HiBOp: a History Based Routing Protocol for Opportunistic Networks

Chiara Boldrini, Marco Conti, Iacopo Iacopini, Andrea Passarella
IIT-CNR, Pisa, Italy
chiara.boldrini@iit.cnr.it, m.conti@iit.cnr.it, iacopo@whoopy.it, a.passarella@iit.cnr.it

Abstract

In opportunistic networks the existence of a simultaneous path between a sender and a receiver is not assumed. This model (which fits well to pervasive networking environments) completely breaks the main assumptions on which MANET routing protocols are built. Routing in opportunistic networks is usually based on some form of controlled flooding. But often this results in very high resource consumption and network congestion. In this paper we advocate context-based routing for opportunistic networks. We provide a general framework for managing and using context for taking forwarding decisions. We propose a context-based protocol (HiBOp), and compare it with popular solutions, i.e., Epidemic Routing and PROPHET. Results show that HiBOp is able to drastically reduce resource consumption. At the same time, it significantly reduces the message loss rate, and preserves the performance in terms of message delay.

1. Introduction

Opportunistic networks are one of the most interesting evolutions of classic Mobile Ad Hoc Networks (MANET). The main assumption of MANET environments is that a sender and a destination are connected to the network *at the same time*. If the destination is not connected when the sender wishes to transmit messages, they get dropped at some point of the network. However, in a pervasive networking environment, nodes will be seldom connectable at the same time through a multi-hop path. For example, devices that users carry with them might be only sporadically attached to the Internet, e.g. when the user moves close to an Access Point. In other words, it is foreseeable a scenario in which a large number of wireless devices and limited-size networks will be *just occasionally* connected to each other. Opportunistic networks aim at make users able to exchange data even in such a disconnected environment, by opportunistically exploiting any nearby device to move messages closer to the final destination. To this end, legacy protocols designed for MANET should be drastically redesigned [2],[3],[10]. Currently, envisioning routing and forwarding protocols[1] for opportunistic networks is one of the most exciting topics [8].

In opportunistic networks, the traditional routing paradigm of Internet and MANET, in which routes are computed based exclusively on topological information, is not adequate anymore. A first approach to routing in opportunistic networks is some variation of controlled flooding: Messages are flooded with limited Time-To-Live (TTL), and delivered to the destination as soon as it gets in touch with some node that received the message during the flood [11]. More advanced proposals replace topological information with higher-level information, trying to limit the cost of flooding. For example, PROPHET [5] forwards messages through nodes with increasing probability of encountering the destination.

More in general, we believe that topological information should be complemented with context awareness. Context is usually quite a loose concept in computer engineering. We see it as a collection of information that describes the reality in which the user lives, and the history of social relationships among users. For example, the context could be defined by personal information about the user (e.g., name), about her residence (e.g., address), about her work (e.g., institution), about her hobbies (e.g., address of the sport facilities she goes to). The routing protocol could, for example, forward via her messages destined to people living in the same place, or in a place nearby. Exploiting such information is somewhat embedded in previous works on this topic. For example, PROPHET

[1]The distinction between routing and forwarding becomes quite fuzzy in opportunistic networks. Therefore, we use these terms interchangeably in the paper.

exploits the frequency of contacts between nodes. MobySpace [4] and MV [1] exploit information about nodes' mobility patterns and places nodes are used to visit. These data can be seen as context information. In this paper we take a more comprehensive approach, and identify the general issues and mechanisms that are required to support context-aware routing policies. HiBOp does not focus on a pre-defined set of context information, but is able to exploit any information users are willing to provide to describe their context. The other protocols that exploit some context information can be seen as special realizations of HiBOp.

We identify two main issues that have to be addressed to collect and exploit context data. Firstly, nodes should be able to automatically learn the context they are currently immersed in, and remember context information they became aware of in the past (Section 3). Secondly, such context data should feed algorithms to decide good next hops towards eventual destinations (Section 4).

In Section 6 we evaluate HiBOp in comparison with Epidemic Routing and PROPHET. Our results show that exploiting context information reduces dramatically the consumption of resources such as memory and bandwidth (and thus, indirectly, energy too), at the cost of a limited increase of the message delay. This is usually fine with the class of applications opportunistic networks should support, i.e., delay-tolerant applications. Furthermore, the message loss rate is significantly reduced, as well.

2. Related Work

Since routing is one of the most compelling issues in opportunistic networks, several research groups are working on this topic. For the sake of space, in this section we only mention Epidemic Routing [11], PROPHET [5], and CAR [7], which are representative of three fundamental approaches to routing in opportunistic networks. The reader can find a comprehensive survey on routing protocols for opportunistic networks in [8].

Epidemic Routing is representative of the simplest type of routing protocols. Routing is based on pair-wise contacts between nodes, during which nodes exchange a *summary vector* containing the list of messages stored at each node. Based on received summary vectors, each node requests those messages it has not yet in its buffer. Messages are delivered to the destination when the destination meets a node carrying the messages addressed to it. Epidemic Routing is representative for dissemination-based routing protocols, which essentially flood (in a controlled way) the network to route messages. HiBOp aims at drastically reduce the cost of such flooding by exploiting context information. From a different standpoint, one could note that one of the routes that Epidemic Routing uses to deliver a message is optimal, in the sense that it is the quickest one to deliver the message. Identifying this route in advance clearly requires an oracle. HiBOp exploits context information to try to identify this particular route, thus approximating the ideal routing algorithm.

Probabilistic Routing Protocol using History of Encounters and Transitivity (PROPHET) is an evolution of Epidemic Routing that introduces the concept of delivery predictability. Delivery predictability is the probability for a node to encounter a certain destination. PROPHET forwarding algorithm is similar to the Epidemic Routing one except that, during a contact, messages are requested only if the receiving node has greater delivery predictability for the destination. PROPHET is representative for a class of routing protocols that exploit *some* context information to limit the Epidemic Routing flood (other examples are MV and MobySpace). HiBOp is able to manage and exploit far richer context information with respect to PROPHET.

