Towards a Geocentric Mobile Syndication System

Adrien Debrie, Pacôme Eberhart, Pierre-Louis Roman, Cécile Le Pape, Olivier Marin

Towards a Geocentric Mobile Syndication System

Adrien Debrie, Pacôme Eberhart, Pierre-Louis Roman, Cécile Le Pape, Olivier Marin
Laboratoire d'Informatique de Paris 6 / CNRS
Université Pierre et Marie Curie
Paris, France
Email: firstname.lastname@lip6.fr

***Abstract*—The present paper proposes a new approach towards filtering and processing the ever growing quantity of data published from mobile devices before it even reaches the Internet. We tackle this issue by circumscribing data to the zone where it is published (geocentricity) and allowing mobile device owners to republish the data they deem relevant (syndication). Results we obtained through simulation show that our solution enables to extract information that is relevant for a majority of users, whilst allowing less relevant data to be exchanged on a more local scale before it disappears entirely.**

***Keywords*-mobile devices; data syndication; recommendation systems**

I. Introduction

The proliferation of mobile devices such as smartphones and tablet PCs leads to the daily production of increasingly huge quantities of data [1]. Presently, these data are mostly exploited via the Internet by social networks, blogs and forums [2], which offer various services for subscribing (feeds), filtering, aggregating, ranking, polling, commenting, and so on.

However, the constant increase in the production of data raises technical and societal issues. For one, the Internet might run out of capacity for absorbing an amount of information that keeps being generated ever faster as new technologies and new online applications become available to ever growing worldwide communities. Another concern is that most elements of data which get published on the Internet only concern small fractions of the connected populations. This generally impedes the access to relevant data. A solution both for facilitating access to data and alleviating the overall information load may be to confine mobile data to particular locations and to limit online publication to *pertinent* data.

In this paper, we propose to filter data being produced on mobile devices before they even reach the Internet. To this end, we design a decentralized data filtering system among mobile users on top of data geolocation. Our work uses syndicated[1], geocentric[2] data in order to provide ways to filter and regroup them as they are being exchanged locally among mobile devices. Devices may publish their own feeds, acquire feeds from other devices and enrich the information with comments added by their owners. Data may be aggregated according to device- or user-specific methods, and then republished. Information with low popularity will thus disappear as it fails to get relayed by enough users, whereas highly popular information gets enriched as it gains relevance amongst mobile users and may thereafter reach the Internet through more classical means.

Syndication over MANETs has been researched before [5], [6], [7] but using data geolocation so as to ascribe the syndication of data to a specific zone remains, to the best of our knowledge, unexplored. We believe there are many potential applications to this approach. For instance, News reports could be extracted from data that got spread over the location of an event. A journalist might collect information locally both by taking pictures and conducting interviews, and by aggregating pictures and comments from other onlookers. Back at the news agency, the information could be filtered by relevance, by popularity, or even by similarity for basic crosschecking. In the context of wide area events such as large conventions or conferences, participants could exchange real time information about local micro-happenings without the need to connect to the Internet; only the most popular items of information would get published afterwards on the event organiser's website.

The main contributions we present in this paper are (i) a new architecture for sharing data feeds among neighbouring mobile devices, and (ii) a mobile network propagation system that promotes data popularity/relevance.

II. A platform for geocentric mobile syndication

The present section depicts the overall architecture of our solution, called GEMS (*GEocentric Mobile Syndication*), which combines data geocentricity with a feed model and a caching scheme in order to filter out data that users deem irrelevant.

The point of geocentricity is to disseminate data amongst mobile devices in a manner that keeps the information contained within a given area. We propose a lazy approach where users located in the area exchange only data in which they manifest interest. Users subscribe to geocentric feeds and acquire new items associated to these feeds in an

[1]Syndicated data authorizes subscribers to republish it.

[2]The maintenance/liveness of geocentric data [3], [4] is circumscribed to the zone where it gets published.

opportunistic manner, every time they come in contact with another user. In order to support data geocentricity, every participating mobile device must be aware of its location. This may be achieved by GPS, but there is no strict requirement on the accuracy of the positioning technique. Another requirement is that devices must be able to exchange data wirelessly amongst themselves, but our approach does not impose any constraint on the communication interface (Bluetooth, Wi-fi Direct, NFC, ...)

