Content-Based Routing in Highly Dynamic Mobile Ad Hoc Networks

ROBERTO BALDONI, ROBERTO BERALDI, LEONARDO QUERZONI
Università di Roma "La Sapienza", Via Salaria 113, I-00198 Roma, Italy
Email: Roberto.Beraldi@dis.uniroma1.it

GIANPAOLO CUGOLA, MATTEO MIGLIAVACCA
Politecnico di Milano, P.zza L. Da Vinci, I-20133 Milano, Italy

Received: October 24 2005

***Abstract*— The decoupling and asynchrony properties of the content-based publish-subscribe paradigm makes it very appealing for dynamic wireless networks, like those that often occur in pervasive computing scenarios. Unfortunately, most of the currently available content-based publish-subscribe middleware do not fit the requirements of such extreme scenarios, in which the network is subject to very frequent topological reconfigurations due to mobility of nodes. In this paper we propose a protocol for content-based message dissemination tailored to Mobile Ad Hoc Networks (MANETs) showing frequent topological changes. Message routing occurs without the support of any network-wide dispatching infrastructure thus eliminating the need of maintaining such infrastructure on top of a physical network continuously changing its topology. The paper reports an extensive simulation study that confirms the suitability of the proposed approach along with a stochastic analysis of the central mechanism adopted by the protocol.**

***Index Terms*— MANET, Publish-Subscribe, Middleware**

I. INTRODUCTION

A Mobile Ad Hoc Network (MANET) is a dynamic collection of wireless mobile devices that are able to communicate and move at the same time through dynamic wireless links [17]. Neither pre-existing infrastructures nor centralized administration functions are required thus self-organization and adaptiveness are important properties. MANETs represent a concrete example of support for pervasive computing.

One of the main issue in MANETs is how to provide the application layer with suitable communication abstractions for the very dynamic nature of the underlying communication network. Content-based publish-subscribe (cb-ps) is a very appealing candidate for such dynamic contexts since it offers a flexible many-to-many communication pattern that decouples components of a distributed application in time, space, and flow [8].

The work described in this paper was partially supported by the Italian Ministry of Education, University, and Research (MIUR) under the IS-MANET and VICOM projects, and by the European Community under the IST-004536 RUNES project.

A preliminary version of this work appeared in the proceedings of the IEEE International Conference on Pervasive Services 2005 (ICPS'05), 11-14 July 2005, Santorini, Greece.

Components of a cb-ps system interact by *publishing* messages and by *subscribing* to the messages whose content matches a given *predicate*. A *dispatching service* is in charge of delivering published messages to all the interested subscribers, thus realizing the decoupling mentioned above.

The implementation of an efficient dispatching service for a MANET is very challenging. In fixed networks, the dispatching service is often realized by a single, centralized server, which stores the predicates that express the interests of subscribers and use them to route messages coming from publishers. Clearly this approach cannot be adopted in MANETs, in which nodes need to communicate without the support of any stable infrastructure.

Recently, many cb-ps middleware have been developed, which adopt a distributed implementation of the dispatching service, composed of a set of *brokers* that collaborate to route messages from publishers to subscribers. In such systems, brokers are connected in an *overlay dispatching network* arranged according to some well known topology, e.g., a tree, to ease routing. In principle, this case is more suitable to MANETs, since a broker could be launched on each mobile node, removing the need of a centralized dispatching service, but the overhead required to maintain the overlay network connecting brokers in the face of topological changes occurring at the networking layer, makes this approach unsuitable for scenarios that exhibit even a moderate degree of mobility.

In this paper we explore a different strategy, whose key aspect is the lack of any predefined logical infrastructure as a support to message routing. We realize a distributed implementation of the dispatching service by running a broker on each mobile node of the MANET but, differently from the previous case, we do not try to keep an overlay dispatching network connecting them. Conversely, we leverage off the broadcast communications available in a MANET to forward messages, letting each receiving broker to autonomously decide if and when re-forwarding a message on the basis of an estimation of its proximity to potential subscribers for that message.

In particular, we use the time elapsed since two nodes went out from each other's transmission range as an estimate of their distance. As we will formally show in Section V, within

a given time interval this represents a good approximation of the distance metric between two nodes. This fact, together with the efficiency of the implicit and indirect forwarding method adopted, results in an effective and scalable routing mechanism in the scenarios we consider, as proven by the results of the simulations we run.

The rest of the paper is organized as follows: Section II discusses the background of traditional cb-ps routing techniques. Section III briefly motivates our work and gives a general description of the routing protocol we propose, while Section IV provides the details of the protocol. In Section V we formally analyze the relationship between the elapsed time and the distance between the brokers, while Section VI presents the results of an extensive campaign of simulations, which validates our approach. Finally, Section VII discusses related work and Section VIII provides some concluding remarks and describes future work.

II. Content-Based Routing

The aim of this section is to provide some background to better understand the domain of our interest and to motivate why new solutions need to be developed. As already mentioned in the introduction, applications exploiting a publish-subscribe middleware are organized as a collection of components, which interact by *publishing* messages and by *subscribing* to the classes of messages they are interested in. The core component of the middleware, the *dispatcher*, is responsible for collecting subscriptions and forwarding messages from publishers to subscribers.

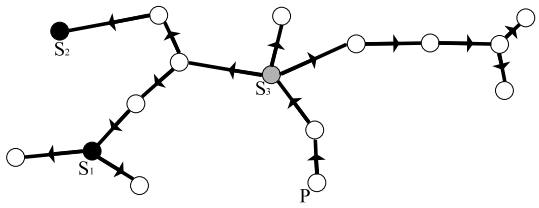
Currently available publish-subscribe middleware differ along several dimensions among which the most relevant are the expressiveness of the subscription language, the architecture of the dispatcher, and the forwarding strategy [8], [1], [3], [16].

The expressiveness of the subscription language draws a line between *subject-based* systems, where subscriptions identify only classes of messages belonging to a given channel or subject, and *content-based* ones, where subscriptions contain expressions (called *predicates*) that allow sophisticated matching on the message content. We guess that the typical scenarios of pervasive computing, where a potential large number of components need to interact very flexibly, justify the latter choice with respect to the more conservative, less expressive, and less scalable solution of subject-based filtering.

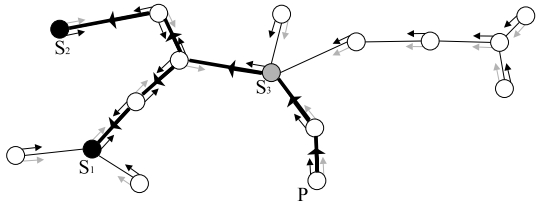
To apply cb-ps in large scale networks, most advanced middleware adopt a distributed dispatcher, where a set of *brokers* are interconnected in an *overlay dispatching network* and cooperatively route subscriptions and messages sent by components attached to them.

Middleware that exploit a distributed dispatcher can be further classified according to the interconnection topology of brokers and the strategy exploited for message dissemination.

The simplest approach is *message forwarding* in which brokers are connected to form an unrooted tree. Publishers send messages to their associated broker, which forwards them to all other brokers by following the tree topology. Moreover, each broker keeps track of the subscriptions coming from the software components directly attached to it into a local *subscription table*, which is used to determine the components, if any, that should receive incoming messages. This solution inevitably results in high overhead as all messages are sent to all brokers, regardless if an attached component has subscribed.



(a) Message forwarding



(b) Subscription forwarding

Fig. 1. Publish-subscribe routing strategies.

