Socially-Aware Routing for Publish-Subscribe in Delay-Tolerant Mobile Ad Hoc Networks

Paolo Costa, Cecilia Mascolo, Mirco Musolesi, and Gian Pietro Picco

***Abstract*—Applications involving the dissemination of information directly relevant to humans (e.g., service advertising, news spreading, environmental alerts) often rely on *publish-subscribe*, in which the network delivers a published message only to the nodes whose subscribed interests match it. In principle, publish-subscribe is particularly useful in mobile environments, since it minimizes the coupling among communication parties.**

However, to the best of our knowledge, none of the (few) works that tackled publish-subscribe in mobile environments has yet addressed intermittently-connected human networks. Socially-related people tend to be co-located quite regularly. This characteristic can be exploited to drive forwarding decisions in the interest-based *routing* layer supporting the publish-subscribe network, yielding not only improved performance but also the ability to overcome high rates of mobility and long-lasting disconnections.

In this paper we propose *SocialCast*, a routing framework for publish-subscribe that exploits predictions based on metrics of *social* interaction (e.g., patterns of movements among communities) to identify the best information carriers. We highlight the principles underlying our protocol, illustrate its operation, and evaluate its performance using a mobility model based on a social network validated with real human mobility traces. The evaluation shows that prediction of colocation and node mobility allow for maintaining a very high and steady event delivery with low overhead and latency, despite the variation in density, number of replicas per message or speed.

***Index Terms*—**

I. Introduction

MODERN communication technologies foster application scenarios where humans exchange information not only through natural means (e.g., verbally), but also through the mediation of computer networks. E-mail is the most evident example of this shift. However, pervasive and ubiquitous computing scenarios are pushing situations where it is the recipient (not the sender) of the information who determines whether and how to seize data flowing in the network. Thus, for instance, services can be freely advertised without *a priori* knowledge about who is going to exploit them: it is up to a given application (or user) to bind to a service based on its description. News and advertisements can be issued without specifying the recipients but only by declaring the type of the message content.

The design of the programming and networking infrastructure enabling these new forms of computer-mediated communication is still a topic of active research. However, the *publish-subscribe* paradigm recently emerged as a particularly promising solution. In such a paradigm, the information producers and consumers are sharply decoupled, as they are fully agnostic of each other. The information producer or *publisher* (e.g., the service advertiser) simply injects a message in the network. Routing protocols no longer revolve around node identifiers, since these are not specified in the message. Instead, the network delivers the message to the interested *subscribers* (e.g., the components interested in services of a given type) based on some characteristic of the message, such as its topic or even its very content.

Thanks to its decoupling properties, publish-subscribe is inherently suited to dynamic environments where the set of communicating parties changes over time and may be disconnected at the time the message is originated [1]. These most notably include mobile delay tolerant systems. However, dealing with the network challenges posed by publish-subscribe makes its practical realisation in this kind of networks difficult. As a result, few approaches exist [2], [3], surveyed in Section VI. Moreover, in these approaches, the *social* angle is never taken into consideration. Information needs are ultimately driven by users, which in a mobile environment exhibit patterns of movement dictated by their social behaviour. These typically lead to *intermittently connected* networks, where, however, connectivity can be ensured precisely by relying on the social ties of users. Users that belong to the same "group" (e.g., co-workers, friends, fans of a sport team) may move far apart from each other and experience long periods of disconnection, but are very likely to *eventually* meet again. This fact can be leveraged to opportunistically deliver messages related with the group's main interest. Section II defines precisely our assumptions about the kind of systems we consider. To give an example, messages containing advertisements of rugby events can be disseminated from host to host by exploiting the social contacts of the fellow people (i.e., people interested in rugby might all be interested in advertisements of rugby events and are bound to meet quite regularly).

In this paper we introduce *SocialCast*, a routing protocol expressly devised to support publish-subscribe in intermittently connected human networks. In a nutshell, *SocialCast* complements the information about the receivers' interests, necessary to routing information, with data about the social ties of people and their consequent predicted movements. The

Manuscript received June 17, 2007; revised January 10, 2008.

Paolo Costa is with the Department of Computer Science, Vrjie University, The Netherlands (e-mail: costa@cs.vu.nl).

Cecilia Mascolo is with the Computer Laboratory, University of Cambridge, UK (e-mail: cecilia.mascolo@cl.cam.ac.uk).

Mirco Musolesi is with the Institute of Security Technology Studies, Dartmouth College, USA (e-mail: musolesi@cs.dartmouth.edu).

Gian Pietro Picco is with the Dipartimento di Ingegneria e Scienza dell'Informazione, University of Trento, Italy (e-mail: gianpietro.picco@unitn.it).

Digital Object Identifier 10.1109/JSAC.2008.0806xx.

dissemination of these interests and social information, as well as its use for message forwarding and buffering, is described in Section III. In *SocialCast*, Kalman filter forecasting techniques [4] are used to predict the future evolution of the movement based on previous observations on some attributes characterising social behaviour (e.g., connectivity changes, colocation), as we illustrate in Section IV. These predictions are used to estimate which hosts are potentially good message carriers, i.e., may enable indirect connectivity by moving into connected portions of the network containing subscribers. *SocialCast* exploits forecasting techniques to identify the best carriers which are also used in CAR [5]. However, CAR is a unicast delay tolerant routing protocol and it does not support group communication. Section V demonstrates the effectiveness of our approach by presenting an evaluation through simulation over a realistic social mobility model [6] validated against real traces [7]. Section VII contains brief concluding remarks, including options for future work.

II. System Model and Assumptions

We assume a network composed of N nodes. For simplicity of treatment we assume they all have the same capabilities, in particular to store messages in a buffer of maximum size β. Nodes are mobile and interconnected by wireless links. The mobility of a node is determined by the user carrying it.

A user, and therefore a node, may act as an information *publisher* or *subscriber*[1]. Publishers and subscribers are in general not aware of each other. A node subscription identifies the node's interest (e.g., "Rugby" or "Computer Science"). We assume that each user in the system has at least one interest. When a message is published (e.g., "Six Nations Results"), it is tagged with the related interest. The goal of our protocol is to deliver the message to the nodes with at least one interest matching the one in the message. As such, delivery is driven by the message content. In this work we base matching on interests specified as message topics, but we conjecture that extensions allowing for more sophisticated and direct matching against the message content can be easily integrated in our approach.

Key to this work is the assumption that users with common interests tend to meet with each other more often than with other users [8]. This can be observed in practice in our everyday life. Examples are people interested in information concerning the department where they work, or friends sharing the same sport interest. In other words, we assume that the mobility of users is driven by their social behaviour that, in turn, is determined by their common interests.

Apart from the aforementioned social behaviour, nodes can move with arbitrary (not necessarily random) directions and speeds, and in doing so they may cause an arbitrary number of network partitions. Furthermore, for what concerns communication we rely solely on the basic ability of a node to communicate within its 1-hop neighbourhood, by broadcasting a message to all the neighbours or unicasting it to a specified one.

[1]A node can be, at the same time, a publisher and a subscriber.

III. Routing in *SocialCast*

In this section, we describe the main characteristics of our routing protocol. This relies on the notion of *utility* for the selection of message carriers in order to enable store-and-forward communication. The utility of a node n with respect to interest i represents how good of a carrier n is for messages matching i. The utility values in *SocialCast* are linked to movement patterns and colocation with other hosts: as the basic assumption is that hosts which have same interest spend time co-located, the *SocialCast* routing aims at exploiting as carrier for messages hosts which have been co-located often with the interested subscribers. The calculation of utilities are described in detail in the next section.

Routing in *SocialCast* consists of three phases: interest dissemination, carrier selection, and message dissemination. The distinction in phases is only for illustration purposes, as in practice each phase is executed one after the other. The whole sequence is repeated periodically after T units, without requiring synchronisation across nodes. Figure 2 contains the pseudo-code for the routing protocol while Figure 1 contains the necessary variable definitions.

During *Interest Dissemination*, each node broadcasts a control message containing the list of its interests to its 1-hop neighbours, along with the corresponding list of utility values as indicated in the pseudocode of Figure 2. These are first locally re-computed based on the current node context before dissemination (Figure 2, function *updateOwnUtilities*). This information is stored in the routing tables of the neighbours, and is key in determining message forwarding decisions. In this phase, the identifiers of the last λ messages received are also piggybacked on the utility message (Figure 2, *received[n]*).

During *Carrier Selection*, the utility of the local node, U_i, is recomputed for all interests i. This utility U_i is compared, for each interest i, against the highest among those communicated by neighbours, say $U_{n,i}$ as reported by a neighbour n. If $U_{n,i} > U_i + \epsilon$, this means that, for interest i, n is a better carrier than the local node (line 3). ϵ is an hysteresis threshold which forbids that the message is bounced back and forward between hosts with similar fluctuating utilities. Otherwise, the local node is still the best carrier for messages tagged with i.

