac{B}{N})^{M(k)} N \Big(\frac{1}{N} \sum_{i=1}^{N} \frac{\lambda_i}{\lambda}^{-\frac{1}{M(k)-1}}\Big)^{-[M(k)-1]}$$

We plot the optimal hit rates for different cache sizes for Zipf-distributed file request rates and $M(k) = 4k^2$ in Fig. 4. These results indicate that increasing the hop limit shows diminishing returns and, hence, the system designer should select a hop limit that is no more than the minimum desired to achieve the target hit rate. To understand if these optimal hit rates can be achieved, we plotted the optimal cache probabilities for one case, $N = 400, B = 20$, in Fig. 5. As we can see, as the hop-limit increases, the optimal cache probabilities begin to become more uniform (since enough nodes are being queried at high hop distances, it is sufficient to have cache probability below 0.1 for even the most popular file to have a very high probability of finding the file).

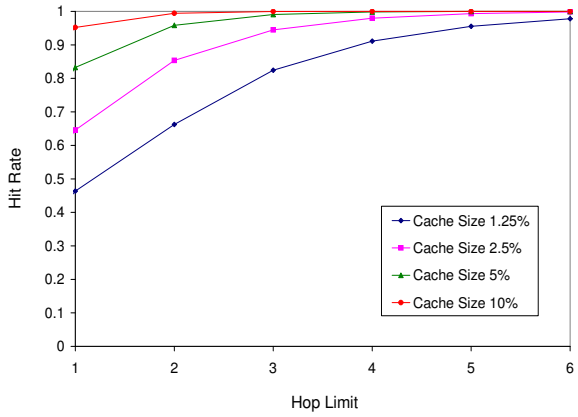

Fig. 4. **Distribution of Optimal Hit Rates with respect to Hop Count for varying Cache sizes with** $M(k) \sim 4k^2$

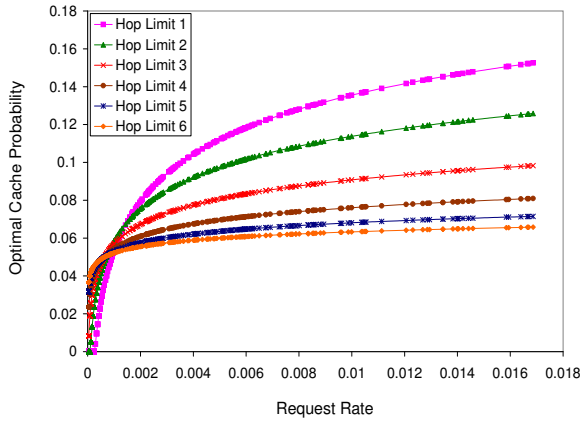

Fig. 5. **Distribution of Optimal Cache Probability with respect to Hop Count for varying request rates with** $M(k) \sim 4k^2$

While this optimization provides us with the best case hit rate performance, as we mentioned earlier, the number of replicas of each file are driven mainly by the file request patterns so it may not be possible to achieve the required cache probabilities needed for the optimal hit rates. Even more importantly, we must bear in mind that finding one source for the searched file (as measured by the hit rate) is not the end-goal. A near-uniform distribution of cache probabilities (as suggested by, say, a hop-limit of 6 in Fig. 5) implies that the sources that have the most popular file will have to serve far more requests than sources that have files that are, say, 10 times less popular as there are almost an equal number of sources of the two files. Such asymmetric distribution of download load is likely to result in queueing delays for the downloads for popular files. As shown in [13], optimizing download time performance requires a linearly proportional replica distribution. Further, in mobile wireless networks, downloading from a single source is unwise as the source peer can go out of range in a short time. Hit rate emphasizes finding one source of a file and gives no weight to finding multiple sources of a file and our design should not be solely guided by the hit rate performance. As [14] discusses, LRU gives a file replica distribution that is close-to-linearly proportional but populates fewer-than-linearly-proportional replicas of popular files. Thus, while LRU may be sub-optimal for download time performance, it is better than a linearly proportional replica distribution for hit rate. Since, LRU is also a popular cache management policy for many other reasons, we evaluate the hit rate performance with LRU next.

B. LRU Model

To determine the probability of finding the desired content in the selected hop limit, we need to find the probability of finding the file at any one node (since the request rates are uniform across nodes, the probability of finding a file is the same across the network). Since each node is same in all respects, an analytical model of the network of LRU-managed caches can be constructed with a single cache that, in addition to serving the local requests, also serves requests from remote nodes. We model this cache from the perspective of a particular file, say, file i - all requests for file i move the file to the top-most position in the storage; a request for any other files moves file i down to one lower position. Reference [3] presents an analytical framework for estimating the hit rate in stand-alone LRU-managed cache. By including the effect of remote requests, their model can be extended to model a network of LRU caches [7].

