Opportunistic Dissemination using Context-Based Data Aggregation over Interest Spaces

Radu-Ioan Ciobanu, Radu-Corneliu Marin, Ciprian Dobre, Valentin Cristea
University Politehnica of Bucharest
Sp. Independentei, 313
Bucharest, Romania
{radu.ciobanu, radu.marin}@cti.pub.ro,
{ciprian.dobre, valentin.cristea}@cs.pub.ro

Constandinos X. Mavromoustakis
University of Nicosia,
46 Makedonitissas Avenue, 1700
Nicosia, Cyprus
mavromoustakis.c@unic.ac.cy

George Mastorakis
Technological Educational
Institute of Crete
Estavromenos 71500, Heraklion,
Crete, Greece
gmastorakis@staff.teicrete.gr

***Abstract*—The traditional pub/sub paradigm is inadequate for dissemination in mobile networks, since any node is able to publish content at any time, thus easily leading to congestion. Therefore, a dissemination paradigm where mobile nodes contribute with a fraction of their resources is needed through the use of opportunistic networks. Moreover, as shown in recent work, a suitable organization for data dissemination in mobile networks should be centered around interests. Thus, we propose an interest-based dissemination framework for opportunistic networks entitled Interest Spaces. We focus on the first step required for implementing it in real life: data aggregation. Furthermore, we propose a method for aggregating data from encountered peers, in order for opportunistic nodes to have an informed view of the network and to avoid storing excessive amounts of information or performing many data exchanges.**

I. Introduction

The publish/subscribe paradigm in traditional client/server and peer-to-peer systems assumes a limited number of publishers, which are well-known and thoroughly advertised. However, with the advent of Web 2.0, and also due to the exponential increase in the number of computationally capable mobile devices on the market, the aforementioned paradigm has suffered significant changes lately. For example, adapted publish/subscribe paradigms assume that any node is able to publish content, so the number of publishers can easily match the number of subscribers. In this situation, congestion can be reached when many nodes publish data on the same channel and the number of subscribers is high. The high number of mobile devices also makes classic publish/subscribe impracticable, since mobile networks require decentralization. Moreover, conventional peer to peer approaches are insufficient, exactly because of the presence of mobile nodes, as well as because of challenged networking conditions. Therefore, a data dissemination paradigm where mobile nodes contribute with a fraction of their resources is needed, which leads to the use of opportunistic networks (ONs).

Whereas classic pub/sub requires brokers to handle subscriptions, we believe that a more suitable organization for mobile networks should be done in a decentralized interest-based manner, as in online social networks. Thus, instead of specifying a channel to which they subscribe, nodes can simply state that they are interested in a certain topic, such as "football". Conversely, nodes that generate information regarding football simply mark it with a "football" tag. This abstraction greatly simplifies publishing and subscribing, since nodes do not need to know who the publishers are, how channels are identified, and they don't need a broker or other central entity.

Data dissemination in ONs can be accomplished successfully through the collaboration of users performing a modified version of publish/subscribe based on interests, as previously shown [1]. For this reason, we propose here an interest-based data dissemination framework for ONs, entitled Interest Spaces, able to disseminate data to interested nodes by taking advantage of their context information (such as location, interests, social connections, encounter history, etc.). Here we present the framework's proposal, including its architecture, components, roles, behavior and functionality. We focus on the first step required for implementing Interest Spaces in real life, namely data aggregation. Thus, we propose a method for aggregating data from encountered peers, in order for opportunistic nodes to have an informed view of the network, and at the same time to avoid storing excessive amounts of information or performing many data exchanges.

II. Related Work

Several context-based models and frameworks for mobile networks have been proposed. For example, Context Spaces [2] is a framework that offers a model for representing context as a constrained view of the world, which can be analyzed in order to decide if certain situations occur. Interest Spaces has a similar way of representing context, but using an ECA (event condition action) paradigm. Thus, situations are not necessarily events that are observed to occur, but rather actions that should be taken if certain conditions are met.

Another example is ContextCast [3], a framework that offers a protocol for maintaining a self-organizing routing backbone which permits geographical addressing and resource discovery. However, one of the main caveats of ContextCast is that it assumes that the nodes have Internet connectivity through TCP at all times. Furthermore, ContextCast groups nodes based on geographical location, but doesn't take advantage of a mobile device's capabilities of detecting which nodes are within wireless communication range. The Interest Spaces framework is able to group nodes not only geographically, but also based on common interests. It is also specifically designed for ONs, so it assumes that nodes can communicate directly when they are in range, or indirectly through other nodes.