During *Message Dissemination*, the content of the buffer is re-evaluated against the new subscriptions and utilities, and messages are forwarded to the interested nodes (line 5) and/or the best carrier (line 8). A copy of messages matching an interest i is immediately sent to all neighbours whose subscriptions contain i. Note how this ensures that nearby interested nodes receive messages, but does *not* imply that these also become a carrier for messages. In other words, messages are delivered to the above application layer but *not* inserted in the nodes' buffer. Indeed, carrier role (and buffer insertion) are determined by the outcome of the previous phase. If the local node is still the best carrier, no action needs be taken. Otherwise, *all* messages tagged with i are removed from the local node's buffer (line 9) and sent to n, the best carrier, where they are inserted in its buffer (line 5 of *receipt of DATA message*). An issue arises if n is also a subscriber for i. In this case, the matching messages can be properly flagged

to inform the receiving carrier n that they must be inserted in its buffer instead of being simply delivered to the application.

To avoid unnecessary traffic, a message is forwarded only if the recipient has not previously received that message. This can be easily verified by checking the list of the last λ messages piggybacked during the message delivery phase (line 4). Moreover, to prevent messages from remaining forever in the system, we rely on a time-to-live (TTL) based on hop counts. Clearly, other solutions are also possible. For instance, in some applications it could be useful to have the publisher explicitly specify an expiration time, (e.g., a concert advertisement is useful only before the time it starts).

Finally, *Message Publishing* consists simply of inserting the published message into the local buffer. The message will then be taken care and forwarded to the interested subscribers as well as "moved" to a better carrier, if and when encountered, according to the routing protocol we described thus far. In other words, *SocialCast* works based on whatever the content of the buffer is, regardless of how such content got inserted. Moreover, to ensure high delivery, a publish operation actually inserts γ copies of the message. Each copy is routed independently, i.e., whenever a better carrier is encountered only *one* copy is removed from the local buffer and sent to the new carrier, to ensure that the copies are spread over time and space across the system. Note that the publisher is the only node that duplicates messages, and does so only at publish time. Therefore, at any time the network contains at most γ copies of the message. This approach to message distribution is shared by other approaches such as Spray&Wait [9].

IV. Computing Utilities from Social Patterns and Mobility

In this section, we illustrate the definition of the utilities used to select message carriers. We argue that social patterns and mobility can be used to measure the suitability of a host as message carrier for subscribers to a given interest.

First of all, we define an *attribute* as a scalar representation of a dimension of the problem that affects the utility of a host as a potential message carrier. The utility is in general a function of multiple attributes representing the different dimensions of the problem (mobility, colocation, battery level, etc.). The primary utility attribute we leverage is the probability of a user to be co-located with another sharing the same interest. In this case, co-location enables direct delivery of messages matching the shared interest. This aspect is captured by the probability of subscriber co-location. However, as in real life, a person meeting many people has more options to disseminate information. Therefore, we also exploit the *change degree of connectivity* as another utility attribute to base forwarding decision upon. A node has a high change degree of connectivity if it frequently changes its neighbour set (e.g., because it is moving, or is static in a very dynamic area).

Knowledge about the current values of these social attributes is helpful, but only to a limited extent. In fact, what really matters are the values that the attributes are likely to assume in the future. We compute these *predicted* values using forecasting techniques based on the Kalman filter [10]. These techniques do not require the storage of the entire past history of the system and are computationally lightweight, making them suitable for a resource-scarce mobile setting. We exploit the fact that colocation patterns with subscribers to a certain topic i are not random, but they are based on the social network that link all the individuals carrying the devices.

Kalman filters are a technique for discrete signal processing that provides optimal estimates of the current state of a dynamic system described by a *state vector*. The state is updated using periodic observations of the system, if available, by a set of *prediction recursive equations*. Our prediction problem can be expressed as a state space model: a time series of observed values is used to represent the evolution of the attributes taken into consideration, from which we can derive a prediction model based on an inner state described by a set of vectors. Formally, given the current input observed value $\mathbf{Y}_t$ and the current state $\mathbf{X}_t$, a predictor based on Kalman filters is able to provide an estimate for the next value of the time series $\widehat{\mathbf{Y}}_{t+1}$.

$$\widehat{\mathbf{Y}}_{t+1} = f(\mathbf{X}_t, \mathbf{Y}_t)$$

We assume that the lag between two subsequent samples $\mathbf{Y}_t$ and $\mathbf{Y}_{t+1}$ of the time series is equal to τ. Trend and seasonal components [4] could be added as well. The prediction is re-evaluated periodically according to the (configurable) value of τ. We use a Kalman filter predictor for each attribute. The filter takes as input the current value at time t of the time series representing a particular attribute and returns the estimated value of the time series at time $t+\tau$ as output.

A summary of the forecasting model is presented in the appendix of this article and a comprehensive presentation of these techniques can be found in [5]. However, it is fundamental to present how a host computes the *input values* to the Kalman filter, i.e., the value of the utility at time t, for which the filter computes the predicted value at time $t+\tau$.

The colocation of h with a subscriber for interest i is

$$U_{col_{h,i}}(t) = \begin{cases} 1 & \text{if } h \text{ is co-located with a subscriber for } i; \\ 0 & \text{otherwise} \end{cases}$$

A value of 1 indicates that h has been co-located with subscribers for i at time t.

The change degree of connectivity of a host h is

$$U_{cdc_h}(t) = \frac{|n(t-\tau) \cup n(t)| - |n(t-\tau) \cap n(t)|}{|n(t-\tau) \cup n(t)|}$$

where $n(t)$ is h's neighbour set at time t. If $|n(t-\tau) \cup n(t)| = 0$ (i.e., the node was isolated in both the previous and current instants of time), $U_{cdc_h}(t)$ is set to 0. The formula yields the number of hosts that became neighbours or disappeared in the time interval $[t-\tau, t]$, normalised by the total number of hosts met in the same time interval. A high value means that h recently changed a large number of its neighbours.

These values are fed into Kalman filter predictors, which yield the predictions $\widehat{U}_{col_{h,i}}$ and $\widehat{U}_{cdc_h}$ of these utilities at time $t+\tau$. These are then composed into a single utility value using results from multi-criteria decision theory [11], as

$$U_{h,i} = w_{cdc_h} \widehat{U}_{cdc_h} + w_{col_{h,i}} \widehat{U}_{col_{h,i}}$$

which represents how good of a carrier h is for messages matching i. The weights w denote the relative importance of

Variables

- *self*: node's own id
- $\mathcal{N}$: set of node's current neighbours
- $\mathcal{I}$: set of node's interests
- $\mathcal{R}$: set of the identifiers of the last λ messages received
- $\mathcal{B}$: the message buffer
- $\mathcal{U}[i]$: node's own utility associated to interest i
- $interests[n]$: the set of neighbour n's interests
- $received[n]$: the set of message received by neighbour n
- $utility[n, i]$: the utility value of neighbour n, associated to interest i

Messages

- DATA$< i, replica, next >$: a data message tagged with interest i. $next$ identifies the neighbour to forward the message to, while $replica$ is a boolean flag indicating whether the message is being forwarded to a new carrier ($replica =$ TRUE) or carried ($replica =$ FALSE).
- CONTROL$< U, I, R, n >$: control message disseminated periodically by each node. It contains the utility values (U), the interests (I) and the identifiers of the last λ messages received (R) of the sender node n.

Functions

- **updateOwnUtilities** (): update node's own utility for interest i based on its mobility and its co-location
- **send** (m, r): send message m to recipient r
- **broadcast** (m): send message m to all 1-hop neighbours
- **deliver** (m): deliver message m to the application

Fig. 1. Pseudo-code definitions of the *SocialCast* routing protocol.

Interest Dissemination

updateOwnUtilities ()
create new message c: CONTROL$< \mathcal{U}, \mathcal{I}, \mathcal{R},$ *self*$>$
broadcast (c)

Invoked on receipt of a CONTROL message from neighbour n.
receive CONTROL$< U, I, R, n >$
$received[n] \leftarrow R$
$utility[n] \leftarrow U$
$interests[n] \leftarrow I$

Carrier Selection

for all $m \in \mathcal{B}$ **do**
  $m.next \leftarrow \perp$
  if $\exists n, i$ **s.t.** $n \in \mathcal{N} \wedge m.i = i \wedge utility[n, i] > \mathcal{U}[i] + \varepsilon \wedge m \notin received[n] \wedge$
  $(\nexists n' \in \mathcal{N}$ **s.t.** $utility[n', i] > utility[n, i] \wedge m \notin received[n'])$ **then**
    $m.next \leftarrow n$

Message Delivery

for all $m \in \mathcal{B}$ **do**
  $m.replica \leftarrow$ FALSE
  for all $x \in \mathcal{N}$ **do**
    if $\exists i$ **s.t.** $m.i = i \wedge i \in interests[x] \wedge m \notin received[x]$ **then**
      send (m, x)
  $m.replica \leftarrow$ TRUE
  if $m.next \neq \perp$ **then**
    send $(m, m.next)$
    $\mathcal{B} \leftarrow \mathcal{B} \setminus \{m\}$

Invoked on receipt of a DATA message from neighbour n.
receive DATA$< i, replica, next >$
if $i \in \mathcal{I}$ **then**
  deliver (m)
$m.TTL \leftarrow m.TTL + 1$
if $m.replica =$TRUE $\wedge\ m.TTL < maxTTL$ **then**
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{m\}$

Message Publishing.

insert γ instances of the published message into $\mathcal{B}$

Fig. 2. Pseudo-code of the *SocialCast* routing protocol.

each attribute. Their value depends on the application scenario and we assess their impact in Section V.

