Interest-Awareness in Data Dissemination for Opportunistic Networks

Radu-Ioan Ciobanu[a], Radu-Corneliu Marin[a], Ciprian Dobre[a,*], Valentin Cristea[a]

[a]*Faculty of Automatic Control and Computers*
University Politehnica of Bucharest
313 Splaiul Independentei, Bucharest, Romania

Abstract

Generally, data dissemination in opportunistic network uses flooding to ensure that content is spread to interested subscribers. However, this might lead to congestion and high overhead, so alternative solutions are required. In this article, we propose leveraging context information such as node interests, social connections and predictions based on contact history, in order to decrease congestion without affecting the network's hit rate and delivery latency. Thus, we first propose a basic interest-based dissemination algorithm, to show that nodes tend to group together based on interests. Then, we present ONSIDE, an algorithm that also uses other types of context information to select the nodes that act as forwarders. Finally, we propose five heuristics for sorting the messages in a node's memory, and show how each of them affects hit rate, delivery latency and congestion.

Keywords: opportunistic, dissemination, interests, social, prediction

1. Introduction

The emergence and wide-spread of new-generation mobile devices, as well as the increased integration of wireless technologies such as Bluetooth and WiFi, create the premises for new means of communication and interaction,

*Corresponding author

Email addresses: radu.ciobanu@cti.pub.ro (Radu-Ioan Ciobanu), radu.marin@cti.pub.ro (Radu-Corneliu Marin), ciprian.dobre@cs.pub.ro (Ciprian Dobre), valentin.cristea@cs.pub.ro (Valentin Cristea)

challenge the traditional network architectures and are spawning an interest in alternative, ad-hoc networks such as opportunistic networks (ONs). ONs consider node mobility an opportunity to exploit, in environments where human-carried mobile devices act as network nodes and are able to exchange data while in proximity. Node mobility creates contact opportunities among nodes, which can be used to connect parts of the network that are otherwise disconnected. Whenever a destination is not directly accessible, data is opportunistically forwarded between nodes, so the peers can bring the message closer to the destination.

An application scenario for ONs is represented by *data dissemination*. Data dissemination implies that the nodes in a network subscribe to certain channels, which publish data. Whenever data is published by a channel, nodes that are subscribed to it must receive the data. Thus, nodes play the roles of "publisher" and "subscriber" at the same time. While in regular networks there is a central entity that keeps track of which channel each node is subscribed to, since ONs are fully decentralized and composed of highly mobile nodes, the subscriptions of a network participant are only known to itself and to nodes it comes in contact with during the ON's lifetime. Thus, to cope with the locality of information, many dissemination strategies flood the network, by sending all channel data and subscription information to every peer a node comes in contact with. However, previous studies show that this not only leads to very high bandwidth usage, but also potentially to congestion [1]. We expand on this, and propose a method to help alleviate such problems, without affecting the hit rate.

As previous studies show, ON nodes tend to encounter other nodes with common interests with a high rate [2, 3, 4]. We believe this happens because humans generally form social groups (i.e. communities) based on similar tastes or preferences. For example, work colleagues, which are in contact for 8-9 hours a day, have common interests regarding the domain they work in. In the same community, people sharing common interests (such as sports) are more likely to "bond" together. This implies that, for the period when a mobile device-carrying person is at work, the corresponding ON node will interact with other nodes with common interests. However, this situation is not only valid for work hours. After work or in weekends, people meet their friends, and it is very likely that these friends are also interested in the same topics. Moreover, participating in an event such as a music concert or a football match leads to even more encounters with nodes with common interests.

Thus, in this article we begin by proposing a basic interest-based dissemination algorithm, showing that it reduces congestion without affecting hit rate, when compared to flooding and contact history-based methods. Afterward, we present *ONSIDE* (*O*pportu*N*istic *S*ocially-aware and *I*nterest-based *D*iss*E*mination), a dissemination strategy that leverages information about a node's social connections, interests and contact history, in order to decrease network overhead and congestion, while not affecting the network's hit rate and delivery latency. This is done by carefully selecting the nodes that act as forwarders, instead of simply flooding every node. Finally, we show that the order in which messages are dropped, when a node's buffer is full and it needs to download a new message, has an effect on various ON-specific metrics. We propose five heuristics for sorting the messages in a node's memory, and show how each of them affects the output. All our solutions are compared with various existing algorithms, using three ON mobility traces.

2. Related Work

Since data dissemination presumes sending a message to multiple nodes, Epidemic [5] is one of the simplest strategies previously proposed. The algorithm simply floods the network with a message, until it reaches all interested subscribers. When two Epidemic nodes meet, they exchange all the messages they carry. This way, assuming that a node can store an unlimited number of messages in its data memory, the maximum hit rate of the network is guaranteed. However, flooding the network with messages can very easily lead to congestion and high overhead, especially in dense networks with high activity.

Although users' interests haven't been employed by too many dissemination solutions in ONs, routing algorithms that take advantage of this information have been proposed over the years. Moghadam and Schulzrinne's interest-aware algorithm [2] is able to analyze a user's history of cached data in order to obtain his interests. Using these interests, the algorithm is able to decide whether a document carried by an encountered node should be downloaded or not when a contact occurs. When a node A meets a node B, it has to decide which documents should be forwarded to B. This is done by mapping each document carried by A into B's interest space and applying cosine similarity. If the result is higher than a predefined threshold, then that document is transferred to B. Thus, documents are only spread to nodes that are interested in their content, which in turn can forward them further on to nodes with similar interests. By using this algorithm, the number of data exchanges in the network decreases dramatically, since a node can only carry a document that it is interested in. Another algorithm for data forwarding in ONs which is based on the assumption that nodes with common interests tend to meet each other more often than regular nodes, is SANE [3]. Similarly to the previous algorithm, nodes with SANE have their interests defined by profiles represented as m-dimensional vectors in an interest space. When two nodes meet, they each compute cosine similarities between their own interest profiles and the relevance profile of each message in the other node's memory. If the cosine similarity for a message is higher than a download threshold, then that message is downloaded. SANE is slightly different than the previous algorithm because it uses two thresholds: one for deciding whether a node is interested in a message, and one for deciding whether a node should download a message. The first threshold is higher than the second one, and is the equivalent of the threshold from the previous algorithm, where a node would only download a document if it was interested in its content. Thus, SANE might lead to more message exchanges in the network, but also to higher hit rates and lower delivery latencies. However, the disadvantage of these two solutions is that they are very restrictive in terms of message exchanges, since data is forwarded only to interested nodes. Thus, if a node never encounters another node that is interested in a data item it stores, then that item might not reach any interested node. This is why, as we show in Sect. 3, ONSIDE leverages other nodes that are not necessarily interested in a data item, but have a high chance of encountering other nodes that are.

Algorithms that combine interest information with other context knowledge (such as social connections) have also been proposed [6, 7]. One example is ML-SOR [8], which exploits three social network layers: the contact network, the online social network and the interest network. The contact network is the proximity graph created through contacts between devices, while the online social network is extracted from virtual contacts. A tuple of multiple such social network layers is defined as a multi-layer social network. ML-SOR thus extracts social network information from multiple contexts, and analyzes encountered nodes in terms of node centrality, tie strength and link prediction on different social network layers. When an ML-SOR node A encounters a node B, it computes a social metric called MLS for all the messages in B's memory, both from its own standpoint, as well as from B's.

If MLS is higher for node A, then it sends a download request for that particular message. The social metric is computed based on three components: CS, TSS and LPS. CS represents the centrality of the nodes in the contact history graph, while TSS and LPS are computed with regard to the message's destination. TSS is the online social network strength between the analyzed node and the destination, and LPS is a link predictor computed on an interest network layer. It counts the number of common interests between the encountered node and the message's destination.

One dissemination solution for delay-tolerant networks (DTNs) is the User-Centric Data Dissemination method proposed by Gao and Cao [9]. This method considers (similarly to ONSIDE) the social contact patterns and interests of the nodes in the network, by using a social centrality metric. This metric allows a node to probabilistically estimate another node's interest in a certain data item. The centrality value of a node for a data item at a given time is defined as the sum of probabilities that the node can forward the data item to all interested nodes (called "interesters" by the authors) within an allotted time slot. Thus, a node's centrality for a data item indicates the expected number of interesters that the node can forward the data item to during the remaining time of data dissemination. There are two methods proposed by the authors for computing a data item's centrality at a node: a local centrality and a multi-hop centrality. The local centrality is calculated by assuming that the inter-contact time between two nodes i and j follows an exponential distribution with a contact rate λ_{ij}, whereas the multi-hop centrality is based on the shortest path on the network contact graph, where the weight of an edge represents the probability that a data item is opportunistically forwarded on that edge within a given time. The multi-hop centrality is also based on the hypoexponential distribution that represents the total time needed to forward data from the source to an interested node. However, there are several drawbacks that the User-Centric Data Dissemination method has, the first of them being that the authors assume an exponential distribution for inter-contact time, although it has been shown previously in [10] that the distribution of inter-contact time follows an approximate power law over a large time range, which may lead to an incorrect approximation of contact times, and thus to wrong values for the centralities. Moreover, Gao and Cao's solution is difficult to apply in practice, because it assumes a priori knowledge of the contact rate λ_{ij} (and, even if it is computed on-the-fly as contacts occur, it might take a while for a node to collect enough information to consider the obtained value relevant). The User-Centric Data Dissemina-

Table 1: Information about the mobility traces used.

