Semi-probabilistic Content-Based Publish-Subscribe

Paolo Costa and Gian Pietro Picco
Dip. di Elettronica e Informazione, Politecnico di Milano
{costa,picco}@elet.polimi.it

Abstract

Mainstream approaches to content-based distributed publish-subscribe typically route events deterministically based on information collected from subscribers, and do so by relying on a tree-shaped overlay network. While this solution achieves scalability in fixed, large-scale settings, it is less appealing in scenarios characterized by high dynamicity, e.g., mobile ad hoc networks or peer-to-peer systems. At the other extreme, researchers in the related fields of multicast and group communication have successfully exploited probabilistic techniques that provide increased fault tolerance, resilience to changes, and yet are scalable.

In this paper, we propose a novel approach where event routing relies on deterministic decisions driven by a limited view on the subscription information and, when this is not sufficient, resorts to probabilistic decisions performed by selecting links at random. Simulations show that the particular mix of deterministic and probabilistic decisions we put forth in this work is very effective at providing high event delivery and low overhead in highly dynamic scenarios, without sacrificing scalability.

1. Introduction

Modern distributed applications are increasingly designed as collections of components interacting through an event-based communication paradigm. Some components observe events and *publish* notifications, which are in turn delivered to those components who *subscribed* to receive them. In the *content-based* incarnation of this publish-subscribe model, the filtering of relevant events is specified by the subscriber using predicates on the event content (e.g., using regular expressions and logic operators), therefore providing additional expressiveness and flexibility.

Content-based publish-subscribe has proven useful in a number of distributed scenarios, due to its high degree of decoupling, asynchronicity, and versatility. Moreover, its inherently reactive style of communication makes it particularly suited for highly dynamic scenarios like those characterizing mobile ad hoc networks (MANETs), peer-to-peer networks, and sensor networks.

The potential for the content-based publish-subscribe *model*, however, can be fully unleashed in these scenarios only if the underlying *system* is designed in a way that is compatible with their requirements. Highly dynamic settings like the aforementioned ones pose unprecedented challenges, mostly determined by the fluidity of the system's network topology. Mainstream systems implementing a distributed infrastructure for publish-subscribe are typically geared to large-scale settings, striving for scalability by routing events on a tree-shaped overlay network,but with few or no mechanisms in place to deal with topological reconfiguration.

Our research group has been particularly active in studying how to design content-based publish-subscribe systems able to efficiently tolerate frequent topological reconfigurations [7, 8, 17]. Nevertheless, the approach we followed thus far also relied on the existence of a tree-shaped overlay network. Indeed, this provided a good starting point that enabled us to build up on well-established results in the field. However, depending on the degree of dynamicity and other parameters characterizing the scenario, maintaining a tree overlay may bring additional overhead and complexity.

On the other hand, approaches in closely related fields, e.g., MANET multicast and subject-based publish-subscribe, cope with dynamicity by routing messages probabilistically, instead of deterministically based on the collection of subscription information. In doing so, they effectively trade delivery guarantees for enhanced scalability, fault tolerance, and resilience to topological changes.

In this paper, we take a different perspective and establish a clear point of departure from our previous work—and, to the best of our knowledge, from that of other researchers' as well. First, our solution relies on an undirected connected graph topology. Given the dynamicity requirements of our target scenarios, a graph structure is not only considerably easier to maintain than a tree, but also provides opportunities for more fault-tolerant solutions by intrinsically providing multiple routes between any two dispatchers. Second, our solution is neither entirely deterministic or probabilis-

tic. Instead, it strikes a balance between the two, since it combines the efficiency of deterministic routing with the resilience to reconfiguration and inherent simplicity of probabilistic approaches.

The core of our approach is very simple. Subscriptions are propagated only in the immediate vicinity of a subscriber, in contrast to most existing systems. Event routing leverages of this subscription information, whenever available, by deterministically routing an event along the link a matching subscription was received from. If no subscription information exists at a given dispatcher, events are forwarded along a randomly chosen subset of the available links. Simulations show not only that our approach is successful per se in providing high delivery rates with low overhead, but also confirm that it performs better than a fully deterministic (or probabilistic) alone. Moreover, nice byproducts of our strategy are a significant reduction of the routing tables size, and easier development and understanding of the actual implementation.

The rest of the paper is organized as follows. Section 2 provides the reader with a concise overview of content-based publish-subscribe. Section 3 presents the details of our approach, by illustrating the strategy for routing subscriptions and events. Section 4 provides an evaluation of our approach through simulation in several scenarios exhibiting different degrees of dynamicity, including mobile ones. Section 5 places our work in the context of related ones. Finally, Section 6 concludes the paper and hints at opportunities for further research on the topic.

2. Content-Based Publish-Subscribe

Applications exploiting publish-subscribe middleware are organized as a collection of autonomous components, the *clients*, which interact by asynchronously *publishing* event notifications (or, shortly, *events*) and by *subscribing* to the classes of events they are interested in. A component of the architecture, the *event dispatcher*, is responsible for collecting subscriptions and forwarding events to subscribers, effectively decoupling the event producers and consumers.

Recently, many publish-subscribe middleware have become available, which differ along several dimensions. In this paper, we focus on systems that realize a *distributed* event dispatcher and enable subscribers to filter events of interest based on their *content* (e.g., by means of regular expressions), rather than their *subject* (i.e., a class of events, defined *a priori*). Among these systems, popular choices are the use of an unrooted tree topology for the overlay network connecting dispatchers, combined with a routing strategy called subscription forwarding.

In a *subscription forwarding* scheme [5], subscriptions are delivered to *all* dispatchers, and are used to establish the routes that are followed by published events. When a client issues a subscription, a message containing the corresponding *event pattern* to be used for filtering is sent to the dispatcher the client is attached to. There, the event pattern is inserted in a subscription table, together with the identifier of the subscriber. Then, the subscription is propagated by the dispatcher, which now behaves as a subscriber with respect to the rest of the dispatching network, to all of its neighboring dispatchers on the overlay network. In turn, they record the subscription and re-propagate it towards all their neighboring dispatchers, except for the one that sent it. This scheme is typically optimized by avoiding propagation of subscriptions for the same event pattern in the same direction. The propagation of a subscription effectively sets up a route for events, through the reverse path from the publisher to the subscriber. Requests to unsubscribe from a given event pattern are handled and propagated analogously to subscriptions, although at each hop entries in the subscription table are removed rather than inserted.