Our protocol relies on predictions about the future values of the attributes. However, in some conditions predictions are not reliable, e.g., because the time series describing a particular attribute is random or exhibit a behaviour that cannot be forecasted with accuracy (i.e., within a given prediction error) using the model used. Therefore, it is important to assess the confidence level of predictions, and modify forwarding decisions accordingly. To assess the quality of predictions we use the technique presented in [12], based on the analysis of the prediction error [13]. A *predictability component* receives in input both the observed value (at time t) of a attribute and the predicted value (computed at $t - 1$). The analysis over time of the difference between these two values enables us to determine whether the prediction model (the Kalman filter in our case) has enough information to predict the next value of the time series with the required accuracy.

Another important note is that the framework used for prediction is general enough for inclusion of other attributes, beyond the ones used (colocation and change degree of connectivity), such as remaining battery power which might be very important for sensor network applications [14].

V. Evaluation

In this section we report about an evaluation of SocialCast based on a social mobility model.

A. Simulation Settings

We evaluated the performance of our protocol using OMNeT++ [15], an open-source discrete event simulator written in C++.

1) Mobility Model: Traditionally, mobile wireless networks simulators assume a mobility model in which nodes move randomly in the space. This, however, does not suit our needs: *SocialCast* exploits prediction of co-location and movement, the use of a purely random mobility model would prevent an effective analysis of the protocol. To this end, we adopted the Community based mobility model [6], characterised by mobility patterns founded on social networks. The model is based on the following observation: in mobile networks, devices are usually carried by humans, so their movement is necessarily based on human decisions and social behaviour. To capture this type of behaviour, the model is based on the social network that links these individuals. In other words, the movements of both groups as well as single hosts is driven by social relationships.

The key problem is the generation of a synthetic social network with realistic characteristics in terms of clustering and average paths between the members of communities (i.e., clusters of nodes present in the social network). Our approach is based on the so-called Caveman model proposed by Watts in [16] to generate a network characterised by a realistic clustering degree. The social network is built starting from a certain number of fully connected graphs representing communities living in isolation, like primitive men in caves. According to this model, every edge of the initial network in input is re-wired to point to a node of another cave with a certain probability p. The re-wiring process is used to represent random interconnections between the communities. The result of this process is a square matrix representing the social network with a row (and a column) for each host. The value in position i, j describes the relationship between the hosts i and j. Real values between 0 and 1 are used to describe the intensity of the relationships among the individuals. 1 represents a very strong relationship, 0 no relationship at all.

The second step is the detection of the communities in the synthetic social network: in order to do so, we use the Girvan-Newman clustering algorithm [17]. The number of these communities may be lower than the initial caves used as seeds of the networks given the re-wiring phase. The generation of the social network and the detection of the communities take place during the setup phase of the simulation (i.e., at time 0).

The simulation area is divided into a grid formed by a certain number of squares. Each group detected using the Girvan-Newman clustering algorithm is then placed in one of these squares. Each host then selects a goal (i.e, a way-point like in the Random Way-Point model) inside the square and moves towards it in a straight line. When this goal is reached, the next goal is chosen inside the square associated to the group of hosts that exerts the highest "social" attraction towards it (including the current one). This group attraction is calculated by summing the values that express the intensity of the relationship between the hosts and the members of the community (extracted from the matrix that describes the social network). The hosts move towards the new goal in a straight line as before until they reach it. Then, we have a new decision point and the selection process of the new goal is repeated.

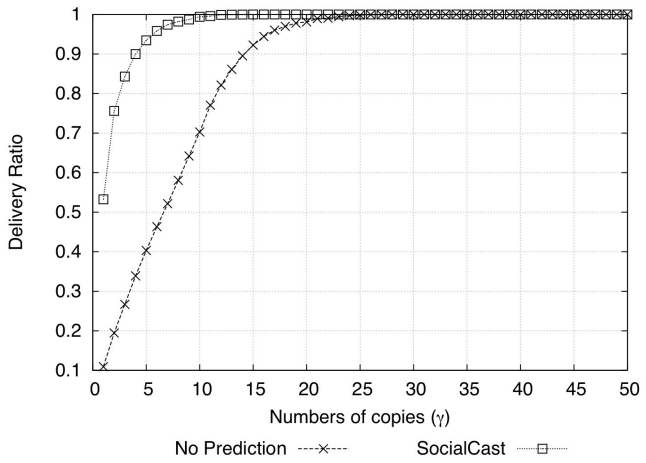
TABLE I
Simulation Parameters

Simulation Environment Parameters	**Default value**
Simulation area	$4\ km \times 4\ km$
Number of hosts	100
Hosts speed	$[1-6]\ m/s$
Transmission range	$250\ m$
Percentage of publishers	50 %
Percentage of subscribers	50 %
Publishing interval	$60\ s$
Number of interests	10
Simulation duration	8 hours
Protocol Parameters	**Default value**
Weight change degree of connectivity utility (w_{cdc})	0.25
Weight colocation utility (w_{col})	0.75
Buffer size (β)	∞
Number of copies (γ)	3
Size buffer last messages IDs received (λ)	100
Retransmission interval (T)	20 s
Hysteresis threshold (ϵ)	0.2

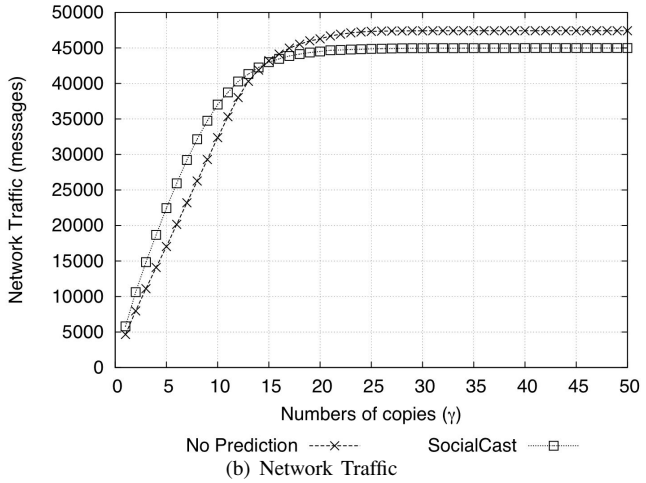
The model was validated against real traces provided by Intel Research [7]; in fact, this model is able to reproduce the distributions of inter-contacts time and contact duration that characterise this set of traces. More specifically, the model generates distributions that are scale-free in certain ranges of values as observed in these traces. At the same time, it is able to take into account the structure of the underlying human network [16].

2) Default Parameters: In real life, people sharing similar interests happen to be co-located more frequently among each others than with others. This property is crucial to our protocol as it can be exploited to perform accurate predictions over future movements of nodes. To reproduce this behaviour in our simulator, we map one interest to each community of the synthetic social network described in the previous section, such that nodes have more probability to be co-located with other nodes having the same interests. Moreover, we assumed that a node can subscribe to at most one interest: since messages belonging to different interests are routed independently, multiple interests do not bring new insights. Finally, publishers are uniformly chosen among all the nodes in the simulation space.

We assume that half of the nodes are subscribers and half, possibly with some overlapping, are publishers. The number of possible interests in the network is 10. The publishing interval is set to 60 s. To enable proper message dissemination, messages are published during the interval $[3000\ s, 3500\ s]$



(a) Delivery



(b) Network Traffic

Fig. 3. Delivery and overhead against number of replicas.

over a total period of $28800\ s$ (8 hours). A summary of all the simulation parameters is presented in Table I. The simulation area $4\ km \times 4\ km$ has been chosen in order to have a sufficiently sparse and disconnected network. The default values of the utility weights w_{cdc} and w_{col} are those providing the best performance in terms of delivery ratio in our simulations.

With respect to the Community based mobility model, in our default simulation scenario, the simulation area is $4km \times 4km$ and is divided into a grid—20×20. In the default scenarios we consider 10 initial caves with a re-wiring probability equal to 0.1 as in [6]. The re-wiring probability is a measure of the connectivity among individuals of different communities (i.e., the initial caves). 0.1 means that the probability of having a relationships between individuals of different communities is 0.1.

Finally, we averaged results over 20 runs, using different seeds for each scenario. We do not show the confidence interval in the graphs, since they are very small. In fact, the maximum standard deviation for the delivery is 0.016, whereas for the overhead is $3,212$.

To provide more insights, we compare *SocialCast* with a variant in which prediction is not used and where the next carrier is selected on a random basis. This variant is implemented as follows. As in *SocialCast*, each node periodically tries to retransmit the copies of the messages contained in its buffer. The mechanisms are identical to *SocialCast* except for the fact the node selects a random entry in the routing table (also considering itself). Then the message is sent to the selected node where it is stored (or is maintained in the buffer of the node) for a subsequent retransmission. Similarly to *SocialCast*, if subscribers are in the neighbourhood, messages are forwarded to them too. This enables us to assess the contribution of prediction.