Trace	**Nodes**	**Duration**	**Type**	**Topics**	
				Total	**Per Node**
Infocom 2006	98	4 days	Conference	27	14.53
Sigcomm 2009	76	4 days	Conference	154	15.61
UPB 2012	66	64 days	Academic	5	3.51

tion method also has the disadvantage that it requires a node to store interest information about all the nodes it encounters, since it has to compute the centrality based on all the other nodes' preferences. ONSIDE, as shown in Sect. 3.3, only requires interest information when a contact occurs, and uses it on-the-fly to make a decision, instead of storing it for future use. This way, forwarding decisions are made quicker and more efficiently. As future work, we wish to implement the User-Centric Data Dissemination algorithm in our opportunistic emulator [11] and compare its results to ONSIDE.

ONSIDE's goal is to offer data dissemination in opportunistic networks through a publish/subscribe interface. Thus, it uses context information of various types (such as social knowledge, contact history, nodes' interests) with the purpose of reducing congestion and achieving high hit rates.

3. Enhancing Data Dissemination with Social Interests

In this section, we propose enhancing data dissemination in opportunistic networks by considering a node's interests. We start from the premise that ON nodes are human-carried mobile devices, which means that the behavior of the network depends on human mobility and interaction. This is an advantage, since (as we show below) people with common preferences tend to meet each other more often than nodes with no interests in common.

3.1. Encounters with Common Interests

It has been shown before in the literature that our assumptions are true [2, 3, 4], but we first prove this holds as well for the kind of situations we are interested in (i.e. not only crowded places with lots of interactions, but also sparse ONs, where the nodes don't interact very often). Thus, we begin by analyzing different mobility traces in terms of contacts between nodes with common interests. We chose three traces for this analysis: Infocom 2006 [12], Sigcomm 2009 [13] and UPB 2012 [14]. We looked for datasets

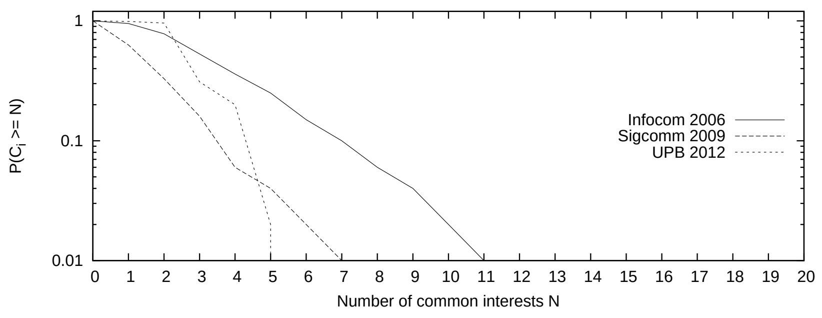


Figure 1: CCDFs of encounters with nodes with common interests (where C_i is the average number of common interests per contact between two encountering nodes).

that offered interest information about the mobile nodes, in order to validate our assumptions and test our algorithms. Table 1 shows an overview of the three traces, including the total number of topics that the participants were able to choose from and the average number of topics a node was interested in. Unfortunately, there aren't many publicly available mobility traces that contain interest information, and the ones we tested with are the most relevant for our needs. There are two separate environments (academic and conference), and the two conference traces differ in terms of total number of interests. The experimental results were obtained by using MobEmu [11], an opportunistic network emulator that is able to replay a trace and apply a desired algorithm when two nodes meet.

Figure 1 shows the average number of contacts between nodes that have at least a given amount of common interests. For the Infocom 2006 trace, only 5% of the total number of contacts are between nodes with no interests in common. Most encounters occur between nodes with two common topics (25%), and the maximum number of similar interests is 15, out of a total of 27 different topics. It can also be seen that, for the Sigcomm 2009 trace, there are more contacts between nodes that have no common topics of interest than for Infocom 2006 (37%), because of the fact that there are far more available topics that the trace participants had to choose from when creating their profile (154, as opposed to 27 for Infocom 2006). Nonetheless, 29% of the contacts are between nodes with one interest in common, while the maximum number of common interests between two encountering nodes is 13. Finally, it can also be seen that, for the UPB 2012 trace (which is sparser than the other two traces and has a much lower number of topics that the participating nodes can be interested in), 99% of encounters are between nodes with at least a common topic of interest. The peak is at two, so 66% of a node's encounters are with other trace participants that it has two interests in common with.

3.2. Basic Interest-Based Dissemination

We begin by presenting a simple method for interest-based dissemination, to show that even such basic algorithms behave better than non-informed ones, and expand on it further on. Since we have shown that ON nodes which have common interests tend to encounter each other at a high rate, we propose taking advantage of this observation by allowing data exchanges only between nodes that have at least a certain number of interests in common. Thus, we define an I interest community as the group of nodes that are interested in I (for example, a "sports" community is composed of devices belonging to people that are sports fans). Thus, a node can belong to any number of communities, since it can have multiple interests. As previously shown, a node has a higher chance of encountering nodes from one of its own interest communities than from other communities.

The interest-based method proposed here, called Interest(N), only allows nodes that belong to at least N common interest communities to exchange messages. Thus, when a node A running Interest(N) encounters another node B, it first checks whether B contains any data that A is interested in. If it does, then A submits a download request for that data. For the next step, A computes the number of communities that both B and itself are a part of, and if that number is greater than a given threshold N, then A submits another download request for all the remaining data in B's memory. This way, data transfers are only performed between nodes with common interests, based on the previous assumption that these nodes encounter each other often and are thus able to successfully deliver channel data to all subscribed nodes. Thus, Interest(N) only disseminates data between the members of a community (i.e. nodes with common interests), unless that data is of direct interest to non-community nodes (in which case it is downloaded anyway).

3.3. ONSIDE: Socially-Aware and Interest-Based Dissemination

We present our improved ON data dissemination strategy, ONSIDE [15]. The algorithm combines the use of interest knowledge for dissemination, with social information about the nodes in the network and contact history, in an attempt to decrease an ON's overall bandwidth consumption and reduce its congestion, while not affecting the average channel hit rate and the delivery latency. It is based on several assumptions, one of them being that nodes which have common interests tend to meet each other more often than nodes that do not, as shown above. The second assumption that ONSIDE is based on states that connections from online social networks (such as Facebook) are respected in an ON node's encounters. We have shown in [11] that a node encounters other socially-connected nodes with a high probability. Not only is this true, but there is also a high chance that a node encounters a second-degree neighbor (i.e. a node with at least one friend in common).

When two nodes running ONSIDE meet, they exchange lists of messages in their data memory (characterized by unique IDs and channel information) and lists of topics of interest. Based on this information, each node analyzes the other node's messages and decides which of them should be downloaded. It then sends a download request for those messages, and starts downloading them one by one until it finishes, or until the two nodes are no longer in contact. The function used by a node A to analyze a message M from a node B and to decide whether it should be downloaded is:

$$\begin{aligned} exchange(A, B, M) = (&common\_interests(A, B) \geq 1) \\ &\wedge (interested(A, M.topic) \\ &\vee (interested\_friends(A, M.topic) \geq thr_f) \\ &\vee (interests\_encountered(A, M.topic) \geq thr_i)) \end{aligned} \tag{1}$$

The result of the *exchange* function is a boolean value that specifies whether a download request should be made to B for message M. The *common_interests* function returns the number of topics that both A and B are interested in. This way, data transfers are only performed between nodes with at least one common interest, based on the previous assumption that these nodes encounter each other often and are thus able to successfully deliver channel data to all subscribed nodes. The second component of the *exchange* function is *interested*, which returns *true* if node A is subscribed to the channel that generated message M (i.e. if it is interested in M's topic). By using this function, a node will download a message and keep it in its cache, even though it might not need it after a while. It will store it for other nodes, since it is highly likely to encounter nodes that have similar interests to its own (as previously shown). The $interested\_friends$ function returns the number of online social network friends of node A that are subscribed to the channel that generated M. By using this component, we assume that a node has access to its social network at any time, which contains at least interest information about the connected nodes. This component of the *exchange* function has the role of speeding up the message's delivery, by requesting a message if a node's social network friends (i.e. nodes that it has a high chance of encountering) are also interested in it. thr_f is a threshold that can be varied according to the density of the ON and of the social network. $interests\_encountered$ is computed based on node A's history of encounters. It returns the percentage of encounters with nodes that are interested in messages similar to M, in terms of channel subscriptions. This function is based on the assumption that a node's behavior in an ON is predictable (as shown in [16]), so that if it encountered many nodes subscribed to a certain channel, it is likely to encounter others in the future as well. thr_i is a threshold between 0 and 1 that can be varied depending on the number of channels in the ON.