Context-Aware Routing (CAR) aims at fully exploit context information, as HiBOp does. CAR assumes an underlying MANET routing protocol that connects together nodes in the same MANET cloud. To reach nodes outside the cloud, a sender looks for the node in its current cloud with the highest probability of delivering the message successfully to the destination. CAR provides a well-stated framework to compute this probability based on context information. HiBOp differs from CAR in a number of ways. Firstly, nodes in CAR compute delivery probabilities proactively, and disseminate them in their ad hoc cloud. Therefore, context is exploited to evaluate probabilities just for those destinations that each node is aware of. HiBOp is more general, as it does not necessarily require an underlying routing protocol, and is able to exploit context also for those destinations that nodes does not know. Furthermore, the definition and management of context information is not addressed in CAR, while it is a core part of HiBOp. Indeed, CAR is more focused on defining algorithms to combine context information (which is assumed available in some way) to compute delivery probabilities. Therefore, a direct comparison between HiBOp and CAR in our scenario is not very interesting, while it is more interesting comparing HiBOp with Epidemic Routing and PROPHET.

Blending together features of HiBOp and CAR is an interesting subject of future work.

3. Context creation and management

The context a user is embedded in can be seen as made up of two main components. The first one describes the *current* context of the user, while the second one is the *legacy* of the context evolution over time.

Personal Information	
Name	Donald
Surname	Duck
Email	d.duck@iit.cnr.it
Phone	340- 343439847837
NID	PLNPPRXX04XX4Y
Residence	
Street	Feather Street, 13
City	Pisa
Work	
Street	Moruzzi Street, 1
City	Pisa
Organization	CNR
Hobbies & Fun	
Address	Sport Street, 10
City	Pisa
Association	SportDuck
System Information	
MAC-Bluetooth	01:23:45:67:89:AB
MAC-802.11	09:00:07:A9:B2:EB
IP-Address	168.0.3.14

Figure 1. Identity Table example

The current context of the user contains, information about the user itself. This information is stored in the Identity Table (IT), an example of which is shown in Figure 1. The current context of a node also includes information about current neighbors of the node, achieved by exchanging ITs during pair-wise contacts. The current context is a snapshot of the local environment the user is currently immersed in. Based on this snapshot, a node could be seen as a good forwarder because, for example, one of its neighbors lives in the same street of the destination. More in general, HiBOp exploits the current context to evaluate the *instantaneous* fitness of a node to be a forwarder.

Taking forwarding decisions based only on instantaneous information would be very limiting. Actually, the current context does not represent users' behaviors and past experiences. For example, a user can be deemed a good forwarder if every morning she passes by the destination's house on her way to work. To exploit this kind of knowledge, nodes should *remember* information about other users *met in the past*. This is achieved through the History table, whose structure is shown in Figure 2. At a high level, the History table records attributes seen during the past in the Identity Table of encountered nodes. The example row reported in Figure 2 tells that the node has seen the attribute "Pisa" (of class "City"). As explained in detail in Section 3.1, the other information stored in the History table allows HiBOp to estimate the probability of encountering that attribute in the near future. It is worth noting that HiBOp remember far more than the mere identity of encountered nodes (as, for example, PROPHET does). All attributes of encountered users let some legacy in the HiBOp history. This is actually a big advantage, because it allows HiBOp to exploit similarities between encountered users and the destination. For example, a node can be deemed a good forwarder because it is very likely to encounter some (unspecified) other user that lives in the same street of the destination. Finally, note that information stored in the History table is periodically refreshed, as explained in Section 3.1.

Aggregate	***Class***	P_c	***H***	***R***
Pisa	City	…	…	…

Figure 2. History table structure

Having laid down the high-level ideas about HiBOp context management, we provide the detailed algorithms in the next section. For space limits, not all details can be thoroughly explained.

3.1. Context-management algorithms

Let us firstly focus on Identity Tables. In general, ITs can contain an extensible set of data, including personal information, such as name and surname, behavioral information, such as job place and hobbies, system information, such as network addresses of node's network interfaces, etc. In general, is up to the user to decide what to expose in the node's IT. Clearly, privacy and security issues are main concerns. We are currently investigating how to address them. One interesting idea could be evaluating matches on attributes without having full information about them. This could be achieved comparing hashes rather than plain text values. However, privacy and security issues need a deeper analysis and remains out of the scope of this paper. HiBOp works with any kind of information stored in Identity Tables (i.e., there is no limitation on what can be stored in ITs). The only requirement is that the set of information (possibly) stored in ITs be unique across the network (e.g., it could be defined by the HiBOp protocol version). This set is defined by the names of the possible attributes of the IT (left-hand side column of Figure 1). We assume that ITs uniquely

identify nodes in the network. In particular, the Node IDentity (NID) field is a hash of the IT, and is used to uniquely name a node in the network.

Nodes learn the environment around them by exchanging ITs during Neighbor Discovery phases, which nodes perform periodically and asynchronously from each other. The *Current Context* (CC) a node is in, is defined by the ITs of its current neighbors, which are stored in the CC table. Specifically, the time interval between two Neighbor Discovery phases is called Signaling Interval. At the end of every Signaling Interval, each node should send either its IT or its NID. If during the last Signaling Interval it received only ITs or NIDs of nodes that are in its Current Context, then it simply refreshes its presence by broadcasting its NID. Otherwise, if it received ITs or NIDs for nodes that are not in its Current Context, it broadcasts its complete IT. In this way, complete ITs are exchanged only among nodes that came in contact during the last Signaling Interval, while stable contacts among neighbors (i.e., contacts lasting for more Signaling Intervals) are refreshed by NIDs. An IT is removed from the CC table when the related node is not in contact anymore. In order to tolerate transitory disconnections or transmission errors, an IT is removed from the CC table after a given number of consecutive Signaling Intervals (after a Death Interval) during which neither ITs nor NIDs are received for that node.