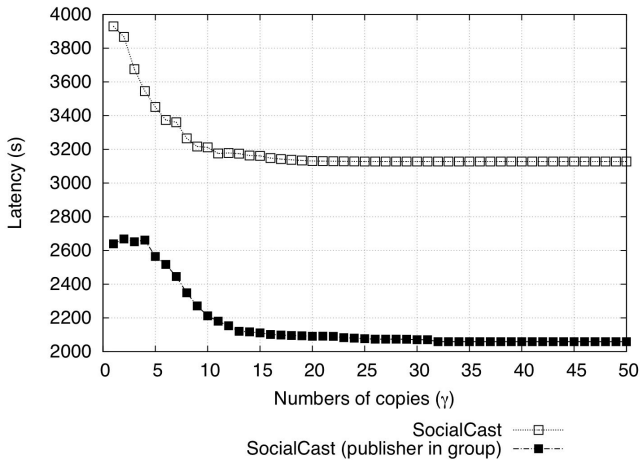


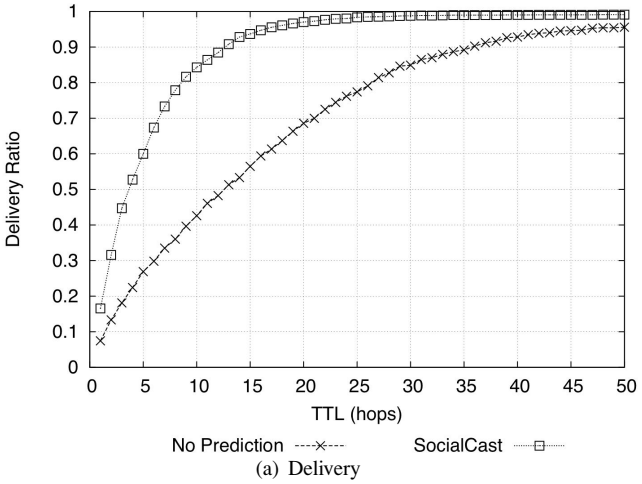
Fig. 4. Latency against the number of replicas.

B. Simulation Results

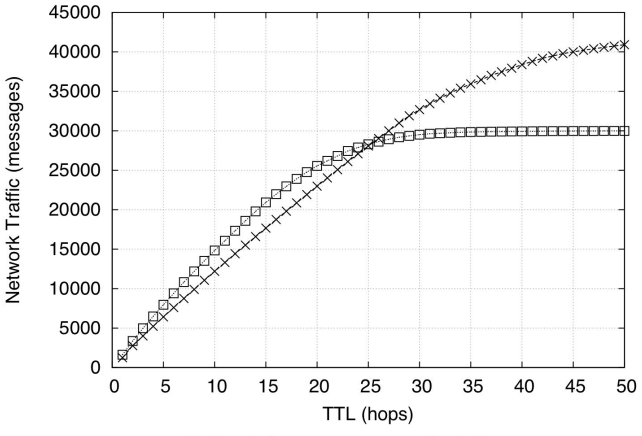
We now present the results of our simulations of *SocialCast*. In all our experiments, we mainly concentrate on message delivery ratio and network traffic. The former accounts for protocol *effectiveness* and is defined as the ratio between the actual number of messages delivered to the interested subscribers and the ideal one. The network traffic, instead, is constituted by the number of forwarded messages and measures the *efficiency* of the protocol.

1) Number of Replicas: The first parameter we studied is the number γ of replicas in the system. This is a key parameter, because it has a large impact on the network traffic. Results in Figure 3(a) show that, through prediction, *SocialCast* is able to achieve high message delivery with less replicas than the ones needed if prediction is not used. Indeed, 5 replicas are sufficient for *SocialCast* to reach more than 90% of subscribers while without prediction three times that number of replicas is needed to obtain similar performance. Notably, although delivery is greatly improved in *SocialCast* (e.g., with $\gamma = 5$ prediction boosts delivery from 40% up to 93%), the network traffic is not increased (see Figure 3(b)). The reason stems from the fact that network traffic strongly depends on the number of replicas. Therefore, since both *SocialCast* and its variant share the same γ, the traffics are similar. However, leveraging off predictions, *SocialCast* can select better carriers which enable reaching more subscribers, thus achieving better performance without increasing the traffic.

In these experiments, we conservatively assumed that publishers are uniformly chosen among all the nodes in the



(a) Delivery



(b) Network Traffic

Fig. 5. Delivery and overhead against time-to-live.

simulation space. However, we also replicated these experiments under a different configuration in which publishers of a given event are selected only among the subscribers for such event. We measured the impact on *SocialCast* performance; if no prediction is used, the performance are identical to the uniformly distributed scenario as the location of the publisher is irrelevant in this case. As expected, we did not observe significant differences in terms of delivery and overhead because, even if the message is published in a random location, our protocol is able to efficiently route it in a few hops towards the subscribers by properly identifying the best carriers. On the other hand, when the publishing node is also a subscriber for the published message, the dissemination occurs much faster because subscribers are naturally good carriers (their co-location is very high). Hence, as depicted in Figure 4, the average latency of *SocialCast* to reach a subscriber decreases considerably.

In the interest of space, from now on we will only show results in the uniformly distributed publishers. Similar trends are observable for the other case.

2) Time To Live (TTL): The Time To Live of a message (TTL) represents the dual parameter of γ, as it provides complementary information. Indeed, γ controls how many instances of the same message are around, while TTL defines

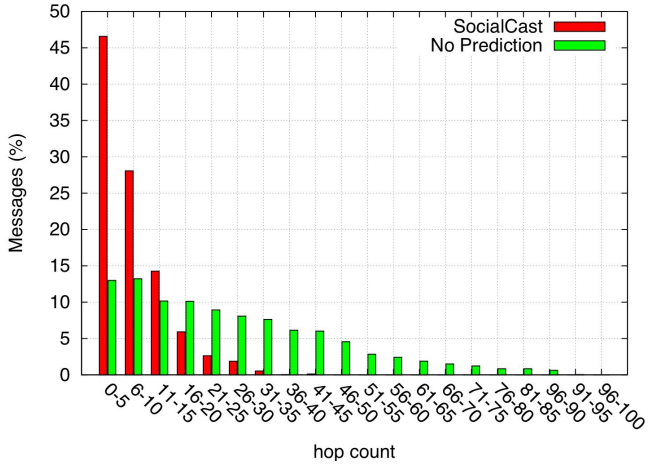


Fig. 6. Hop count distribution.

for how far, in terms of hops, a message will be around. Clearly, given a fixed γ, by increasing the number of possible hops, there are more chances to reach other subscribers. Unfortunately, this comes at the price of a higher overhead because the message will stay around longer. Figure 5 shows the performance of our protocol against different values of TTL. As expected, prediction enables decreasing the TTL because a message is forwarded only when needed, i.e., when a better carrier or a subscriber is encountered. Conversely, without prediction, messages are forwarded in a random fashion and hence more hops are needed to successfully contact the subscribers. This is confirmed in Figure 5(a): 15 hops per message are enough to *SocialCast* to reach more the 90% of subscribers while the variant without prediction requires at least 35 hops per message. Clearly, the two traffic curves show similar trends because the TTL (as well as γ) directly influences the traffic. Notably, however, the network traffic generated by *SocialCast* saturates for $TTL > 25$. Indeed, when all the best carriers and the subscribers have been reached (i.e., the delivery hits 100%), the messages are not replicated further. This happens for a value of TTL equal to 25. The traffic generated by the variant without prediction, instead, increases linearly with the TTL because it continuously forwards messages, also when not needed. This is a result of paramount importance because it demonstrates that our protocol does not waste resources generating additional traffic when not needed.

This behavior is also confirmed in Figure 6 in which we plot the distribution of the number of hops needed to reach a subscriber, when no TTL is used. As expected, in *SocialCast* the vast majority of messages are delivered to subscribers after few hops and, notably, none required more than 31 hops. Conversely, when no prediction is used, the number of hops increases significantly (up to 93 hops).

3) Buffer Size: Beside network traffic, another key factor to observe is represented by memory consumption. Since *SocialCast* is meant to run on handheld devices, optimising memory is a mandatory task. To investigate this aspect, in Figure 7 we plot a snapshot of the distribution of buffer sizes after[2] $10,000s$. Results confirm our expectations: since *SocialCast*

[2]We took snapshots also at other instants and we observed similar distributions.