3.4. Improving ONSIDE with Message-Sorting Heuristics

One of the main disadvantages of ONSIDE (and ON routing or dissemination algorithms in general) is that they are applied on nodes that have limited data memory. Even with GB-range memory, if said nodes belong to a dense network with many contacts and messages sent, they run the risk of congestion, especially if the routing or dissemination algorithm is not sufficiently informed to avoid congested nodes. Therefore, when a node has a full data memory and another node wants to send it a message for relaying, either it won't be able to accept that message, or it will have to drop an existing message from its cache. Most routing and dissemination algorithms in the literature make no mention of how the decision of which messages to drop is made, so the assumption is that generally the oldest message in terms of received time is dropped when the memory is full. This isn't necessarily optimal, since the node might be a good forwarder for that message. For example, if the node belongs to a certain social community and the oldest cache message is of great interest to other nodes from the community, there is a great chance that the carrier will encounter community members which are interested in the message. If the message is dropped, then it might not be delivered to the other community members, or they might receive it from another node, but with a much greater delay. For this reason, we propose here a set of five message-sorting heuristics that can be employed in ONSIDE, and show how they affect the overall behavior of the algorithm in Sect. 5.

The first two heuristics are based on a message's timestamp, i.e. the time a message has been generated. The *increasing timestamp (IT)* heuristic sorts the messages in a node's memory from lowest to highest timestamp, i.e. from the oldest to the newest message. When a node must remove a message from memory, it does so from the start of the message list, so the first message that gets removed is the oldest one. This is based on the assumption that, if a message has been generated a long time ago, it may have become irrelevant. Moreover, the message might have been delivered to its intended recipients in the meantime, so keeping it in memory might occupy space with no purpose. On the other hand, the *decreasing timestamp (DT)* heuristic assumes that, if a message is old, it must be delivered as soon as possible in order to avoid high delays, so the newer messages are the ones that have to be dropped first.

The third heuristic is called *topic interest (TI)* and is based on the previous assumption that ON nodes tend to encounter other nodes with similar interests. Therefore, this heuristic sorts messages according to whether the carrier node is interested in the topic of the message or not. Messages that the node is not interested in are at the head of the list, so they are dropped first whenever the memory is full and a new message must be stored. This way, messages of interest to the node's interest community are kept in memory, because there is a high chance that the node will deliver them to at least one other node, as shown previously.

The fourth heuristic is not only based on the current node's interests, but also on the interests of other nodes it is socially-connected with (i.e. friends on social networks). Thus, the *interested friends (IF)* heuristic sorts the messages in a node's memory by the number of the carrier node's friends that are interested in them. If a node has a friend that has some different interests from it, this heuristic would help keep messages of interest to the friend in memory, to be forwarded when there is an encounter. The more friends are interested in a message's topic, the more important that message is, since it has the chance of being delivered to more nodes.

Finally, the *interests history (IH)* heuristic is based on the assumption that, if a node has many encounters with nodes interested in a specific topic, then messages with that topic are more important and should not be deleted from the memory. Therefore, a node stores a history of past contacts in terms of interests (i.e. it counts the number of times each interest is "met"), and messages are sorted based on the percentage of encounters with nodes interested in their topics, from lowest to highest.

3.5. A Use Case Scenario and Potential Limitations

The algorithms that we have proposed above can be used in an opportunistic interest-based publish/subscribe framework. For example, let's assume that two friends carrying mobile devices meet in a bar for a drink. One of them is a football fan, and will go to a match after the two friends separate, while the other one likes rock music and is preparing to go to a concert of his favorite band. In the bar they meet in, there are many other people with mobile devices, each of them having different tastes. In order to have an efficient data dissemination using only the mobile devices present in the bar, each of them must be able to correctly detect the context of its user (such as preferences, behavior patterns, position, etc.), in order to infer the interests for which that user is a good forwarder. Consequently, the mobile device belonging to the friend that likes football should start collecting all football-related information from the devices nearby (belonging to the other people in the bar). Similarly, the rock music fan's phone should gather all rock-related data. When the two friends separate, the football fan goes to the match, where all the information his device has collected will be useful for a large number of people (since probably all the people in the stadium are football fans). Something similar should happen for the rock fan.

The scenario proposed above would be useful in an ON with many interests, where epidemic methods would require large storage capabilities for devices and fast connections. In such a situation, nodes may end up carrying a lot of useless data (from their and their connected nodes' point of view). Moreover, if contacts are not extremely long, nodes might not get the chance to exchange all data when a contact occurs, so the messages they exchange might not be relevant. This is where ONSIDE aims to improve data dissemination, since nodes will only download messages of relevance for them and their social and interest connections. However, there are some limitations for our method if the interests are not defined properly. For example, if the interests are very broad and their number is low, congestion might still appear, as shown in Sect. 5 for the UPB 2012 trace. This happens because, as stated before, there are many messages tagged with certain interests, and nodes will end up having to exchange a lot of the data at a contact. The opposite problem may also appear, where the interests might be too specific, and thus nodes might only specialize in forwarding a small subset of the interest space, thus leading to lower hit rates. In Sect. 5, we show the results of applying ONSIDE to the three traces presented in Table 1, where the total number of interests and the number of interests per node vary.

4. Theoretical Analysis

For our approach, we assume a highly dynamic mobile network, so structures such as multicast trees or broker-based publish/subscribe overlays can't be exploited. As such, the forwarding decision is based on a gossiping-like approach. Nodes advertise topics their users are subscribed to, upon making contact with other nodes. With this, a node uses a utility function when it encounters another node, as shown in Equation 1, for each data object either stored locally or in the peer's cache. This function has the role of identifying the set of data objects that can be stored in the local cache (i.e. that fit in the cache, or that fulfil similar other constraints [17, 18]). Moreover, each relay only has a limited buffer space. Thus, according to these assumptions, there is a function p_e that represents the probability of an object being exchanged between two nodes (i.e. the probability that $exchange(A, B, M)$ is 1). Consequently, the forwarding decision in our analysis can be modeled similarly to solving a multi-constraint knapsack problem (MKP), to identify the optimal set of data objects that do not violate a set of constraints (i.e. don't exceed the cache's size, and prioritize the probability of successful delivery). Formally, the MKP has the following form:

$$\max \sum_k U_k x_k, s.t. \sum_k c_{jk} x_k \leq 1, j = 1, \ldots, m, x_k \in \{0, 1\}, \forall k \tag{2}$$

In Equation 2, k denotes the k-th object that the node can select, U_k its utility (i.e. probability of successful delivery if forwarded through the encountered node), c_{jk} the consumption of resource j related to fetching and storing object k (normalized to the maximum allowed consumption of that resource, i.e. $0 \leq c_{jk} \leq 1$ holds true), m the number of constrained resources (i.e. memory size), and $x_k \in \{0, 1\}$ the MKP's variables.

We introduce the exchange probability function, p_e, to be modeled as a product between the access probability to the data object (p_{ac}) by a measure of the delivery cost (p_{av}), normalized by the object's size (s). The rationale of this definition is that the utility of selecting one particular object (i.e. $exchange(A, B, M)$ is 1) should be high if only if the node has a high chance of successful delivery (i.e. the cost of objects should be inversely proportional to the probability of their successful delivery). Normalizing by the size is just an approximation of the knapsack problem (Equation 2). With this, p_e models the probability of selecting a particular data object when two nodes meet.

In the network, users of any node cache are either data consumers (a node carries the data because it has an interest in it), or carriers/mules for other nodes (the node carries the data for other encountered nodes). Thus, p_e has two components: one for the local user ($u^{(l)}$, modeling the probability of the local user being interested in the message) and one for any social community the local user is in contact with ($u^{(s)}$, modeling the probability of the node carrying the message for other users in the network). Formally, this is expressed as:

$$p_e(exchange = 1) = u^{(l)} + u^{(s)} = p_{ac}^{(l)} p_{av}^{(l)} + p_{ac}^{(s)} p_{av}^{(s)} \tag{3}$$

In Equation 3, $p_{ac}^{(l)}$ represents the probability that the local user is interested in a data object (corresponding to the $interested(A, M.topic)$ component from Equation 1), $p_{ac}^{(s)}$ represents the probability that any encountered node is interested in a data object (related to $interests\_encountered$ in Equation 1), $p_{av}^{(l)}$ represents the probability that the local node "sees" a data object in the caches of any encountered node ($common\_interests$ in Equation 1), and $p_{av}^{(s)}$ represents the average probability over encountered nodes that those nodes "see" a data object in the cache of any other node they encounter (i.e. the popularity of a particular topic, which generally follows a normal distribution, and is given by the $interested\_friends$ component in Equation 1).

With this construction, we can model data dissemination according to a Markov process. A similar simplified analysis is presented in [19], where authors assume a user subscribed to a particular interest receives all the data objects for that topic, and that availability indexes are aggregated over all objects disseminated over the same topic. Thus, the utility of all the data objects for the same topic, computed by a given node, is the same. With this, our proposed exchange function maps over Equation 2 in [19]: $utility = u^{(l)} + u^{(s)}$, with the difference that in [19] authors model utility as a monotonically decreasing function of object's availability in the network, while in Equation 3 the utility depends on the capacity of the node of maximizing the chances of successful delivery, if selecting the next node. The modeling effect is similar (see [9], and the details below). Consequently, according to Theorem 1 in [19], the data dissemination process always reaches one of two possible stationary regimes: either nodes store only data objects for the topics they are subscribed to (the *greedy behavior*), or nodes store data objects for other topics (the *oscillating regime*). Details on the Theorem (i.e. demonstration and conditions) are in [19].