The second building block of HiBOp context representation is the History table (Figure 2), that stores values the node has seen in ITs of neighbors met in the past. For example, if a node receives an IT with a row <City, Pisa>, then there will be a row in the History table whose Aggregate filed is "Pisa". The Class field is the corresponding *name* of the attribute in the Identity Table ("City" in the example). The reason why we store classes will be clear later on. Three counters are bound to each aggregate, i.e., the Continuity Probability (P_c), the Heterogeneity (H), and the Redundancy (R). P_c represents the probability of encountering a node that carries that value in its IT. The H field contains the average number of distinct encountered nodes, which stored that aggregate. This field is a sort of fault tolerance index, because high heterogeneity means that there are several distinct chances of encountering that aggregate on distinct nodes. The R field contains the average number of occurrences of the aggregate *within the same* IT. The redundancy information is valuable, because if a node stores the same aggregate several times in its IT, then its link towards that aggregate is very high.

Aggr	*Class*	*Carriers*	*Cont Count*	*Het Count*	*Red Count*
Pisa	City	A,B,C	2	1	2

Figure 3. Repository table structure

The History table is built as the legacy of the evolution of the Current Context. To dynamically update its content, an intermediate data structure is used, called Repository table (whose structure is shown in Figure 3). This table has an entry for each attribute the node has recently seen. The evolution of the Repository table can be characterized through two time interval: the Signaling Interval marks the update time of the Repository, while every Flushing Interval the content of the Repository is merged into the History table. At the end of every Signaling Interval, HiBOp scans its Current Context, and adds a new row in the Repository table for attributes in the Current Context that have not yet a corresponding row in the Repository table. All the other fields for such new rows are set to 0. Both for new and old rows, the values of related counters are then updated. In more details, for each attribute with a corresponding row in the Repository field, HiBOp executes the following steps:

- the Continuity Counter is incremented. Therefore, the Continuity Counter stores how many times that attribute has been seen in the Current Context during a Flushing Interval;
- if the node whose IT stores that attribute is not listed in the Carriers list, the Heterogeneity counter is incremented, and the NID of the node is added to the Carriers list. In addition, the Redundancy counter is incremented by the number of times that attribute appears in the IT. Therefore, the Heterogeneity counter stores on how many different neighbors that attribute has been seen (during the current Flushing Interval), while the Redundancy counter stores the total number of entries in the Current Context that contain that particular attribute (during the current Flushing Interval).

Once every Repository Flushing Interval (which is an integer number of Signaling Intervals), HiBOp uses the data in the Repository table to update the History table. For each value of attribute in the Repository table we compute the corresponding Continuity Probability, Heterogeneity, and Redundancy as explained below. Next, we combine these results with the corresponding values in the row associated with that attribute in the History Table as shown in Equations (1), (2), and (3). Specifically, a sample of Continuity Probability is computed as

$$p_c^{(rep)} = \frac{ContCount}{M},$$

where M is the number of Signaling Intervals in a Repository Flushing Interval. The sample of the Continuity Probability is thus computed as the probability of having seen that attribute during the previous Flushing Interval (recall that *ContCount* is the number of Signaling Intervals during which that attribute has been seen in the Current Context). The Continuity Probability in the History table is then updated as follows:

$$P_c \leftarrow \delta \cdot P_c + (1-\delta) p_c^{(rep)} \quad (1)$$

,

where δ is a classic smoothed average parameter ($0 \le \delta \le 1$). In a similar fashion, the Heterogeneity and the Redundancy are updated as follows:

$$H \leftarrow \delta \cdot H + (1-\delta) \cdot HetCount \quad (2)$$

$$R \leftarrow \delta \cdot R + (1-\delta) \cdot \frac{RedCount}{HetCount} \quad (3)$$

.

The computation of the Heterogeneity is self-explanatory from Equation (2). As far as the Redundancy (Equation (3)), we compute a redundancy sample as $\frac{RedCount}{HetCount}$, and then apply again a standard smoothed average. Dividing *RedCount* by *HetCount* means computing the "average redundancy" of the attribute during the Flushing Interval, i.e, the average number of times that attribute has been seen in a *single* Identity Table during the Flushing Interval.

4. Using the context for forwarding operations

At a high level, forwarding is based on the concept of *opportunity* to reach a certain destination, measured in term of probability of carrying the message closer to the destination. Messages are forwarded only to nodes with higher probability of getting them closer to the destination. This policy is not new. The novelty of HiBOp is *how context is exploited* to evaluate these probabilities. The main idea is that message sender includes more information about the destination than a simple network address. The sender should include (any subset of) the destination's Identity Table. Delivery probabilities are evaluated based on the *match* between this information and the context stored at each encountered node (as described in Section 4.2). High match means high similarity between the node's and the destination's context. Actually, delivery probabilities can be seen as a measure of this similarity.

Besides this, it should be noted how HiBOp controls message replication, which is a major advantage over state-of-the-art solutions. Specifically, only the sender of a message is allowed to create multiple copies of the message (following the algorithm described in Section 4.1). Other nodes that carry a message compute the delivery probabilities of encountered nodes, and do *not* keep copies of forwarded messages. This allows HiBOp to control and drastically reduce message flooding.