SICC [4] is a framework for MANETs that can be used to describe the context of an application by extending it to encompass a neighborhood within the network. This allows applications to define their area of effect, giving them control over the dissemination process. However, MANETs need routing information to function correctly. In a dense network with thousands of devices, the amount of storage space required to store all the routes of a node becomes extremely high. The advantage of the Interest Spaces framework is that it is built for ONs, where disconnections are the rule, rather than an undesired effect, and routes are opportunistically built depending on a node's encounters. Moreover, while SICC allows nodes to define a context composed of peers in their geographic vicinity, applications using our proposed framework define interest spaces, which can contain nodes located anywhere, but sharing common interests.

Another framework that offers services similar to Interest Spaces is Floating Content [5]. It allows applications to define anchor zones, which are geographical areas where nodes enter, spend a certain amount of time, and leave. In this case, the nodes are mobile devices belonging to humans, and the anchor zones are relatively small-sized areas where many people congregate, which represent the boundaries of an ad hoc ON. While inside the anchor zone, nodes may copy the data either if they need it for themselves, or if they transport it for the benefit of others. If a node is interested in a certain data item from the anchor zone, it replicates it. Data dissemination using Floating Content is done to all nodes in an anchor zone, so a node receives messages regardless of whether they are required or not, which may lead to congestion and unnecessary transfers. On the contrary, Interest Spaces only delivers data to nodes that have declared their interest in it, or to cache nodes, which are able to speed the dissemination process for other nodes.

Regarding merging or aggregating data, Pietropaoli et al. [6] propose methods for propagating belief functions from one frame of discernment to another, in order to compute context more efficiently. This way, the same evidence is not computed unnecessarily multiple times. Based on the authors' observations, we propose using aggregation in Interest Spaces as well, in order to decrease the amount of data stored per node, as well as the quantity of computations needed when defining a node's context. Another way of performing data aggregation is proposed in [7], through employing cognitive heuristics when disseminating in ONs. Thus, dissemination information is reduced by compression into an aggregate metric that allows the probabilistic identification of whether the data items of a channel have been disseminated enough. Similarly to this idea, we also employ a probabilistic approach. The aggregation method proposed in this paper has the benefit of being performed for multiple types of context data (such as contact history or social information). Moreover, it is based on the relationships between nodes. Its goal is not only to reduce storage space and data transfers, but also to improve the dissemination decision through a higher degree of information.

III. Interest Spaces

A. Architecture

There are four layers in the Interest Spaces framework, as shown in Figure 1. On the highest level there is the application layer, which provides the API for all applications using our framework. The API provides the basic calls for publishing a data item marked with at least one tag (interest) and for specifying that the current node is interested in receiving data items marked with the given tags (the equivalent of subscribing to channels in classic pub/sub).

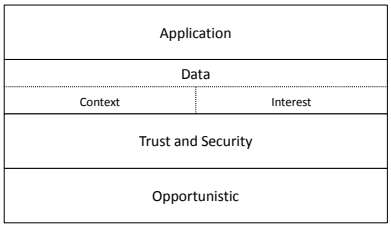


Fig. 1. Architecture of the Interest Spaces framework.

The second layer is the data layer and has two components: interest and context. The interest component handles data publishing and subscriptions based on interests, as a response to application requests. At this level, interests are represented internally, a node's data memory is organized based on interests and data items are packed with the necessary information, as shown further on. The context component is responsible for collecting context information, both about the current node (such as social connections, location, etc.), as well as about encountered nodes (encounter history, interest history, etc.). Since most nodes in opportunistic networks are mobile devices with relatively small memories, there is a limit to the amount of data that can be stored. In order to reduce the amount of information kept on-device (especially when there are many interests and nodes in the network), the context component also handles data aggregation. The context is also used in performing caching and forwarding decisions: nodes decide if they are suitable for carrying data marked with a certain tag based on context information collected opportunistically through encounters with other nodes.

The next layer of the Interest Spaces architecture handles trust and security. It deals with data encryption, trust representation, anti-forging mechanisms, etc. Because opportunistic networks are decentralized, there is no central node that can act as a certified entity used for storing keys and certificates. Instead, nodes have to use gossiping mechanisms and similar techniques to decide which nodes to trust on the fly.