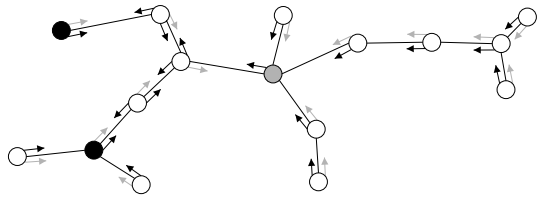

Figure 1. A dispatching network with subscriptions laid down according to a subscription forwarding scheme.

Figure 1 shows a dispatching network with two dispatchers subscribed[1] for a "black" pattern, and one for a "gray" pattern. (Hereafter, we ignore the presence of clients and consider only dispatchers.) Arrows denote the routes laid down according to these subscriptions, and reflect the content of subscription tables. As a consequence of the subscription forwarding process we described, the routes for the two separate subscriptions are laid down on the single tree constituting the dispatching network. This choice is typical of content-based systems and is motivated by the fact that a single event may match multiple patterns. Therefore, routing on multiple independent trees, as typically done by subject-based systems, would lead to inefficient duplication of events along the separate trees.

3. Semi-probabilistic Content-Based Routing

Several alternatives exist to the mainstream scheme we just described, as we discuss in Section 5. The approach we present in this paper, however, takes a distinctly different and novel perspective on the problem, motivated by the requirements we address.

[1] Content-based systems allow rather sophisticated expressions. For instance, the two subscriptions could be {`Software* OR Appl*s`} and *B* {`Distributed AND ?pplication?`}, and events containing `"Distributed Applications"` would match both.

In our target application scenario the connectivity configuration of hosts, and therefore dispatchers, can change freely and frequently. This requirement encompasses application scenarios like mobile ad hoc networks, peer-to-peer networks, and many variants of sensor networks. Our earlier work on topological reconfiguration of publish-subscribe showed that it is possible to reconcile routing information [17] and recover events lost during reconfiguration [7] efficiently. However, it still assumed the availability of an underlying tree-shaped overlay network. The management of such an overlay network introduces considerable complexity in the implementation and, when the setting is highly dynamic, is a possible cause of inefficiency. Moreover, a tree provides exactly one route among any two dispatchers and, worse, among any two subtrees. Therefore, when a link fails the message flow is severely disrupted.

In this paper we set out to devise a routing strategy that:

- tolerates arbitrary reconfigurations of the connectivity among dispatchers;
- provides high, and steady, rates of event delivery with low overhead;
- is scalable;
- makes minimal assumptions on the underlying overlay and physical networks;
- is simple, therefore leading to small-footprint software artifacts, easily deployable on resource-constrained devices, possibly down to sensors.

To achieve these goals, first of all we abandon the tree-shaped overlay network and simply assume that dispatchers are able to communicate along the links of the connectivity graph. On fixed networks, this graph can be easily built as an overlay, e.g., by reusing Gnutella-like solutions. On mobile networks, existing protocols (e.g., [2]) based on network- or application-level beaconing can be used. A major advantage of basing our approach on a graph is that it is an inherently more fault-tolerant structure than a tree.

Nevertheless, the deterministic routing strategy we described in Section 2 would not work on a graph, as it would create plenty of loops. Therefore, some adaptation is required. Moreover, we contend that virtually *any* fully deterministic strategy is going to experience severe drawbacks in the highly dynamic scenario we target, where routing information quickly becomes stale.

On the other hand, probabilistic approaches like *epidemic* (or *gossip*) algorithms [1, 10] are known to satisfy many of the aforementioned requirements in the context of multicast communication. Inspired by the spreading of diseases, these algorithms forward information at random towards a small subset of available nodes, and rely on the availability of multiple routes to ensure that the "infection" carrying the information extends to a sufficient percentage of the receivers. Epidemic algorithms essentially trade the absolute guarantees provided by deterministic approaches for probabilistic ones, yielding in turn increased scalability and resilience to change, as well as reduced complexity.

The solution we present in the following combines the two approaches in the context of content-based publish-subscribe. On one hand, we still maintain deterministic information about subscriptions but only in the vicinity of a dispatcher, therefore reducing the likelyhood of loops and yet providing accurate information for routing events. On the other hand, in the portion of the network where this localized information is unavailable we complement it with probabilistic routing decisions, by routing events at random along a small subset of the available links. The remainder of this section describes our approach in more detail. Section 4 shows that this simple strategy is indeed successful in meeting all of our aforementioned requirements.

Subscription Propagation. As we already mentioned, in a highly dynamic network it may become impractical to maintain subscription information about every node in the network. In our approach, each dispatcher knows only a limited portion of the interests of the other dispatchers, determined by the *subscription horizon* ϕ. The value of this parameter represents the number of hops a subscription is propagated away from the subscriber.

Propagation occurs similarly to subscription forwarding. When a subscription request is issued by a dispatcher, the corresponding message is forwarded to all of its neighbors, which update their subscription tables accordingly. If $\phi = 1$, no further action is taken. Otherwise, each dispatcher forwards the subscription message to all of its neighbors, except the one who sent the message. A subscription is never forwarded twice along the same link, unless an unsubscription occurs in between the two. Differently from subscription forwarding, however, the subscription tables maintain information not only about which subscription was received on which link, but also about the distance of the subscriber, ranging between zero (for a local subscription) and ϕ. Figure 2 shows the layout of subscriptions for a case where $\phi = 1$.

Topological reconfigurations, i.e., the appearance of a new link or the vanishing of an existing one, must induce a proper reconfiguration of subscription information. However, this is easily accomplished by relying on (un)subscription operations, as discussed in [17]. When a dispatcher detects the presence of a new link, it simply sends a subscription message along that link. Similarly, when a link vanishes, the dispatcher behaves as if it received an unsubscription message for all patterns associated to that link. These (un)subscriptions are then propagated based to the extent determined by ϕ.

Event Propagation. Event propagation is where probabilistic decisions may come into play. An event received on

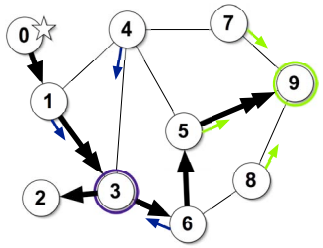

Figure 2. Semi-probabilistic routing with $\phi = 1$ and $\tau = 0.5$. Numbered circles represent dispatchers. A colored circle around a dispatcher denotes it as a subscriber. The short colored arrows represent subscription information, and indicate the forwarding path for matching events. Dispatcher 0 publishes an event that gets forwarded either deterministically (double-headed thick arrows) or probabilistically (single-headed thick arrows).

a dispatcher's link is always forwarded only along a subset of the remaining $l - 1$ links, being l the dispatcher's degree in the graph interconnecting dispatchers. The selected links may change from time to time, but the overall percentage of links exploited for forwarding is fixed, and determined by the *event propagation threshold* τ. This parameter allows to compute the number f of links to be used for forwarding as $f = \lceil \tau(l-1) \rceil$, where the ceiling operator is necessary to guarantee that the event is always forwarded along at least one link.