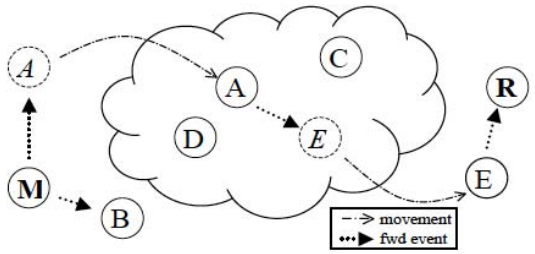

Figure 4. HiBOp forwarding process

The HiBOp forwarding process can be thus decomposed in three phases (see Figure 4):

- *Emission*: the sender injects the message in the network, replicating it for the sake of reliability.
- *Forwarding*: exploiting nodes' mobility and contacts, each copy of the message proceeds in the network towards the destination.
- *Delivery*: when a node carrying the message finds the destination the process stops.

The third phase of the process is trivial and is not discussed further. The rest of this section is thus devoted to the Emission (Section 4.1) and Forwarding (Section 4.2) phases. Also in this case, space constraints do not allow us to provide a thorough description of all details.

4.1. Emission phase

In opportunistic networks it is clearly impractical to manage reliability via ARQ mechanisms like in the legacy Internet (or in MANET too). Techniques such as message replication or network coding look more suitable. HiBOp addresses reliability by replicating messages at the sender only. HiBOp assumes that the application notifies a reliability requirement in terms of maximum tolerable message loss, p_l^{max}. Following the mechanisms described in Section 4.2, a sender node gets from its neighbors the probabilities of successfully delivering the message to the final destination. Let us denote them as $p_{succ}^{(i)}$, where i denotes the i-th

neighbor, ordered by decreasing delivery probability, and let us denote the delivery probability of the sender as $p_{succ}^{(0)}$. Assuming that these probabilities are independent, the number of neighbors (k) to which the message is forwarded by the original sender is evaluated as follows:

$$k = \min\left\{ j \mid \prod_{i=0}^{j} (1 - p_{succ}^{(i)}) \le p_l^{max} \right\}.$$

Basically, the sender forwards the message to the minimum number of neighbors such that the joint loss probability is below the maximum threshold specified by the application. If not enough neighbors are currently available, the message is forwarded to the available neighbors, is queued at the sender, and new neighbors are used as soon as they become available. Note that, to avoid flooding in case of too low delivery probabilities, neighbors are used as forwarders just if their delivery probability is above a threshold (set to 0.001 in our experiments).

4.2. Forwarding phase

4.2.1. Weighting attributes. The main idea of HiBOp forwarding is evaluating delivery probabilities based on matches between the sender information in the message, and context information available on nodes. It should be noted that matches should be *weighted* based on the class of the matching attribute. Class weights should represent the *precision* of that class in identifying the destination. For example, a match on the destination's name gives far *more precise* information than a match on the residence city of the destination.

Figure 5 visually represents the qualitative ranking of class precision we have defined for HiBOp (bigger circles represent lower precision). The definition of the weights we have used reflects this qualitative ranking.

Several functions can be used to assign weights to classes based on the ranking above. In our case, named w_0 the weight of the least significant class, we have computed other weights as $w_{i+1} = w_i + r_i \cdot \beta$, where β is defined as the weight increase parameter, and r_i is the maximum redundancy of the i-th class. The main idea is that i) weights should be monotonically increasing, and ii) the relative difference between classes should increase if the less significant one allows for a higher redundancy, because higher redundancy usually means lower significance.

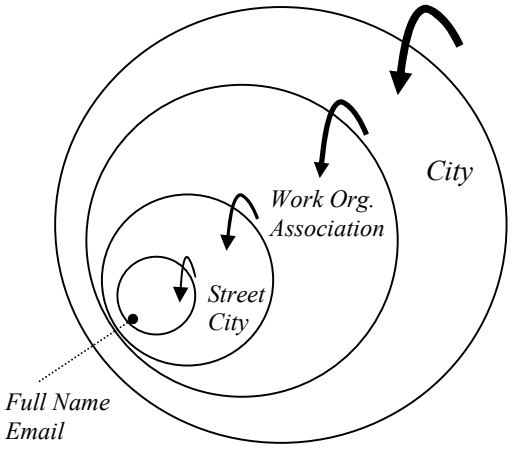

Figure 5. Precision of attribute classes

4.2.2. Forwarding based on Delivery Probability. A node wishing to forward a message broadcasts a message containing the destination information along with its own delivery probability. Nodes that receive such message evaluate their delivery probability and send it back to the inquiring node (with a unicast transmission) if it is higher than the inquiring node's one.

At each node, the delivery probability is computed from three components, related to i) the node's Identity Table, ii) the node's Current Context, and iii) the node's History. In the following we describe how HiBOp currently exploits context information to compute these values. Investigating alternative policies is an interesting subject of future work.

As far as the Identity Table, the node finds those attributes in the destination information that matches with attributes in its IT. The delivery probability from the IT is then evaluated as the ratio between the sum of the weights of matching attributes, and the sum of the weights of attributes specified in the destination information, i.e.:

$$P_{IT} = \frac{\sum_{j \in \{match\}} w_j}{\sum_{j \in \{dst\_info\}} w_j}. \quad (4)$$

As far as the Current Context, recall that it is made up of ITs of current neighbors. For each such IT the node evaluates P_{IT}, and the delivery probability related to the Current Context is the maximum over these probabilities:

$$P_{CC} = \max_{j \in CC} P_{IT}^{(j)}. \quad (5)$$

Evaluating the contribution of the History to the delivery probability requires more steps. Recall that

each aggregate in the History table comes with three indices, i.e., the Continuity Probability (P_c), the Heterogeneity (H), and the Redundancy (R). First of all, HiBOp selects the aggregates that match with destination information. For such aggregates, the R and P_c indices are combined as follows:

$$P_{op}^{(j)} = P_c^{(j)} \cdot \frac{R^{(j)}}{r},$$

where r is the maximum possible redundancy for the class of the j-th matching aggregate. Essentially, P_c is scaled according to the potential redundancy that the aggregate could achieve. Similarly to the contribution related to the IT, the contribution related to the History is evaluated based on the weighted mean of the $P_{op}^{(j)}$ values, computed as:

$$P_H{}' = \frac{\sum_{j \in \{match\}} P_{op}^{(j)} \cdot w_j}{\sum_{j \in \{dst\_info\}} w_j}.$$