An alternative and more widely used strategy is *subscription forwarding*, which limits the overhead of message forwarding by spreading knowledge about subscriptions beyond the first broker, i.e. the broker where the publisher is attached, along the unrooted tree connecting brokers. Specifically, when a broker receives a subscription from one of the application components connected to it, not only it stores the associated predicate into its subscription table as in message forwarding, but also it forwards such a predicate to the neighboring brokers. During this propagation, each broker behaves as a subscriber with respect to its neighbors[1]. This process effectively sets up routes for messages through the reverse path followed by subscriptions.

In Fig. 1 the above strategies are compared by showing the same setting, characterized by a distributed dispatcher composed of 16 brokers. Two of them, namely S_1 and S_2, have components connected to them (not shown to avoid cluttering the figure) that subscribed to the same predicate, represented as a black color, while broker S_3 received a "gray" subscription. Finally, broker P received a message matching the black predicate but not the gray one. The path followed by this message is shown through thick, directed lines, while black and gray arrows represent the content of subscription tables. More specifically, each broker has a colored arrow oriented towards another broker if it received the corresponding subscription

[1]This basic scheme can be optimized, e.g., by exploiting the notion of "coverage" among predicates, or by aggregating them, as described in [1].

from that broker. Figure 1(a) shows how message forwarding incurs in the highest overhead at publishing time, while it does not require subscriptions to be propagated. Subscription forwarding (Fig. 1(b)) fills the subscription tables of each broker but offers the best performance at publishing time.

Apart from the two strategies above, which are the most common especially in presence of content-based subscription languages, other, more complex strategies are possible, in which brokers are connected in a more complete graph to improve routing efficiency and increase system availability and fault tolerance.

All these solutions are characterized by a permanent network-wide structure, be it a tree or a different kind of graph joining a set of brokers, which supports message and, optionally, subscription forwarding. These distributed techniques where originally motivated by scalability issues in large networks and are quite successful in quasi-static, wired scenarios. While distributed dispatchers are also a necessity in MANETs, it is easy to argue that a naive application of such structure-based approaches to mobile networks is very inefficient, since it requires to maintain a set of logical connections among mobile brokers. Moreover, due to mobility, it may be often the case that the topology of the overlay network of brokers doesn't reflect the actual position of the nodes, and consequently the topology of the physical network.

III. Proximity-Driven Routing: An Overview

Message routing based on a distributed set of brokers interconnected in an overlay dispatching network is hard to implement efficiently in a MANET due to the cost required to cope with the frequent changes in the topology of the physical network.

To succeed in a MANET, and particularly in those including fast moving nodes, a cb-ps protocol should not require any predefined network-wide structure. Starting from this observation we developed a diffusion protocol, dubbed *proximity-driven routing protocol*, whose general concepts are described in this section. Details are given in the next section.

A. Assumptions

In our description we assume that the MANET is composed of N mobile nodes, each running a broker. When necessary, to stress the difference between nodes and brokers, we will use the notation n_i to indicate the i-th mobile node of the network, and b_i to refer to the broker running on that node.

Each broker b_i acts as an entry point to the cb-ps dispatching service for every application running on node n_i. When a component running on a node n_i wants to receive some message, it subscribes to b_i, which then stores the predicate associated with the subscription into its subscription table. Similarly, to publish a message, a component running on a node n_i sends it to the broker b_i.

The protocol does not rely on any network layer protocol; rather it only assumes the availability of a local broadcast communication primitive, which allows a node to unreliably send a message to all its one-hop neighbors via a single transmission – a fundamental and always satisfied assumption in MANETs.

Finally, we assume that the interests of all the application components connected with a broker b_i can be condensed into a single *predicate*, which reflects the content of b_i's subscription table[2].

B. Some General Considerations

To develop our protocol we started by observing that if each broker knew the Euclidean distance of its neighbors from the recipients of a message (i.e., the subscribers) as well as its own distance from them, then the message could be forwarded by letting a broker b_i to send it only to the neighbors closer than itself to the subscribers.

To put such a kind of geographical routing into practice we must solve three problems: how to calculate the list of recipients of a message; how a broker determines its distance from the others; how a broker determines the set of neighbors that should process the message.

To calculate the list of the final recipients of a message a broker should know the subscriptions issued by all other brokers in the network, something that is clearly not reasonable. Consequently, we decided to relax this requirement by collecting information about subscribers as the message is being forwarded and appending them as control information in the message.

As for the second of the problems above, we excluded the use of a location service support, e.g., based on positioning device like GPS, and decided to measure the distance between two brokers by exploiting the time elapsed since they were most recently *adjacent* (i.e., in direct communication range) to each other. This estimation technique is very simple (a beacon signal is sufficient for this purpose) and reasonably accurate, provided that the elapsed time is not too long. Section V further elaborates on this issue.

Finally, as we wanted to keep any routing decision as simple and "distributed" as possible, we decided to abandon the idea of each broker collecting the distance information from its neighbors and using it to *explicitly* choose the set of neighbors that have to further forward a message. Conversely, we adopted an *implicit* routing mechanisms, which uses broadcast communication to reach every neighbor, letting them autonomously decide if re-sending the message or not, based also on the observed behavior of other neighbors. In particular, this realizes a *store, delay, and cancel-or-forward* approach, which represents an efficient technique for routing: it exploits the broadcast nature of the wireless transmissions to send multiple copies of the same message via a single transmission; it eliminates the need of collecting and maintaining information (i.e., distance data) about neighboring brokers, it avoids the burden of link breakage detection and, overall, it provides an intrinsic resilience to the topological changes caused by the mobility of the nodes.

[2]Note that this assumption is realistic for content-based publish-subscribe systems whose subscription language is usually powerful enough to allow it.

C. The Protocol

Let now consider how the basic message forwarding scheme works. Each broker b_i periodically broadcasts a beacon message containing the predicate that summarizes its own subscription table. A broker b_j, which is adjacent with b_i, receives this message and stores the predicate together with the time it received the beacon into its *proximity table*. This mechanism allows each broker to determine the time elapsed since it lost contact with any other broker. This value, which is infinite if the two brokers never came in contact and zero if they are still adjacent, is the basis to calculate the *proximity value* (or simply "proximity") p_{ji} of b_j with respect to b_i.

Each message m carries a *destination list*: the (estimated) list of brokers interested in receiving the message, each coupled with the lowest proximity computed by the brokers that forwarded the message so far. As an example, the destination list of a message m includes a couple $< i, p >$ if broker b_i is known to be interested in receiving the message (i.e. m matches a subscription issued by some subscriber attached to b_i) and p is the lowest proximity from b_i calculated by all the brokers that forwarded m. The message has also a unique network-wide identifier provided by the source broker, we will refer to it with the notation $m.id$.

Suppose now that at time t the broker b_i receives a message m for the first time. It will resend the message if (i) it is aware of some new broker not mentioned in the destination list carried by m or (ii) its proximity table holds for some broker b_k a proximity lower than that associated to the same broker b_k into m's destination list.

Such a condition is in general not sufficient to trigger the actual transmission of the message. The broker b_i, in fact, schedules the transmission of the message after a delay proportional to p_{ik} (the lowest value is considered if such a condition holds for more than one broker, see later). If during such a time interval it doesn't hear the same message (i.e. a message with the same identifier) again, then the transmission will take place. Otherwise b_i silently drops the message. The rational behind this decision is to avoid that two adjacent brokers will forward the same message and also to let brokers closest to some destination to "suppress" transmission of adjacent brokers less close.