Upon receiving an event, in principle[2] the following processing occurs. First, the subscription table is inspected for subscriptions matching the event. If a match is found, the event is routed along the link associated to the subscription. Subscriptions selection is prioritized according to ϕ: an event is forwarded based on a subscription at distance d only if there is no matching subscription at distance $d - 1$. As we verified through simulation, this strategy reduces the likelihood of forwarding the event along a stale route.

This step is iterated until the number of links used for propagation is greater than f. If the number of matching subscriptions is not sufficient, the propagation threshold is met by forwarding the event along as many links as needed to reach f, randomly selected among those that have not been used in the current forwarding step. The only exception is constituted by links associated to subscriptions at distance $d = 1$; a matching event is forwarded along *all* of these links, regardless of the propagation threshold. The rationale is the fact that subscriptions at $d = 1$ represent the most accurate routing information, and the most direct route towards the corresponding subscribers. Finally, it is important to note that, during the overall process, an event is never forwarded twice along the same link.

Figure 2 shows an example. Let us assume that $\tau = 0.5$ and that an event matching both subscriptions is published by dispatcher 0. This dispatcher has only one link and no subscription information: therefore, the event gets forwarded to dispatcher 1 as this is the only alternative. At dispatcher 1, two links are available. Nevertheless, the link towards 3 is associated with subscription information: it is therefore selected for forwarding and no further action is taken since the threshold is met. At dispatcher 3, the event is delivered locally. Moreover, the links towards 2, 4 and 6 are all viable routing options, and the event must be forwarded along $f = 2$ links. No deterministic information is available, therefore the decision is done entirely at random. The figure shows the case where the event is forwarded towards 2, where it stops propagating, and 6. There, the same situation occurs, with the links towards 5 and 8 as viable options. The figure shows the case where 5 is selected. At this dispatcher, the presence of deterministic information "captures" the events and steers it towards 9, where it gets locally delivered.

It is interesting to note that, at each hop, an event may be routed according to different criteria. As we already mentioned, "holes" in the dissemination of subscription information are bypassed by relying on random selection of links. However, the very nature of content-based systems is an asset for our routing approach, because an event matching multiple subscriptions may leverage of a bigger set of subscription information during its travel. Again, this is exemplified in Figure 2, where the event not only is routed by a mixture of deterministic and non-deterministic decisions, but deterministic ones (i.e., the hops from 1 to 3, and from 5 to 9) are generated by different subscriptions.

Dealing with loops. With reference to Figure 2, a choice of $\phi = 2$ would have created a routing loop among the nodes 4, 5, 7, and 9. Routing loops are easily detected by relying on a unique identifier for every event, easily implemented using the identifier of event publisher and the value of a counter incremented at the publisher each time it publishes an event. Therefore, an event received is actually propagated by a dispatcher only if it has never been received before.

More sophisticated loop avoidance and detection algorithms are available in the literature. However, on one hand they are likely to be impractical in the highly dynamic scenario we target, while on the other hand they would introduce a lot of complexity in our algorithm, which instead we want to keep as lightweight as possible.

Avoiding unnecessary propagation. In Figure 2, we note how event forwarding does not really stop at dispatcher 9, since there is no way to know that no other subscriber exists in the system. Without a way to cease forwarding, events would be forwarded indefinitely—more precisely, until a loop is detected. Indefinite propagation is dealt with by attaching a time-to-live (TTL) field to each event message, and by decrementing its value at every hop. When an event

[2]This sequence of steps serves only for illustration purposes: a number of optimizations are possible in reality.

is duplicated at a dispatcher along multiple routes, all the copies retain the same TTL value. Therefore, an event is propagated only if its TTL is greater than zero.

A more refined mechanism consists of associating different TTLs to the two form of routing we exploit, therefore defining a *deterministic TTL* (TTL_d) and a *probabilistic TTL* (TTL_p). With this scheme, propagation ceases when either of the TTLs reaches zero. The advantage of this scheme is that it provides a direct way to control both aspects of propagation, therefore enabling a more accurate tuning of the performance of our approach.

4. Evaluation

In this section we evaluate our approach through simulation. The metrics we analyze are event delivery rate and overhead. The former is defined as the ratio between the number of subscribers that should receive a given event and those who actually get it. The overhead is constituted by subscription messages and by event messages that are either duplicated, never received by a subscriber, or routed along unnecessarily long routes. These contributions are difficult to separate, and in any case do not provide significant insights. Therefore, we analyze overhead by simply plotting the overall number of messages flying in the system.

After describing the simulation setting, we first focus on a static scenario. This allows us to understand the behavior of our solution without the bias introduced by reconfiguration, and to verify that indeed it performs well when the system is stable. Then, we move to a scenario where reconfiguration occurs, and distinguish between a milder setting where each network change occurs in isolation, and an extreme one where reconfiguration occurs at a very high rate. Finally, we evaluate our approach against reconfiguration patterns determined by using ANSim [13], a generator of mobility scenarios. All of our simulations are developed with OMNeT++ [19], a free, open source discrete event simulation tool.

Simulation Setting. Events are represented as randomly-generated sequence of integers, determined using a uniform distribution. An event pattern associated to a subscription is represented by a single number. An event matches a subscription if it contains the number specified by the event pattern in the subscription. Each dispatcher is subscribed to two event patterns, drawn randomly from the overall number of patterns available in the system, which in our simulation is set to 512. For each event, the percentage of receivers is about 10% of the overall number N of dispatchers in the system as this is a commonly accepted "rule of thumb" for content-based systems (see [6]).

Our simulations are run with dispatchers continuously publishing events on a network with stable subscription information, where no (un)subscriptions are being issued. The frequency at which the events are published by each dispatcher determines the system load in terms of event messages to be be routed. We choose a rather high publishing load scenario with about 5 publish/s per dispatcher. Therefore, the number of the published events grows linearly with the size of the system.

Parameter	Default value
Network size	$N = 300$
Graph degree	$l = 5 \pm 1$
Patterns per node	2
Available patterns	512
Percentage of receivers	10%
Published events/s per dispatcher	5

Table 1. Simulation parameters and their default values.