The delivery probability related to the History is evaluated by modifying $P_H{}'$ according to the H indices of matching attributes. Specifically, HiBOp increases $P_H{}'$ of a factor Δ_{max} at most, scaling this factor according to the average heterogeneity of matching attributes ($\bar{h}$):

$$P_H = \max\left\{1, P_H{}' + \Delta_{max} \cdot \left[1 - e^{-(\bar{h}-1)}\right]\right\}. \qquad (6)$$

Note that, since Δ_{max} is scaled according to an exponential law, the same average heterogeneity increase results in a higher P_H increase for small values of average heterogeneity (e.g., increasing $\bar{h}$ from 1 to 2 has a greater effect than increasing $\bar{h}$ from 10 to 11).

The delivery probability is finally computed by combining Equations (4), (5), and (6), as follows:

$$P = \alpha \cdot P_H + (1-\alpha) \cdot \max\{\eta \cdot P_{CC}, P_{IT}\}. \qquad (7)$$

Equation (7) is made up of two components, weighted with a smoothing factor α ($0 \le \alpha \le 1$). The first component is P_H, which describes the *legacy* of the node's past history. The second component describes the *current* status of the node's environment. The α factor gives more weight to past history or to the current environment. The node's current environment is jointly described by P_{IT} and P_{CC}, which are therefore combined together in Equation (7). The η factor (η<1) scales down P_{CC} with respect to P_{IT}, because P_{CC} is related to a neighbor, while P_{IT} is related to the local node. Let assume two potential next hops A and B, and let assume that $P_{CC}^{(A)} = P_{IT}^{(B)}$. Let us also assume that $P_H^{(A)} = P_H^{(B)} = P_{IT}^{(A)} = P_{CC}^{(B)} = 0$ Thus, $P^{(A)} = \eta \cdot P_{CC}^{(A)}$ and $P^{(B)} = P_{IT}^{(B)} > P^{(A)}$ hold, i.e. the η factor makes node B preferable as a next hop. This is correct, because forwarding through A surely requires a further hop to give the message to the A's neighbors that generated $P_{CC}^{(A)}$.

5. Simulation setup

The performance of the HiBOp protocol has been evaluated in terms of delay, buffer occupation, message loss and amount of traffic generated in the network. HiBOp performance has been compared to that of Epidemic Routing (Epidemic, for short) and PROPHET.

In order to evaluate the performance of HiBOp we developed a custom simulator. The goal of the simulation study is understanding which is the impact of using context information in opportunistic networks. Therefore, we assumed ideal wireless links with infinite bandwidth and negligible transmission delay. This is clearly unrealistic, but allows us to isolate the effect of context awareness from networking effects such as congestion, transmission errors, etc. Since HiBOp exploits context to reduce messages' spread in the network, we can anticipate that neglecting networking congestion favors Epidemic and PROPHET. Indeed, we are neglecting additional delays and message losses related to network congestion, which might significantly increase delays and message drop rates under very high traffic load [9].

Our simulation scenario was a square of size 1250x1250m, divided in a 5x5 grid. The number of nodes was set to 80, and the transmission range was set to 100m. Nodes moved according to the traces generated by Community based Mobility Model [6]. This model is quite different from traditional random models and mimics real human movement patterns. Every node belongs to a social community. Nodes that are in the same social community are called friends, while nodes in different communities are non-friends. Nodes' movements are determined by the attraction of other nodes: each node moves (with a uniformly distributed speed) towards the cell in which it has more friends. Therefore, it's more likely that a node will be in contact with nodes of its community, because they spend more time together. CMM also includes the notion of travelers that do not always move in the cell where they have more friends. From time to time, they

move to the second most attractive cell (i.e., to the cell in which they have the second highest number of friends), and then get back to the most attractive cell afterwards. Once in a while a reconfiguration occurs, during which all groups change cell. During a reconfiguration nodes of different groups have chances to meet. For example, CMM allows us to model a typical campus scenario, in which communities are people attending the same classes, or a typical working environment, in which communities are working departments.

The context used in our simulation was the personal information of the user, and the information about its working place. To make the context coherent with CMM, similar attributes were given to nodes belonging to the same group. Simulation results show that HiBOp is already able to outperform Epidemic and PROPHET also with such limited context information.

We considered a messaging application, and the set of senders was chosen uniformly at random at the beginning of the experiment. The interval between the generation of two consecutive messages at the same sender was modeled according to an exponential distribution, with average 300s. Message destination was a friend node with 50% probability, and a non-friend node with 50% probability. Among the friends and non-friends, the destination was chosen uniformly at random. Messages expired after 18000s.

Each simulation ran for 90000 seconds. To gather accurate measures about the message loss, senders stopped generating messages 18000s before the end of the simulation. This way, messages that had not been delivered at the end of a simulation run had certainly been dropped due to timeout expiration. We replicated each simulation configuration 5 times with independent seeds. Unless otherwise stated, results presented hereafter are the average over the 5 replicas.

Finally, the parameters related to CMM were set as shown in Table 1, while the parameters related to HiBOp were set as shown in Table 2. When assessing the sensitiveness of HiBOp to a particular parameter, we set the other parameters as shown in this table.