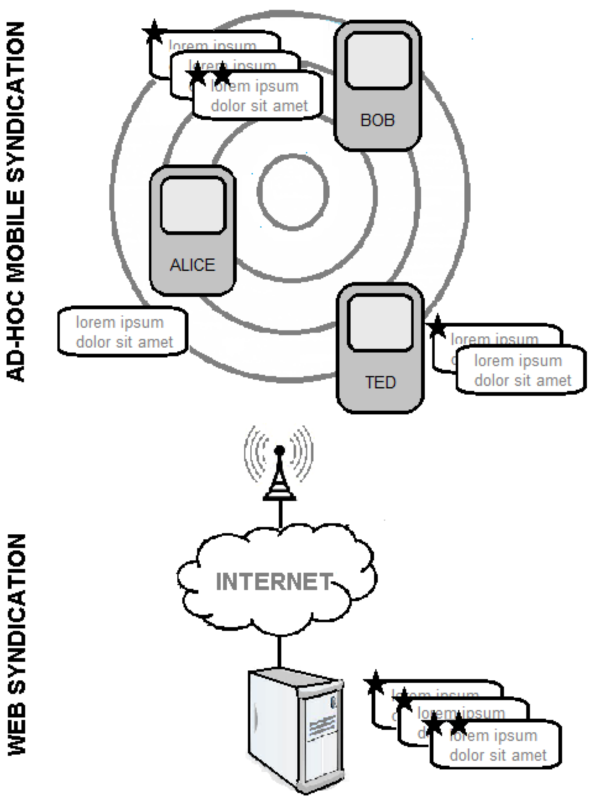

Figure 1. Data storage and exchange in the GEMS architecture.

Users exchange information through data feeds: every data item in a feed is a multimedia message (text, picture, video, ...) published by a user. Users can read and add comment for every item of the feed. To enable user subscription, feed descriptions can be made available online on web servers, or within mobile ad-hoc service discovery protocols, or even advertised with QR codes.

Figure 1 presents our overall architecture, where mobile devices form a store-and-forward dissemination organization. Users post original feed items on the spot of their location and share them. The publication mechanism associates every item with a zone that is centered on the postage location. When entering the zone, other devices may acquire the item and store a copy in their local cache. They may also republish it when establishing contact with another device inside the zone. The probability of exchanging items depends on a combination of metrics, among which the distance from the center of the zone.

If there is no available device to replicate an item, then the latter may disappear. In order to improve the availability of feed items within the mobile network, our solution uses replication and caching techniques detailed in Section V. Every mobile device dedicates a fraction of its storage capacity for the sharing of items. Upon every contact, up to several megabytes of data may be transferred between devices, depending on the contact duration and the size of the exchanged items. Some feeds may not have been synchronized when the communication ends, possibly because of a unilateral disconnection, a network error, or low bandwith. A missing feed item can be downloaded during a later contact with the same device or with another one, or from a backup server if it becomes popular enough.

Users also exchange votes and expose items together with their popularity, along with other metadata such as lifespan, publication date and location; this aspect of our solution is detailed in Section III. An important hypothesis of our approach is that popular data items are more likely to be relevant for a wider audience. User comments are exchanged between devices until items become popular enough to be published on the Web, for instance in a regular RSS or ATOM feed. Less popular items are eventually deleted from every device. The overall system acts as a collaborative filter for shared items based on their popularity.

Our architecture does not require any fixed IP infrastructure in order to function. However, Internet servers may also be used for backup support in order to increase data availability and as a global publishing extension for internet feed readers. Such a server infrastructure may be centralized, or distributed over a P2P overlay; its purpose is to provide persistence for the most popular items among mobile device users.

A user might be interested in storing some unpopular items locally, for example to read them later or to force their publication on the web. A user may also want to enrich such items with comments to other more popular items. For this purpose, our architecture supports another level of persistence: a user can store favorite/relevant items locally and then delete them manually. A user may also redefine arbitrarily the lifespan of an item stored locally. However, user profiling falls outside the scope of the present paper.

III. Feed model

Users exchange information through data feeds. The main assumption of our approach is that people living in the same area, working together or sharing social events are more likely to share common interests. Sharing feeds associated with geographic areas follows this assumption. Thus a GEMS feed is an ordered sequence of geolocalized *news*. More