At the bottom level of the architecture, there is the opportunistic layer. It handles the communication between nodes, based on the decisions taken at the layers above. The opportunistic layer also has the role of performing the exchange of context data between nodes, necessary for making more informed decisions.

B. Application Layer

A node's interest in Interest Spaces is expressed as a tag. There are two types of tags that a message can have: general and specific. A general tag represents a broad domain that can encompass a lot of data, but is used if the application level cannot decide on a more specific topic. On the other hand, specific tags represent more informed topics. Each specific tag is a subset of a general tag, and the reunion of all specific tags

pertaining to a general tag would contain all data marked with the general tag. For example, a general tag could be "sports", which would be formed of specific tags such as "football", "tennis", "basketball", "hockey", etc. The general tags are useful for reducing the size of messages that would otherwise contain a long list of specific tags.

The application layer represents the interface between the Interest Spaces framework and the applications that use it. It offers various capabilities through its API, as shown below:

- *set_general_tag(name)*: adds a general tag
- *set_specific_tag(name, general)*: adds a specific tag with the specified general tag
- *publish(content, priority, tags)*: publishes a message with the given content, priority and tags
- *subscribe(tag, callback)*: subscribes to receive messages marked with the given tag, calling the function given as parameter when this happens
- *unsubscribe(tag)*: cancels subscription to receive messages marked with the given tag

C. Data Layer

1) Message Format: Data items in Interest Spaces are sent between nodes as messages, which, aside from the actual content, also contain metadata used to uniquely identify them. For example, the timestamp specifies the time when the message was generated. Some messages may have a limited life, so the timestamp is used to help compute this life and decide whether delivering those messages becomes a priority. The priority is used to represent the importance of a message, since some messages (such as breaking news) should arrive faster than others at interested nodes. This is a dynamic measure that can change over time given other context conditions, and will be detailed in future work.

The tags are represented as strings, which are attached to a message's representation. There is no limit to the number of tags a message can have, the only restriction is that a message cannot be marked with a specific tag and with the general tag corresponding to the specific one at the same time.

2) Message Storing: Internally, a node should store the messages it has received or generated in a manner that would make access to them easy. Generally, a request for data is made with a tag, so this is why the tags act as keys in a dictionary. However, since a message can have multiple tags, there is no association between a tag and a message (i.e. data object), but between a tag and a pointer to a data object, so multiple tags can point to the same object. General tags also have pointers to their composing specific tags, so it is easier to reach them. A general tag can have any number of composing specific tags, whereas a specific tag only has one general tag parent. Each tag has a list of pointers to messages marked with it, and messages in each tag list are ordered by priority, so if a contact is short and not all messages marked with a certain tag can be sent, then the most important ones will be delivered first.

3) Node Types and Behavior: There are three types of nodes in Interest Spaces: publishers, subscribers and cache nodes. One node can act as any type. The application level can only control if a node is a publisher and/or a subscriber, but the decision of whether a node becomes a cache is left solely to the framework.

In order to disseminate messages, publishers must specify tags for the data objects they publish. Other nodes that are subscribed to those tags are able to receive data generated by the publishers. The purpose of Interest Spaces is to maximize the hit rate of the data items, which means that as many interested nodes as possible should receive the data they are interested in. Moreover, this has to be done as quickly as possible, while avoiding node and network congestion. A node becomes a publisher when the application decides that it needs to send a message, associated with at least one tag. When a message is generated, it is stored into the node's memory and the interest and context components are used to decide what should be done with that message (i.e. store it, send it to a node, delete it, etc.). When a message must be forwarded to an encountered node, it passes through the trust and security layer, after which is given to the opportunistic layer, which then handles the actual data exchange.

Subscribers are nodes interested by information marked with certain tags. They are able to specify the tags they are interested in at any time, as well as to unsubscribe from certain tags. Messages that subscribers are interested in should arrive as soon as possible, especially in environments where data can become stale quickly. The application layer is not aware of which nodes cache which data and how this is done. Instead, it only specifies the tags that the current node is interested in, and the opportunistic layer performs the exchange with encountered nodes that contain data of interest. The data layer is used to help decide if a certain data item is needed. If such an item is received by a node, it is sent directly to the application layer.