Table 1. CMM parameters

Number of nodes	80
Simulation area	1250x1250m
Cells in the grid	5x5
Node speed	$U \in$ [2-9]m/s
Number of groups	8
Reconfiguration interval	9000s
Travelers speed	5m/s
Number of travelers	8

Table 2. HiBOp parameters

Signaling Interval	5s
p_l^{max}	0.05
Repository Flushing Interval	1800s
δ	0.5
η	0.95
Δ_{max}	1
α	0.5
Death Interval	10s
Default buffer size	50 messages
Default number of senders	20
Default message size	50000 B

6. Simulation Results

6.1. Unlimited buffers

In this section we don't put any limit on the nodes' buffer size, which is clearly the best possible configuration for Epidemic and PROPHET. The evolution over time of the buffer size (averaged over all nodes) is plotted in Figure 6. It clearly shows that Epidemic and PROPHET require about one order of magnitude more space than HiBOp. Even though memory is cheap nowadays, messages in Opportunistic Networks are usually far larger than messages in IP networks, and whole files can be accommodated in a single message [10]. For an average message size of 1MB, nodes running Epidemic should reserve about 400MB just for routing purposes! It is therefore worth to consider how buffer limitations impact on these protocols' performance. Specifically, in the following we will use a FIFO replacement policy for managing buffers. Investigating other (smarter) policies is out of the scope of this paper.

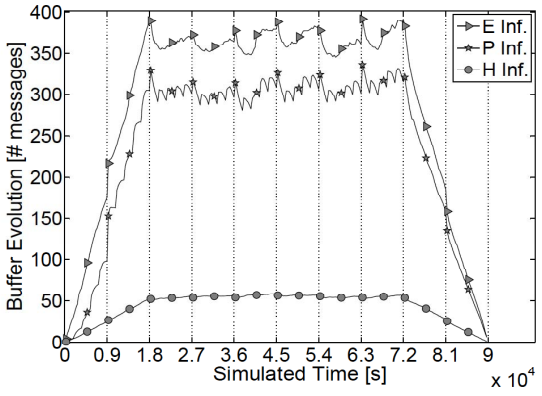

Figure 6. Buffer evolution with unlimited size

Before going on, it is worth noting that the message loss experienced by all protocols was negligible

(below 0.1%), and the HiBOp average delay was 1.71x and 2.25x the average delay of PROPHET and Epidemic, respectively (Table 3). Even though there is a clear delay increase, HiBOp performance remains acceptable even in this extremely favorable scenario for Epidemic and PROPHET.

Table 3. Average delays with unlimited buffers (10^3s)

Epidemic	PROPHET	HiBOp
0.6466	0.8489	1.4538

6.2. Resource Consumption

In this section we analyze the resource consumption of HiBOp in comparison with Epidemic and PROPHET, in terms of buffer occupation, and traffic overhead. This evaluation is fundamental, since one of its main goals is reducing the overhead of previous routing.

Figure 7 shows the average buffer occupancy of the three protocols during time, for three selected values of the maximum buffer size, i.e., 20, 50 and 100 messages. The plot highlights that, as expected, both Epidemic and PROPHET saturate the buffers. Specifically, after an initial startup phase, and before the final cool-down phase (the last 18000 seconds in which no new message is generated), buffers are almost always 100% full. Since the figure plots the average buffer occupation over all nodes, this means that *all* buffers in the network are saturated. HiBOp is much less greedy in using buffer resources. The fact that the average occupation is much lesser than the maximum buffer size, means that the probability of HiBOp saturating buffers is very low. As it is shown in the next sections, the number of messages delivered by HiBOp is even higher than the number of messages delivered by Epidemic and PROPHET. Therefore, this buffer occupancy comparison is even somewhat unfair to HiBOp.

Finally, note that buffer occupancy in all cases drops every 9000 seconds. This is because a reconfiguration occurs every 9000 seconds. Since during a reconfiguration nodes of different groups have more chance to meet, this results in a message delivery peak.

Figure 8 shows the resource consumption in terms of networking overhead. Specifically, it plots the ratio between the total number of bytes exchanged over the network, and the total number of bytes successfully delivered to destinations. Therefore, it shows how many bytes have to be generated, on average, for each successfully delivered byte. Note that the total number of bytes generated includes not only the application-level messages to be forwarded, but also the whole routing and forwarding traffic generated by the protocols.

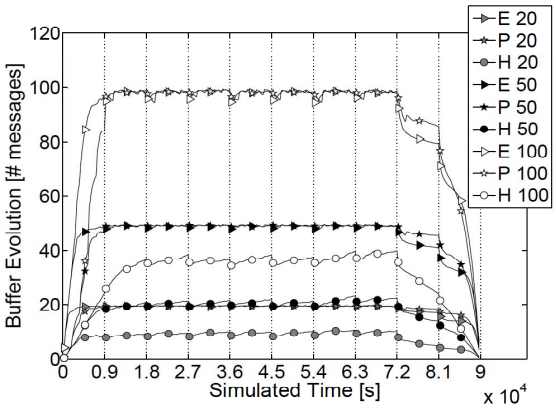

Figure 7. Buffer occupation

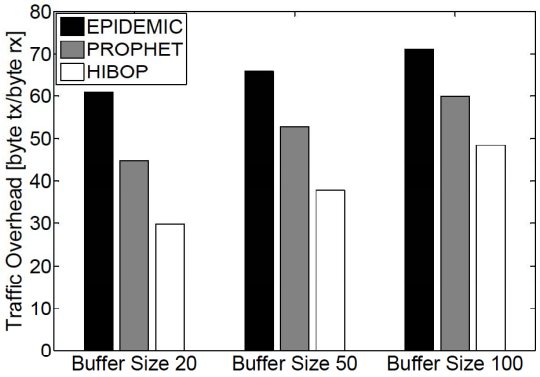

Figure 8. Traffic overhead

Therefore, this index also accounts for the effect of exchanging Identity Tables and using long message headers in HiBOp, which is an additional overhead with respect to Epidemic and PROPHET. In Figure 8, three groups of bars are plotted for selected maximum buffer sizes (20, 50, 100 messages). First of all, HiBOp significantly reduces the networking overhead of Epidemic and PROPHET. Specifically, the reduction is in the range [32%,51%] with respect to Epidemic, and [19%,34%] with respect to PROPHET. The networking overhead increases with the maximum buffer size. As shown by Figure 7, the buffer occupancy increases with the maximum buffer size, meaning that – on average – each node stores an increasing number of messages to be forwarded. Therefore, the traffic generated either to forward these messages, or to exchange information about delivery probabilities, increases with the maximum buffer size.