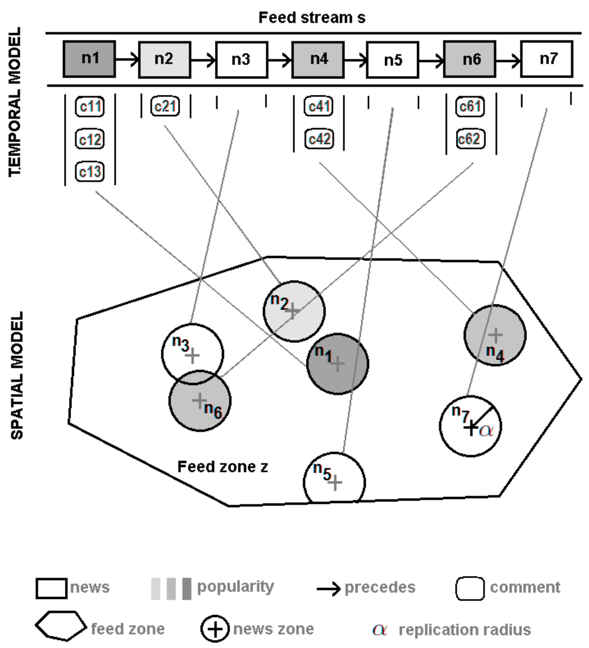

Figure 2. Temporal and spatial representation of a GEMS feed.

precisely, as shown in figure III, a GEMS feed is a tuple $f = (f\_id, t_f, d, z, s)$, where f_id is the identifier of the feed, t_f is a lamport timestamp[8], d is the feed description, z is the feed zone and s is a stream of news.

A. Definition of a GEMS feed

The logical timestamp t_f is used to maintain loose causal ordering between news within a feed stream (details in Subection III-B). t_f is initially set to 0 and incremented every time a news item is added to the feed. When receiving fresh news associated with a timestamp t', the local feed timestamp t_f is set to $max(t_f, t') + 1$.

Users consult the feed description d in order to decide whether they wish to subscribe to the feed. d is a tuple which represents the usual RSS/Atom feed properties: title, description, publication date, authorship, key words, URL, etc. The URL is derived from title and publication date properties.

The feed zone z is a geographic area (e.g. building, campus, street, town). The *zone* is represented by a geometry (e.g. a polygon or a circular area) based on spatial coordinates. A GEMS feed is geocentered within its zone : it contains news published within the zone by mobile users. A feed is *active* for a mobile node when the node is localized within the feed zone, and *inactive* otherwise.

B. Definition of a GEMS stream

The contents published by users into a given feed are inserted in a data stream: a GEMS stream s is a data stream of annotated news items. It is defined as a (possibly infinite) set of news $s = \{n_i\}$. Each news n_i is a tuple $(n\_id, d, t, loc, \alpha, s, C, \sigma)$, defined as follows:

- each news n_i is identified by a hash value n_id based on the author identifier and the publication date;
- t is the timestamp of the news item; its value is set to that of the local feed timestamp at creation time, allowing to maintain a weak causal order between news: $n_i \prec_n n_j \iff t_i <= t_j$.
- loc is the location (e.g. GPS coordinates) of the spot where the news item was created; a replication radius α is also associated to every news at creation time : the circular area (loc,α) is called the *news replication zone* and is used for cache management, as presented in Section V;
- d is the content of the news published by a mobile user; d is a tuple which contains a multimedia file and text description, plus metadata such as publication date and authorship;
- C is a list of comments; each comment is a tuple which contains a comment identifier, a small text and authorship.
- the signature σ of each news is defined as a bloom filter based on its comments. The signature is an array of M bits (typically $M = 64$) used to test whether some new data is a member of a set. Every data is associated with k bits computed with k different hash-functions (typically, $k = 3$ or 4). A data is a member of the set if each k hashed value of the data is already set in the filter. False positives are possible but false negatives are not; although this may lead to missing data, it prevents duplicates.

We also define the *popularity* of a news item n_i, denoted $\gamma(n_i)$. Many popularity functions have been proposed before, mostly based on explicit votes or on user behaviour. In order to minimize the overhead induced by its computation, we introduce a simplistic popularity function : the popularity value of a news item n_i is equal to the number of comments attached to n_i.

IV. MOBILE SYNDICATION

The GEMS syndication protocol follows a publish/subscribe communication pattern. A mobile device user begins by subscribing to GEMS feeds in order to be able to create or comment on news associated with these feeds. Whether online or through an ad-hoc mobile feed discovery protocol, the subscription method is beyond the scope of this paper. Periodically, the mobile device (acting as client) initiates a publishing phase with one of its reachable mobile neighbors (acting as server). Application views are then automatically refreshed on the client side. Finally, a web publishing phase occurs in order to send the most popular news on web servers. The present section details the GEMS ad-hoc mobile publishing algorithm.