Finally, the cache nodes represent the backbone of the Interest Spaces framework. Their task is to cache and transport data items for the benefit of others. Nodes are able to become cache nodes for certain tags when they are in the vicinity of other nodes interested in such tags, or when they are known to interact often with such nodes. They store data of interest to these nodes until they encounter them and deliver the data. A node is a cache node for a certain context, which is computed on the fly and can change very quickly.

4) Caching Decision Function: The decision of whether a node should become a cache for a certain tag is done at the context level and is naturally based on the node's context. Information that might be useful at this phase includes encounter and interest history, current and encountered node interests, online social network, remaining battery, available storage space, geographical location, etc. This decision is taken using a function C_t (where t is the tag for which the function is called) defined as follows:

$$C_t : ctx \rightarrow [0, 1]$$

The domain (ctx) is the context of the node, while the result is a value between 0 and 1 which represents the probability of a node becoming a cache for tag t. This function is called at every contact or data exchange, as well as periodically,

since the context is ever-changing. C_t is called for every tag encountered in the last time frame, similar to a sliding window.

Upon a contact, nodes exchange metadata about the messages they carry, and then request the ones they are interested in. Thus, when a node becomes a cache for a certain tag, it will start requesting data marked with that tag from every node it encounters, and will provide it to every encountered node interested in it. When a node decides that it cannot act as a cache any more for a tag, there are multiple ways in which it can proceed. It can either instantly drop any data with that tag (assuming the data is not marked with other tags that the node is a cache for), it can drop it after a predefined period has passed, or it can store it until it encounters a node that announces itself as a node cache for that tag and then deliver it to that node. Moreover, if a node decides that it cannot act as a cache because of conditions such as low battery or storage space, it can request an encountered node to take its place, even if that node would not have become a cache by itself. This may be useful especially if the two nodes would meet again, since the encountered node would basically act as a cache for the current node (it would store the data for a while, and then return it to the original node).

5) Node Memory Organization: An Interest Spaces node's data memory is split into several sections. Firstly, there is the message cache, containing messages generated by the node itself or received from other encountered nodes. Then, nodes also contain identification data, which includes a unique node ID (such as the device's MAC address or IMEI code), the social network information (which can have multiple layers, depending on the social network used), interests in the various tags (i.e. the tags the application requires, and the ones that the nodes are caches for), as well as a list of counts for each carried tag (updated at every contact, in order to avoid parsing the entire list of messages at each encounter). The current location of the node is stored as well, and it is updated whenever it changes. Finally, a node keeps context information that is aggregated with similar information received from encountered nodes. This context information is updated at every contact with another node. The contact history contains encounters with other nodes, while the tag history counts each tag seen in messages from encountered nodes. The interest history represents the interests that encountered nodes have shown, and the location history keeps track of the geographical coordinates where the current node has been.

6) Context Data Aggregation: ONs are mainly composed of mobile devices such as smartphones, tablets, sensors, etc. Generally, these devices have limited memory, as well as battery. For this reason, it is important to optimize data storage, especially since large networks can lead to a great deal of context data needing to be stored. On the other hand, caching and forwarding decisions need to be as informed as possible, because they optimize the hit rate, delivery latency and even congestion of the network. In order to strike a balance between these two issues, we propose aggregating the context data a nodes stores with the data of encountered nodes, instead of keeping them separate per encountered node.

IV. Data Aggregation in Interest Spaces

In order for a node to decide whether to become a cache for another node, it should have as much information about the network as possible. Lack of information can lead to erroneous decisions, which may in turn lead to nodes becoming caches and then not being able to deliver messages to their destinations, or nodes not becoming caches even though they are the most suitable candidates. For this reason, nodes need to collect as much data as possible about encountered nodes, from number of contacts with others, to social connections or interests. However, storing this data for all the nodes encountered, especially in large and massively populated ONs, can easily lead to congestion due to the high number of data transfers needed. Furthermore, the storage of opportunistic network devices is limited, so the information stored should also be minimized as much as possible.