We consider, similarly to [20], pairwise node inter-contact times as being exponentially distributed. The contacts between two nodes i and j then form a homogeneous Poisson process (we validated this assumption in [21]) with the contact rate λ_{ij}, which we are able to approximate using the online (computationally feasible) algorithm proposed in [16].

If a user subscribes to a topic j, then the local access probability is $p_{ac}^{(l)} = \mathbf{1}_{\{j\}}(i)$, where $\mathbf{1}_{\{\cdot\}}(\cdot)$ is the standard indicator function. The social access probability, in this case, is the probability of meeting a node subscribed to topic i. If we assume nodes compute exactly p_{av} parameters, then in Equation 3 both $p_{av}^{(l)}$ and $p_{av}^{(s)}$ become equal to $\frac{n_i}{M}$, where n_i is the number of nodes considering topic i useful (either they are subscribers, or they can disseminate to subscribers), and M is the number of nodes in the system. Therefore, the utility computed by a node subscribed to j for the topic i becomes:

$$U_{ij} = (\mathbf{1}_{\{j\}}(i) + z_i) \cdot p_{e_{\forall k}}(k, j, i) \cdot e^{-\lambda \frac{n_i}{M}} \tag{4}$$

We assume that the probability that any given node can subscribe to topic i (denoted by z_i) and all terms in Equation 3 consist of independent variables following a stochastic process with mean $\frac{1}{\lambda}$ (assuming the Poisson inter-arrival process). With this, combining Equation 3 and Equation 4, the utility function is strictly monotonically decreasing with an object's availability (p_{av}), and according to Theorem 1 in [19], it follows that there are boundary regions for the parameter space describing the network where the data dissemination process converges to a stationary regime (either greedy or oscillating). ■

We can now generalize to the case of relaying data when nodes have limited buffer space. In this case, the proposed algorithm can be decomposed into two stages: (i) *relay selection* for each data to maximize the dissemination utility function, and (ii) *data item selection* on a relay if its buffer is not enough to carry all the data items. For this case, our analysis resembles the one presented in [9]. This stands because what the authors define as the *centrality value* of a node i for some data item d_k is well approximated by the probability function described in Equation 3, i.e. the expected number of subscribers that node i can disseminate the data to, and what authors in [9] define as *multi-hop centrality* can be well approximated by Equation 4. In this case, the *exchange* function selects a node i by a node j for the data item d_k, iff $p_e(i, j, d_k) \geq \frac{N_I^k}{N_R^k}$, where N_R^k is the number of relays currently selected for data d_k, and N_I^k is the estimation of the number of subscribers that will receive d_k. The demonstration is straightforward: (i) if node i is a subscriber, then by selecting it we have one more subscriber being served, (ii) otherwise, if node i is selected, then it either has friends that it can serve data d_k to ($interested\_friends \geq thr_f$), or its circle of visited nodes includes subscribers to the data item's topic ($interests\_encountered \geq thr_i$), so N_I^k most likely increases with at least one (node i most likely will be able to deliver to at least one other subscriber). Thus, the behavior of the utility function used in the relay selection (phase (i)) is similar to the centrality function assumed [9].

For the second phase (data item selection), the heuristics proposed in Sect. 3.4 are designed to select a data item only if its contribution to data dissemination increases with the selection. Thus, the selection process can be formulated as a knapsack problem (see Equation 2). Data with higher popularity is preferred, but the preference diminishes when the number of relays for the same data items increases (we want to ensure that data items with lower popularity are also fairly disseminated, when the popular data items have already been carried by a number of relays). With this, all hypotheses from [9] are met, and following the analysis there, we can conclude that:

- *At any given time t, the number of potentially reached subscribers is higher than the number of relays.* This follows Lemma 2 in [9], where $\frac{N_I(t)}{N_R(t)} \geq (1 - e^{-s_G(T-t)}) \cdot p_{min}$, where $N_I(t)$ means N_I at time t, s_G is a parameter depending on the inter-contact time distribution and the actual size of the network, p_{min} depends on the probability function of nodes being actually interested in various topics, and T is a maximum time predicted for data to traverse the mobile network (proof in [9]).

- *The probability for the dissemination function to increase its utility after some time period is upper bounded.* This follows naturally from the demonstration of Theorem 1 in [9]. Also, the following implication applies: the utility function for disseminating a data item is proportional to the contact capability of relays and to the data popularity (proof in [9]). $\blacksquare$

5. Experimental Setup and Results

This section presents the experimental results obtained when running the solutions proposed in Sect. 3 on various mobility traces.

5.1. The Interest Algorithm

Since we are dealing with dissemination in opportunistic networks, data is generated through channels that nodes are able to subscribe to. When a node is subscribed to a channel, it is interested in any data generated by that channel that it hasn't received yet. Because we have interest information in the three traces used for testing, we consider that a channel is represented by a topic. Therefore, there are 27 channels for Infocom 2006, 154 for Sigcomm 2009 and 5 for UPB 2012 (as shown in Table 1). Every node that is interested in a certain topic can generate information on the corresponding channel, but not on channels that don't match its interests. The generation of messages is modeled as closely as possible to real-life situations, so each node that has at least one interest generates 30 messages per day in the two-hour interval when the most contacts happen. A node that is interested in multiple topics is able to generate data for each of the corresponding channels, so the channel for each of the 30 messages generated by a node per day is chosen randomly out of its interests. For now, we assume that a node's data memory is unlimited, and that nodes are able to exchange all the data they carry in a single contact.

Regarding the metrics we analyze, we first look at hit rate, which is the ratio between the number of messages that have successfully arrived at nodes subscribed to the corresponding channels, and the total number of messages generated, each of them multiplied by the number of subscribers to the channel. The second metric is delivery cost, defined as the ratio between the total number of messages exchanged, and the number of generated messages multiplied by the number of corresponding channel subscribers. We are also interested in hop count, which is computed as the number of nodes that carried a message until it reached an interested destination on the shortest path. Another metric that we use in our analysis is fairness, which shows whether each channel is treated equally in terms of message routing. Similarly to [22], fairness is computed according to Jain's fairness index, using the hit rate as the measure of the service level obtained by each channel. Our goal, when compared to Epidemic (which has a very high fairness level, since it is able to obtain the maximum hit rate), is to affect the fairness as little as possible.

We also want to show that splitting nodes into communities based on topics of interest offers better overall results than splitting them into communities based on the time they spend in contact with each other. Therefore, aside from Epidemic, we also compare our results to a community-based dissemination algorithm. This algorithm is based on the idea that each community has a leader (generally the node with the highest centrality), which intermediates

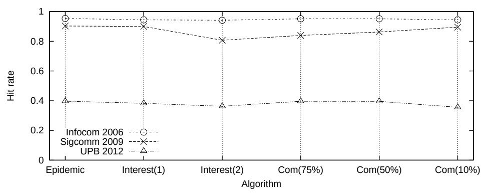
Table 2: Average community sizes for three k-CLIQUE versions.

Trace	**Com(10%)**	**Com(50%)**	**Com(75%)**
Infocom 2006	8.19	56.89	69.40
Sigcomm 2009	8.86	34.41	54.17
UPB 2012	6.03	18.66	22.12

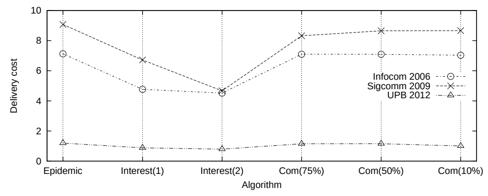
the communication between disjoint communities. The community-based algorithm we compare Interest with, simply entitled Community, functions similarly to the Socio-Aware Overlay [23] and BUBBLE Rap [12]. When two nodes A and B running the algorithm meet, each of them verifies if it is a leader in its own community, by computing its local centrality (using the S-window algorithm) and comparing it with the centrality of other nodes in its own community that it has recently encountered. The node with the highest centrality in each community is its leader. For community detection, the k-CLIQUE algorithm [24] is used. If node A is not a leader, it only epidemically exchanges data with B if both nodes belong to the same community. If A is a leader, it exchanges data with node B if they are both in the same community or if node B is also a leader in its own community. This way, leader nodes act as the gateways between communities, with the bulk of data dissemination being performed inside a community. However, when A encounters B and B carries data that A is interested in, that data will be transferred to A, regardless of whether the two nodes belong to the same community or not. This algorithm is based on the idea that nodes belonging to the same community meet each other very often. However, we attempt to prove that such a solution is not necessarily the best one, and that forming communities based on interests rather than time spent in contact is more useful in terms of data dissemination.

For the Community algorithm, we test with various k-CLIQUE parameters, to analyze how the average size of a community affects dissemination. Therefore, for each of the three mobility traces, we split into communities whose sizes are in average 75%, 50% and 10% of the total number of nodes in the trace. For UPB 2012, which is sparser and where contacts are rarer, the maximum size of a community obtained by k-CLIQUE is lower than 75%. The average community sizes for the three k-CLIQUE versions are shown in Table 2.