The graph constituting the overlay network is built with a constant degree l, to eliminate as much as possible the bias induced by a random shape. This requirement about a constant degree is maintained also in the case of a dynamic network, although the actual number of links of a dispatcher is allowed to vary between $l - 1$ and $l + 1$.

Finally, each simulation was run 10 times with different seeds and the values averaged. The simulation parameters are shown in Table 1, together with their default values.

Static Network. We first analyze a static scenario, without topological reconfigurations. This allows us to analyze the behavior of the various approaches without the perturbations induced by reconfiguration. Also, we initially set TTL=∞ to first analyze the behavior of the system without limiting event propagation.

The top chart in Figure 3 shows the event delivery rate for a configuration with an event propagation threshold $\tau = 0.25$ and with a subscription horizon varying between $\phi = 0$ (purely probabilistic) and $\phi = 3$. The chart evidences that indeed deterministic information boosts delivery, which doubles when moving from a purely probabilistic routing to one with a 1-hop subscription information. Nevertheless, the improvement clearly cannot be linear: $\phi = 2$ brings an additional 8% improvement, and there is no appreciable difference against $\phi = 3$. The reason for the overlapping of these two curves is that, unlike $\phi = 1$, they are subject to the limitation on propagation set by τ. As we verified through simulation, if we were to use *all* the deterministic information, the delivery of these curve would not only be different but also better. On the other hand, the overhead would be also huge, due to the formation of a high number of loops, and therefore duplicate events.

The overhead is shown in the bottom chart of Figure 3. The various approaches are compared against an additional "ideal" curve whose value is computed by routing events along the shortest routes between the publisher and each receiver, derived using Dijkstra's algorithm. This is a very

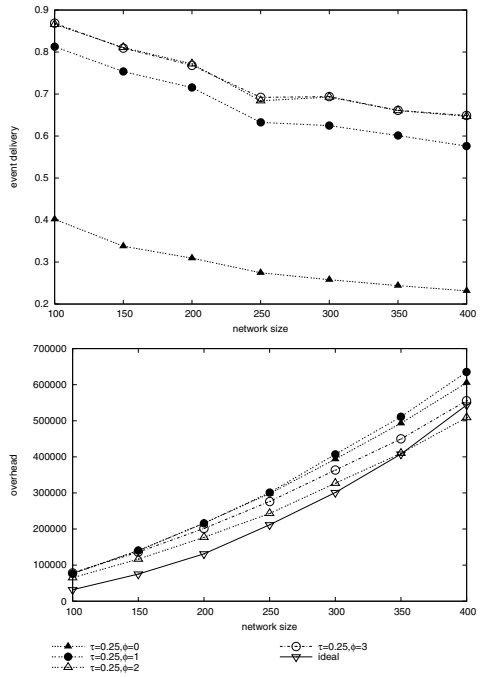

Figure 3. Event delivery and overhead in a static network.

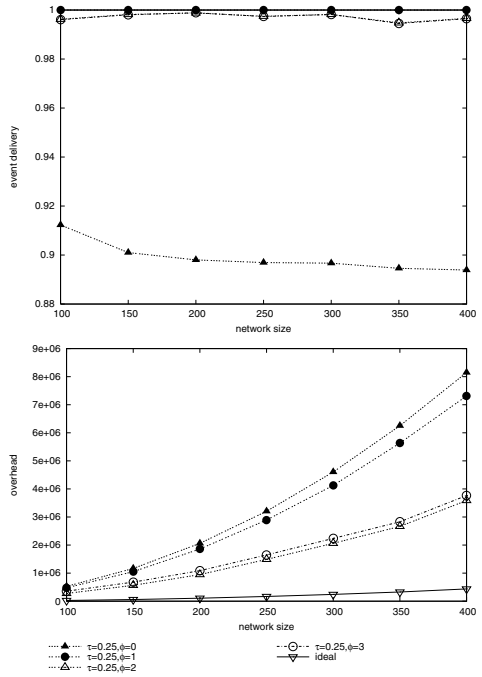

Figure 4. Effect of a higher fanout ($f = 2$).

conservative measure because events are unlikely to follow the shortest path and, more important, the curve does not take into account the messages required to find out, set up and maintain the corresponding routes. Effectively, this ideal curve provides a lower bound for our approach. The upper bound is instead provided by flooding ($\tau = 1$), which is not plotted because it generates an extremely high number of messages (e.g., 11,958,600 at $N = 300$). Therefore, Figure 3 shows that our overhead is extremely far from the upper bound, and quite close to the ideal lower bound[3]. Indeed, more deterministic information ($\phi > 1$) enables savings up to 20% w.r.t. pure probabilistic and $\phi = 1$. Moreover, increasing ϕ further does not provide advantages: $\phi = 2$ generates less messages than $\phi = 3$, due to the increased number of routing loops in the latter. We also verified that higher values of τ quickly bring the system to 100% delivery. This is clearly true for flooding. Also, $\tau = 0.5$ already brings all the curves to full delivery except for $\phi = 0$, which remains at about 96%. Nevertheless, in this latter case the overhead is an order of magnitude higher (around 4 million messages instead of 400,000), although the relative performance among the curves with $\tau = 0.5$ is unchanged, with $\phi > 1$ providing smaller overhead.

Looking at Figure 3, one could notice how delivery drops as the scale increases. Nevertheless, it is worth recalling that in the charts above we assume that each dispatcher added to the system is also a publisher emitting 5 events per second, and that the fraction of receivers for each event is always 10% of the dispatchers in the system. Instead, both the dispatcher's degree l and the event propagation threshold τ remain constant: the fanout f (i.e., the number of links along which events are forwarded) therefore remains constant as well. As a consequence, while the number of dispatchers and receivers increases the ability of the system to spread messages decreases. In a real deployment setting, the increase in scale should be compensated by increasing f, i.e., by intervening either on τ or l. In Figure 4 we show the effect of increasing the degree to $l = 9$ while retaining $\tau = 0.25$, yielding $f = 2$. The ability to spread messages over more links boosts delivery which becomes close to (and for $\phi = 1$ exactly) 100%, except for the purely probabilistic routing that is stuck at around 90%. The bigger fanout augments the likelihood of routing loops, which explains why $\phi > 1$ does not reach 100%. On the other hand, the overhead is also largely increased and reaches the same order of magnitude of the configuration with $l = 5$ and $\tau = 0.5$. This is not surprising, since the product $\tau \cdot l$, which defines the number f of links available for routing and therefore ultimately constrains the effectiveness of routing, is roughly the same in both scenarios.