Based on Figure 7 and Figure 8, HiBOp significantly reduces resource consumption in terms of

buffer occupancy and networking overhead. From a complementary standpoint, this also shows that HiBOp's context-aware features act as an effective *congestion control* system for opportunistic network. This is a key point, as currently adopted routing protocols tend to be very greedy in resource usage, thus resulting in high resource congestion.

6.3. User perceived QoS

In this section we investigate the QoS perceived by users in terms of message delay and message loss. Specifically, Figure 9 shows the message loss of the three protocols for varying buffer sizes. As expected, the message loss drops as the buffer size increases, because messages can live longer in nodes' buffers. It is interesting to note that HiBOp message loss is always lower than Epidemic and PROPHET message loss. This is not trivial, because HiBOp drastically reduces message replication (as shown by the lower buffer occupancy), i.e., it explores less paths towards the destinations. Potentially, this could results in *greater* message loss, if the wrong paths are chosen. Since the message loss actually decreases with HiBOp, this tells that i) HiBOp allows messages to live longer in the buffers thanks to low replication, thus letting more time for them to be delivered, and ii) HiBOp is able to choose correct paths based on its context-aware rules.

The performance in terms of message delay is analyzed jointly in Table 4 and Figure 10. To make delay distributions related to different protocols comparable, we added samples equal to the maximum message lifetime (18000s) for lost messages. These samples are not plotted in the CCDFs of Figure 10, which results in truncating the plots before the 100th percentile.

First of all, it should be noted that these delay values (in the order of tens of minutes) are typical in opportunistic networks, even though they might look fairly high. In more detail, delay is the only figure for which HiBOp performs generally worse than Epidemic and PROPHET. For medium (50) and large (100) buffers, the HiBOp additional delay is respectively 14% and 84% with respect to Epidemic, and 14% and 55% with respect to PROPHET. Actually, HiBOp outperforms both Epidemic and PROPHET for small buffers, because of the greater share of messages having delays greater than 18000s. By focusing on the tail distributions, one notices that HiBOp is actually able to *reduce* the high percentiles, unless for large buffers (100). This is another effect of the greater HiBOp reliability in delivering messages. These delay figures can be seen as more than acceptable for HiBOp indeed. The average delay increase is tolerable, especially by recalling that protocols like Epidemic are not very likely to be widely adopted, due to their excessive overhead.

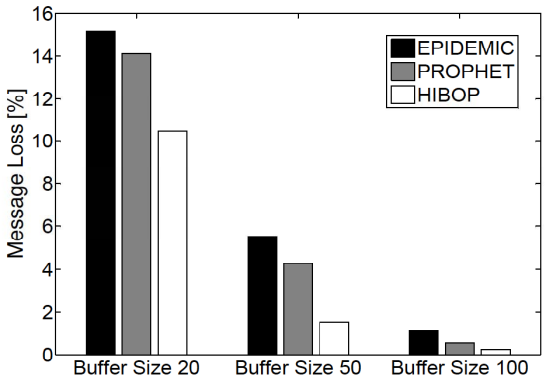

Figure 9. Message loss

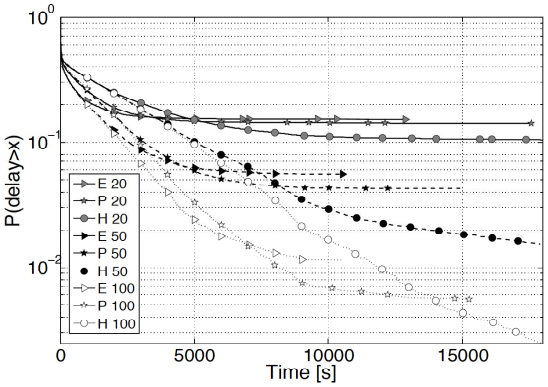

Figure 10. Delay distributions

Table 4. Average message delay (10^3s)

	Epidemic	PROPHET	HiBOp
Buff = 20	2.94	2.89	2.73
Buff = 50	1.43	1.42	1.62
Buff = 100	0.79	0.93	1.44

6.4. Effect of the emission phase configuration

In this section we investigate the impact of the p_I^{max} parameter, which is used by the sender during the emission phase to decide how widely to replicate the message. Specifically, we consider three p_I^{max} values, i.e., 5% (the default), 50% and 80%. Increasing p_I^{max} actually means replicating the message less aggressively, and exploring less paths. The expected effect of increasing p_I^{max} is thus to further reduce

resource consumption, and to worsen delay and message loss performance.