Algorithm 1 GEMS publishing algorithm

```
Require: a feed $f$, a mobile peer acting as a server $s$
Ensure: $f$ is refreshed with news and comments from $s$
 1: if feed $f$ is active then
 2:     send the feed identifier $f\_id$ to $s$
 3:     receive identifiers of local news from $s$, ordered by decreasing popularity
 4:     compute the list of missing news
 5:     compute the list of common news
 6:     for each $news\ n_i$ in common do
 7:         send the signature of $n_i$
 8:         receive missing comments from $n_i$
 9:     end for
10:     send the list of missing news identifiers
11:     receive the list of missing news
12: end if
13: return $f$
```

Mobile devices can independently add, share and delete content within the feed. Each device maintains a partial view of the global feed. When a contact occurs between a mobile device acting as $client$ and another mobile device acting as $server$, the client attemps to download missing news and comments from the server. Algorithm 1 describes the GEMS publishing protocol that a mobile device follows for every feed f in its subscriptions list upon contact with a server device.

The GEMS publishing algorithm is *stateless* : each message is treated as an independent transaction, and therefore the server retains no information about previous contacts. The client first checks for an active feed f to refresh. For every active news n_i in the feed cache, the server computes its *popularity* value $\gamma(n_i)$. The server sends its local list of news identifiers, ordered by popularity. The client splits the list in two : *diff* is the list of news that are yet unknown to the client; *common* is the list of news that the client already knows but that may have new comments.

New comments are uploaded first : the signature s_i of each news n_i in the list *common* are sent to the server, which returns comments on n_i that are not in s_i. The client adds the comments to their associated news with respect to $\prec_c$, the precedence order among comments. The popularity of each refreshed news increases.

Then the missing news are uploaded : the client sends the list *diff* of missing news identifiers to the server, which replies with the requested news. The client adds the news to the feed with respect to $\prec_n$, the precedence order amongst news.

Since connections between devices may be short-lived due to their mobility, ordering news items by popularity allows to transfer the most popular news first.

The above feed publication algorithm allows device users to update their local view for data they are explicitly interested in. However, this scheme does not suffice for the propagation of all data amongst mobile nodes. GEMS integrates a cache management scheme alongside feed publication in order to enforce an extensive dissemination of news items within $feed\ zones$.

V. CACHE MANAGEMENT

One of the goals of our approach is to maintain the data published by users within a particular area, and replicate it amongst the other devices in the same area so as to guarantee data availability. Thus, the replicated information forms a distributed cache in which the mobile nodes carry a set of news, depending on the geographic origin of these data. Upon contact with peers met in their vicinity, and following the principles of opportunistic communications, users try to copy data from these neighbour peers, according to an appropriate data management policy.

A. Cache model

Our approach relies on a two-level data cache: the first level is the *hot* data cache, the second one is the *probation* data cache. The hot cache contains the data items that have to be shared with the highest priority: the most popular ones and the newly created ones. The probation cache is a "purgatory" where less popular messages still have a chance of reaching the hot cache.

Both caches are defined by a number of slots on every mobile device: one cache slot holds exactly one news item, regardless of its size. Since multimedia item sizes may vary greatly, organizing the caches into slots enables to set cache sizes dynamically. By imposing a limit on the item size in the application, it is possible to tune the number of slots on any device so as to adapt the maximum cache size to the capacity . This allows to account for very different hardware configurations: typically, the storage capacity is much larger on laptops than on smartphones.

B. Cache policy

Initially, both the hot cache and the probation cache on a mobile device are empty.

Insertion in the hot cache occurs in three cases.

1) *When there is a free slot available in the hot cache.* If the hot cache isn't full, then any message to be stored locally is inserted there.
2) *When a new item is created locally.* A new item from the mobile device user is always directly inserted in the hot cache, whether the latter is full or not. This policy guarantees that new information gets a starting chance: an opportunity to spread amongst mobile users and gain popularity.
3) *When the popularity of an item is high enough.* Items with high popularity values may be received from other mobile devices; items in the probation cache may gather enough popularity to justify their promotion to

the hot cache. Whether it was received from another device or whether its popularity increased, an item gets promoted to the hot cache when its popularity value is higher than that of the least popular item in the hot cache.

Inserting an item in the hot cache when it is already full implies freeing a slot. To do so, an item in the hot cache gets demoted to the probation cache. Demotion to the probation cache is always imposed on the least popular item from the hot cache. In a situation where several items share the minimum popularity value in the hot cache, then the one with the least recent timestamp is moved to the probation cache.