The impact of the event propagation threshold can be appreciated in Figure 5, where we plotted event delivery and overhead against increasing values of τ. We chose a graph with $l = 11$, to increase the number of links available and therefore increase the number of meaningful values of τ. In this scenario, a small value of $\tau = 0.1$ (i.e., routing on

[3]Interestingly, $\phi = 2$ generates less traffic than the ideal curve. This behavior can be understood by looking at the upper chart in Figure 3: $\phi = 2$ yields a rather low event delivery while the ideal approach delivers all the events.

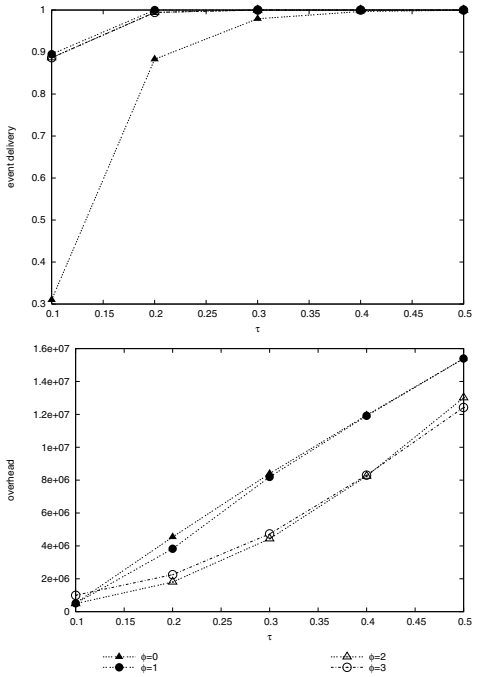

Figure 5. Impact of the event propagation threshold τ.

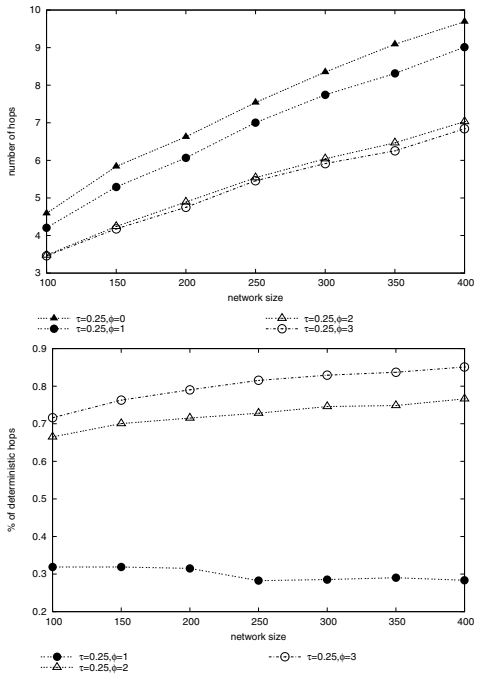

Figure 6. Average length of event routes and percentage of deterministic hops.

1-2 links) is not sufficient to guarantee a decent delivery in a purely probabilistic approach, while the use of some deterministic information already yields a delivery of about 90%. This fact has a huge impact on overhead. A delivery rate of 100% can be achieved with 1-hop information $\phi = 1$ using $\tau = 0.2$. To match the same performance, a purely probabilistic approach must raise the propagation threshold to $\tau = 0.5$, which leads to an overhead that is almost 4 times higher. Also, an increase in the amount of deterministic information ($\phi = 2$) by keeping $\tau = 0.2$ generates a very small decrease in delivery (which remains over 99%) but enables an additional reduction of the overhead, which becomes 60% less than the case with $\phi = 1$, and a remarkable 90% reduction—an order of magnitude—over the purely probabilistic one.

These considerations confirm that complementing a probabilistic approach with small amounts of deterministic information enables high delivery rates with low overhead.

Route Characteristics. An interesting insight on the behavior of our approach and an additional validation of our previous considerations are provided by the analysis of the routes followed by events. The top chart of Figure 6 reports the average length of these routes[4], in number of hops, using the parameters of Table 1 and $\tau = 0.25$. Flooding yields the shorter routes, due to the high probability of generating loops, but at the cost of huge overhead, as we mentioned earlier. Instead, pure probabilistic routing generates longer routes, immediately followed by the case with $\phi = 1$. This is consistent with the fact that events are largely forwarded at random. In turn, $\phi > 1$ generates routes that are about half as long as the previous ones, due to the ability to "steer" events quicker towards their recipients.

This is confirmed by the bottom chart of Figure 6, which plots the percentage of deterministic hops travelled by events in the various cases. As one would expect, this increases with ϕ, albeit not proportionally. Also, as the system scale increases, the percentage of deterministic hops increases for $\phi > 1$ since more routes become available, but decreases for $\phi = 1$, because the average distance between two dispatchers (and therefore also between subscribers) increases while $\tau \cdot l$—and therefore f—remains constant.

Dynamic Network. We now analyze the performance of our approach in a setting where the network topology undergoes reconfiguration, and show that indeed it tolerates very well high degrees of dynamicity. The simulation parameters and the notion of reconfiguration we use here are similar to what defined in [7, 17].

A topological reconfiguration consists of a link breakage, followed by the appearance of a new link. Graph repair is performed after a time interval (that we arbitrarily set to $0.1s$) modeling the delay necessary to the underlying layers to find the replacement link. When a link breakage occurs, our simulator looks for two nodes with a degree lesser than or equal to l, to maintain the average degree as steady as possible. Likewise, a link is selected for removal only if its endpoints' degree is greater than or equal to l. As in [7, 17], we simulated two scenarios, one where the

[4] An event route is the sequence of hops crossed by a given event until its propagation is halted by a dispatcher that received it twice.

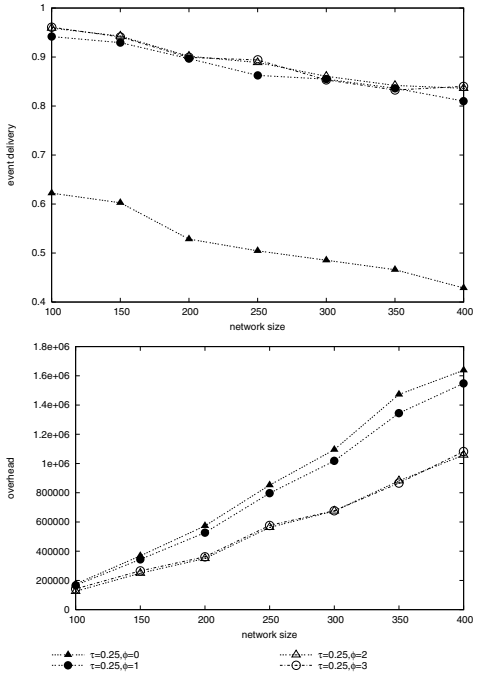

Figure 7. Event delivery and overhead in a dynamic network with reconfigurations every $\rho = 0.03s$.