To clarify this basic mechanism, let us consider Fig. 2, which shows a set of nodes (the black circles) together with their transmission ranges (the gray circles surrounding the node). Imagine broker b_0 publishes a message matching broker b_4's subscriptions. The message is sent via broadcast and received both by b_1 and b_2. Assume that b_0 and b_4 have never came in contact so that the destination table carried by m is initially empty. Assume that b_2 missed $p_{24} = 5$ beacons from b_4. The broker b_2 schedules the transmission with some delay proportional to 5. However, b_1 is adjacent to b_4 (i.e., $p_{14} = 0$) and immediately sends the message. Broker b_2, on receiving the message from b_1 aborts the scheduled transmission and silently drops m. Moreover, since the proximity carried by the message sent by b_1 is zero, the broker b_3 ignores the message (by definition zero is the lowest possible proximity).

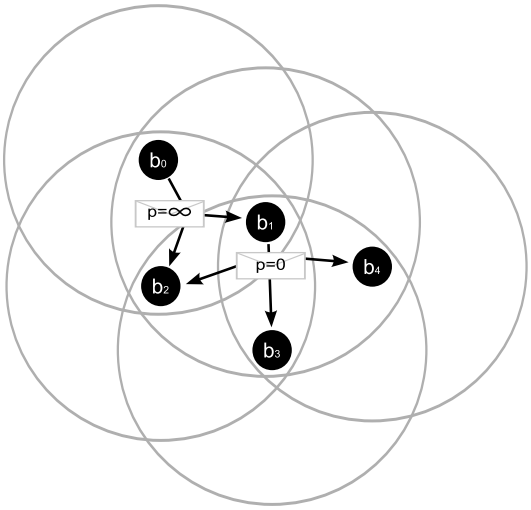


Fig. 2. The basic coordination mechanism.

```
Broker state
  st: Array of ⟨pred, id⟩
  pt: Array of ⟨id, pred, time⟩
Every Δt seconds do begin
    pred ← st.summarize()
    broadcast(pred)
    pt.cleanUp()   // removes entries older than 10 * Δt
       seconds
end
predicateReceived(pred, brokid) begin
    if ∃k such that pt[k].id = brokid then
        pt[k].pred ← pred
        pt[k].time ← currentTime
    else
        pt.append( ⟨brokid, pred, currentTime⟩)
    end
end
forward(m) begin
    if message m' s.t. m'.id = m.id was already received then
        de-schedule transmission of m'
        return
    end
    foreach ⟨pred, id⟩ in st do
        if pred.matches(m) then
            m.setProximity(myId,0)   // Update m's
               destination list
            sendToClient(m, id)
        end
    end
    minProximity ← 1.0
    matched ← false
    foreach ⟨id, pred, time⟩ in pt do
        p ← pt.proximityFor(id)
        if pred.matches(m) and (id ∉ m.destinationList or
        p < m.getProximity(id)) then
            matched ← true
            m.setProximity(id,p)
            if minProximity > p then minProximity ← p
        end
    end
    if not matched and m.credit > 0 then
        m.credit ← m.credit − 1
        matched ← true
    end
    if matched then
        schedule m for transmission with a delay proportional to
        minProximity
    end
end
```

Fig. 3. The Proximity-Driven Routing Protocol.

IV. Proximity-Driven Routing: Protocol Details

The pseudo-code of our Proximity-Driven Routing protocol, is reported in Fig. 3. Each broker maintains the following data

structures:

- A subscription table that holds information about the subscriptions issued by application components running on the same node. This table is organized as an array st of pairs $\langle pred, id \rangle$, where $pred$ is the predicate carried by a subscription and id is the identifier of the component that issued the subscription.
- A proximity table organized as an array pt of triples $\langle id, pred, time \rangle$, where id is the identifier of a broker, $pred$ is the predicate received from that broker, which summarizes its subscription table, and $time$ is the time when the predicate was received.

Every ΔT seconds each broker b_i beacons a summary of the predicates stored into its subscription table using a broadcast packet. A broker b_j that is within the transmission range of b_i receives such a beacon and executes the procedure `predicateReceived` of Fig. 3 to update its proximity table. If the same predicate was already received from the same broker, then the entry is refreshed, i.e. the time associated to the entry is set to the current time. Otherwise a new element is appended to the table. After a timeout, experimentally set to $10\Delta T$ seconds, entries are deleted from the table (procedure `cleanUp` in Fig. 3). In other words, information about brokers for which more than 10 beacons have been missed, are dropped. This reflects the general intuition, also confirmed by the model provided in the next section, that too large proximity values are not correlated with the effective distance between brokers.

The information stored in the proximity table, together with the fact that the beacon interval ΔT is known globally, allow each broker b_i to calculate the proximity value p_{ij} at time t with respect to any other broker b_j as follows: p_{ij} is infinite if b_j is not present into b_i's proximity table; otherwise it is a value in the range $[0..1]$ calculated as the number of b_j's beacons missed by b_i divided by 10.

Remembering from previous section that each message carries a unique identifier and a destination list composed of couples $\langle id, proximity \rangle$, we can describe how message forwarding proceeds (see procedure `forward` in Fig. 3). On receiving a message m a broker checks if the same message, i.e., a message with the same identifier, has been received before[3]. If this is the case, the message is removed from the list of messages scheduled for trasmission (if present) and it is dropped without any further processing.

If m was never received before then the broker checks if it matches some predicate into its subscription table. If this is the case, the broker delivers m to the corresponding subscriber and set the proximity for itself into m's destination list to 0. This step will avoid to trigger further transmissions aiming at hitting the broker, as clarified next. Furtherly, the broker determines if it has to re-forward the message. This happens when m matches at least a predicate advised by a broker b_i such that: (1) b_i doesn't belong to the destination list of the message or (2) the proximity value for b_i computed by the receiving broker according to its proximity table is strictly lower than the one carried into the message.

[3]This check can be easily accomplished by remembering the identifiers of the messages received so far.

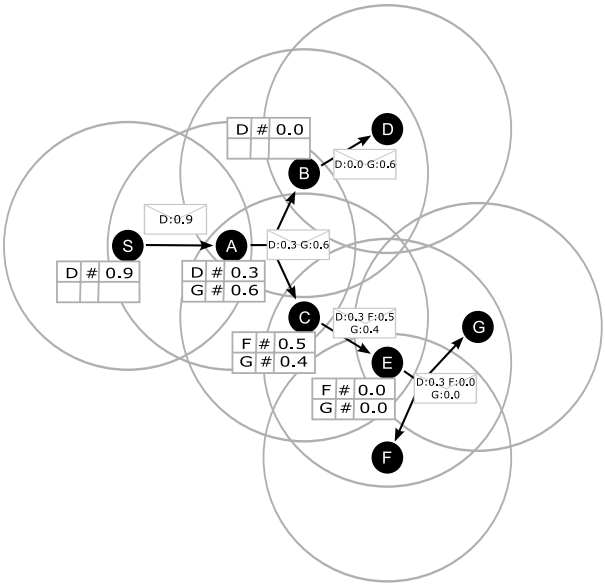


Fig. 4. An example of message routing.

In both cases the retransmission of the message m is scheduled after a delay proportional to the proximity for b_i owned by the receiving broker. When more than one broker exits that satisfies the conditions above, the delay is determined by the lowest proximity.

If none of the above cases hold, message should be dropped, but to increase delivery at the price of some more traffic, a new chance is given to the message for being forwarded. To this end, a message also carries an integer value, called the *credit* of the message, which represents the number of times a broker can force the retransmission of the message despite the fact that the conditions stated above about proximity do not hold. As shown in Fig. 3, if such a case occurs, the message is scheduled for transmission with the delay associated to the maximum possible proximity value, i.e. one. This way, forwarding due to credit tends to be cancelled by forwarding due to proximity.

Figure 4 portraits an example of message forwarding. The proximity table of a node is reported close to the node, while messages show the destination list they carry. For the sake of simplicity instead of storing the absolute time when the node received a beacon message, the last column of the proximity table stores the proximity value computed as explained above.