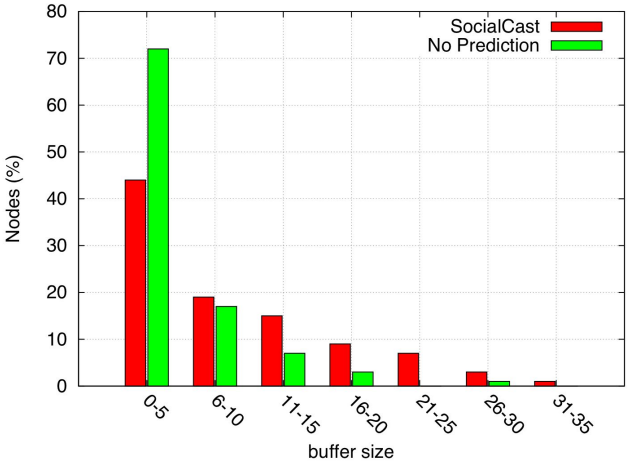


Fig. 7. Buffer size distribution.

is able to identify the best carriers for each message, messages will be stored only on a small subset of nodes, thus enabling an efficient usage of memory resources. Indeed, almost half nodes store less than 5 message in their buffer while only the 2% of nodes hold more than 30 messages. This shows that our protocol is able to select the best carriers and to limit message buffering only on few specific nodes. Instead, as far as the random protocol is concerned, as expected, the large majority of the hosts has a buffer occupation under 10. This means that there are no particular hosts that are selected for their characteristics as good carriers.

4) Host Speed: We also evaluated our approach with respect to the speed of nodes. Fast carriers are preferable over slow ones because they can significantly increase message dissemination. Rather than the node degree, it is the number of encountered nodes per unit of time which makes the difference. This is taken into account by the change degree of connectivity, shown in Figure 8(c). In any case, faster nodes ensure better performance in terms of delivery latency. In particular, the two weights w_{cdc} and w_{col} play a major role in this respect because the former accounts for node mobility (change degree of connectivity) while the latter focuses on node co-location. To assess their impact, we performed a number of experiments with various combinations of weights, using different speeds. We can appreciate the impact of the choice of the values of the weights only with respect to speed variations. To recreate the heterogeneity typical of real scenarios, we set nodes' speed in a range between $1m/s$ and an upper bound S which we varied from 1 to $20m/s$. Results are provided in Figure 8. Not surprisingly, co-location is essential to ensure proper delivery. Indeed, given the social network underneath, if a node has been co-located with another node with a given interest, it is highly likely that it will be co-located again in a near future with another node sharing the same interest. This statement is fully supported by Figure 8(a): when co-location is not taken into account (i.e., $w_{col} < 0.5$) delivery drops[3]. On the other hand, network traffic is not significantly impacted by weights because it mostly depends on the value of γ and TTL (see Figure 8(b)). This yields that,

[3]Note that $w_{col} = x$ means $w_{cdc} = 1 - x$ since we considered only two weights normalized in order to have a sum of weights equal to 1.

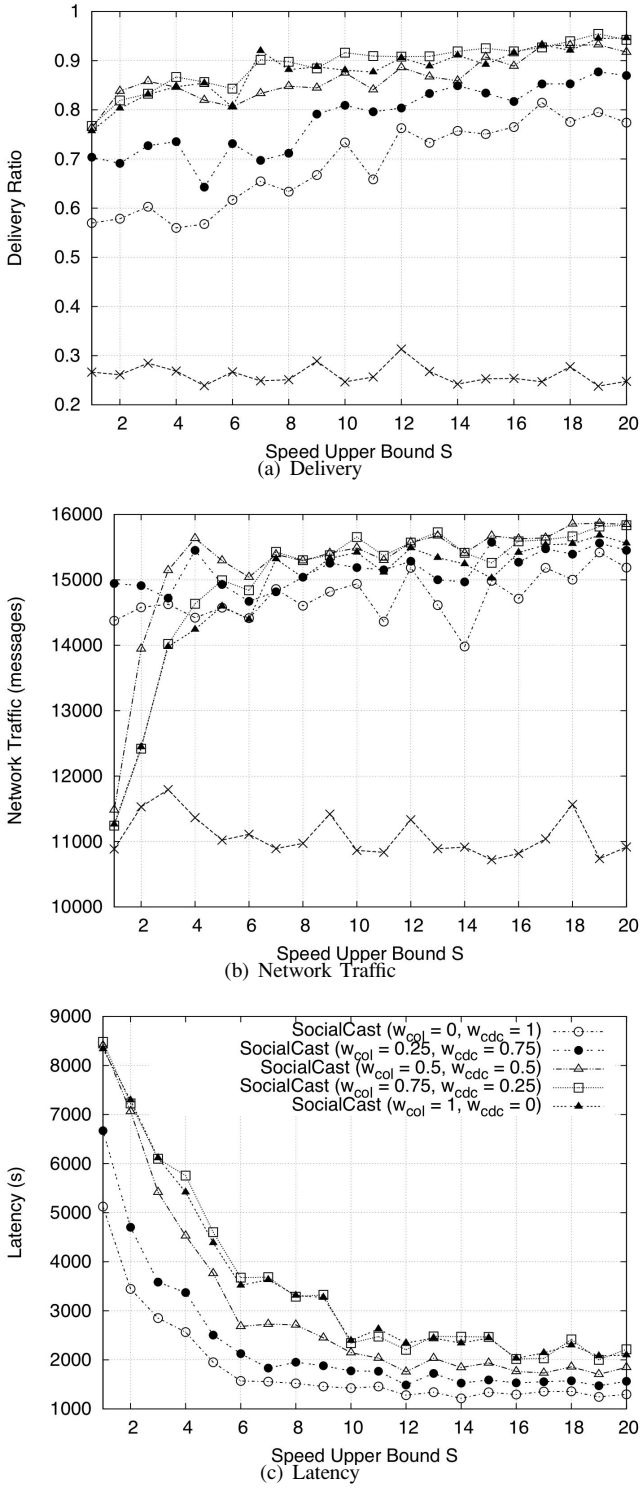
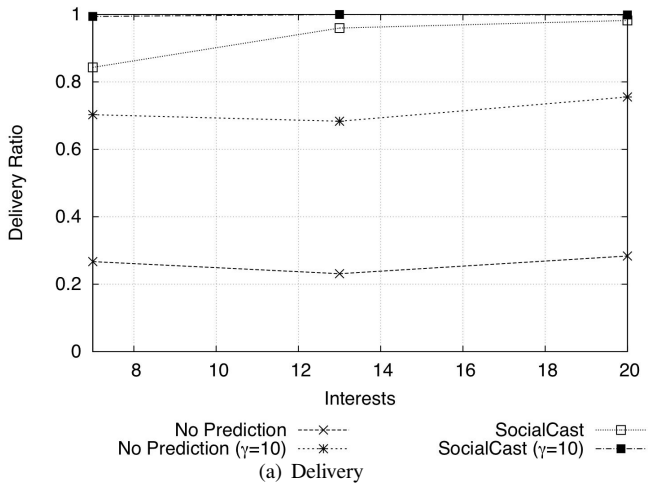


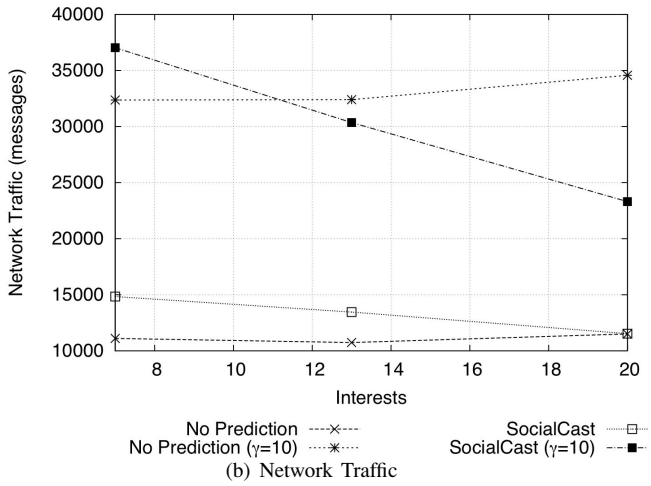
Fig. 8. Delivery and overhead against the lower bound of the speed (lower is always 1).

while not significantly impacting on the traffic, co-location is of paramount importance to correctly steer messages towards subscribers.

Looking at these results, one might argue that the utility related to the change degree of connectivity (U_{cdc}) is of no use, because in terms of delivery and network traffic there is no appreciable difference between $w_{cdc} = 0.5$ (corresponding to $w_{col} = 0.5$) and $w_{cdc} = 0$ (corresponding to $w_{col} = 1$). Never-

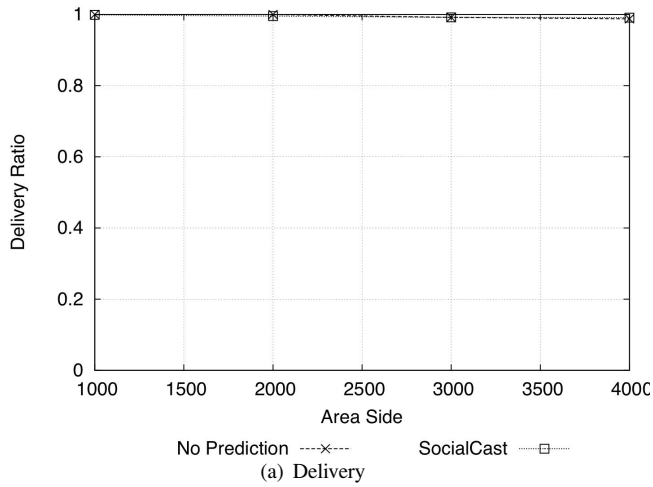


(a) Delivery

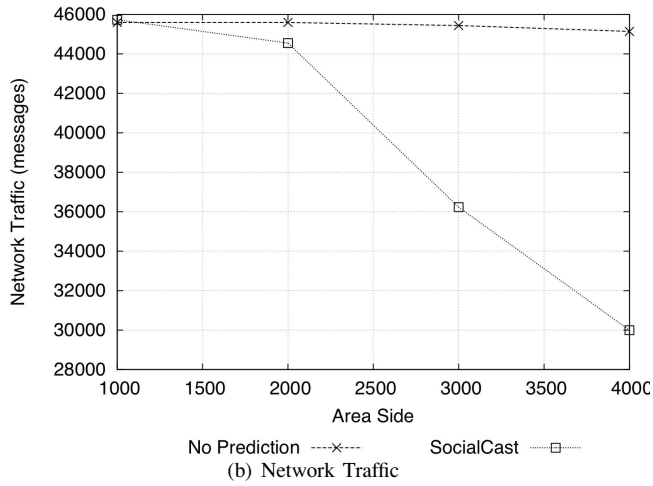


(b) Network Traffic

Fig. 9. Delivery and overhead against the number of interests.