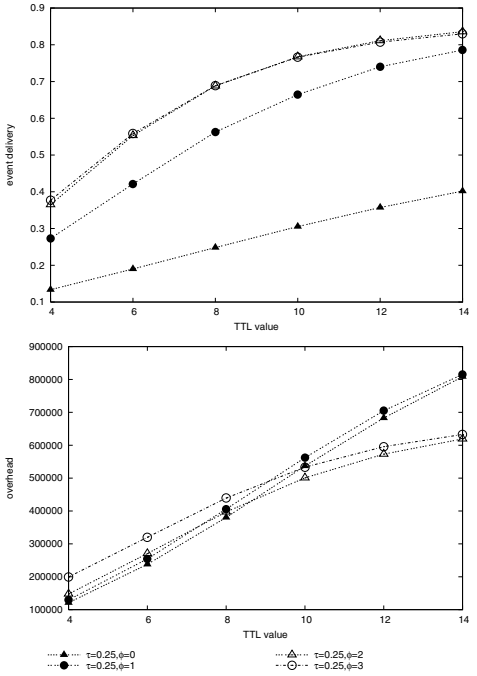

Figure 8. Impact of TTL on a dynamic network.

time between two reconfigurations is $\rho = 0.3s$ and another with $\rho = 0.03s$. In the first scenario, reconfigurations effectively occur in isolation, since a link is always replaced after $0.1s$, i.e., before a new reconfiguration occurs. In the second scenario, instead, the system is undergoing continuous and frequent reconfiguration[5].

The results we show in Figure 7 refer to this second case, with the parameters in Table 1. In the figure, one can easily see that the performance of our algorithms is not reduced by dynamicity, whose perturbation is easily absorbed by the probabilistic component of routing, and by the redundancy of the graph. As a matter of fact, event delivery even improves. This apparently counterintuitive result can be interpreted in the light of well-known results on probabilistic algorithms, which suggest that for "infection" to be effective, random propagation should not occur only locally, but also infect far away nodes. Therefore, dynamicity, by continuously restructuring the connectivity among dispatchers, effectively helps spreading information, by allowing nodes to suddenly becoming in contact with a different set of neighbors, and spread messages from another point. Indeed, the cases that experience the biggest improvements are those that rely most on probabilistic information, i.e., the purely probabilistic and the one with $\phi = 1$. Another contribution to this increase is given by the way we reconfigure the graph, which causes the degree to become $l = 5 \pm 1$. Since $f = \lceil \tau(l-1) \rceil$, the dispatchers who got an additional link end up having $f = 2$ links available, while those who lost a link remain at $f = 1$ as in the static case, by virtue of rounding. The combined effect of the two phenomena is the key to interpreted also the overhead.

Finally, the performance of the milder scenario with $\rho = 0.3s$, whose charts are omitted here, is somewhere in between Figure 3 and 7.

Limiting Propagation. Thus far, we never restrained propagation and assumed a TTL$= \infty$ for the sake of analyzing the behavior of our approach. In practice, however, introducing a TTL enables considerable savings in overhead, as we discussed in Section 3. Figure 8 shows the impact of TTL on a dynamic network with the parameters of Table 1 and $\rho = 0.03s$. Clearly, low values of the TTL may constrain propagation too much, and negatively impact event delivery. For instance, Figure 8 shows that in order to match the event delivery we showed in Figure 7 for $N = 300$ and TTL=∞, we need to set TTL=14. Interestingly, with this value the curves with $\phi > 0$ remain at about the same delivery rate, except for a small reduction with $\phi = 1$, while the purely probabilistic routing gets worse, with a reduction in delivery of about 20%. This is not surprising, given that the deterministic approaches lead to routes that are significantly shorter, as we discussed earlier, and therefore are not significantly affected by the TTL limitation. Moreover, the overhead drops significantly, as we expected. For $\phi > 1$, overhead is reduced of about 8%, while for the others reduction is more significant, around 30%.

In Section 3 we mentioned the possibility of using two

[5]Each reconfiguration involves two dispatchers, the link endpoints. At 300 reconfigurations per second, with a network size of $N = 300$ each dispatcher changes two neighbors per second. Since our simulations are run for over two seconds, at least 4 neighbors out of 5 get replaced.

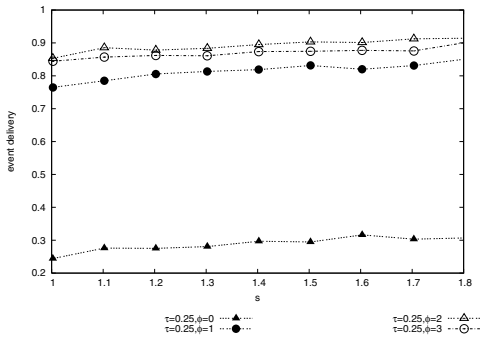

Figure 9. TTL_d =10 and TTL_p =8 on a dynamic network.

different values TTL_d and TTL_p to impose different limits to propagation, with the benefit of enabling a more accurate performance tuning. The impact of such a scheme is shown in Figure 9, where we used TTL_d =10 and TTL_p =8, with the rest of the parameters unchanged w.r.t. Figure 8. Note that, differently from those we showed thus far, this chart represents a single run plotted against (simulated) time. The values we chose for TTL_d and TTL_p enable, for $\phi > 0$, delivery rates a little bit higher than those obtained with the single TTL=14 in Figure 8. However, the tradeoffs in terms of overhead among the various solutions are profoundly different. The overhead of $\phi > 1$ is slightly increased, although justified by the small increase in the delivery rate. On the other hand, the overhead of $\phi = 1$ is greatly improved, dropping from about 820,000 messages to about 640,000 (more than 20% less), while $\phi = 2$ and $\phi = 3$ are at about 680,000 and 730,000, respectively. This configuration makes routing with $\phi = 1$ more appealing than in earlier scenarios, making it a valid alternative to $\phi = 2$. A different choice for TTL_d and TTL_p, e.g., further increasing the gap between the two, would favor routing with $\phi > 1$.

Stability of Event Delivery. Plotting the chart in Figure 9 against time enables us to evidence also another interesting phenomenon whose generality goes beyond the use of TTL, that is, the event delivery is quite stable over time. This is particularly relevant especially if compared against similar results obtained by approaches that rely on a tree, e.g., [17]. There, event delivery has wide and very frequent excursions, ranging from 100% down to 40%. The reason for the remarkable improvement of our approach can be attributed to its use of a graph and its ability to exploit alternative routes thanks to the probabilistic component.