The data aggregation method we propose here is based on the idea that strongly-connected nodes (in terms of interactions, social connectivity, common interests, etc.) can work together to deliver data to interested destinations. Since most of the context data is numerical, we first propose an aggregation weight function defined as follows (with N representing the set of all nodes in the network):

$$A_W : N \rightarrow [0, 1]$$

When A encounters B, it has to compute $A_W(B)$, and the result is a weight with which B's data is merged into A's. Then, an aggregation function is called depending on the type of context data, which is defined as follows, where D is the domain for each context data type (e.g. $\mathbb{N} \times \mathbb{N} \times ... \times \mathbb{N}$ for the number of contacts with other nodes):

$$A_F : [0, 1] \times D \times D \rightarrow D$$

A_F is called by A when encountering B after calling A_W, and takes three parameters: the weight computed by A_W, the value of the context data stored by A, and the context data value of B. The result is an aggregated value of the context data that will be stored at node A (e.g. a vector representing the aggregated number of contacts with each node).

As stated above, the types of context data that a node stores are different, so a specific aggregation function has to be defined for each of them. For this paper, we focus on the four types of context data that we consider to be the most relevant in terms of dissemination in opportunistic networks: history of contacts, social connections, interests and encountered interests.

For the history of encounters, tags, or interested nodes, the values are numbers that represent the count (i.e. number of encounters with a given node per a time window, number of times a certain interest has been spotted, etc.). Therefore, the aggregation formula that we propose is as follows:

$$A_F(w, c_A, c_B) = max(c_A, w \times c_B)$$

w is the weight for node B as computed by the A_W function, c_A is the value of the context data of node A (e.g. a vector of contacts with each node), c_B is the value for node B, and the result of function A_F replaces the value of c_A at node A. As an example, let's assume that node A has met node

C 10 times in a given time window, and node D 5 times in the same window. Node A encounters node B, which had met node C 6 times and node D 12 times prior to the encounter, and let's assume that A computes $A_W(B)$ as 0.5. Then, A will end up with the value of encounters with node C as 10 (i.e. $max(10, 0.5\times6)$) and with node D as 6 (i.e. $max(5, 0.5\times12)$). The same can be done for the history of encountered interests, since it's also represented as a vector.

For other context information, such as the social network graph or the list of interests, a different approach should be taken, because the representation is not a single number. For the social network data, which is represented as a list of boolean values for each node that the current node has encountered, we propose using the following aggregation function:

$$A_F(w, c_A, c_B) = \begin{cases} 1 & \text{if } c_A = 1 \text{ or } w \times c_B > t_a \\ 0 & \text{otherwise} \end{cases}$$

In this case, A_F is 1 only if node A is connected to the corresponding node, or if node B is connected and the weight computed by A_W is higher than a predefined aggregation threshold t_a. Something similar can be done for a node's list of interests, assuming that a value of 1 means that a node is interested in a tag, and 0 that it is not. In this case, if the aggregation weight computed by A for B is higher than a threshold, B's interests are added to A's.

V. Experimental Results

Although this paper is centered around the Interest Spaces framework, we chose to focus on data aggregation as the first step. We begin this section by describing our experimental setup and motivating our decisions. Then, we highlight the amount of data that can be saved by using our proposed aggregation methods, as well as the reduction in network transfers when nodes come into contact. In order to show that storing context data about encountered nodes is useful, we apply our proposed aggregation method to an existing dissemination algorithm and show that the overall hit rate and delivery latency of the opportunistic network are improved.

A. Experimental Setup

The experiments were performed using MobEmu [8], an opportunistic network emulator that is able to replay a mobility trace and apply a desired algorithm when two nodes meet. We chose a real-life trace instead of a simulation because it offers a more realistic behavior, and because the trace we used also contains information about the social connections between the nodes and their interests. This allows us to test the proposed aggregation methods for all four context data types, in a realistic scenario. The chosen trace is UPB 2012 [9], and was collected by using an Android application for a period of 64 days in an academic environment at the University Politehnica of Bucharest, with the participants being students, assistants and teachers from the faculty. Contact information was obtained through Bluetooth discovery messages and AllJoyn interactions, and the social data was collected from the Facebook accounts of the participants, which were also used for generating the interest of each node. Thus, there are five potential interests in the network, and each node can have any number of them. Since not all participants started the tracing application on their phones when getting to the faculty as instructed, some of the nodes have very few contacts. Moreover, some participants didn't have a Facebook account, so their social and interest data is missing. In order to be able to apply our solution correctly, we have removed from the trace the nodes without social and interest data, as well as those with few contacts. Thus, we ended up with 24 nodes, each having at least 43 contacts (with an average of 249 per node), and an average number of social connections of 6.7.