(a) Delivery



(b) Network Traffic

Fig. 10. Delivery and overhead against side length.

theless, taking into account also node mobility is beneficial in terms of latency. Indeed, as charted in Figure 8(c), the ability to identify faster carriers allows for a faster delivery. For instance, when the maximum speed is $6m/s$, using $w_{cdc} = 0.5$ instead of $w_{cdc} = 0$ provides a reduction of the average delivery latency from $4000s$ to less than $3000s$. Clearly, when the average speeds become higher (i.e., when S increases), the role of U_{cdc} becomes less evident because all nodes move reasonably fast.

5) Interests: An important feature of our approach is the ability to adapt to network conditions and to avoid generating traffic when not needed. We already shown in Figure 5(b) that *SocialCast* does not forward additional messages unless required. To further demonstrate this, we run a set of experiments with different numbers of interests. Since in our simulations we assumed at most one interest per node, increasing the number of interests yields a lower number of recipients per message. The aim is to show that our protocol is able to autonomously tune the traffic generated according to the popularity of the interest, which, of course, is unknown to the publisher.

Charts in Figure 9 support this claim. While delivery is always well above 80%, *SocialCast*, in the default configuration ($\gamma = 3$), has considerably smaller overhead, $11,000$ instead of $15,000$, when the number of interests increases from 7 to 20. Note that this difference is not caused by the smaller number of messages forwarded to subscribers but, instead, it is a consequence of the fact that the fewer the subscribers are the fewer nodes have been co-located with them and, hence, the fewer good carriers will be around. This can be proved by looking at the performance when no prediction is used: the generated traffic remains constant because, without prediction, the protocol has no indications about when (an to which neighbour) a node needs to forward a buffered message.

Looking at the chart in Figure 9(a), one might wonder why delivery increases with the number of interests. The reason lies in the number of interested nodes per message: the fewer interests there are, the more nodes must receive the message. Hence, $\gamma = 3$ is not sufficient to contact all the interested nodes. Indeed, increasing γ to 10 enables contacting all the interested nodes[4]. Conversely, when the number of interest increases (i.e., when the number of recipients per message decreases), $\gamma = 3$ is enough to achieve full delivery. It is interesting to note, however, that even with $\gamma = 10$, we have the same decreasing trend in the network traffic, confirming what we argued above.

[4]As shown in Figure 5(a), increasing TTL instead of γ would have achieved the same result.

6) Density: Finally, the last parameter we investigated is the area density. Clearly, in a dense area the number of forwarded messages will be higher than in a sparse one because multiple routes will be available. Hence, to enable a fair comparison among the different scenarios, we run our experiments without imposing any TTL to remove any bias. Interestingly, since messages can propagate for ever, both protocols, with and without prediction, achieve full delivery (as shown in Figure 10). Nevertheless, *SocialCast* outperforms the variant with no prediction because, although the forwarded messages are not limited, it is able to decrease the network traffic when not needed, i.e., when the network becomes sparse and it is crucial to identify the good carriers.

All the aforementioned results confirm the suitability of *SocialCast* for our scenarios and demonstrate the improvements introduced by our prediction mechanisms. Indeed, thanks to prediction, *SocialCast* performs more accurate selection and provides a more efficient usage of resources, both in terms of network traffic and memory. In addition, the ability to predict over the co-location and the node mobility allows for maintaining a very high and steady event delivery with a reasonably low traffic and latency.

VI. Related Work

To our knowledge, there is very little work addressing the use of publish-subscribe paradigms for delay-tolerant networking. Most research projects about routing in Delay Tolerant Networks (DTN) [18] have focused on unicast [5], [19] and on opportunistic infrastructure-less dissemination [7], [20]. In [21], Lindgren et al. propose a probabilistic routing approach to enable asynchronous communication among intermittently connected groups of hosts. The calculation of the delivery probabilities is based, somewhat simplistically, on the period of time of colocation of two hosts and not on a forecasted colocation probability. Zhao et al. in [22] discuss the so-called Message Ferrying approach for message delivery in mobile ad hoc networks. The authors propose a proactive solution based on the exploitation of highly mobile nodes called ferries. These nodes move according to pre-defined routes, carrying messages between disconnected portions of the network. Our approach does not assume the a priori knowledge of the movement of the potential carriers, but it is able to infer it by evaluating the history of colocation with the other hosts. The Context-aware Adaptive Routing (CAR) protocol is presented in [5]. The approach has quite a refined model of prediction over time series which has inspired the forecasting-based techniques adopted in this work. However, CAR deals exclusively with unicasting and its forwarding model is based on addresses and not on interests. Recently, the key issue of the selection of message carriers has been addressed in [23] by formulating it as a resource allocation problem.

Approaches based on the replication of messages on all hosts have also been presented, such as epidemic-style techniques [24] that exploit a virus-spreading metaphor to indicate the dissemination of information to all. The problem of broadcasting in delay tolerant networks has also been studied in [3], where an analysis with different mobility models is presented. Solutions which exploit a more constrained number of message copies have exploited erasure coding techniques [25] to improve performance in terms of delivery ratio given a certain degree of redundancy in the system [26], [27].

While broadcasting has attracted a lot of the researchers interest, the work presented in [2] concentrates on DTN multicast routing and temporal issues for delay tolerant networking, trying to account for temporal group membership. More recently, Yoneki et al. in [28] discuss the design of a publish-subscribe communication overlay based on the distributed detection of social groups by means of centrality measures [29]. This system relies on the detection of communities for event notification, whereas our approach is based on contacts between pairs of hosts only. In other words, it assumes a previous knowledge of all the social ties between all the individuals carrying the devices and the emerging community structures before starting the communication process.

In terms of opportunistic unicast networking in human networks, social ties have been exploited to support communication in Pocket-Switched Networks: for example, LABEL [30] exploits clustering algorithms to group nodes in communities by evaluating their colocation patterns. Messages sent to a certain recipient are forwarded to hosts of the same community, since these have a higher probability of getting in reach of the recipient in the future. However, the model requires that every node of the network is statically "tagged", i.e., associated to a certain community. Another example of unicast routing in intermittently connected mobile ad hoc networks founded on social network concepts is SOLAR [31] that exploits macro-mobility patterns between groups of nodes that are detected using machine learning techniques. More recently in [32] the authors use social network analysis to extract communities ties among the individuals carrying the devices. This a priori knowledge of the structure of the underlying social network is then used as a basis of the routing decisions. *SocialCast* instead implements a one-to-many communication paradigm and is based on colocation and movement patterns rather than on an explicit analysis of the emergent clustering of hosts into communities.

Solutions to support communication by means of a publish/subscribe paradigm have been proposed for large-scale wired networks. Most of these approaches target fixed networks of brokers, with some notable exceptions (e.g., [33]) supporting also client mobility. These solutions, however, are clearly not suitable to our scenario, in which no fixed infrastructure can be assumed. In the close area of MANETs, there has been a consistent body of work concerning multicast communication. However, results are not directly reusable given the peculiarity posed by content-based routing, which instead has been addressed by very few works in literature. Content Based Multicast (CBM) [34] and STEAM [35] provide a notion of *spatial scope* which defines the area messages are propagated within. In particular, CBM allows publishers to specify the direction and the distance a message is spread. Similarly, STEAM limits the message propagation to a proximity area, inside which messages are broadcast and locally matched against subscriptions. With respect to these approaches, our protocol enjoys wider applicability, as messages are delivered throughout the network, based on node interests, regardless of their locations. Autonomous

Gossip [36] shares an idea similar to ours, by pushing messages towards potential receivers in a content-based fashion, according to node "similarities". However, the authors neither give details on how this notion of similarity is actually computed and disseminated, nor provide quantitative analysis on the performance of their protocol. Another closely related approach is discussed in [37], where nodes keep track of the last time they have been in range of others with the same interests. This information is used to select the best forwarders to deliver a message. This prediction mechanism is more primitive and coarse-grained than ours, therefore implying more inaccurate message forwarding. Moreover, the approach assumes a very simple mobility model and offers no support for disconnected operation. In [38], an approach to publish-subscribe for MANETs based on a combination of probabilistic and deterministic information dissemination techniques is presented: *SocialCast* instead uses a prediction based mechanism to drive the spreading, combined with store-and-forward to cope with intermittently connected networks.