A critical component of this framework is $r(i, j, k)$, the rate at which file i is pushed down from position j to position $j + 1$ when the hop limit for scoped flooding is k hops. Let $p_{local}(i, j, k)$ be the probability of finding file i in top j positions in the cache when the hop limit is k hops. The probability of finding file i in local cache given a hop limit of k is then $p_{local}(i, B, k)$. $p_{local}(i, j, k)$ can be expressed in terms of $r(i, j, k)$ [3] by:

$$p_{local}(i, j, k) \approx \frac{r(i, j, k)}{\sum_{i=1}^{N} r(i, j, k)}$$

At steady-state, the push-down rate for file i from position j to $j+1$, $r(i,j,k)$, must equal the rate at which file i is brought into top j positions of the LRU stack (otherwise the probability of finding the file in these top j positions becomes unbounded). This conservation of flow principle helps us compute $r(i,j,k)$. File i is brought into top j positions under two conditions: (i) a local request for file i when file i is not in top j positions: the file may be brought to the top position from positions $j+1 \cdots B$ of the local cache if it is available there or it may be brought from a remote node (if a node within the search range has the file), this is $r_{local}(i,j,k)$; (ii) a remote request for file i: since the file i is not in top j positions, it must be in the remaining $j+1 \cdots B$ positions in the local cache for it to show up in top j positions on a remote request. Thus, we can write the following equations:

$$r_{local}(i,j,k) = \lambda_i[1 - p_{local}(i,j,k)]$$
$$[1 - (1 - p_{local}(i, B-j|j,k)(1 - p_{remote}(i,j,k)]$$

$$r_{remote}(i,j,k) = \lambda_i[1 - p_{local}(i,j,k)]$$
$$p_{local}(i, B-j|j,k)$$

where

$$p_{local}(i, B-j|j,k) = \frac{p_{local}(i,B,k) - p_{local}(i,j,k)}{1 - p_{local}(i,j,k)}$$

and

$$p_{remote}(i,j,k) = [1 - (1 - p_{local}(i,j,k))^{M(k)}]$$

A node sends out a file request only when it does not have the file. Thus, the rate at which the other $M(k)-1$ nodes send a file request for this file to the peer-to-peer network is $\lambda_i[1 - p_{local}(i,B,k)]$, where $p_{local}(i,B,k)$ is the probability that the file i is available at a node. The nodes that have file i in their cache satisfy these requests for file i sent to the peer-to-peer network. Assuming that the requests are uniformly distributed over the nodes that have the file, the request rate for file i served by a node that has file i on account of requests from other nodes equals $\frac{(M(k)-1)r_{remote}(i,j,k)S}{M(k)p_{local}(i,B,k)}$. Thus,

$$r(i,j,k) = r_{local}(i,j,k) +$$
$$\frac{(M(k)-1)r_{remote}(i,j,k)S}{M(k)p_{local}(i,B,k)}$$

Starting with $p_{local}(i,1,1) = \lambda_i$ we can iteratively solve the above equations until the value of $p_{local}(i,B,k)$ converges. The complexity is $O(NB)$ and, in our computations, the value of p converged in only a few iterations.

Given $p_{local}(i,B,k)$, we can compute the hit rate for file i in the k-hop neighborhood as $P(i,B,k) = [1 - (1 - p_{local}(i,B,k))^{M(k)}]$ and the overall hit rate (across all searches) as:

$$P(B,k) = \sum_{i=0}^{N} \frac{\lambda_i}{\lambda}[1 - (1 - p_{local}(i,B,k))^{M(k)}]$$

Among the inputs to our model, the cache size B and the hop limit k are the design choices while λ_i, the file request rate distribution, and $M(k)$, the number of nodes in the k-hop neighborhood, are inputs that the system designer must provide for the specific application scenario being investigated.

We show the LRU performance for the dense, urban scenario and the sparse highway scenario in Figs. 6 and 7 respectively for Zipf-distributed file request rates. In Figures 6, 7, the cache ratio refers to the size of the individual node cache with respect to the total number of files in the network. So for example, a cache ratio of 0.1 means, an individual nodes' cache can store 10% of the total files in the network.

We find that with increasing hop count, the marginal gain in hit rates diminishes. This effect is even more pronounced as the cache ratio increases. Our analytical framework can be used to tune the query flood to achieve required levels of hit rates, and consequently the performance of AdTorrent by suitably adjusting the hop limit of the query flood. So, for example, if 80% hit rate was a satisfactory level of performance measure, our results suggest that a query hop limit of 4 will yield satisfactory performance in the dense-urban scenario irrespective of the cache size (as long it is above a certain threshold). Recall that our mobility model simulations in Section IV also suggested a hop limit of 4 to obtain a reasonable average connectivity duration which would facilitate robustness in protocol performance.