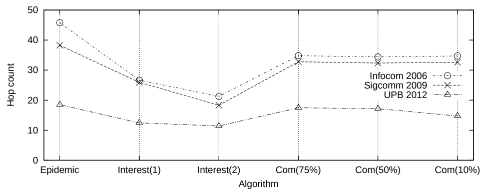
Figure 2(a) shows the hit rate that was obtained when running Epidemic, Interest(1), Interest(2), Community(10%), Community(50%) and Commu-



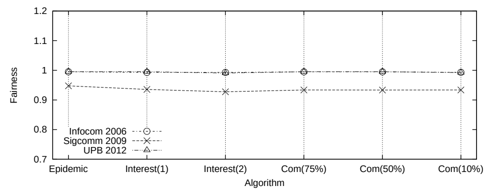
(a) Hit rate



(b) Delivery cost



(c) Hop count



(d) Fairness

Figure 2: Results for the Interest algorithm.

nity(75%) for the three traces specified in Sect. 3. Since Epidemic assumes that the nodes in an ON have unlimited memory and bandwidth and are thus able to exchange all the stored messages when an encounter occurs, the hit rate obtained by the algorithm is the maximum hit rate that can be achieved in the network. Therefore, we compare with Epidemic in order to have a superior limit to the hit rate, and to show that the Interest algorithm not only improves congestion and bandwidth usage, but that it doesn't affect the network's hit rate much. This is clear from Fig. 2(a) for all three traces: for Infocom 2006, the hit rate drops with 0.92% for Interest(1) and 1.20% for Interest(2), compared to Epidemic; for Sigcomm 2009, it decreases by 0.35% and 9.60%; finally, running Interest on UPB 2012 only decreases the network's hit rate by 1.42% and 3.44%, respectively. Using a community-based algorithm also leads to the hit rate decreasing. For Infocom 2006 and UPB 2012, the smaller the community, the lower the hit rate is. However, for the Sigcomm 2009 trace, when we decrease the number of nodes in a community, the hit rate actually increases. This most likely happens because there are far more available topics of interest in this trace than in the others, and the chance of two nodes that belong to the same k-CLIQUE community being interested in the same topics is low. Therefore, when the community size is lower, there are more leaders in the network, which can move data to different communities. This result proves that organizing nodes into communities based on the number of encounters between two nodes and the time they spent being in contact is not necessarily a good approximation of a node's interest "neighbors", and that in publish/subscribe systems new ways of organizing nodes based on common interests should be employed.

The delivery cost and hop count can be seen in Fig. 2(b) and Fig. 2(c). For all three mobility traces, both Interest(1) and Interest(2) behave better than not only Epidemic, but also than Community. The delivery cost is higher for Community because, if the nodes in a community do not share many interests in common, they will keep on exchanging messages between each other although few of them are interested in those messages, until they reach the leader, which is the only node from the community that can disseminate the messages towards nodes that are actually interested in their content. The same reason leads to higher hop counts for Epidemic, because messages bounce from uninterested node to uninterested node, until they exit the community through the leader. The improvements brought by Interest are significant in terms of network and node congestion, as well as bandwidth used. For example, for the Sigcomm 2009 trace, the delivery cost drops from 9.08 for Epidemic to 4.67 for Interest(2). Since there are 25,998 total data items generated in the network (value obtained by counting the number of interested nodes for each message), the total number of data items transferred during the trace decreases by 114,651. Assuming a data item has 5 MB, this means that using an interest-based algorithm leads to transferring 559 less GB in four days (i.e. the duration of the trace).

We also analyze the fairness of the three algorithms, with regard to a channel's hit rate. Our proposed algorithm should decrease the fairness as little as possible, because all channels must be served as fairly as possible. Figure 2(d) shows that Interest doesn't affect the average channel fairness almost at all. Thus, the fairness level drops by 0.20% for both Interest versions on the Infocom 2006 trace, by 1.21% and 2.01% for Sigcomm 2009, and by 0.01% and 0.45% for UPB 2012. This means that all channels are treated almost equally, which leads to a high satisfaction level for the nodes interested in the corresponding topics.

5.2. ONSIDE

Regarding the way nodes publish data and subscribe to channels, the experiments were done identically to what was shown above for Interest. In order to highlight the benefits of ONSIDE, we compare it to Epidemic. However, since the classic Epidemic algorithm is not entirely feasible in real-life (given that it assumes an unlimited data memory), we also compare ONSIDE to a limited-memory version of Epidemic (Limited Epidemic), that behaves exactly like the original implementation, except that, when the data memory is full and a new message should be downloaded, an existing message must be deleted from memory.

Aside from Epidemic, we compare ONSIDE to a dissemination-modified version of ML-SOR as well. We have chosen this algorithm since it is also interest-based and socially-aware like ONSIDE. The basic ML-SOR algorithm, as described in Sect 2, computes a social metric MLS as a function of three utility scores: CS, TSS and LPS. Both TSS, as well as LPS, are computed based on information about the two encountering nodes and the destination of the current message. However, in data dissemination we can't talk about a message's destination, since there isn't a single destination, and the node that generates the message isn't aware of the channel's subscribers. Therefore, we have modified ML-SOR to compute the social metric using information about the source of the message, instead of the destination. This means that, regarding the TSS component, when a node A encounters a node B, it will forward a message M to it if B is socially connected to M's source on more online social networks than A. This is based on the assumption that a node is more likely to encounter nodes from its social community than regular nodes, and that nodes from the same community share common interests. Similarly, when analyzing the LPS component, a node A will forward a message M to B if B has more interests in common with M's source than A does. This is plausible, since a node is likely to encounter nodes sharing its interests, so B has a better chance of further disseminating M than A does. The third ML-SOR component (CS) remains unmodified, since it is only computed in regard to A and B.

In order to analyze how the various algorithms we test with behave in different conditions, we vary a node's data memory size. Thus, a node is able to store either 20, 100, 500 or 4500 messages at once in its memory. Assuming that a message has an average size of 5 MB, this means that the range of node memory we test with is 100 MB - 22 GB. We consider these to be appropriate values for different kinds of mobile devices, ranging from older smartphones to top-of-the-line devices. For the first set of experiments, for all three memory-limited algorithms we test with (i.e. Limited Epidemic, ML-SOR and ONSIDE), whenever a node decides to download a message but its memory is full, it deletes the oldest message and replaces it with the new one. In the second set of tests, ONSIDE is run using the five heuristics presented in Sect. 3.4. Aside from the metrics previously described, we are also interested in the delivery latency, which is the duration between the time a message was generated and the time it was delivered.

As shown in Sect. 3, ONSIDE has two thresholds: thr_f (for the number of friends interested in a topic) and thr_i (for the percentage of encounters with nodes interested in a certain topic). We compute thr_f by taking into account the average number of friends a node has for each trace. This is because we wish to avoid exchanging too many messages, which can happen when the threshold is low and a node has many friends (because the $interested\_friends$ function will yield a value greater than thr_f with a high probability). On the other hand, we also don't wish the inequality to always be false, so thr_f mustn't be too large in comparison with the number of interested friends. Since we have previously shown that nodes tend to meet other nodes with common interests, as well as nodes they are socially connected with, we use the following formula to approximate the interested friends threshold:

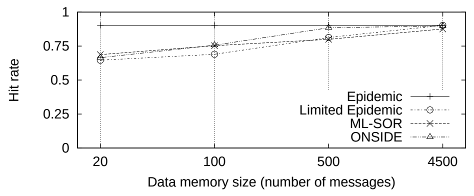
$$thr_f = [\frac{friends}{2}] \tag{5}$$

In Equation 5, $friends$ is the average number of friends for the corresponding trace. Thus, since $friends$ for Sigcomm 2009 is 3.63, thr_f is 1, while for UPB 2012 (where each node has in average 10.49 friends), thr_f is 5. The reason thr_f is higher for UPB 2012 is that, since the trace is taken in an academic environment instead of a conference, most of the participants are socially connected since they are colleagues. Having a low thr_f would lead to more messages being exchanged, and thus the potential of congestion. Finally, since Infocom 2006 doesn't contain information about the social network, thr_f is 0.