Suppose broker S generates a message matching subscriptions on brokers D, F, and G. S is only aware of the subscriptions at D, for which it holds a proximity of 0.9. It then sends the message with destination list $C : 0.9$. On receiving the message, broker A decides to forward it. Indeed, it knows another broker, broker G, which is interested in the message. Moreover, the proximity for D calculated by A is lower than 0.9. Brokers B and C both receive the message sent by A. Broker B re-forwards the message since it calculates a proximity 0.0 for D. Similarly, broker C re-forwards the message because it is aware of new broker F and also has a proximity for G (0.4) lower than that included into the

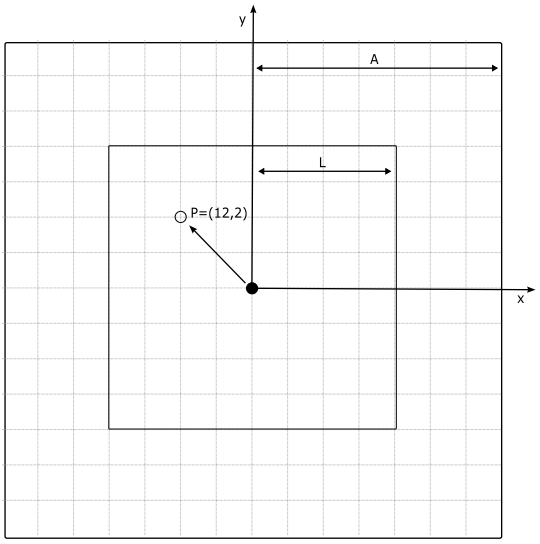


Fig. 5. A 14 × 14 grid ($A = 7$) with an example of relative position and transmission area for $L = 4$.

destination list of the message (0.6). Finally, broker E re-forwards the message since it calculates two proximity values for F and G lower than those included into the message (0.0 against 0.5 and 0.4, respectively).

V. Elapsed time distance correlation

The proximity-driven protocol assumes that the time elapsed since two nodes were most recently adjacent to each other is a measure of the chances of the two nodes being still close to each other, i.e., the lower the time the closer the nodes. In this section we provide a stochastic model that supports such a claim.

In particular, we developed a mobility model discrete in time and space, which captures the main short therm characteristic of a physical movement and calculates the conditional expected distance between two nodes, given the time elapsed since they lost direct connection. A similar analysis has been carried out in [15] for a Manhattan-like topology. The model presented here is more general as it removes the constrains on the movements, i.e. a node can move in any direction, including diagonals.

The field we consider (see Fig. 5) is a $2A \times 2A$ two-dimensional grid, wrapped along both directions, i.e., forming a torus. Two nodes move in this field by jumping from one point of the grid to an adjacent one. Fixing the coordinates system on one of the two nodes allows to express the position of the other as $P = (x, y)$, where $x, y \in [0, 2A)$ are the coordinates of node measured along the two axis. Analogously, we define the distance between the two nodes as the norm: $||x, y|| = max\{min\{x, 2A - x\}, min\{y, 2A - y\}\}$. As for connectivity, we suppose that a wireless link exists between the two nodes when $||x, y|| < L$. This means that the transmission area is a square with edge $2(L - 1)$ points. In the previous figure $L = 4$.

Movements change the relative position of the two nodes along the two axis independently from each other. The change can be of at most one grid point, in each direction, so that the relative movement can be described by a pair $M = (m_x, m_y)$, where $m_x, m_y \in \{-1, 0, 1\}$. Movements occur at regular discrete time ticks. Hence, if P is the current relative position of the two nodes and P' was the previous one, we can write $P = P' \otimes M = (x' \otimes m_x, y' \otimes m_y)$, where we define $a \otimes b = (a+b) \, mod \, 2A$ if $a+b \geq 0$ and $2A+(a+b)$ otherwise. According to this formula, movement at time t determines the position of the node at time t from its previous position at time $t - 1$.

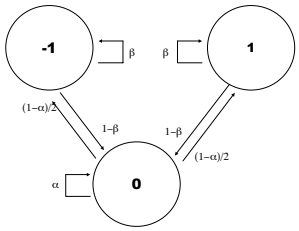


Fig. 6. Markov chain associated to movements.

To model realistic patterns of mobility while keeping the analysis tractable, we assume that the current movement, M, only depends on the previous one M', according to the following rules:

$$Pr\{m_x|m'_x\} = \begin{cases} \alpha & \text{if } m'_x = 0, m_x = 0 \\ (1-\alpha)/2 & \text{if } m'_x = 0, m_x \neq 0 \\ \beta & \text{if } m'_x \neq 0, m_x = m'_x \\ 1-\beta & \text{if } m'_x \neq 0, m_x = 0 \\ 0 & \text{otherwise} \end{cases}$$

a similar relationship also applies between m_y and m'_y.

These are the transition probabilities of the Markov chain portrayed in Fig. 6; α can be interpreted as the probability that the relative position along one axis doesn't change given that it was also unchanged in the previous time tick, while β represents the probability that the movement along one direction repeats again. Clearly α and β have to be less than 1. Also note that we do not allow m_x and m_y to change directly from 1 to -1 and vice versa; hence abrupt changes are avoided.

The average value of the module of the relative speed of the two nodes along an axis, denoted by s, is given by the stationary probability π_1 $(= \pi_{-1})$ of the chain:

$$s = \frac{1-\alpha}{(1-\alpha)+(1-\beta)} = \frac{1-\alpha}{2-(\alpha+\beta)}$$

Let $D(k)$ be the distance at time k between the two nodes. Its expected value depends on the initial "state" $s_0 = (x_0, y_0, m_{x_0}, m_{y_0})$ observed just after the wireless link between the nodes is broken. Specifically, if $Pr\{s_0\}$ is the probability of observing the state s_0, we can write:

$$\mathbf{E}[D(k)] = \sum_{s_0} \mathbf{E}[D(k)|s_0] Pr\{s_0\}$$

From our model of connectivity we have that the position after a breakage must belong to the square with edge L. Similarly, from how we defined the concept of movement, M,

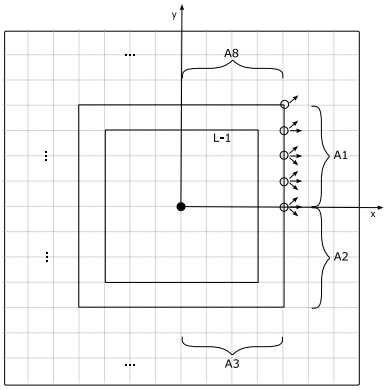


Fig. 7. Possible initial states after a link breakage.

we have that m_{x_0} and m_{y_0} cannot be arbitrary numbers but must be compatible (see later) with the current position of the two nodes and the event of link breakage. In particular, due to the symmetry of our model, we can divide the square with edge L into eight equivalent segments, $A_1, .., A_8$, as done in Fig. 7. They are equivalent in the sense that the conditional expected distance given the node exits from segment A_i, say $\mathbf{E}[D(k)|s_0 \in S_i]$ is the same for any segment (S_i being the set of possible states just after the link is broken due to the node exiting from segment A_i). Moreover, a node has the same probability of exiting from any segment. Hence:

$$\mathbf{E}[D(k)] = \sum_{i=1}^{8} \mathbf{E}[D(k)|s_0 \in S_i] Pr\{s_0 \in S_i\}$$

$$= \sum_{i=1}^{8} \mathbf{E}[D(k)|s_0 \in S_i] \frac{1}{8} = \mathbf{E}[D(k)|s_0 \in S_1]$$

To compute $\mathbf{E}[D(k)|s_0 \in S_1]$ we first observe that (x, y, m_x, m_y) can be an initial state, i.e. it belongs to S_1, only if the following conditions hold: (i) $x = L$, $y = 0..L$ and (ii) $m_x = m_y = 1$ if $y = L$; $m_x = 1$,$m_y = \in [0, 1]$ if $y = L - 1$; $m_x, m_y \in [-1, 0, 1]$ if $y < L - 1$. We say that a movement (m_x, m_y) is compatible with the position (x, y) if it satisfies the above conditions; they are depicted as small arrows near the possible exit positions in Fig. 7.