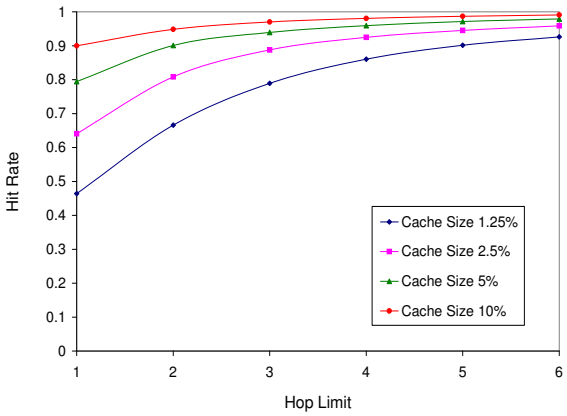

Fig. 6. **Hit Rate vs. Hop Count with LRU, for varying Cache sizes and** $M(k) \sim \alpha k^2$

C. Effect of Spreading Factor

As already discussed, downloading from multiple sources in mobile wireless networks is preferable. However, since parallel downloading affects cache probabilities, we wish to check the effect of increasing the spreading factor on the hit rates. In Fig. 8, we show the hit rates for different spreading factors. As we can see from the figure, the spreading factor has no effect on the hit rate so our choice of the spreading factor is not limited by hit rate considerations.

D. Hop Limit Selection

We can see from Figs. 6 and 7 that the cache size is an important factor in determining the hit rate and, thus in determining the appropriate hop limit. For example, if the

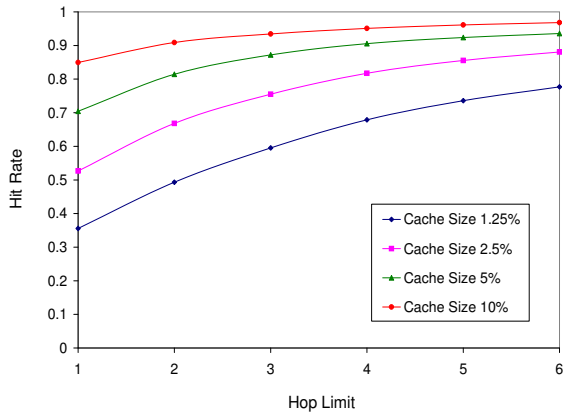

Fig. 7. **Hit Rate vs. Hop Count with LRU, for varying Cache sizes and sparse node growth** $M(k) \sim \alpha k^{1.4}$

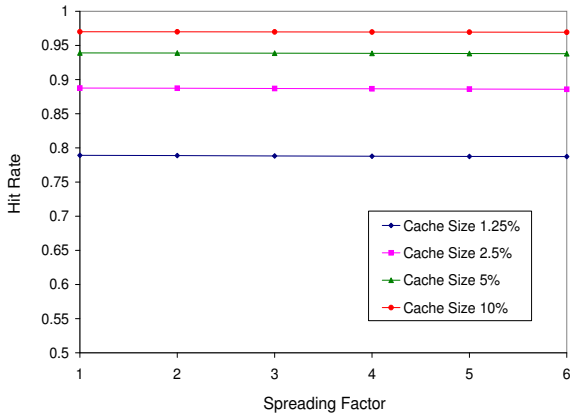

Fig. 8. **Effect of spreading factor on hit rate,** $M(k) \sim \alpha k^2$**, Hop Limit 3**

cache size is only 1.25% of the total number of files, we need information on about 60 nodes to achieve better than 75% hit rate and, as our content list aggregation gives us information on one extra hop, we need a hop limit of 6 in the sparse highway scenario (notice that, in the denser and faster-growing urban scenario, a hop limit of 3 would be sufficient to achieve this hit rate). In contrast, if the cache size was 5% of the total number of files, even with a hop limit of 2 in the sparse highway scenario (i.e. with querying only 11 peers per query) we get a hit rate of 80%.

E. Effect of Skew in File Request Rates

Skewed data access patterns improve the system performance in caching applications. To determine the sensitivity of hop limit selection on the skew in request rates, we computed the hit rate for different request rate distributions using our LRU model. The results are shown in Fig. 9. As expected, decrease in skew does decrease the overall hit rate but, unless the request rate has very little skew (e.g. has a zipf-parameter of 0.5), it may not be necessary to increase the hop limit or the cache size.