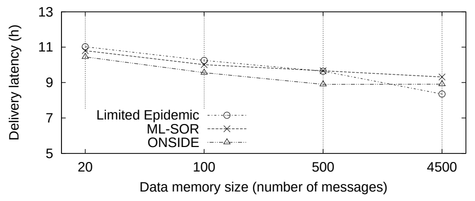
The second threshold (thr_i) depends on the average number of topics per node and the total number of topics available, as shown in Table 1. When the topics per node / total topics ratio is high, the threshold should be lower, since otherwise there would be too many message exchanges. For this reason, we use the following formula for the interested nodes threshold:

$$thr_i = max(0.9 - \frac{per\_node\_topics}{total\_topics}, 0) \tag{6}$$

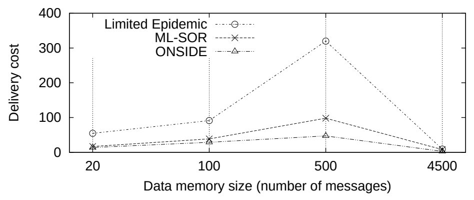
The result of Equation 6 is thus a value between 0 and 0.9 (we have chosen to subtract the ratio from 0.9 instead of 1 in order to avoid having a threshold



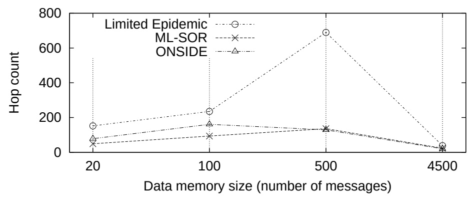
(a) Hit rate



(b) Delivery latency



(c) Delivery cost



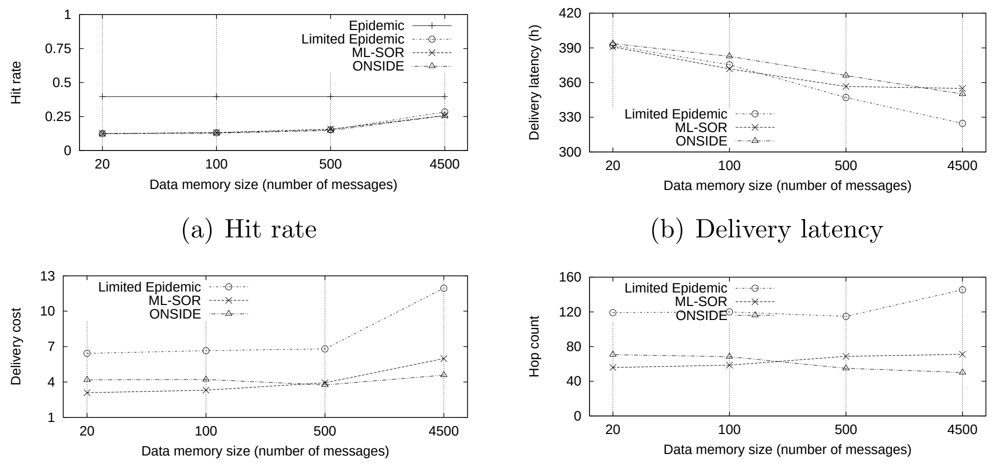
(d) Hop count

Figure 3: Results for ONSIDE on the Sigcomm 2009 trace.

that is too high). For Sigcomm 2009, thr_i is 0.75, for UPB 2012 it is 0.2, and for Infocom 2006 thr_i is 0.3. As can be seen, we have approximated the results of Equation 6 to only two decimals, for simpler computations.

5.2.1. Basic ONSIDE

Figure 3 shows the results obtained by applying Epidemic, Limited Epidemic, ML-SOR and ONSIDE on the Sigcomm 2009 trace. It can be seen in 3(a) that, in terms of hit rate, ONSIDE generally performs better than both ML-SOR, as well as Limited Epidemic (for a 500-message data memory, the ONSIDE hit rate is 7% better than Limited Epidemic's and 8% better than ML-SOR's). For a 4500-message memory, ONSIDE yields a hit rate close to the maximum value, which is obtained when running Epidemic and Limited Epidemic. Limited Epidemic is able to achieve maximum hit rate because the memory size is large enough to fit all the messages generated in the trace, since Sigcomm 2009's duration is only three days. The delivery latency results, presented in 3(b), show similar results: ONSIDE is able to achieve a better delivery latency than ML-SOR regardless of the memory size (with a maximum improvement of up to 1.7 hours), whereas Limited Epidemic only outperforms our algorithm when the data memory is large enough to store all the messages generated in the trace. However, the downside of Epidemic-based algorithms is evident from Fig. 3(c) and Fig. 3(d),

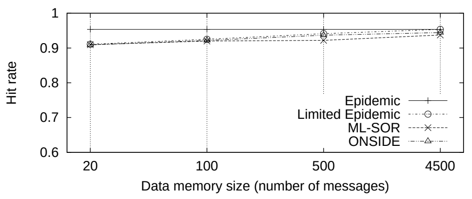


(a) Hit rate
(b) Delivery latency
(c) Delivery cost
(d) Hop count

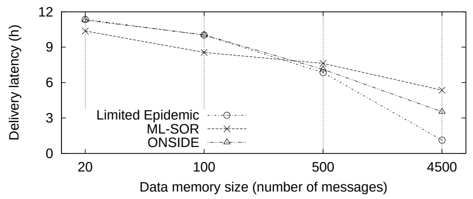
Figure 4: Results for ONSIDE on the UPB 2012 trace.

where it can be seen that the bandwidth used and the network and node congestion are really high. ONSIDE's delivery cost is lower than the ones obtained by Limited Epidemic and ML-SOR for all memory sizes, yielding an improvement of up to 272 compared with Limited Epidemic and 40 when compared to ML-SOR (for a data memory that can store 500 messages). Since there are 25,998 total data items generated in the network, ONSIDE manages to decrease the total number of data items transferred during the trace by 7,086,015 compared to Limited Epidemic and by 1,320,958 compared to ML-SOR. Assuming a data item has 5 MB, this means that using ONSIDE leads to transferring 32 and 12 less TB in three days. Regarding hop count, ONSIDE again clearly outperforms Limited Epidemic for all data memory sizes, but it does yield a higher hop count than ML-SOR for lower data memory sizes (like 20 and 100). This shows that ML-SOR might be a bit more suitable than ONSIDE for older devices that have limited memory, but these days a smartphone's Flash storage is in the order of GBs.

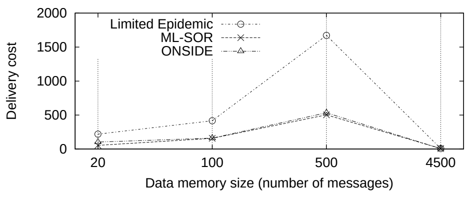
The UPB 2012 results are shown in Fig. 4. Since the duration of the trace is much higher than Sigcomm 2009's, there are a lot more messages generated in the network, so the maximum hit rate is harder to achieve with a limited data memory. This is clear from Fig. 4(a), where even Limited Epidemic doesn't manage to achieve a very high hit rate. The ONSIDE algorithm performs similarly to both Limited Epidemic, as well as ML-SOR.



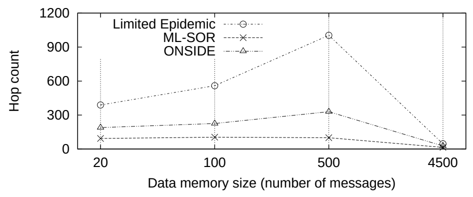
(a) Hit rate



(b) Delivery latency



(c) Delivery cost



(d) Hop count

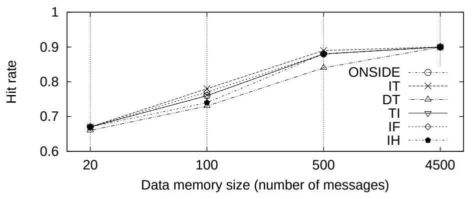
Figure 5: Results for ONSIDE on the Infocom 2006 trace.

Because this trace has a much longer duration that Sigcomm 2009 and the ON is relatively sparse (with few contacts), the delivery latency values are very high. Regardless of this, the values, shown in Fig. 4(b), are similar for all three algorithms, with a slight edge for Limited Epidemic for higher data memory sizes. However, this comes with the cost of increased congestion and overhead, as seen in Fig. 4(c) and Fig. 4(d), where Limited Epidemic performs much worse than ONSIDE and ML-SOR. Regarding delivery cost, ONSIDE performs the best out of all three algorithms for data memory sizes of 500 and 4500, leading to improvements of up to 7.5. For lower memory sizes, ML-SOR performs better, which further cements the statement made previously, that it is better suited for older devices with small memories. The situation regarding hop count is identical: ONSIDE has the better results for memories of 500 and 4500 messages, while ML-SOR is better for a lower memory size. The disadvantage of ONSIDE for UPB 2012 is that there are only five very broad topics of interest reported by the trace. This means that there is a very high chance (99.32%, as shown in Sect. 3) that, when two nodes encounter each other, they have at least one interest in common. This leads to a lot of data exchanges in ONSIDE, which is why the results are not so good. For future work, we wish to apply machine learning techniques to the trace, in order to infer more user interests from mobility and collocation information, as proposed by Noulas et al. [25].

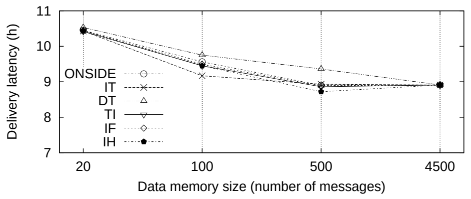
Finally, the results for the Infocom 2006 trace are presented in Fig. 5. Infocom 2006 has the disadvantage of not containing social information about the participating nodes. Therefore, the TSS component from ML-SOR will always be 0. Similarly, ONSIDE doesn't use the $interested\_friends$ component, since thr_f is set to 0. Figure 5(a) shows that ONSIDE yields better hit rates than ML-SOR for all data memory sizes except 20, keeping close to the two Epidemic versions, especially when the data memory size is higher. Regarding delivery latency, shown in Fig. 5(b), ONSIDE fares better than ML-SOR for large memory sizes. Limited Epidemic has the best behavior in terms of latency, but it pays for it with a higher degree of congestion, as seen in Fig. 5(c) and Fig. 5(d). The hop count obtained by Limited Epidemic is extremely large, about three times larger than the one ONSIDE yields. However, ML-SOR performs better in terms of hop count than both the other two solutions. The delivery cost is lower for ML-SOR and ONSIDE, with our solution faring much better when the data memory size is higher.