Density of Receivers vs. System Scale. In Figure 3 and 7 we analyzed event delivery and overhead when the scale is increased by adding new publishers and receivers to the system and showed that, despite the resulting heavy load, our approach still yields good performance. As a final dimension of evaluation, it is worth investigating here how the properties of our approach are affected by the density of receivers in the network. We setup a scenario where the

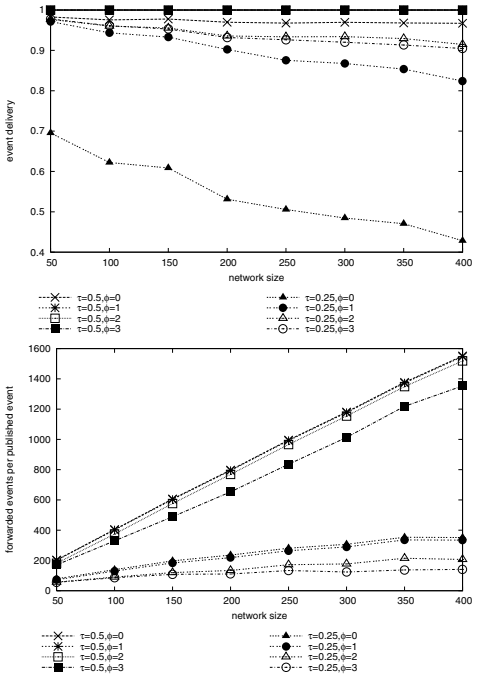

Figure 10. Event delivery and overhead in a dynamic network with a fixed number of receivers.

number of dispatchers (still all publishing at 5 event/s) is increased while the number of receivers per event (10 in our case) remains constant. The density of receivers therefore decreases linearly, from 20% for $N = 50$ down to 2.5% for $N = 400$. This scenario elicits new issues w.r.t. the one we examined previously. With a constant density of receivers and a growing scale, we need to increment the number of forwarded events to reach a larger set of receivers, and therefore increasing the fanout is a viable solution. Instead, here the number of receivers does not change: therefore, what we are assessing is how "selective" is our routing towards the receivers.

The simulation parameters are those of Figure 7. Results are in Figure 10, where we plot both the event delivery and the number of forwarded events divided by the number of published events. This latter metric characterizes the effort, in terms of forwarded events, required to deliver a single event to a fixed set of receivers in a growing network[6].

Figure 10 shows that a high fanout ($\tau = 0.5$) always achieves high delivery but basically saturates the network by reaching almost every dispatcher, thus increasing the traffic linearly with scale. However, even in this case deterministic information ($\phi = 3$) achieves some savings, as it enables a more selective routing. Instead, a lower fanout ($\tau = 0.25$) yields a very different behaviour. Event delivery is a lower than with $\tau = 0.5$, since the probability to reach a receiver depends on the product of the probabilities to se-

[6]We do not consider the traffic generated by (un)subscriptions since in our heavy publishing scenario it is negligible w.r.t. the number of events.

τ, ϕ	0	1	2	3
0.25	2,480,085	2,805,589	1,858,628	5,890,292
0.5	6,383,210	6,944,038	6,516,774	10,246,982

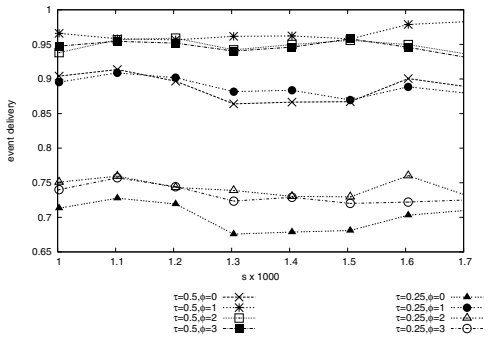

Figure 11. Event delivery in a MANET.

lect the right neighbour at each hop—which in turn depends on fanout, $f = 1$ in this case—and therefore decreases as the routes connecting publishers to receivers become longer. However, while this negative effect is evident for pure probabilistic routing, it is almost entirely compensated in terms of delivery by the deterministic information available when $\phi > 0$, which reduces the reliance on random forwarding by "steering" forwarded events more efficiently through the network. This increased efficiency is mirrored in the overhead chart, where the number of forwarded events per published event still increases, due to the longer routes towards receivers, but this time remains well below a linear trend.

A Mobility Scenario. To validate our approach in a less controlled simulation environment, we evaluate it in a mobile scenario generated with ANSim [13], a scenario generator for MANETs. The sequence of topological reconfigurations caused by mobility is recorded and fed into our simulator, removing the limitation on the degree. The mobility model we chose is Random Direction model [4], where nodes move along a direction until they reach one of the edges. Since in this scenario the degree is no longer constant, we optimize our algorithm by forwarding an event to a "leaf" dispatcher (i.e., one connected to the rest of the graph by a single link) only if it is a subscriber.

The chart in Figure 11 shows the event delivery and overhead for $N = 200$ nodes confined in a square area with a side of 2 Km, with a wireless communication range of 250 m, each moving at a velocity of 2 m/s. These values, which are common in the research field, keep our graph connected. Moreover, after having analyzed the performance with TTL=∞ we set TTL_d =12 and TTL_p =8. To provide a more general validation, we plotted the results with $\tau = 0.5$ and $\tau = 0.25$.

In this setting, pure probabilistic routing with $\tau = 0.5$ provides a reasonable event delivery of a little less than 90%. This is comparable to the one obtained with $\tau = 0.25$ and $\phi = 1$, which however has a much smaller overhead—about 66% less. All the other variants with $\tau = 0.25$ perform poorly in terms of delivery: the number of links used for forwarding is too low, and the dynamicity is too high for deterministic information at more than one hop to provide reliable support for routing. In this case, the variant with $\phi = 1$ bears the double advantage of being allowed to exploit deterministic information regardless of τ, and of exploiting more reliable routing information. On the other hand, all the other variants with $\tau = 0.5$ improve delivery over their purely probabilistic variant, with $\phi = 3$ being the most expensive in terms of overhead.