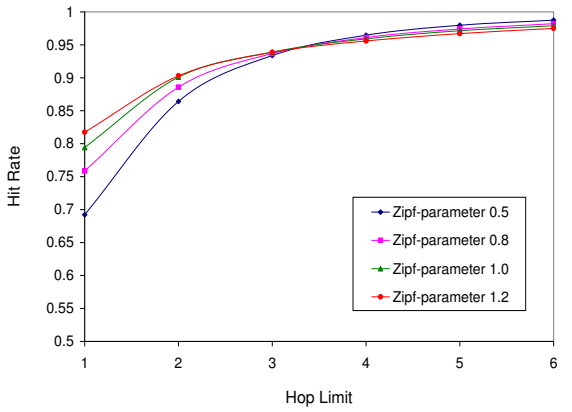

Fig. 9. **Effect of skew in request rates on the hit rate,** $M(k) \sim \alpha k^2$**, Cache Size 5%**

VII. Related Work

This section summarizes previous work related to cooperative data transfer protocols for the wired settings as well as vehicular environments. BitTorrent is a popular [2] file-sharing tool, accounting for a significant proportion of Internet traffic. There are two other peer-peer bulk transfer protocols namely, CarTorrent and Coopnet. CarTorrent [8] is a recent work that extends the BitTorrent protocol to the vehicular networks scenarios addressing issues such as intelligent peer and piece selection given the intermittent connectivity and limited bandwidth of the wireless medium.

Peer-to-peer networking in cooperative mobile environments has been proposed by several others. However, the constraint of limited buffers at the peers is discussed by very few others. [7] analyzed epidemic information dissemination to support web accesses with limited buffers per peer. Our analytical model to study the trade-off between the query hop limit and the overall hit rate is very similar to theirs. Since our problem setting is different from theirs, some of our assumptions are different. As discussed earlier, our model (as well as that of [7]) is an extension of the analytical model for a stand-alone LRU cache given in [3]. Limited buffers in case of peer-to-peer networks in wired scenario are also discussed in [14] which also uses a similar model for network of LRU caches but their end goal is different from ours.

VIII. Conclusion

In this paper we presented a novel application involving search and location aware content delivery(in our case advertisements/deals) to the nodes in a Vehicular Adhoc Network. We proposed a efficient keyword search on this content overlay. To aid system designers in selection of the design parameters in their implementation of AdTorrent application, we present a realistic mobility model for the urban, vehicular scenario and an analytical model of the epidemic query dissemination to evaluate the impact of the scope of the query dissemination on the hit rate. We derived the optimal hit rate as a function of the cache size and the hop limit and then developed a model for performance with LRU cache

management when swarming-based content delivery is used. System designers can use our analytical framework to estimate the required cache size and scope of the query dissemination based on user performance requirements. In our evaluation of some local scenarios, we found that a hop limit of 4 hops gives an adequate hit rate and that the incremental gain from increasing the scope of the query flood beyond 4 hops was minimal. These results are very encouraging in that they show the feasibility of AdTorrent deployment in urban scenarios.

REFERENCES

[1] B. Bloom, Space/time tradeoffs in hash coding with allowable errors. CACM, 13(7):422-426, 1970.

[2] B. Cohen, *Incentives Build Robustness in BitTorrent*, in IPTPS 2003.

[3] A. Dan and D. Towsley, *An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes*, in Proceedings of ACM SIGMETRICS 1990.

[4] *Dedicated Short Range Communication Architecture*, www.astm.org/SNEWS/MAY_2004/dsrc_may04.html

[5] Omprakash Gnawali, *A Keyword-Set Search System for Peer-to-Peer Networks*, Masters Thesis, Massachusetts Institute of Technology, 2002.

[6] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang, *A group mobility model for ad hoc wireless networks*, in Proceedings of ACM International Workshop on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM), August 1999.

[7] C. Lindemann and O. P. Waldhorst, *Modeling Epidemic Information Dissemination on Mobile Devices with Finite Buffers*, in Proceedings of ACM SIGMETRICS 2005.

[8] A. Nandan, S. Das, G. Pau, M.Y. Sanadidi and M. Gerla, *Cooperative Downloading in Vehicular Wireless Ad Hoc Networks*, In Proceedings of Wireless On-Demand Networks and Services, St. Moritz, Switzerland, Jan 2005.

[9] *QualNet user manual* www.scalable-networks.com

[10] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, Handling Churn in a DHT, in Proceedings of USENIX 2005

[11] A. Saha and D. Johnson, *Modeling mobility for vehicular ad-hoc networks* in Proceedings of ACM VANET 2004

[12] K.Tang, M. Gerla and R. Bagrodia, *TCP Performance in Wireless Multi-hop networks*, in Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications, 1999.

[13] S. Tewari and L. Kleinrock, *On Fairness, Optimal Download Performance and Proportional Replication in Peer-to-Peer Networks*, in Proceedings of IFIP Networking, May 2005.

[14] S. Tewari and L. Kleinrock, *Proportional Replication in Peer-to-Peer Network*, to appear in Proceedings of IEEE INFOCOM 2006.

[15] B. Zhou, K. Xu and M. Gerla, *Group and swarm mobility models for ad hoc network scenarios using virtual tracks*, in Proceedings of MILCOM 2004