Items received from other mobile devices get inserted in the probation cache if their popularity is too low to justify demoting another item from the hot cache. If the probation cache is full, the insertion of an item requires discarding some probationary item. The selection process for items to be discarded from the probation cache relies on their timestamps: the least recent pieces of information are abandoned first. For two items of equal age, the least popular one is discarded first.

Algorithm 2 Cache management algorithm

```
Require: a message $m$ to store locally, the hot cache $hc$
    and the probation cache $pc$
Ensure: $m$ is inserted in the proper cache
 1: if a slot is free in $hc$ then
 2:   insert $m$ in $hc$
 3: else
 4:   if $m$ is a newly created item then
 5:     demote least popular message from $hc$ to $pc$
 6:     insert $m$ in $hc$
 7:   else
 8:     //$m$ is a received message
 9:     if $\exists m_2 \in hc \mid \{m_2$ less popular than $m$, or equally
        popular but with inferior timestamp$\}$ then
10:       demote $m_2$ from $hc$ to $pc$, insert $m$ in $hc$
11:     else
12:       //$m$ can't be inserted in $hc$
13:       if a slot is free in $pc$ then
14:         insert $m$ in $pc$
15:       else
16:         drop oldest message from $pc$, insert $m$ in $pc$
17:       end if
18:     end if
19:   end if
20: end if
```

Algorithm 2 sums up the cache management policy on every mobile device. The whole policy aims at guaranteeing that those data which have survived a long time in the system without gathering enough popularity are the first ones to be deleted.

C. *Cache content propagation protocol*

Due to the mobility of the devices, contact duration is bounded. Once the GEMS publishing algorithm presented in Section IV has been carried out, there may not be enough time left to transfer all the requested news items. Therefore determining the order in which data items will be transferred can be critical.

Several schemes can be used to enforce priorities amongst items. Random or FIFO ordering would be very efficient, but will preserve neither the geocentricity of the data nor its popularity. Since both of these notions are essential in our context, we introduce a heuristic to take them into account.

As described in Section III, every news item is associated with a publication location loc and a replication radius α. Both values define a round-shaped area called the $replication\ zone$ of the data. Data storage is location-centric: data must be contained within a $feed\ zone$, and kept as close to its origin as possible. In order to achieve this, every data item has a probability $prob$ of being replicated on another device during an opportunistic contact. The value of $prob$ depends on the current location of the server device. Given the value $dist$ of the distance between the server device and the publication location of the data item, $prob$ is computed as follows:

$$prob = \begin{cases} 1 & \text{if } dist \leq \alpha \\ (\alpha/dist) & \text{if } dist > \alpha \\ 0 & \text{outside feed zone} \end{cases}$$

We compute the pertinence $\nu(n_i)$ of a news item n_i as follows:

$\nu(n_i) = prob \times \gamma(n_i)$

with $\gamma(n_i)$ the popularity value of n_i as presented in Section III. For obvious reasons, we impose that a news item has no pertinence outside its feed zone.

Upon transferring its cache data to another device, every mobile device orders the requested items by pertinence. Hence, the most popular items that are closest to their publication location get a better chance of being replicated. This protocol is called *MIF*, which stands for *Most Important First*. The MIF protocol benefits the most popular news items without any moderation, which may prevent less popular items from ever being transferred; the latter may therefore never get an opportunity to become popular. As a consequence we introduce a second protocol, called *MIFProba*, which picks a *probationary* news item at random and places it amongst the most popular items.

VI. RESULTS

This Section presents a preliminary evaluation of our approach on top of the Opportunistic Network Environment simulator [9], which allows to simulate the mobility of devices on a map.

A. Simulation settings

We ran the default scenario proposed by the simulator: several pedestrians[3], each of them a mobile device user, wander randomly in a [3400m x 4500m] portion of the city map of Helsinki, Finland. Users choose random points on the map and follow the streets to reach their destination. Their movement speed varies from 0.5 to 1.5 m/s and they use wireless Bluetooth interfaces (2 Mbps, with a range of 10m) to communicate. At the beginning of the simulation every device generates 10 news items, each one of them 100Kb in size but with a different popularity in the range [1..10]. The radius of the replication zone around every publication location is 10m. The cache size of every device is set to 5 slots: a maximum of 5 *hot* news items and 5 *probationary* news items can coexist on the same device.

We assessed the behaviour of three different cache content propagation protocols: *Random*[4], *MIF*, and *MIFProba*. All three protocols were tested on the same map, with varying densities of devices. A nighttime and a daytime simulation populate the map respectively with a total of 50 devices and 300 devices, leading respectively to a low density of 2.6 devices/km^2 and a fair density of 19.6 devices/km^2.