5. Related Work

The majority of content-based publish-subscribe systems are built upon a tree-shaped overlay, with some of them addressing the easier problem of supporting client mobility. Recent work by the authors [7, 8, 17] deals instead with reconfigurations affecting the dispatchers in the tree. Other recent approaches [6, 18] exploit a graph-based topology upon which a set of dispatching trees are superimposed. However, these papers do not provide any detail about if and how dynamicity is taken into account, and at which cost. To our knowledge, only CBM [20] and STEAM [16] explicitly address content-based publish-subscribe in wireless networks, although they focus on combining it with location information. CBM exploits position-based routing to constrain the propagation of matching events along a specified direction, while STEAM limits the event propagation to a proximity area, inside which events are broadcast and locally matched against subscriptions. Instead, our work focuses on the provision of a general-purpose content-based publish-subscribe. In the related area of content dissemination over MANETs, autonomous gossiping [9] provides the ability to push content towards potential receivers, by exploiting epidemic dissemination of data and user profiles.

In the context of MANET routing, none of the approaces is directly reusable because of the peculiar challenges posed by content-based routing, but some rely on similar ideas. In the Zone Routing Protocol (ZRP) [12] for unicast, a node proactively maintains routing information about its neighborhood, and reactively requests information about destinations outside of it. In our approach, long distance propagation is instead achieved in a probabilistic way. On the other hand, route driven gossip [15] exploits epidemic algorithms to maintain and disseminate a localized view of the system, enhanced with routing information.

Finally, we believe our work can be adapted to content dissemination in sensor networks. Current approaches (e.g., [3, 14]) spread nodes' interests across the whole network to create a reverse path from a publisher to receivers, but no details are provided about how to deal with a dynamic network, which indeed is relevant in many applications.

6. Conclusions and Research Opportunities

The solution we presented and its evaluation confirm that we meet the goals stated in Section 3. The combination of deterministic information and probabilistic forwarding enables us to rejoin the conflicting goals of providing high event delivery and low overhead, without sacrificing scalability, and for scenarios with very high degrees of dynamicity. The choice of a graph not only gives us a more resilient communication infrastructure but also one that can be more easily maintained. Finally, the resulting algorithm is very simple, easing understanding and simplifying deployment. We contend that the very concept of content-based routing concurs to these significant results, because a single event may match multiple subscriptions and therefore may be routed by different deterministic information.

Nevertheless, our ultimate goal is to devise a mechanism that adapts to network conditions, for which the results in this paper provide the foundation. When a perturbation is observed in the network, the parameters of our solution are automatically adjusted to guarantee a good tradeoff between delivery and overhead, e.g., by reducing the amount of deterministic information. When the network is instead stable, deterministic information is propagated to a greater extent, with mechanisms to disseminate it efficiently (e.g., to avoid loops) that may even end up relying on a backbone tree and subscription forwarding. We are currently studying mechanisms to achieve this goal.

Another subject of ongoing research is the optimization or customization of the solution presented here to specific application domains. For instance, use in MANET environments would probably benefit from an alternative scheme (reminiscent of [11]) where events and subscriptions rely on the inexpensive one-hop broadcast and where probability does not determine the likelihood to route along a link, rather the probability of a node to rebroadcast a message. Similar opportunities may exist for other environments, e.g., content dissemination in sensor networks.

Finally, our immediate goal is the implementation of our solution and its validation in real applications.

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

References

[1] K. P. Birman et al. Bimodal multicast. *ACM Trans. on Computer Systems*, 17(2):41–88, 1999.

[2] D. Blough et al. The k-neigh protocol for symmetric topology control in ad hoc networks. In *MobiHoc'03: Proc. of the 4^{th} ACM Int. Symp. on Mobile ad hoc networking and computing*, pages 141–152, 2003.

[3] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proc. of the 1^{st} Int. Wkshp. on Wireless Sensor Networks and Applications*, pages 22–31, 2002.

[4] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communications & Mobile Computing (WCMC)*, 2(5):483–502, 2002.

[5] A. Carzaniga, D. Rosenblum, and A. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. *ACM Trans. on Computer Systems*, 19(3):332–383, Aug. 2001.

[6] A. Carzaniga, M. Rutherford, and A. Wolf. A routing scheme for content-based networking. In *Proc. of INFOCOM*, Mar. 2004.

[7] P. Costa, M. Migliavacca, G. Picco, and G. Cugola. Epidemic Algorithms for Reliable Content-Based Publish-Subscribe: An Evaluation. In *Proc. of the 24^{th} Int. Conf. on Distributed Computing Systems (ICDCS04)*, pages 552–561, Mar. 2004.

[8] G. Cugola, D. Frey, A. L. Murphy, and G. P. Picco. Minimizing the Reconfiguration Overhead in Content-Based Publish-Subscribe. In *Proc. of the 19^{th} ACM Symp. on Applied Computing (SAC04)*, pages 1134–1140, Mar. 2004.

[9] A. Datta, S. Quarteroni, and K. Aberer. Autonomous Gossiping: A self-organizing epidemic algorithm for selective information dissemination in mobile ad-hoc networks. In *Proc. of , Int. Conf. on Semantics of a Networked World*, June 2004.

[10] A. Demers et al. Epidemic algorithms for replicated database maintenance. *Operating Systems Review*, 22(1):8–32, 1988.

[11] Z. Haas, J. Halpern, and L. Li. Gossip-Based Ad Hoc Routing. In *Proc. of INFOCOM*, pages 23–27, June 2002.

[12] Z. Haas and M. Pearlman. The Zone Routing Protocol (ZRP) for Ad Hoc Networks. IETF draft, June 1999.

[13] H. Hellbrück. ANSim Web page. `www.i-u.de/schools/hellbrueck/ansim`.

[14] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proc. of MobiCom*, 2000.

[15] J. Luo, P. Eugster, and J. Hubaux. Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks. In *Proc. of INFOCOM'03*, April 2003.

[16] R. Meier and V. Cahill. STEAM: Event-Based Middleware for Wireless Ad Hoc Networks. In *Proc. of the 1^{st} Int. Wkshp. on Distributed Event-Based Systems*, July 2002.

[17] G. P. Picco, G. Cugola, and A. L. Murphy. Efficient Content-Based Event Dispatching in the Presence of Topological Reconfigurations. In *Proc. of the 23^{rd} Int. Conf. on Distributed Computing System s (ICDCS03)*, pages 234–243, 2003.

[18] P. Pietzuch and J. Bacon. Hermes: A Distributed Event-Based Middleware Architecture. In *Proc. of the 1^{st} Int. Wkshp on Distributed Event-Based Systems*, July 2002.

[19] A. Varga. OMNeT++ Web page. `www.omnetpp.org`.

[20] H. Zhou and S. Singh. Content based multicast (CBM) in ad hoc networks. In *Proc. of the Workshop on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, Aug. 2000.