If we call $C(x, y)$ the set of movements that are compatible with the position (x, y) we can write:

$$Pr\{(x, y, m_x, m_y)\}$$

$$= Pr\{(x, y)\} Pr\{(m_x, m_y)|(m_x, m_y) \in C(x, y)\}$$

hence:

$$Pr\{(x, y, m_x, m_y)\}$$

$$= \frac{1}{L+1} \frac{Pr\{m_x\} Pr\{m_y\}}{\sum_{(m_x, m_y) \in C(x,y)} Pr\{m_x\} Pr\{m_y\}}$$

Let now $P_{s_0}(x, y, m_x, m_y, k)$ be the probability that at time k the position is (x, y) and the movement is (m_x, m_y), given that at all previous times $k' < k$ the position (x', y') was such that $||x', y'|| \geq L$ (i.e. the nodes never established the link again) and that at time $k = 0$ the state was s_0. We can write the recurrent equation as shown in Fig. 8.

$$\mathbf{E}[D(k)] = \sum_{i=1}^{8} \mathbf{E}[D(k)|s_0 \in S_i] Pr\{s_0 \in S_i\} = \sum_{i=1}^{8} \mathbf{E}[D(k)|s_0 \in S_i] \frac{1}{8} = \mathbf{E}[D(k)|s_0 \in S_1]$$

Fig. 8. Recurrent equation.

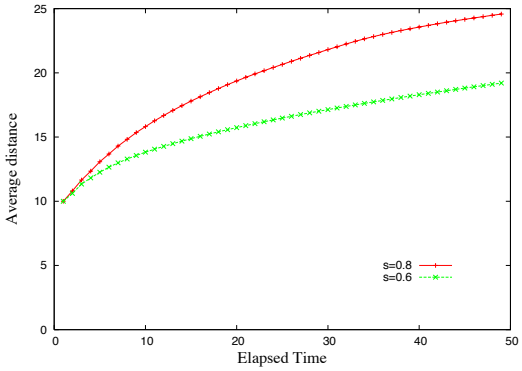


Fig. 9. Expected distance as a function of elapsed time after a link breakage, $A = 20, L = 10$.

Hence:

$$\mathbf{E}[D(k)] = \mathbf{E}[D(k)|s_0 \in S_1] = \sum_{s_0 \in S_1} Pr\{s_0\}$$

$$\sum_{||x,y||=\psi, |m_x| \leq 1, |m_y| \leq 1} \psi P_{s_0}(x, y, m_x, m_y, k)$$

In Fig. 9 we report the expected distance $D(k)$ obtained by numerically resolving the previous recurrent equation, as a function of k and with the speed s given as a parameter. We can observe how, for an elapsed time below a given threshold value, the lower the time the lower the expected distance. This means that for a given speed, the chances of the destination being close to a node are higher if the node has seen recently the destination and that such a chance increases as the speed is decreased. Thus, the elapsed times can be used to estimate the relative proximity of two nodes wrt a destination.

VI. Evaluation

To asses the performance of our proximity-driven routing protocol we simulated it using J-Sim [5], an open-source network simulator that provides a full simulation of the 802.11 protocol stack as well as a fairly complete and detailed signal propagation model. Simulation allows us to test the performance of our protocol in very large and complex scenarios, involving hundreds of mobile nodes, something very hard to achieve using real devices.

The main figures we measured were *delivery* and *overhead*. In a cb-ps application, the former is defined as the average ratio between the number of subscribers that received a message and the total number subscribers interested in the message. As for the overhead, we defined it as the average number of link-layer packets generated for each delivered message (including the beacon packets generated by our protocol).

As a baseline to evaluate the performance of our protocol, we used a pure gossip protocol, which represents the simplest structure-less approach for cb-ps message dissemination. In

Parameter	Default Value
Number of nodes	$N = 100$
Field area	$A = 1000 \times 1000\ m^2$
Minimum speed	$S_m = 10\ m/s$
Maximum speed	$S_M = 20\ m/s$
Number of publishers	$N_p = 2$
Publishing rate (for each publisher)	$Pr = 0.5\ msg/sec$
Number of subscribers	$N_s = 10$
Beacon interval	$\Delta t = 5\ sec$
Message credits	$Cr = 0$
Forwarding probability	$p = 0.5$

Fig. 10. Default simulation parameters.

the gossip protocol we considered, brokers delivers messages using the broadcast facility provided by the MAC layer (i.e., 802.11 in our simulations), adopting a forwarding probability $p \in (0, 1]$. This means that the broker running on the same node of the publisher, i.e., the first one forwarding the message, delivers it using an 802.11 broadcast packet, while the receiving brokers, independently from the content of their subscription tables, re-forwards it with a probability $p \in (0, 1]$.

A. Simulation Settings

The reference scenario we considered is that of a MANET composed of a number of nodes dispersed in a square field, which move around according to a random waypoint mobility model [9]. Each node randomly chooses a destination and starts moving toward it at a random speed. Once the destination has been reached, the node randomly determines another destination, and continues in that direction with a new randomly chosen speed.

The total number N of nodes, the area A of the field, and the minimum S_m and maximum S_M speed nodes can move at are the main physical parameters that characterize the simulated scenario.

As for the wireless protocol, we used the 802.11 network model provided by J-Sim. To reflect a realistic open field scenario, we choose a two rays ground propagation model with a random transmission range varying between 100 and 200 meters.

A broker runs on each node and it has either a single publisher, or a single subscriber attached to it, or it acts as a pure forwarder. We assume that N_p publishers produce messages of interest for N_s subscribers at a publishing rate of Pr msg/s. These parameters characterize the cb-ps application model.

Finally, the main parameters that characterize our protocol are the beaconing interval Δt and the number of credits Cr initially assigned to a message. Similarly, the gossip protocol we use for comparison is characterized by the probability of re-forwarding P.

Unless otherwise stated all the simulation parameters above assume the default values listed in Table 10.

B. Simulation Results

To have a baseline to start evaluating our protocol we first simulate the gossip protocol in our reference scenario (see Fig. 10) varying the forwarding probability p. Results are reported in Figure 11. It is worth observing how the delivery

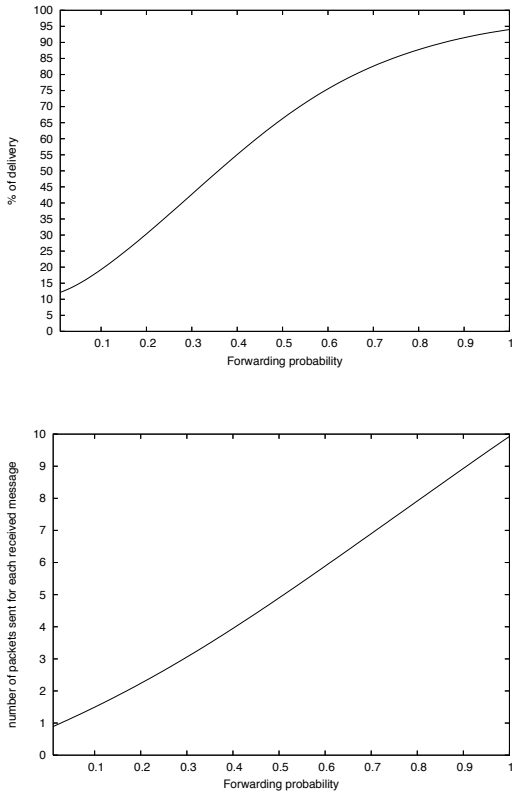


Fig. 11. Impact of forwarding probability on delivery (left) and overhead (right).

exhibits the typical bimodal behavior of gossip protocols [7]. We also note how 100% delivery is never reached due to collisions at the MAC layer and network partitioning, while a reasonable percentage of delivery, say more than half of the number of interested subscribers, can be achieved at the cost of at least 5 packets per delivered message.