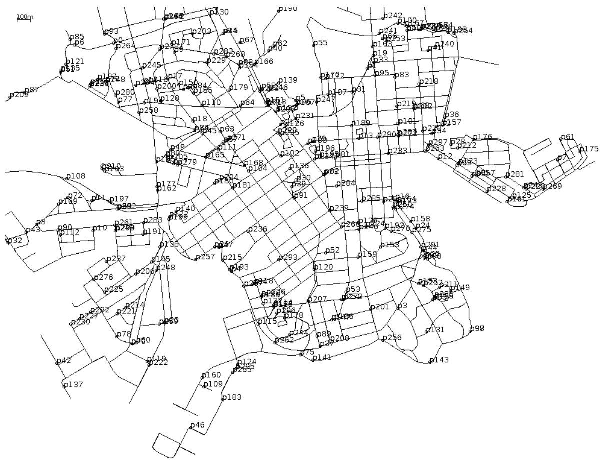

Figure 3. Map of a simulation with 300 devices.

B. Evaluation results

Figure 4 shows the impact of popularity on the average number of times a news item gets transferred when the device density is low. *Random* propagates news uniformly, regardless of their popularity. The consequence is that every news item gets the same coverage amongst mobile devices: a low average of 15 transfers at night. Conversely, *MIF* increases this value above 40 for the most popular news, ensuring that these will get disseminated amongst virtually all the devices on the map. This is important because

[3]We removed cars and trams from the simulation.

[4]The *Random* protocol orders news items at random before transfer.

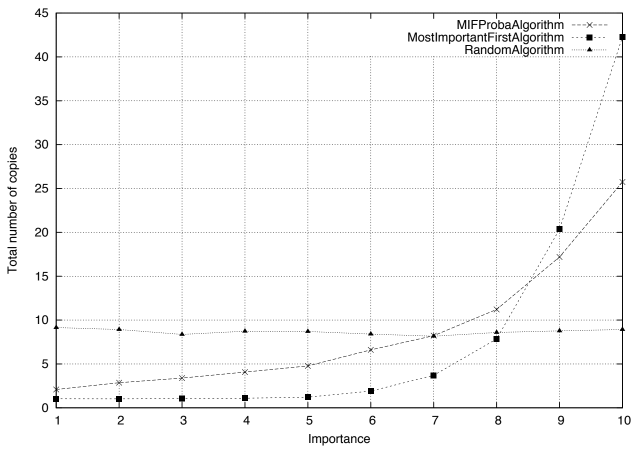

Figure 4. Nighttime average number of transfers per popularity.

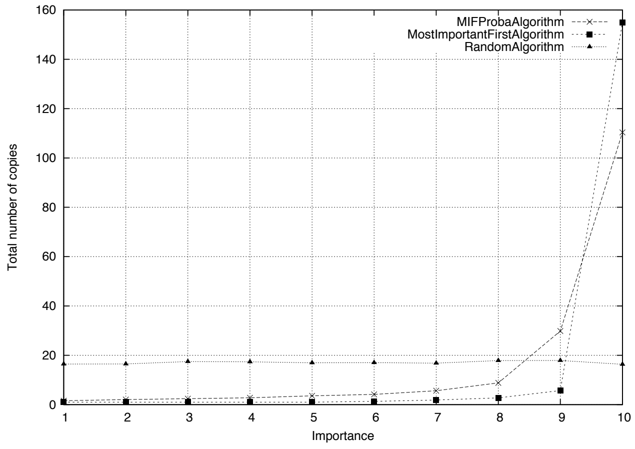

Figure 5. Daytime average number of transfers per popularity.

low device density increases the probability of losing data. Hence MIF completes our objective of guaranteeing the resilience of news items proportionally to their popularity. As expected, *MIFProba* moderates the effect of popularity on news propagation, but retains the benefit to higher popularities.

In a daytime fair-density context (Figure 5), the results confirm the behaviors observed during the nighttime scenario. The main difference is that *MIFProba* offers a much better dissemination rate (220/300 = 0.73 transfers/device) than in low-density area (0.57 transfers/device).

Admittedly our evaluations are only in their early stages. Yet the preliminary results we obtained demonstrate that, by taking into account the semantic – popularity and publication location – of data within feeds, mobile syndication enables to filter content efficiently.