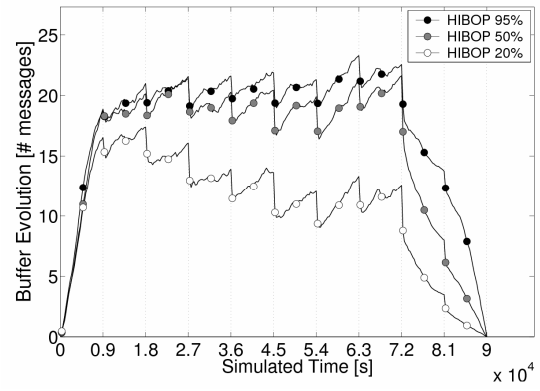

Figure 11. Buffer occupation

Table 5. Message Loss and Traffic overhead

p_l^{max}	5%	50%	80%
Msg loss	1.53%	1.60%	2.11%
Overhead	37.70	33.91	27.14

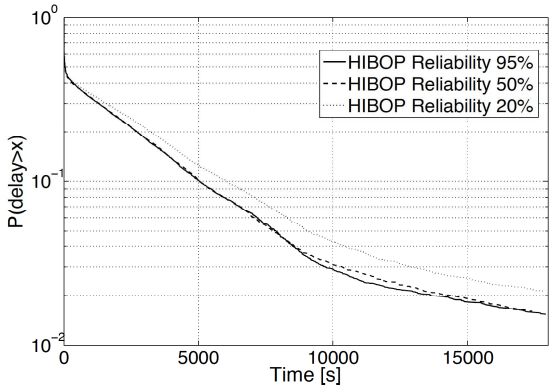

Figure 12. Delay distribution

Figure 11 and Table 5 show that HiBOp resource consumption decreases as p_l^{max} increases. Accordingly, the message loss (Table 5) and the delay (Figure 12) increase with p_l^{max}. Even though the observed behavior matches intuition, the performance worsening in terms of message loss and delay is lower than what one could expect. For example, increasing p_l^{max} from 5% to 80% results in additional message loss of just about 0.7%. On the positive side, HiBOp is able to guarantee very high reliability even for loose bounds on p_l^{max}. However, one might wonder how p_l^{max} is precise in controlling the *end-to-end* HiBOp reliability. This point is one of the features we are currently investigating more deeply, in order to improve HiBOp.

7. Conclusions and future work

In this work we have proposed a context-based routing framework for opportunistic networks, and a particular routing protocol (HiBOp). We have evaluated its performance across a range of parameters' values, in comparison with Epidemic Routing and PROPHET. We have shown that HiBOp is able to drastically reduce the resource consumption, in terms of network traffic and nodes' buffer occupation. At the same time, HiBOp is significantly more reliable than Epidemic and PROPHET, as it reduces the message loss rate. This is paid, in some configuration, with a delay increase, that is however tolerable for typical applications of opportunistic networks. Finally, HiBOp is able to reduce the probability of having very large delays.

These results clearly indicate that context-based forwarding is a very interesting way of finding paths in opportunistic networks. Actually, it looks like a strong alternative to classical flooding-based protocols, and can be seen as an effective means of controlling congestion in these networks.

Despite this, there are a number of directions along which HiBOp can be further investigated. Several control knobs exists in its design, whose effects have to be clearly investigated. For example, how to achieve fine-grain control on end-to-end reliability is an interesting topic. Furthermore, finding analytical bounds for the performance of context-based forwarding is a very exciting point. All of these aspects are clearly out of the scope of this paper. Nevertheless, we have already shown that exploiting context is a powerful idea in opportunistic networks, and we have provided a general framework for managing context, and exploiting it to find correct paths.

Another interesting aspect is how to "bootstrap" a network with HiBOp, i.e., how to route data when context information is not widely available in the network. Purposely, we have let this point out of this paper. Indeed, in our simulations we have discarded the initial transient phase required to spread context information in the network, to precisely evaluate the advantage of exploiting context information. The performance of HiBOp when context information is not spread widely is actually analyzed in [12]. In that paper we deeply investigate the performance of HiBOp with respect to a number of parameters describing the social behavior of users. One of the main outcomes highlighted in [12] is the fact that HiBOp becomes

efficient (i.e., it routes data with comparable delay with respect to Epidemic and with lower overhead) even in configurations in which context information is just partially spread in the network. Clearly there are corner cases in which context information cannot circulate due to particular mobility patterns. In those cases, HiBOp becomes less efficient than epidemic approaches. An interesting direction to pursue is how to design an integrated protocol that be able to exploit context information as soon as it becomes available, but falls back to epidemic-like routing when such information is not available.

8. References

[1] B. Burns, O. Brock, and B. N. Levine, "MV Routing and capacity building in disruption tolerant networks", *Proc. of IEEE INFOCOM 2005*, Miami, FL, March, 2005.

[2] K. Fall, "A delay-tolerant network architecture for challenged internets", *Proc. of ACM SIGCOMM*, 2003.

[3] S. Jain, K. Fall, and R. Patra, "Routing in a delay-tolerant network", *Proc. of ACM SIGCOMM*, 2004.

[4] J. Leguay, T Friedman, and V. Conan, "Evaluating Mobility Pattern Space Routing for DTNs", *Proc. of IEEE INFOCOM 2006*.

[5] A. Lindgren, A. Doria, and O. Schelèn, "Probabilistic routing in intermittently connected networks", *ACM Mobile Computing and Communications Review*, 7(3), July 2003.

[6] M. Musolesi, C. Mascolo, "A Community Based Mobility Model for Ad Hoc Network Research*", in Proceedings ACM/SIGMOBILE REALMAN*, 2006.

[7] M. Musolesi, S. Hailes, and C. Mascolo, "Adaptive Routing for Intermittently Connected Mobile Ad Hoc Networks", *Proc. of IEEE WoWMoM*, 2005.

[8] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic Networking: data forwarding in disconnected mobile ad hoc networks", *IEEE Communications Magazine*, 44(11), Nov. 2006.

[9] A. Spuropoulos, K. Psounis, and C. Raghavendra, "Multi-copy routing in intermittently connected mobile networks", Tech. Rep., CENG-2004-12, USC, 2006.

[10] J. Scott, P. Hui, J. Crowcroft, C. Diot, "Haggle: A Networking Architecture Designed Around Mobile Users" *Proc. of IFIP WONS*, 2006.

[11] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks", *Tech. Rep. CS-2000-06*, CS Dept., Duke University, 2000.

[12] C. Boldrini, M. Conti, A. Passarella, "Impact of Social Mobiltiy on Routing Protocols in Opportunistic Networks", *Proc. of IEEE WoWMoM 2007,* Helsinki, June 2007.