In all the experiments presented here, data (in the shape of messages) 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. We consider that a channel is represented by a topic of interest (or a tag), so there are 5 channels for the UPB 2012 trace. Every node interested in a certain topic can generate information on the corresponding channel, but not on channels that do not match its interests. Each node that has at least one interest generates 30 messages per day. A node interested in multiple topics is able to generate data for each of the corresponding channels, by choosing randomly.

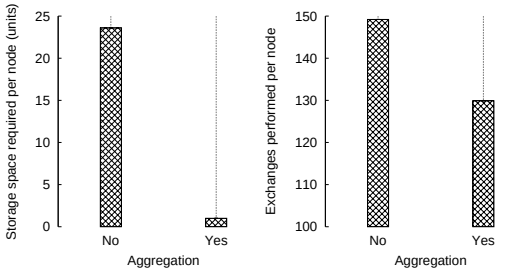
When aggregating social networks and interests, we chose the values for the t_a threshold empirically, by analyzing the distribution values of the aggregation function. Thus, t_a's value for social networks is 0.5, whereas for interests it is 0.1. The reason t_a is so small for interests is that the UPB 2012 trace has only five topics to choose from, so aggregating too often would lead to all the nodes ending up having all five interests. This would in turn lead to congestion in the network, since nodes would exchange messages very often. We are currently working on a method of dynamically varying the parameters of a dissemination algorithm (such as t_a) based on a node's view of the network. This will be presented as future work.

B. Saving Data through Aggregation

Through the aggregation method described in this paper, nodes do not need to separately store data about other nodes. Instead, they simply aggregate that data over their own data, thus reducing the necessary storage space. Therefore, we represented the necessary storage space for the Interest Spaces aggregation method as a single unit (e.g. a list of nodes, an array of social connections, etc.), and the required space for a non-aggregation algorithm as a multiple of this unit. Normally, if data weren't aggregated, a node would need to keep information about all the encountered nodes, as well as about nodes that the encountered nodes had previously met. If the network is very dense, a node might even end up keeping information from all nodes in the network.

The situation is similar for data exchanges. When aggregating using the method we proposed, nodes only need to perform a single data exchange at each contact. However, when keeping data separately per node, a node A would not only have to receive the information about an encountered node B, but also about all the other nodes encountered by B that have not been encountered by A (or even about nodes that A had encountered before B met them, in order to keep the information fresh).

Figure 2 shows the advantages of using aggregation. In Figure 2(a), it can be seen that, by not aggregating data, 23



(a) Storage requirements (b) Data exchanges

Fig. 2. Storage requirements and data exchanges with/without aggregation.

times more storage space would be needed per node. This means that, in networks where nodes meet often, a node would end up storing information about all the other nodes in the network. In real-life ONs with thousands of users, storing such a large amount of data would not be plausible. Figure 2(b) shows that a node also has to perform fewer data exchanges when aggregating, since it only requires the aggregated information about the encountered node. Reducing the number of exchanges leads to a lower congestion level in the network and to a decrease in the number of collisions (since the nodes communicate wirelessly).

C. Applying Data Aggregation to an Existing Algorithm

In the previous subsection we have shown that data aggregation is able to reduce the number of necessary network transfers, while at the same time decreasing storage requirements. However, we also wish to show that aggregating data is useful for improving the dissemination process in terms of hit rate, as well as delivery latency. Thus, we applied the proposed aggregation method to an existing algorithm that makes use of contacts and interest history, node interests and social information when performing dissemination decisions.

The chosen algorithm is ONSIDE [1], a dissemination strategy that leverages information about a node's social connections, interests and contacts history, in order to improve hit rate and delivery latency. This is done by carefully selecting the nodes that act as forwarders, instead of simply flooding every node. When two ONSIDE nodes meet, each node analyzes the other's messages and decides which of them should be downloaded by using a utility function. 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:

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

The result is a boolean value that specifies whether a download request should be made to B for message M. $common\_interests$ returns the number of topics that both A and B are interested in, and $interested$ returns $true$ if node A is subscribed to the channel that generated message M (i.e. if it is interested in M's topic). $interested\_friends$ returns the number of online social network friends of node A that are subscribed to the channel that generated M, whereas thr_f is a threshold that can be varied according to the density of the opportunistic and social networks. $interests\_encountered$ 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. thr_i is a threshold between 0 and 1 that can be varied depending on the number of channels in the network. For our experiments, we used the same values for thr_f and thr_i as in [1] for the UPB 2012 trace.