Finally, we believe that *SocialCast* can be an example of protocol supporting new abstractions for modern communication systems inspired by the publish/subscribe paradigm like those proposed in [1].

VII. Conclusions and Future Work

This paper presented *SocialCast*, a interest-based routing protocol to support delay tolerant communication in human networks. The approach assumes that socially bound hosts are likely to be co-located regularly: these colocation patterns are then used to efficiently route the messages from publishers to interested subscribers. The social ties selection is made by taking into account predictions about contextual parameters (e.g., colocation and mobility patterns), based on previous observations. We have evaluated our approach in realistic scenarios with disconnections to demonstrate the advantages of the prediction and store and forward strategies in terms of message delivery, delay and overhead. We would like to be able to test the performance of *SocialCast* with a higher number of interests: this implies the re-engineering of the mobility framework of our simulator. We also plan to perform an analytical evaluation of the protocol. The key issue is to characterise network connectivity given the fact that the underlying connectivity graph is time-variant dependent on the mobility model.

We plan to implement *SocialCast* on the Haggle platform [39] as a Forwarding Algorithm. Up to our knowledge this will be the first available protocol to support one-to-many communication for this framework. We also plan to refine the subscription model, allowing for more refined content representation, and to design filtering mechanisms based on it. The integration of temporal validity constraints in the subscription semantics is among our current research directions. However, we observe that these aspects are orthogonal to the forwarding mechanisms presented in this article. Future work will also address the dynamic adaptation of the number of replicas to network conditions, as well as the inclusion of additional contextual information in our predictions as encompassed by the general framework we described in Section IV. This will enable further improvements, as well as the deployment of our technique to related fields, e.g., wireless sensor networks.

Acknowledgment

The work is partially supported by projects EPSRC CREAM and ESF MINEMA. Mirco Musolesi is supported from a research program in the Institute for Security Technology Studies at Dartmouth College, under award 60NANB6D6130 from the U.S. Department of Commerce. The statements, findings, conclusions, and recommendations are those of the authors and do not necessarily reflect the views of the National Institute of Standards and Technology (NIST) or the U.S. Department of Commerce.

Appendix

In this appendix we present the forecasting model used for the prediction of context information in our protocol.

A state space model for a time series $\mathbf{Y}_t$ consists of two equations. The first one called the *observation equation* is the following

$$\mathbf{Y}_t = G_t\mathbf{X}_t + \mathbf{W}_t \quad t = 1, 2, ...$$

with $\mathbf{W}_t$ defined as[5]

$$\mathbf{W}_t = WN(0, R_t)$$

This equation defines the w-dimensional observation $\{\mathbf{Y}_t\}$ as a linear function of a v-dimensional state variables $\{\mathbf{X}_t\}$ and a noise term. The second one is the *state equation* defined as follows

$$\mathbf{X}_{t+1} = F_t\mathbf{X}_t + \mathbf{V}_t \quad t = 1, 2, ...$$

with $\mathbf{V}_t$ defined as

$$\mathbf{V}_t = WN(0, Q_t)$$

This equation determines the state $\mathbf{X}_{t+1}$ at time $t + 1$ in terms of the previous state $\mathbf{X}_t$ and a noise term. Let w as the dimension of $\mathbf{Y}_t$ and v as the dimension of $\mathbf{X}_t$, $\{G_t\}$ is a sequence of $w \times v$ matrices and $\{F_t\}$ is a sequence of $v \times v$ matrices. We assume that $\{\mathbf{V}_t\}$ is uncorrelated with $\{\mathbf{W}_t\}$, even if a more general form of the state space model allows for correlation between these two variables. Analytically, we can rewrite this condition as follows

$$E(\mathbf{W}_s\mathbf{V}_t^T) = 0 \quad \forall\, s, t$$

We also assume that the initial state $\mathbf{X}_1$ is uncorrelated with all of the noise terms $\{\mathbf{V}_t\}$ and $\{\mathbf{W}_t\}$.

With the notation of $P_t(\mathbf{X})$ we refer to the best linear predictor (in the sense of minimum mean-square error) of $\mathbf{X}$ in terms of $\mathbf{Y}$ at the time t. $P_t(\mathbf{X})$ is defined as follows

$$P_t(\mathbf{X}) \equiv \left[\ P_t(X_1) \quad ... \quad P_t(X_v)\ \right]^T$$

where

$$P_t(X_i) \equiv P(X_i|\mathbf{Y}_0, \mathbf{Y}_1, ..., \mathbf{Y}_t)$$

[5]WN stands for White Noise, which is a sequence of uncorrelated random variables X_t, each with the same mean and variance σ^2. Therefore, it is also an example of stationary time series. More specifically, the notation $WN(0, \{R_t\})$ indicates white noise with zero mean and variance R_t.

$P(X_i|\mathbf{Y}_0, \mathbf{Y}_1, ..., \mathbf{Y}_t)$ indicates the best predictor of X_i given $\mathbf{Y}_0, ..., \mathbf{Y}_t$. We can also observe that $P_t(\mathbf{X})$ has the following form

$$P_t(\mathbf{X}) = A_0\mathbf{Y}_0 + ... + A_t\mathbf{Y}_t$$

since it is a linear function of $\mathbf{Y}_0, ..., \mathbf{Y}_t$. It is possible to prove [4] for the state space model discussed in the previous section that the one-step predictor

$$\widehat{\mathbf{X}}_t \equiv P_{t-1}(\mathbf{X}_t)$$

and their error covariance matrices

$$\Omega_t = E[(\mathbf{X}_t - \widehat{\mathbf{X}}_t)(\mathbf{X}_t - \widehat{\mathbf{X}}_t)^T]$$

are determined by these initial conditions

$$\widehat{\mathbf{X}}_1 = P(\mathbf{X}_1|\mathbf{Y}_0)$$

$$\Omega_1 = E[(\mathbf{X}_1 - \widehat{\mathbf{X}}_1)(\mathbf{X}_1 - \widehat{\mathbf{X}}_1)^T]$$

and these recursive equations

$$\widehat{\mathbf{X}}_{t+1} = F_t\widehat{\mathbf{X}}_t + \Theta_t\Delta_t^{-1}(\mathbf{Y}_t - G_t\widehat{\mathbf{X}}_t)$$

$$\Omega_{t+1} = F_t\Omega_tF_t^T + Q_t - \Theta_t\Delta_t^{-1}\Theta_t^T$$

where

$$\Delta_t = G_t\Omega_tG_t^T + R_t$$

$$\Theta_t = F_t\Omega_tG_t^T$$

As estimation model, we use a basic state space model composed of the following two scalar equations

$$Y_t = X_t + W_t \quad t = 1, 2, ...$$

with

$$W_t = WN(0, Q_t)$$

and

$$X_{t+1} = X_t + V_t \quad t = 1, 2, ...$$

with

$$V_t = WN(0, R_t)$$

With respect to the Kalman filter prediction, we can consider a mono-dimensional system with

$$G_t = [1]$$

$$F_t = [1]$$

Therefore, we can derive the recursive equations of the Kalman filter for the prediction of the values of this series. Given the previous observed value Y_t and the predicted value at time t, $\widehat{X}_t$, the recursive equation for the determination of the predicted value at time $t+1$ is

$$\widehat{X}_{t+1} = \widehat{X}_t + \frac{\Omega_t}{\Omega_t + R_t}(Y_t - \widehat{X}_t)$$

with

$$\Omega_{t+1} = \Omega_t + Q_t - \frac{\Theta_t^2}{\Omega_t + R_t}$$

Since in this case

$$\Omega_t = \Theta_t$$

we can also write

$$\Omega_{t+1} = \Omega_t + Q_t - \frac{\Omega_t^2}{\Omega_t + R_t}$$

REFERENCES

[1] M. Demmer, K. Fall, T. Koponen, and S. Shenker, "Towards a Modern Communications API," in *Proc. 6^{th} Workshop on Hot Topics in Networks (HotNets-VI)*, Atlanta, GA, November 2007.

[2] W. Zhao, M. Ammar, and E. Zegura, "Multicasting in delay tolerant networks: semantic models and routing algorithms," in *Proc. 2005 ACM SIGCOMM workshop on Delay-tolerant networking (WDTN'05)*. New York, NY, USA: ACM Press, 2005, pp. 268–275.

[3] G. Karlsson, V. Lenders, and M. May, "Delay-tolerant broadcasting," in *Proc. 2006 SIGCOMM workshop on Challenged networks (CHANTS'06)*. New York, NY, USA: ACM Press, 2006, pp. 197–204.

[4] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*. Springer, 1996.

[5] M. Musolesi, S. Hailes, and C. Mascolo, "Adaptive Routing for Intermittently Connected Mobile Ad Hoc Networks," in *Proc. 6th International Symposium on a World of Wireless, Mobile, and Multimedia Networks (WoWMoM'05). Taormina, Italy.* IEEE press, June 2005.