5.2.2. ONSIDE with Message-Sorting Heuristics

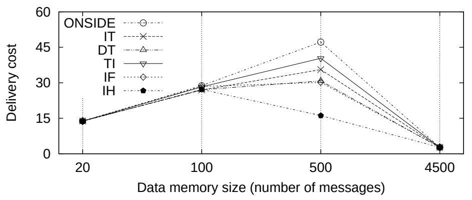
Figure 6 shows the results of running ONSIDE with its five heuristics on the Sigcomm 2009 trace. From Fig. 6(a), it can be seen that the hit rate is barely affected by any of the heuristics. The best improvement (2.61%) over basic ONSIDE is brought by IT, for a data memory of 100. For higher data memories, ONSIDE is already very close to the maximum hit rate achievable, so the results are equal regardless of heuristic. It can also be seen that the heuristic which performs the worst is DT. This means that, for this trace, dropping messages that were generated recently is not a good idea, since there is the risk of losing them completely, especially if they haven't had a chance to be relayed to other forwarding nodes. This way, messages may completely disappear from the network, so the hit rate becomes lower as a result. Regarding delivery latency, it can be seen in Fig. 6(b) that varying the message-sorting heuristic doesn't affect the overall delivery latency of the network too much. The best-performing heuristics are IT and IH, but they only manage to reduce the latency by less than an hour. However, when discussing delivery cost and hop count, as shown in Fig. 6(c) and Fig. 6(d), the situation is different, since employing a message-sorting heuristic has a great effect on these two metrics. Firstly, it can be seen that there is no heuristic that doesn't outperform basic ONSIDE in terms of delivery cost and hop count, and this is especially clear for a data memory of 500. This happens because, for lower data memory sizes, there are many overflows,



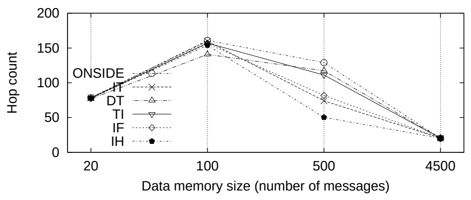
(a) Hit rate



(b) Delivery latency



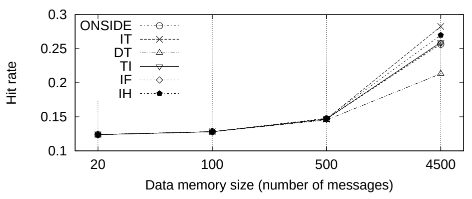
(c) Delivery cost



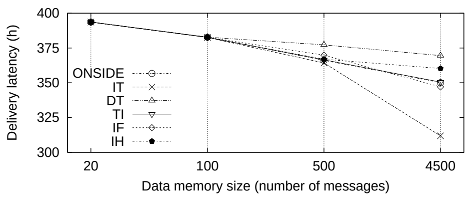
(d) Hop count

Figure 6: Results for ONSIDE with message-sorting heuristics on the Sigcomm 2009 trace.

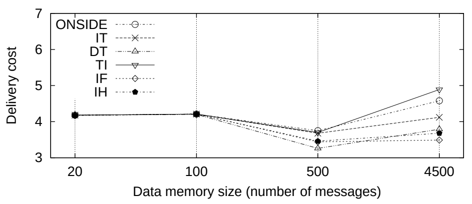
which lead to a lot of messages being dropped. This way, it doesn't matter much which heuristic is used, since a message doesn't get to be stored too much in a node's memory. For a higher memory size, such as 4500, most of the messages in the network can be stored at the same time in a node's memory, so there really isn't a need for removing messages, since there barely are any overflows. The best-performing heuristic in terms of delivery cost and hop count is IH, which is able to reduce the delivery cost by up to 31, leading to 3 times fewer messages being exchanged in the network, drastically reducing congestion. Keep in mind that ONSIDE was already able to reduce the delivery cost of the network when compared to Limited Epidemic, so the overall delivery cost improvement of IH when compared to Limited Epidemic is of 303, with almost 20 times fewer messages exchanged in the network. The IH heuristic manages similar results regarding hop count: for a data memory of 500 messages, IH reduces the average number of hops a message passes through until it reaches its destination by 78. Thus, the conclusions for the Sigcomm 2009 trace, as presented in Fig. 6, are that the message-sorting heuristic doesn't affect the network's hit rate and delivery cost too much, but it is able to drastically decrease the congestion. This isn't necessarily a rule applicable for all mobility traces (as we will show next, for the other two traces), depending instead on the specific characteristics of each particular opportunistic network.



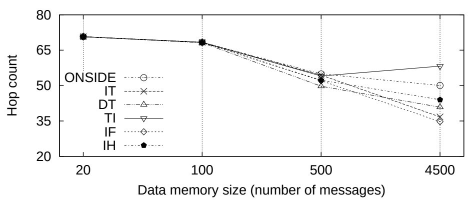
(a) Hit rate



(b) Delivery latency



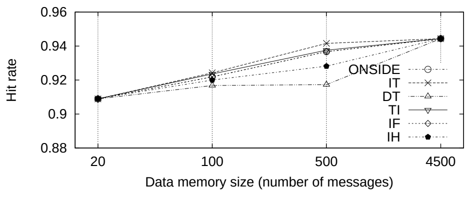
(c) Delivery cost



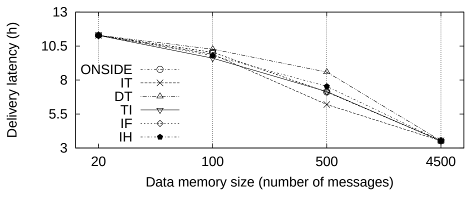
(d) Hop count

Figure 7: Results for ONSIDE with message-sorting heuristics on the UPB 2012 trace.

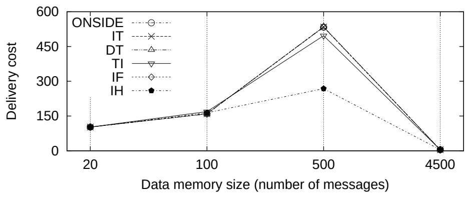
The results for the UPB 2012 trace can be seen in Fig. 7. Figure 7(a) shows that, as opposed to Sigcomm 2009, for UPB 2012 employing the right heuristic for sorting messages can lead to an increase in hit rate. There are four heuristics that perform better than basic ONSIDE: IT, TI, IF and IH, with the greatest improvement being brought by IT (almost 3% for a data memory of 4500). The reason why selecting a good heuristic for this trace works better than for Sigcomm 2009 in terms of hit rate is that there are a lot more messages in the network for UPB 2012, and its duration is much longer, leading to more data memory overflows. Therefore, correctly sorting the messages leads to a more optimal behavior in term of keeping the ones that the carrier node has a high probability of delivering to the destination. IT works better because it is able to spread new messages quicker in the network, which increases the probability of bootstrapping (i.e. avoiding early extinction of messages during the initial phase of generation). In terms of delivery latency, shown in Fig. 7(b), IT is also the most suitable heuristic, being able to decrease the average latency with up to 38 hours for a data memory of 4500 messages, when compared to ONSIDE. This shows that IT is not only able to increase the hit rate, but it is also able to deliver the messages quicker. It can also be seen in Fig. 7(b) that not all heuristics help deliver messages faster, as DT and IH perform worse than basic ONSIDE. Figures 7(c) and 7(d) show that IT performs very well when compared to



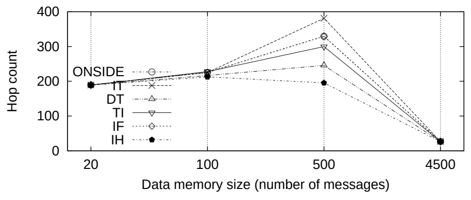
(a) Hit rate



(b) Delivery latency



(c) Delivery cost



(d) Hop count

Figure 8: Results for ONSIDE with message-sorting heuristics on the Infocom 2006 trace.

basic ONSIDE for delivery cost and hop count as well (in both cases, the obtained values are lower than for ONSIDE). However, it doesn't perform the best out of all tested heuristics. Instead, IF performs best, because the UPB 2012 trace has a strong connection between the social network of participants and the real-life interactions. This is true because the nodes belong to students attending classes at University Politehnica of Bucharest. Students participating in the same courses thus meet on a regular basis, and they get to interact and know each other, so most likely they end up being Facebook friends. For this reason, the IF heuristic reduces the delivery cost, since nodes store messages that are of interest to their Facebook friends, and end up meeting them after relatively short intervals, so the messages don't need to bounce from node to node that much.

Finally, Fig. 8 presents the results for the Infocom 2006 trace, and it can be seen that the two best-performing heuristics are IT (hit rate and delivery latency) and IH (delivery cost and hop count). IT helps increase the hit rate and decrease the delivery latency for the same reasons presented for UPB 2012, namely that new messages have the chance of spreading quicker in the network, and thus are able to reach suitable forwarding nodes or interested subscribers faster. On the other hand, predicting future contacts based on the encounter history helps decrease the congestion in the ON, because nodes may know with a good probability when they will meet a message's destination, so they don't forward it to other nodes if that moment will arrive in the near future. It is also possible that combining IT and IH could lead to even better results overall, but we will leave this as future work.