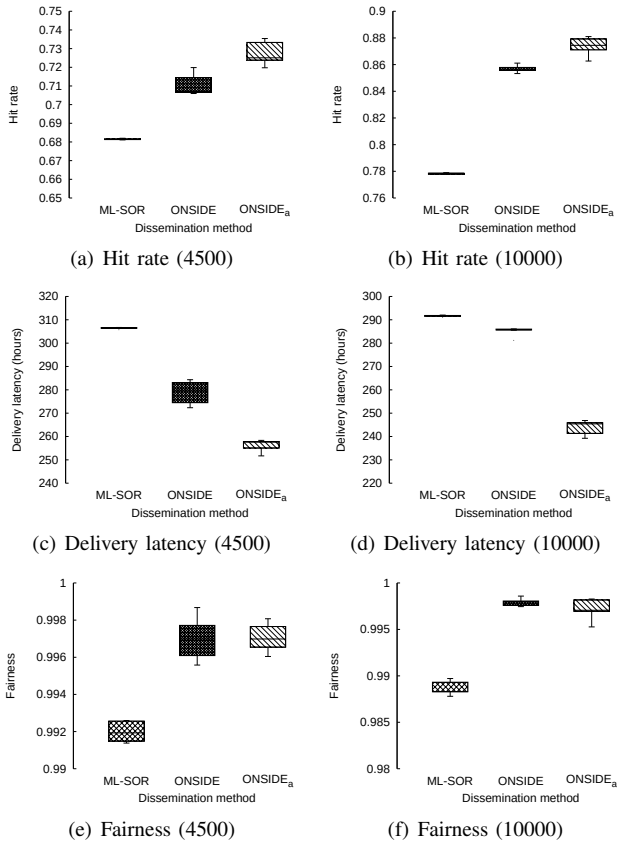
As can be seen, ONSIDE uses all the types of context data we aggregate, so it is the ideal candidate for our experiments. Instead of using only data from the current node, ONSIDE with aggregation (called ONSIDE$_a$) uses aggregated information updated at each contact. For the A_W function, we used a weighted sum of the parameters that we considered as the most important ones able to characterize the relationship between two ON nodes: similarity (number of common neighbors on social networks), friendship (number of common interests), connectivity (whether nodes are social network friends or not) and number of contacts. All these parameters are normalized to the maximum value computed so far, and we used equal weights. Thus, the aggregation weight of a node A towards a node B is computed as follows:

$$\begin{aligned} A_W(A,B) = w_1 \times sim(A,B) + w_2 \times friendship(A,B) \\ + w_3 \times connectivity(A,B) + w_4 \times contacts(A,B) \end{aligned}$$

In [1], we showed that ONSIDE is able to outperform other algorithms (such as ML-SOR [10]) in terms of hit rate and delivery latency. However, since the conditions of our experiments differ from [1], we also show the ML-SOR results here. ML-SOR is an algorithm which 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. ML-SOR has been adapted for dissemination as described in [1].

Figure 3 shows the results obtained by running ML-SOR, ONSIDE and ONSIDE$_a$ on the UPB 2012 trace, for node memory sizes of 4500 and 10000 messages (i.e. 4.5 GiB and 10 GiB for 1 MB messages). Figures 3(a) and 3(b) show not only that ONSIDE outperforms ML-SOR in terms of hit rate, but also that ONSIDE$_a$ yields better results than basic ONSIDE. The reason that data aggregation increases the hit rate is that the dissemination decision is more informed. Through aggregation, a node A may act on behalf of a strongly-connected peer B, downloading messages of interest to B. Since the two nodes encounter each other often, messages for B downloaded by A have a higher chance of reaching their destinations than if they had been cached by other nodes.

Furthermore, not only does ONSIDE$_a$ increase the number of nodes that receive messages they are interested in, but it also decreases the time it takes to receive them. Figures 3(c) and 3(d) show that ONSIDE$_a$ is able to reduce the average delivery latency of the network with up to 50 hours, which represents an improvement of about 17%. Moreover, this reduction in delivery latency happens while increasing the hit rate, so even though some more remote nodes are reached, the data is delivered faster when using ONSIDE$_a$, as opposed to ML-SOR or basic ONSIDE. It can also be observed that the



(a) Hit rate (4500) (b) Hit rate (10000)

(c) Delivery latency (4500) (d) Delivery latency (10000)

(e) Fairness (4500) (f) Fairness (10000)

Fig. 3. ML-SOR, ONSIDE and ONSIDE$_a$ results.

latency values for ONSIDE$_a$ do not have any outliers, so the results over several runs tend to be similar.