[6] M. Musolesi and C. Mascolo, "Designing mobility models based on social network theory," *ACM SIGMOBILE Mobile Computing and Communication Review*, vol. 11, no. 3, July 2007.

[7] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of Human Mobility on the Design of Opportunistic Forwarding Algorithms," in *Proceedings of INFOCOM'06*, April 2006.

[8] M. Mcpherson, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, vol. 27, no. 1, pp. 415–444, 2001. [Online]. Available: http://arjournals.annualreviews.org/doi/abs/10.1146/annurev.soc.27.1.415

[9] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proc. WDTN'05*. New York, NY, USA: ACM Press, 2005, pp. 252–259.

[10] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. ASME J. Basic Engineering*, March 1960.

[11] R. Keeney and H. Raiffa, *Decisions with Multiple Objectives: Preference and Value Tradeoffs*. Wiley, 1976.

[12] M. Musolesi and C. Mascolo, "Evaluating context information predictability for autonomic communication," in *Proc. 2nd IEEE Workshop on Autonomic Communications and Computing (ACC'06). Co-located with 7th IEEE Int. Symp. WoWMoM'06*. Niagara Falls, NY: IEEE Computer Society Press, June 2006.

[13] C. Chatfield, *The Analysis of Time Series An Introduction*. Chapman and Hall, 2004.

[14] B. Pasztor, M. Musolesi, and C. Mascolo, "Opportunistic Mobile Sensor Data Collection with SCAR," in *Proc. 4th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'07)*. Pisa, Italy: IEEE Press, October 2007.

[15] A. Varga, "The OMNeT++ discrete event simulation system," in *Proc. ESM'2001*, Prague, 2001.

[16] D. J. Watts, *Small Worlds The Dynamics of Networks between Order and Randomness*, ser. Princeton Studies on Complexity. Princeton University Press, 1999.

[17] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, February 2004.

[18] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of SIGCOMM'03*, August 2003.

[19] S. Jain, K. Fall, and R. Patra, "Routing in a Delay Tolerant Network," in *Proceedings of SIGCOMM'04*. ACM Press, 2004, pp. 145–158.

[20] J. Su, A. Goel, and E. de Lara, "An Empirical Evaluation of the StudentNet Delay Tolerant Network," in *Proc. MOBIQUITOUS'06*, San Jose, California, July 2006.

[21] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic Routing in Intermittently Connected Networks," *Mobile Computing and Communications Review*, vol. 7, no. 3, July 03.

[22] W. Zhao, M. Ammar, and E. Zegura, "A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks," in *Proc. MobiHoc'04*, May 2004.

[23] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN Routing as a Resource Allocation Problem," in *Proc. SIGCOMM'07*, August 2007.

[24] A. Vahdat and D. Becker, "Epidemic routing for Partially Connected Ad Hoc Networks," Department of Computer Science, Duke University, Tech. Rep. CS-2000-06, 2000.

[25] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proc. SIGCOMM'98*, 1998, pp. 56–67. [Online]. Available: citeseer.ist.psu.edu/byers98digital.html

[26] Y. Wang, S. Jain, M. Martonosi, and K. Fall, "Erasure-coding based routing for opportunistic networks," in *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay Tolerant Networking (WDTN'05)*. New York, NY, USA: ACM Press, 2005, pp. 229–236.
[27] S. Jain, M. Demmer, R. Patra, and K. Fall, "Using redundancy to cope with failures in a delay tolerant network," in *Proc. SIGCOMM'05*. New York, NY, USA: ACM Press, 2005, pp. 109–120.
[28] E. Yoneki, P. Hui, S. Chan, and 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)*. New York, NY, USA: ACM, 2007, pp. 225–234.
[29] P. Hui, E. Yoneki, S.-Y. Chan, and J. Crowcroft, "Distributed community detection in delay tolerant networks," in *Proc. ACM SIGCOMM MobiArch'07*, August 2007.
[30] P. Hui and J. Crowcroft, "How small lables create big improvements," in *Proc. IEEE ICMAN'07*, March 2007.
[31] J. Ghosh, S. J. Philip, and C. Qiao, "Sociological Orbit aware Location Approximation and Routing (SOLAR) in MANET," *Elsevier Ad Hoc Networks Journal*, vol. 5, no. 2, pp. 189–209, March 2007.
[32] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant MANETs," in *Proceedings of the 8th ACM International Symposium on Mobile ad hoc networking and computing (MobiHoc'07)*. New York, NY, USA: ACM Press, 2007, pp. 32–40.
[33] I. Burcea, H.-A. Jacobsen, E. de Lara, V. Muthusamy, and M. Petrovic, "Disconnected Operation in Publish/Subscribe Middleware," in *Proceedings of the 5^{th} IEEE International Conference on Mobile Data Management (MDM'04)*, Los Alamitos, CA, USA, 2004.
[34] H. Zhou and S. Singh, "Content based multicast (CBM) in ad hoc networks," in *Proc. MobiHoc'00*, August 2000.
[35] R. Meier and V. Cahill, "STEAM: Event-Based Middleware for Wireless Ad Hoc Networks," in *Proc. 1^{st} International Workshop on Distributed Event-Based Systems*, Jul. 2002. [Online]. Available: http://computer.org/proceedings/icdcsw/1588/15880639abs.htm,http://citeseer.nj.nec.com/550381.html
[36] A. Datta, S. Quarteroni, and K. Aberer, "Autonomous Gossiping: A self-organizing epidemic algorithm for selective information dissemination in mobile ad-hoc networks," in *Proc. International Conference on Semantics of a Networked World*, June 2004.
[37] R. Baldoni, R. Beraldi, G. Cugola, M. Migliavacca, and L. Querzoni, "Content-based routing in highly dynamic mobile ad hoc networks," *J. Pervasive Computing and Communication*, vol. 1, no. 4, 2005.
[38] P. Costa and G. P. Picco, "Semi-probabilistic Content-Based Publish-Subscribe," in *Proc. 25^{th} IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, Columbus (Ohio, USA), June 2005.
[39] J. Su, J. Scott, P. Hui, E. Upton, M. H. Lim, C. Diot, J. Crowcroft, A. Goel, and E. de Lara, "Haggle: Clean-Slate Networking for Mobile Devices," Tech. Rep. UCAM-CL-TR-680, January 2007.

Paolo Costa is currently a Postdoctoral Researcher with the Department of Computer Science, Vrije Universiteit, Amsterdam. He holds an MSc (2002) and a Ph.D. (2006) degree in Computer Engineering from the Politecnico di Milano, Italy. His research interests include large scale distributed systems, wireless sensor networks, vehicular information dissemination, and gossip-based protocols. Further details are available at http://www.cs.vu.nl/∼costa.

Cecilia Mascolo is an EPSRC Advanced Research Fellow and a University Lecturer in the Computer Laboratory, University of Cambridge. Prior to this, she was with the Department of Computer Science, University College London. She holds an MSc (1995) and a Ph.D. (2001) in Computer Science from University of Bologna (Italy). She has been a visiting fellow in Washington University in St. Louis in 1998. She has published extensively in the areas of opportunistic mobile network routing, realistic mobility models exploiting social theory, mobile sensor networks, middleware for pervasive and context- aware systems. Dr. Mascolo is currently working on EPSRC, EU and industry funded projects on opportunistic routing for mobile and sensor networks and embedded systems middleware, with applications in wildlife monitoring, emergency rescue operations and vehicular information dissemination. Dr. Mascolo has served as a Technical Programme Committee member in many middleware, software engineering, mobile system, delay tolerant network ACM and IEEE conferences and workshops. More details of her profile are available at http://www.cl.cam.ac.uk/∼cm542.

Mirco Musolesi is a Postdoctoral Researcher at the Department of Computer Science, Dartmouth College, NH, USA. He is a Fellow of the Institute of Security Technology Studies at Dartmouth. Previously, he has been a Research Fellow at University College London, University College London (2005-2007). He also spent a research period at INRIA Rocquencourt, France in 2003. He holds a PhD in Computer Science from University College London, United Kingdom (2007) and a MSc in Electronic Engineering from the University of Bologna, Italy (2002). His research interests include delay tolerant networking, mobile networking and systems, wireless sensor systems, mobility modeling and social network based systems. More information about his profile and his research work can be found at http://www.cs.dartmouth.edu/∼musolesi.

Gian Pietro Picco is an Associate Professor in the Dipartimento di Ingegneria e Scienza dell'Informazione (DISI) at University of Trento, Italy. Previously, he has been on the faculty of Washington University in St. Louis, MO, USA (1998-1999) and Politecnico di Milano, Italy (1999-2006). He holds a Ph.D. from Politecnico di Torino, Italy (1998). The goal of his current research is to ease the development of modern distributed systems through the design and implementation of appropriate programming abstractions and of communication protocols efficiently supporting them. His work spans the research fields of software engineering, middleware, and networking, and is oriented in particular towards wireless sensor networks, mobile computing, and large-scale distributed systems. More information at http://disi.unitn.it/∼picco.