VII. Related work

Research in mobile content communication generally considers either stable wired networks, or fully connected

MANETs. [10], [11], and [12] propose to construct and maintain content-based routing structures in order to forward messages efficiently between publishers and subscribers. The underlying routing structure for mobile nodes needs to be frequently changed, inducing high maintenance costs and reduced reliability while restructuring.

The work presented in this paper is more related to opportunistic mobile data dissemination. Mobile users are potentially interested in every data in the feeds they subscribe to, depending on the current device position. Opportunistic mobile data dissemination allows portable device users to communicate in a natural and effective way. Opportunistic networks are designed for delay and disruption tolerance, with the assumption that there are no predictable mobility patterns. They take advantage of locality and mobility to multiply information exchange opportunities.

A. Opportunist probabilistic dissemination

The simplest mechanism for data dissemination within a network is broadcasting [13]. Flooding is a simplistic form of broadcasting, in which every node – or all nodes in a localized area – retransmits distinct packets exactly once. Flooding is reliable and reasonably efficient in sparse networks. As the neighbor degree gets higher, flooding may increase resource *broadcast storm problems* [14]: contention, heavy contention and collision. Several mechanisms allow to limit redundant broadcasts until every node receives the information. An intuitive way is to use probabilistic rebroadcasting. On receiving a broadcast message for the first time, a host will rebroadcast it with probability p. This offers better performances than pure flooding but it introduces latency for every flooded message, and some messages are never rebroadcast. Probabilistic approaches make decisions solely on the basis of local information.

There are a number of variants of this basic idea [15], [13]. In counter-based methods, a counter is used to keep track of the number of times the broadcast message is received. A node rebroadcasts a message if it hasn't been received from a minimum number of neighbors. In the distance-based scheme, each device determines the additional coverage it can offer to the original source of the message. The estimation of the distance between mobiles is either based on the strength of the radio signal or on positioning devices such as GPS.

Location-based probabilistic rebroadcasting shares our objective of disseminating data within a predetermined geographic area. [16], [4] and [3] keep data at physical locations within an anchor area. Nodes cooperate to maintain the locality of data using collaborative caching techniques. Instead of storing data on a particular node, it is replicated to whatever nodes are currently near the data home location. The goal is to maintain data as close as possible to the location where it was generated so that it can easily be found later on. Data gets replicated across and deleted from nodes with a probability depending on the distance from the publication location. Data is stored in cache as long as possible; however, when there is no available storage space left, the data that is farthest away from its home location is dropped from the user device. Beyond its replication zone, data gets deleted either during cache replacement or when its TTL expires.

In addition to an anchor zone around the publication location, GEMS keeps replicas within a feed zone specified at creation time. Data replication does not only take into account the distance from the publication location, but also the popularity of the data within the application.

B. Opportunistic content-based dissemination

[6], [5] and [17] perform a content discovery phase before replicating data from one node to another when a contact occcurs. Nodes exchange their local data indexes, and then replicate singletons.

A simple approach is to associate data with compact identifiers and share complete lists of identifiers. For instance, Hycast [6] exchanges lists containing hash codes of the known podcast URIs. This list is considerably smaller than a list containing the complete podcast feeds. Such an approach is well suited for small datasets but does not scale in high data-throughput applications.

In order to reduce network usage, content-matching is often based on Bloom-filter comparison. BDP [18] uses Bloom filters as compact storage for the data version information, thus efficiently identifying version differences among data items with the same key. Publish/subscribe systems[5][17] built on top of the BDP protocol exchange bloom filter hash indexes populated with the identifiers of available feeds and their entries at mobile publishers.

GEMS matches feeds by exchanging lists of feed identifiers. Since feeds are associated with geographic areas, the number of active feeds remains small at a node location. We also use bloom filters to perform news matching within an active feed. We use caching to limit the publication window, and by extension the probability of a collision within bloom filters. Eventually, the most relevant data within the feed zone remains in the cache, and is hence available for publication.

VIII. CONCLUSION AND FUTURE WORKS

This paper proposes GEMS, an architecture for mobile geocentric feed syndication. GEMS associates shared feeds with geographic areas, and thus promotes local communities of interest amongst mobile device users. News feeds are replicated within a $feed\ zone$ among mobile devices by means of opportunistic communications; priority is given to the most popular news items. Eventually, the most relevant data survives long enough to get published on the web. Since mobile devices have limited storage capacities, news are kept within a cache and may be deleted independently by each device. GEMS enforces a cache management policy as a natural filter for published news: as less popular news are deleted from node caches, mobile devices eventually collect the most popular news within a feed zone, as demonstrated by our first simulations.