Figures 3(e) and 3(f) present the fairness of the three algorithms. This metric shows whether each topic of interest is treated equally in terms of message dissemination. Fairness is computed according to Jain's fairness index, using the hit rate as the measure of the service level obtained by each topic. It can be observed that all three algorithms have a fairness very close to maximum, meaning that messages tagged with all interests are distributed equally in the network. However, the ONSIDE algorithms outperform ML-SOR, regardless of the data memory size. Moreover, it can be seen that, for a data memory of 4500, ONSIDE$_a$ tends to fluctuate the least out of the three solutions for multiple runs.

VI. Conclusions and Future Work

In this paper, we have proposed Interest Spaces for opportunistic networks, an interest-based dissemination framework that spreads data by taking advantage of nodes' context information. We have described the framework's architecture and presented in detail its most important layers. Then, we focused on the first step in implementing Interest Spaces in real life, namely data aggregation. We proposed a method for aggregating various context data types, such as contact and interest history, social connections and node interests, and showed not only that aggregating data leads to less storage space required and fewer data exchanges, but also to a better performance in terms of hit rate and delivery latency.

Since we have only implemented a small part of the Interest Spaces framework, for future work we wish to carry on by proposing a cache decision function and analyzing its behavior. This function is arguably the most important component of the framework, since it is used to decide which nodes are data carriers. An optimal caching function would lead to high hit rates and low delivery latencies, which is why we consider it to be an important component of Interest Spaces.

Acknowledgment

The work presented in this paper is co-funded by the European Union, Eurostars Programme, under the project 8111, DELTA "Network-Aware Delivery Clouds for User Centric Media Events". The research is partially supported by COST Action IC1303 AAPELE, and by project MobiWay, PN-II-PT-PCCA-2013-4-0321.

References

[1] R.-I. Ciobanu, R.-C. Marin, C. Dobre, V. Cristea, and C. Mavromoustakis, "Onside: Socially-aware and interest-based dissemination in opportunistic networks," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, May 2014, pp. 1–6.

[2] A. Padovitz, A. Zaslavsky, and S. W. Loke, "A unifying model for representing and reasoning about context under uncertainty," in *In Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, 2006.

[3] D. Heutelbeck, "Context spaces - self-structuring distributed networks for contextual messaging and resource discovery," in *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*. London, UK, UK: Springer-Verlag, 2002, pp. 248–265. [Online]. Available: http://dl.acm.org/citation.cfm?id=646748.704403

[4] C. Julien, G. catalin Roman, I. C. Society, and Q. Huang, "Sicc: Source-initiated context construction in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, pp. 401–415, 2008.

[5] J. Kangasharju, J. Ott, and O. Karkulahti, "Floating content: Information availability in urban environments," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, March 2010, pp. 804–808.

[6] B. Pietropaoli, M. Dominici, and F. Weis, "Propagation of Belief Functions through Frames of Discernment : Application to Context Computing," in *The 26th International FLAIRS Conference*, St. Pete Beach, États-Unis, May 2013. [Online]. Available: http://hal.inria.fr/hal-00927088

[7] L. Valerio, M. Conti, E. Pagani, and A. Passarella, "Autonomic cognitive-based data dissemination in opportunistic networks," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, June 2013, pp. 1–9.

[8] R. I. Ciobanu, C. Dobre, and 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*, ser. ADHOC-NOW'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 69–82. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31638-8_6

[9] R.-C. Marin, C. Dobre, and 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. [Online]. Available: http://dx.doi.org/10.1109/EIDWT.2012.29

[10] A. Socievole, E. Yoneki, F. De Rango, and J. Crowcroft, "Opportunistic message routing using multi-layer social networks," in *Proceedings of the 2Nd ACM Workshop on High Performance Mobile Opportunistic Systems*, ser. HP-MOSys '13. New York, NY, USA: ACM, 2013, pp. 39–46. [Online]. Available: http://doi.acm.org/10.1145/2507908.2507923