Figure 12 shows the performance of our protocol as a function of the number of credits under the same reference scenario. Although the maximum delivery is slightly lower than the one obtained by gossip, reasonable high values can be reached at a much lower cost. As an example, without using any credit, our protocol delivers 70% of the messages published using less than a half packets with respect to gossip (respectively 2.5 and 5.5 packets per delivered message). Gossip shows the same overhead of our protocol with zero credits (i.e., 2.5 packets per delivered message) while delivering only 30% of the published messages. By increasing the number of credits the delivery can be increased while still keeping a high convenience with respect to gossip.

The next point to evaluate is how the number of subscribers affects the protocol's performance. Figure 13 shows the delivery and overhead as a function of the number of subscribers, measured under a different number of credits. The performance of the gossip protocol are also reported varying the gossip probability. It is interesting to note the effectiveness of the credits mechanism as a way to increase the delivery, which is particular effective when the number of subscribers

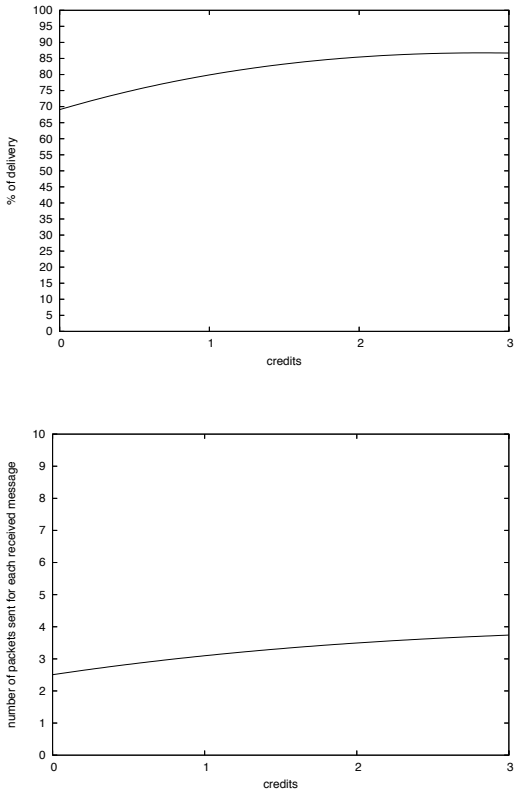


Fig. 12. Impact of credits on delivery (left) and overhead (right).

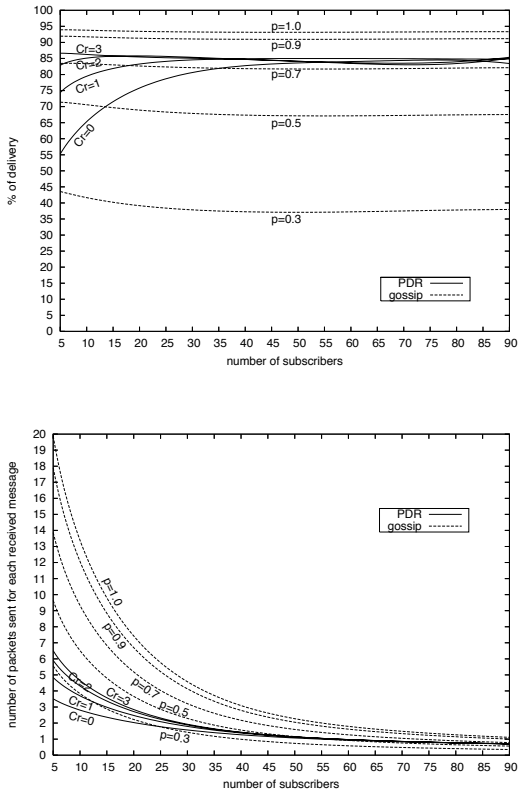


Fig. 13. Effect of increasing the number of subscribers on delivery (left) and overhead (right).

is low. Our protocol is always able to assure a high delivery fraction (more than 85%) independently of the number of subscribers and at progressively decreasing cost. As expected, the efficiency of the gossip protocol increases with the number of subscribers.

Another parameter that could impact performance is the rate of published messages. As shown in Fig. 14, our protocol is only very marginally influenced by this parameter, while gossip is much more sensible, expecially when p increases. This can be explained by remembering, from previous simulations, that gossip loads the network much more than our protocol. As a consequence, when the publishing rate increases, gossip suffers from a relevant number of collisions, which do not occur when our protocol is used. It is worth noticing that an increase in the publishing rate also decreases the overhead of our protocol. To understand this behavior we have to remember that as part of the overhead we also count the beacon packets produced by our protocol. The number of such packets is fixed (it depends on the beaconing interval, only) and it represents a lage fraction of the overhead when a few messages are published. When the number of messages published (and consequently received) grows, this contribute to the overhead becomes negligible.

Next step is to study how mobility affects the performance of our protocol. Figure 15 shows how delivery and overhead change when the speed of nodes increases, showing also the impact of adopting different beaconing intervals (i.e., 2.0, 4.0, and 6.0 seconds) at different speeds. At fist one could think that a shorter beaconing interval would provide the best performance (not considering the impact on the overhead). The number of missed beacons, in fact, is the key parameter we use to estimate the distance between brokers, and ultimately to guide our protocol. A shorter beaconing interval should provide the best accuracy and consequently the best performance. On the other hand, we have to remember that in order to make sure that the "missed-beacons to distance" correlation is valid for the entries stored in the proximity table, we delete them after 10 missed beacons. As a consequence, under low mobility a short beaconing interval results in removing valid entries from the table (i.e. those for which the correlation is still valid). Convrsely, under a high mobility degree a long beaconing interval does not provide enough accuracy. This explains the results in Fig. 15. The same beaconing interval cannot account for every situation. Each range of speeds has an "ideal" beaconing interval, while other choices reduce performance.

To analyze how our protocol performs when the size of the network grows, and to compare it against the gossip, as before, we varied the number of nodes in the network, while keeping their density constant, i.e., by also increasing the size of the simulated area. Given the high impact on the protocols' performances when the density of subscribers changes, as shown in Fig. 13, we fixed the percentage of subscribers with respect to the total number of nodes N to 10%. For the same rason we fixed the percentage of publisher to 2% of N and we

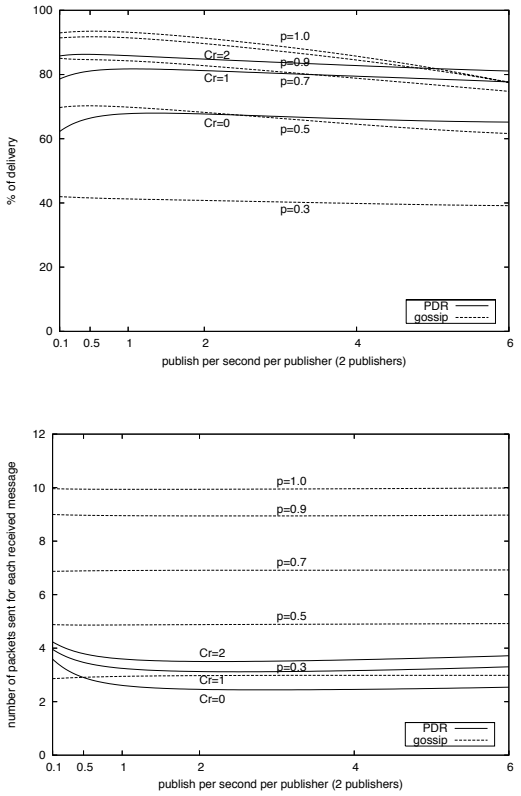


Fig. 14. Effect of increasing the publishing rate on delivery (left) and overhead (right).