6. Conclusions and Future Work

In this article, we have shown that using node interests in data dissemination strategies for opportunistic networks helps reduce congestion, without affecting the hit rate and delivery latency. We began by presenting Interest, a simple interest-based algorithm, and showed that it matches Epidemic in terms of hit rate, but drastically decreases congestion. We have also shown that it clearly outperforms (both in terms of congestion, as well as hit rate) a community-based algorithm that groups nodes based on their contacts.

Then, we presented ONSIDE, which uses information about a node's social connections, interests and contact history, and compared it with Epidemic and ML-SOR. The results showed that, generally, both ONSIDE and ML-SOR barely affect the overall hit rate (with a slight advantage for ONSIDE), while decreasing the congestion and overhead. For most of the situations, ML-SOR seems to work better for lower data memory sizes, while ONSIDE fares well for nodes that are able to store more messages. This happens because ONSIDE has a higher point of balance, when the messages stored in a node's memory reach a local maximum in terms of utility, making subsequent data transfers unnecessary. For ML-SOR, this point is lower, but the equilibrium is less stable, eventually leading to frequent data exchanges.

Finally, we have also proposed five heuristics for storing the messages in a node's data memory. As we stated previously and confirmed through the analysis of the results, there is no best heuristic that suits all types of opportunistic networks. Instead, which heuristic should be chosen depends on various characteristics of the ON such as density, average contact and inter-contact times, etc. Generally, IT (increasing timestamp) and IH (interests history) have yielded the best results, which leads us to believe that new messages that enter the ON should be relayed quickly by the publisher (in order to avoid the risk of them being dropped due to congestion and being lost for good), and that behavior in opportunistic networks is fairly predictable based on the history of encounters.

For future work, we wish to test our proposed solution on a synthetic trace obtained from a mobility model such as HCMM [26], because it would allow us to control the number of nodes and their behavior better. However, these traces don't contain interest information (which is needed by ONSIDE and all related solutions), so it would have to be generated based on node relationships. Something similar is done in [25], where the authors apply machine learning techniques to existing traces, in order to infer user interests for mobility. The main limitation of this method is, however, that it requires prior knowledge of at least some node interests, which isn't available on a synthetic trace.

We also wish to find more existing mobility traces that contain interest information, since unfortunately not many are available. Moreover, we wish to perform another tracing experiment at our faculty, to continue the work done in [14]. However, we wish to collect user interests in a more fine-grained fashion, in order to achieve better results for ONSIDE.

Acknowledgements

The research presented in this paper is supported by national project MobiWay, Project PN-II-PT-PCCA-2013-4-0321. We would like to thank reviewers for their constructive comments and valuable insights.

References

[1] M. Mordacchini, L. Valerio, M. Conti, A. Passarella, A cognitive-based solution for semantic knowledge and content dissemination in opportunistic networks, in: World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a, IEEE, 2013, pp. 1–6.

[2] A. Moghadam, H. Schulzrinne, Interest-aware content distribution protocol for mobile disruption-tolerant networks, in: World of Wireless, Mobile and Multimedia Networks Workshops, 2009. WoWMoM 2009. IEEE International Symposium on a, 2009, pp. 1–7.

[3] A. Mei, G. Morabito, P. Santi, J. Stefa, Social-aware stateless forwarding in pocket switched networks, in: INFOCOM, 2011 Proceedings IEEE, 2011, pp. 251–255.

[4] P. Costa, C. Mascolo, M. Musolesi, G. Picco, Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks, Selected Areas in Communications, IEEE Journal on 26 (5) (2008) 748–760.

[5] A. Vahdat, D. Becker, Epidemic routing for partially connected ad hoc networks (2000).

[6] C. X. Mavromoustakis, G. Mastorakis, A. Bourdena, E. Pallis, Energy efficient resource sharing using a traffic-oriented routing scheme for cognitive radio networks, IET Networks Journal Accepted, to appear 2014.

[7] C. X. Mavromoustakis, Mitigating file-sharing misbehavior with movement synchronization to increase end-to-end availability for delay sensitive streams in vehicular p2p devices, International Journal of Communication Systems 26 (12) (2013) 1599–1616.

[8] A. Socievole, E. Yoneki, F. De Rango, J. Crowcroft, Opportunistic message routing using multi-layer social networks, in: Proceedings of the 2Nd ACM Workshop on High Performance Mobile Opportunistic Systems, HP-MOSys '13, ACM, New York, NY, USA, 2013, pp. 39–46.

[9] W. Gao, G. Cao, User-centric data dissemination in disruption tolerant networks, in: INFOCOM, 2011 Proceedings IEEE, IEEE, 2011, pp. 3119–3127.

[10] J. C. C. D. R. G. Augustin Chaintreau, Pan Hui, J. Scott, Pocket Switched Networks: Real-world mobility and its consequences for opportunistic forwarding, Tech. rep., Computer Laboratory, University of Cambridge (Feb. 2005).

[11] R. I. Ciobanu, C. Dobre, V. Cristea, Social aspects to support opportunistic networks in an academic environment, in: Proceedings of the 11th international conference on Ad-hoc, Mobile, and Wireless Networks, ADHOC-NOW'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 69–82.

[12] P. Hui, J. Crowcroft, Bubble Rap: forwarding in small world DTNs in ever decreasing circles, Tech. Rep. UCAM-CL-TR-684, University of Cambridge Computer Laboratory (May 2007).

[13] A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, C. Diot, MobiClique: middleware for mobile social networking, in: Proceedings of the 2nd ACM workshop on Online social networks, WOSN '09, ACM, New York, NY, USA, 2009, pp. 49–54.

[14] R.-C. Marin, C. Dobre, F. Xhafa, Exploring Predictability in Mobile Interaction, in: Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference on, IEEE, 2012, pp. 133–139.

[15] R.-I. Ciobanu, R.-C. Marin, C. Dobre, V. Cristea, C. X. Mavromoustakis, ONSIDE: Socially-aware and interest-based dissemination in opportunistic networks, in: The Sixth IEEE/IFIP International Workshop on Management of the Future Internet, ManFI, IEEE, 2014.

[16] R. I. Ciobanu, C. Dobre, Predicting encounters in opportunistic networks, in: Proceedings of the 1st ACM Workshop on High Performance Mobile Opportunistic Systems, HP-MOSys '12, ACM, New York, NY, USA, 2012, pp. 9–14.

[17] A. Bourdena, C. X. Mavromoustakis, G. Kormantzas, E. Pallis, G. Mastorakis, M. B. Yassein, A resource intensive traffic-aware scheme using energy-efficient routing in cognitive radio networks, Future Generation Computer Systems Accepted, to appear 2014.

[18] C. X. Mavromoustakis, C. D. Dimitriou, G. Mastorakis, E. Pallis, Real-time performance evaluation of f-btd scheme for optimized qos energy conservation in wireless devices, in: Globecom Workshops (GC Wkshps), 2013 IEEE, IEEE, 2013, pp. 1151–1156.

[19] C. Boldrini, M. Conti, A. Passarella, Modelling data dissemination in opportunistic networks, in: Proceedings of the third ACM workshop on Challenged networks, ACM, 2008, pp. 89–96.

[20] S. Ioannidis, A. Chaintreau, L. Massoulié, Optimal and scalable distribution of content updates over a mobile social network, in: INFOCOM 2009, IEEE, IEEE, 2009, pp. 1422–1430.

[21] R. I. Ciobanu, C. Dobre, V. Cristea, Social aspects to support opportunistic networks in an academic environment, in: Ad-hoc, Mobile, and Wireless Networks, Springer, 2012, pp. 69–82.

[22] C. Boldrini, M. Conti, A. Passarella, Design and performance evaluation of contentplace, a social-aware data dissemination system for opportunistic networks, Comput. Netw. 54 (4) (2010) 589–604.

[23] E. Yoneki, P. Hui, S. Chan, J. Crowcroft, A socio-aware overlay for publish/subscribe communication in delay tolerant networks, in: Proceedings of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems, MSWiM '07, ACM, New York, NY, USA, 2007, pp. 225–234.

[24] P. Hui, E. Yoneki, S. Y. Chan, J. Crowcroft, Distributed community detection in delay tolerant networks, in: Proceedings of 2Nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture, MobiArch '07, ACM, New York, NY, USA, 2007, pp. 7:1–7:8.

[25] A. Noulas, M. Musolesi, M. Pontil, C. Mascolo, Inferring interests from mobility and social interactions, in: Proceedings of Workshop on Analyzing Networks and Learning with Graphs (NIPS 2009), 2009.

[26] C. Boldrini, A. Passarella, Hcmm: Modelling spatial and temporal properties of human mobility driven by users' social relationships, Comput. Commun. 33 (9) (2010) 1056–1074. doi:10.1016/j.comcom.2010.01.013. URL `http://dx.doi.org/10.1016/j.comcom.2010.01.013`