Our current efforts for the improvement of this work include a complete performance assessment. Although our first simulation results are promising, they need to be extended in order to validate our approach. In particular, we want to examine the impact of the cache size and of the density of mobile devices, and to experiment on dynamic workloads. In parallel, we are implementing the GEMS system on Android for demonstration purposes, and will use it to test its efficiency with a live application on campus.

Finally, our solution raises two scientific issues we wish to address. Firstly, any mobile user can artificially increase the popularity of a news item by adding fake comments. We intend to work on a reputation system on top of GEMS to prevent such behavior. Secondly, users may wish to find specific types of information before they get published online, yet without travelling to a specific $feed\ zone$. A mobile ad hoc search protocol could integrate news popularity as a parameter within its requests.

REFERENCES

[1] M. Hilbert and P. López, "The world's technological capacity to store, communicate, and compute information," *Science*, vol. 332, no. 6025, pp. 60–65, 2011.

[2] C. Aquino, "Social networking on-the-go: U.s. mobile social media audience grows 37 percent in the past year," *ComScore Press Release*, October 2011.

[3] J. Ott, E. Hyytia, P. Lassila, T. Vaegs, and J. Kangasharju, "Floating content: Information sharing in urban areas," in *Proc. IEEE 9th International Conference on Pervasive Computing and Communications (PerCom'11)*, 2011, pp. 136 – 146.

[4] R. C. Nathanael Thompson and R. Kravets, "Locus: A location-based data overlay for dtns," in *Proc. ACM 5th Workshop on Challenged networks (CHANTS'10)*, 2010, pp. 47–54.

[5] V. Lenders, M. May, G. Karlsson, and C. Wacha, "Wireless ad hoc podcasting," *Mobile Computing and Communications Review*, vol. 12, no. 1, pp. 65–67, 2008.

[6] M. B. A. Andronache and S. Rothkugel, "Hycast-podcast discovery in mobile networks," in *Proc. ACM 3th Workshop on Wireless multimedia networking and performance modeling (WMuNeP'07)*, 2007, pp. 27–34.

[7] C. K. Fung, P. C. K. Hung, W. M. Kearns, and S. A. Uczekaj, "Dynamic workflow generation with interoperable security alerts in manet," in *Proc. IEEE Int. Conf. on Services Computing (SCC'07)*, 2007, pp. 418–426.

[8] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, no. 7, pp. 558–565, Jul. 1978.

[9] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *Proc. of ICST SIMUTools*, 2009.

[10] P. Costa and G. P. Picco, "Semi-probabilistic content-based publish-subscribe," in *Proc. IEEE 25th Int. Conf. on Distributed Computing Systems (ICDCS'05)*, 2005, pp. 575–585.

[11] M. Petrovic, V. Muthusamy, and H. arno Jacobsen, "Content-based routing in mobile ad hoc networks," in *Proc. IEEE 2nd Int. Conf. on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'05)*, 2005, pp. 45–55.

[12] F. Guidec and Y. Maheo, "Opportunistic content-based dissemination in disconnected mobile ad hoc networks," in *Proc. IEEE Int. Conf. on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM'07)*, 2007, pp. 49–54.

[13] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *Proc. ACM 3rd Int. Symp. on Mobile ad hoc networking & computing MobiHoc (MobiHoc'02)*, 2002, pp. 194–205.

[14] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proc. ACM/IEEE 15th Int. Conf. on Mobile Computing and networking (MobiCom'99)*, 1999, pp. 151–162.

[15] S. Pleisch, M. Balakrishnan, K. Birman, and R. van Renesse, "Mistral: efficient flooding in mobile ad-hoc networks," in *Proc. of the ACM int. symp. on Mobile ad hoc networking and computing*, 2006, pp. 1–12.

[16] A. A. V. Castro, G. D. M. Serugendo, and D. Konstantas, "Hovering information - self-organising information that finds its own storage." in *In proc. of IEEE SUTC*, 2008, pp. 193–200.

[17] . R. Helgason, E. A. Yavuz, S. T. Kouyoumdjieva, L. Pajevic, and G. Karlsson, "A mobile peer-to-peer system for opportunistic content-centric networking." in *Proc. of ACM MobiHeld*, 2010, pp. 21–26.

[18] T. Chen, D. Guo, X. Liu, H. Chen, X. Luo, and J. Liu, "Bdp: A bloom filters based dissemination protocol in wireless sensor networks," in *Proc. IEEE 6th Int. Conf. on Mobile Adhoc and Sensor Systems (MASS'09)*, 2009, pp. 593–602.