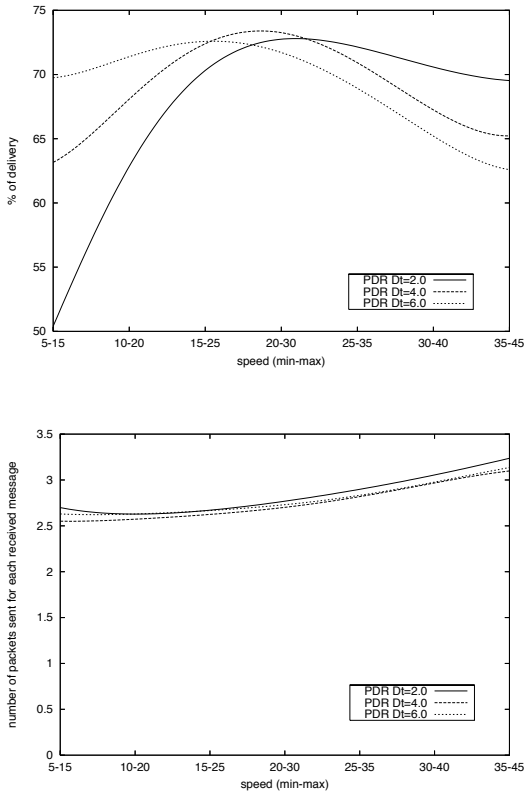


Fig. 15. Delivery and overhead versus speed at different beaconing intervals.

kept the publishing rate (per publisher) constant. As shown in Fig. 16 both our protocol and gossip scale very well, with gossip decreasing slightly its delivery as the network size grows, and our protocol marginally increasing it.

The second scalability test we run, see Fig. 17, consists in increasing the number of nodes N while keeping the area A constant, hence producing an increase in the node density. Here, we observe an interesting phenomenon, which is due to the increasing number of collisions: a low gossiping probability provides better performance as the density increases, while using a higher probability performance starts decreasing after a given number of nodes. Our protocol is much more resilient to collisions because of the suppression mechanism it uses, which can be considered a form of auto-adaptation to the density of the network. Here, as usual, the overhead of our protocol is far better than gossip and is rather constant with respect to the increasing density of nodes.

VII. Related Work

As mentioned in Section II, the last decade saw the development of several cb-ps middleware platforms, which embody different routing algorithms to implement content-based publish-subscribe facilities for fixed networks (for a detailed comparison see [8], [1], [3], [16]).

At the same time, in the last few years we have seen a continuous growth in the dynamism of networks, motivated by peer-to-peer overlays and wireless networking, which originated a number of challenging issue for routing in general and for content-based routing in particular. Our research group have addressed these issues within the cb-ps context, by adapting traditional tree-based cb-ps routing techniques, optimal from a pure traffic standpoint in stable networks, with the addition of advanced tree maintenance mechanisms for both fixed peer-to-peer networks [14], and mobile ad-hoc networks [13]. These experiences convinced us that tree-based routing techniques cannot be applied in very dynamic scenarios like those we consider in this paper.

To overcome the intrinsic fragility of trees, other researchers experimented mesh based solutions. Among them, Yoneki and Bacon [19] proposed using the On Demand Multicast Routing Protocol (ODMRP) for building an optimized dissemination mesh by applying techniques developed for multicast routing in MANETs to the context of a cb-ps system. In particular, they used bloom filters to summarize subscriptions. As a result, the cb-ps scheme is actually approximated to a topic based one, and the cost of this approximation is clearly an intrinsic limitation to such a solution.

However, as mobility and size of the network grow, overlay state maintenance, whichever form it takes, tends to pose severe scalability problems. As a way to alleviate them, Content Based Multicast [20] and STEAM [12] introduce *spatial scopes* as a way to limit the diffusion of messages to a restricted geographical zone, thus avoiding the burden of maintaining a global routing topology. In Content Based Mul-

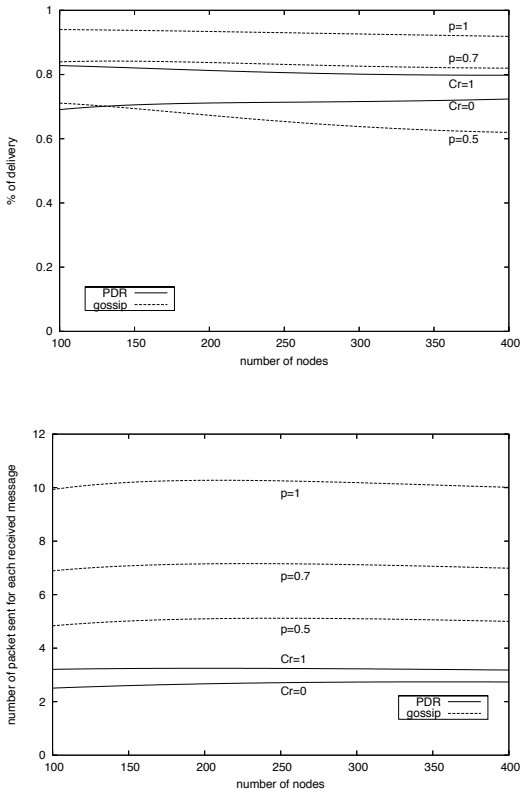


Fig. 16. Delivery and overhead as network size grows.

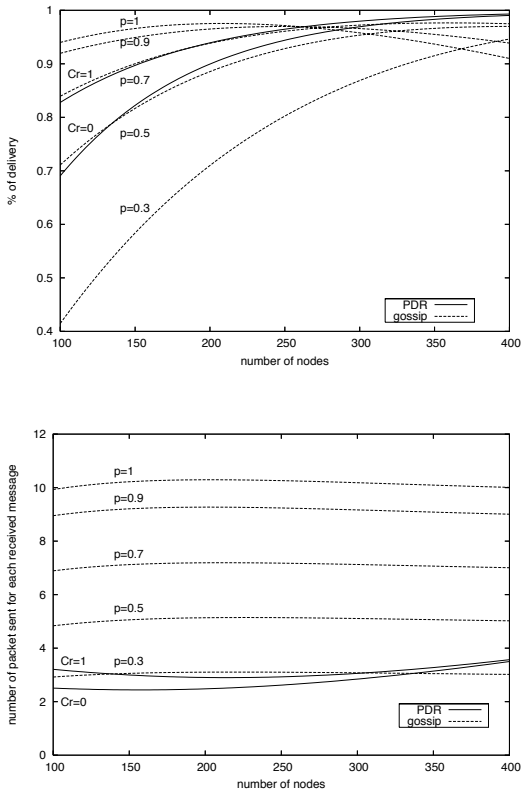


Fig. 17. Delivery and overhead as node density increases.

ticast messages are generated and spread in a given direction up to a given distance. Nodes interested in information about a given area send pull requests (i.e, subscriptions) in that area, where the matching actually occurs. Similarly, STEAM offers a content-based service to applications and implements it using a "proximity-group" communication mechanism. Proximity groups are defined by a proximity area, which defines the validity of messages and the type (the subject) of the filter, while the full content based filters are not used for routing but kept by subscribers. Moreover, the group communication middleware used by STEAM [10] provides strong guarantees about message delivery and ordering, but of course it will scale poorly if scopes grew beyond a given scale.

Besides spatial scoping, which intrinsically provides content-based routing only in a limited area around the publisher, other mechanisms were recently proposed to limit the amount of state brokers needed to keep while still providing content-based routing network-wide. In [2], a TTL is used to limit the propagation of subscriptions. Published messages "follow" the routes defined by subscription propagation (as in subscription forwarding routing — see Section II) when they are available, otherwise they are propagated using pure probabilistic gossip techniques. Similarly, the authors of Autonomous Gossip [4] propose a completely stateless, bio-inspired, self organizing mechanism to disseminate informations in a content-based fashion. Each mobile node of a MANET has a profile, which can be thought of as the node's subscriptions, and a destination, which is the place where the node is going, which is assumed to be known to the routing algorithm. Messages are also labelled with a profile, a destination, or both. Messages migrate from node to node according to their "similarity" with the node's profile and destination. Depending on the hospitality (based on similarity) received at the present node, messages decide to either continue to reside, migrate, or replicate to another node with a more suitable profile and/or goal destination. While similar to our algorithm in the idea of a totally structure-less dissemination based only on local information, Autonomous Gossip has several peculiarities. Indeed, with our algorithm messages follow the "traces" left by mobile nodes, which spread their interests through beacons that populate the proximity tables used for routing. Conversely, Autonomous Gossip messages reside in the routing nodes indefinitely, waiting to find interested targets.

Location-aided routing, also called geographical routing, or position-based routing, is a different approach, which also holds the promise of providing cb-ps in a structure-less fashion. Currently, this kind of routing has been used for addressing unicast messages in MANETs in a way that avoids keeping and maintaining routing tables by exploiting the knowledge about the geographical position of the destinations. This allows to keep the routing decision entirely local to the forwarding node, by propagating messages along a path of decreasing distance to the destination, in a similar way our own routing algorithm does. Recently, geographical routing have been extended to provide multicast services, e.g., see [11] and [18], while, to the best of out knowledge, it has never been

applied to content-based dissemination. Clearly, our algorithm can be viewed as a special form of geographical routing, even if it is characterized by several peculiarities. First of all the domain of application, content-based routing, which is new in the area. Moreover, all the previous geographical routing approaches we are aware of, rely on some sort of self-localization mechanism to compute the current position of nodes (e.g., through a GPS) and on a location service, used to determine the position of message recipients. Our approach relaxes both these requirements, in creasing the applicability of the approach.

Finally, the idea of exploiting the elapsed time since two nodes were most recently neighbors as a proximity metric has been originally proposed in [6] in the framework of on-demand path discovery. This idea has been further elaborated in [15].

VIII. Conclusions

In this paper we explored a new approach to content-based routing in Mobile Ad Hoc Networks. Our protocol doesn't require any network-wide structure to support routing decisions. Rather, it uses broadcast to efficiently send a message to all neighboring nodes and defers to them the decision to forward the message based on an estimation of their distance from a potential subscriber of the message.

The protocol is very resilient to topological changes and can thus be profitably used in settings characterized by a high mobility degree. We have shown through simulations that messages can be delivered with high probability to the interested subscribers at a low cost. We are currently investigating how to improve the performance by increasing the accuracy of the estimations by taking other information, e.g. the permanence of a node close to another, into account.

Acknowledgments

The authors would like to thank Gian Pietro Picco and Paolo Costa for the interesting discussions about content-based publish/subscribe systems in mobile ad-hoc networks.

References

[1] A. Carzaniga, D. S. Rosenblum and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Trans. on Computer Systems*, 19(3):332–383, August 2001.

[2] P. Costa and G. P. Picco. Semi-probabilistic content-based publish-subscribe. *Proc. of the 25^{th} Int. Conf. on Distributed Computing Systems*, Columbus, OH, June 2005. IEEE Press.

[3] G. Cugola, H. Di Nitto and A. Fuggetta. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. *IEEE Trans. on Software Engineering*, 27(9):827–850, September 2001.

[4] A. Datta, S. Quarteroni and K. Aberer. Autonomous gossiping: A self-organizing epidemic algorithm for selective information dissemination in wireless mobile ad-hoc networks. *In Proceedings of the International Conference on Semantics of a Networked World (IC-SNW04)*. LNCS, Springer Verlag, 2004.

[5] J-Sim Web Page. `http://www.j-sim.org`.

[6] H. Dubois-Ferriere, M. Grossglauser and M. Vetterli. Age matters: efficient route discovery in mobile ad hoc networks using encounter ages. *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pp. 257–266, Annapolis, MD, 2003. ACM Press.

[7] Z. Haas. *et al*. Gossip-based ad-hoc routing. *Proceedings of IEEE INFOCOM 2002*, New York, 2002. IEEE.

[8] P. T. Eugster, P. A. Felber, R. Guerraoui and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[9] D. B. Johnson. Routing in Ad Hoc Networks of Mobile Hosts. *Proc. of the Workshop on Mobile Computing Systems and Applications*, pp. 158–163, 1994.

[10] M. Killijian, R. Cunningham, R. Meier, L. Mazare and V. Cahill. Towards group communication for mobile participants. *Proceedings of the 1st ACM Workshop on Principles of Mobile Computing (POMC 2001)*, pp. 75–82, 2001.

[11] M. Mauve, H. Föler, J. Widmer and T. Lang. Position-based multicast routing for mobile ad-hoc networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):53–55, 2003.

[12] R. Meier and V. Cahill. Steam: Event-based middleware for wireless ad hoc networks. *Proceedings of the 1st International Workshop on Distributed Event-Based Systems (DEBS '02)*, Vienna, Austria, July 2002.

[13] L. Mottola, G. Cugola and G. P. Picco. Tree overlays for publish-subscribe in mobile ad hoc networks. Technical report, Dipartimento Elettronica e Informazione, Politecnico di Milano, 2005. Submitted for pubblication at Percom'06.

[14] G. P. Picco, G. Cugola and A. L. Murphy. Efficient Content-Based Event Dispatching in Presence of Topological Reconfiguration. *Proc. of the 23^{rd} Int. Conf. on Distributed Computing Systems (ICDCS03)*, pp. 234–243. ACM Press, May 2003.

[15] R. Baldoni, R. Beraldi and L. Querzoni. A hint based probabilistic protocol for unicast communications in manets. *Ad Hoc Networks (to appear)*.

[16] D. S. Rosenblum and A. L. Wolf. A Design Framework for Internet-Scale Event Observation and Notification. *Proc. of the 6^{th} European Software Engineering Conf. held jointly with the 5^{th} Symp. on the Foundations of Software Engineering (ESEC/FSE97)*, LNCS 1301, Zurich (Switzerland), September 1997. Springer.

[17] C.-K. Toh. *Ad hoc Mobile Wireless Networks*. Prentice Hall PTR, Upper Saddle River, 2002.

[18] M. Transier, H. Föler, J. Widmer, M. Mauve and W. Effelsberg. Scalable Position-Based Multicast for Mobile Ad-hoc Networks. *Accepted for Proc. of the First International Workshop on Broadband Wireless Multimedia: Algorithms, Architectures and Applications (BroadWim 2004)*, San Jose, CA, October 2004.

[19] E. Yoneki and J. Bacon. An adaptive approach to content-based subscription in mobile ad hoc networks. *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW04)*. IEEE, 2004.

[20] H. Zhou and S. Singh. Content based multicast (cbm) in ad hoc networks. *MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pp. 51–60, Piscataway, NJ, 2000. IEEE Press.