Distribute, Store and Retrieve Management Policies in Wireless Ad-Hoc Networks using the Content Delivery Publish/Subscribe Paradigm

Vasilis Sourlas*, Paris Flegkas*, Georgios S. Paschos†, and Leandros Tassiulas*
*Department of Computer & Communication Engineering
University of Thessaly, Greece.
†CERTH-ITI, Greece.
Email: vsourlas, pflegkas, gpasxos, leandros @uth.gr

***Abstract*—Policy management is a management paradigm that has been extensively studied for the case of fixed networks but limited work can be found for migrating it to mobile environments. It enables dynamic adaptation of the network behavior to current conditions based on high-level business and operational objectives. Publish/subscribe has become an important architectural style for designing distributed systems and especially for mobile environments due to the loose coupling of the components involved namely the publishers and the subscribers. In this paper, we present a policy-based management system for wireless ad-hoc networks using the publish/subscribe paradigm for distributing policies to the managed nodes ensuring this way that all nodes will receive the related defined policies, in an asynchronous and loosely coupled manner, achieving the desired network-wide behavior. Moreover, we enhance the publish/subscribe system with a novel request/response mechanism for tackling the problem of how newly joined nodes will retrieve previously introduced policies. Finally, we describe our initial design and implementation of the proposed mechanism, evaluate it through simulation and testbed experiments and give pointers to our future work.**

I. Introduction

Management of Wireless ad-hoc networks has recently attracted a lot of attention due to the proliferation of mobile/pervasive devices and their ability and need to communicate. The nature and their inherent characteristics such as topology changes, mobility of nodes and limited terminal capabilities of such networks pose new challenges and requirements for their management. Traditional management approaches in fixed networks based on centralized approaches (SNMP - Simple Network Management Protocol) cannot be applied in ad-hoc networks. The highly dynamic environment of wireless ad-hoc networks can benefit from the self-management capabilities provided by Policy-based Management (PBM). PBM has been applied to fixed networks using also a centralized architecture where a Policy Decision Point (PDP) resides in a management server and is responsible for retrieving policy rules. Policy rules are defined by the operator in the Policy Management Tool (PMT), stored in the Policy Repository (PR) and translating them to management operations on the network elements i.e. Policy Enforcement Points (PEPs) [1]. In order to apply PBM in ad-hoc networks, a hierarchically distributed management approach has to be followed as described by related work [2].

The publish/subscribe paradigm has become an important architectural style for designing distributed systems. Applications that exploit a publish/subscribe communication paradigm are organized as a collection of autonomous components (clients), which interact by publishing events and by subscribing to the classes of events they are interested in. The event dispatcher (broker) is responsible for collecting subscriptions and forwarding events to subscribers. In publish/subscribe systems the selection of a message is determined entirely by the client, which uses expressions (filters) that allow sophisticated matching on the event content.

Network management has always been one of the most popular applications using an event-based paradigm where managers subscribe to events published by the managed elements. In this paper, we present a policy-based management architecture for mobile ad-hoc networks based on the publish/subscribe communication paradigm for distributing policy rules from the points they are introduced to all the responsible components for their enforcement. We show through simulations and testbed experiments that the dynamic environment of wireless ad-hoc networks can benefit from such a communication paradigm. Policy distribution is a critical task in managing wireless ad-hoc networks since there is a need to assure that all nodes comply with policies introduced by the operator leading the system to stability. Moreover, we enhance the publish/subscribe paradigm with a request/response mechanism so that nodes that join the network can retrieve previously introduced policies. This is important since nodes might not have joined the network when the operator defines a policy and current publish/subscribe systems do not support retrieval of previous or missing events.

The rest of the paper is organized as followed. Section II describes related work in the area of management of wireless ad-hoc networks and publish/subscribe systems. Section III presents our proposed policy management architecture based on the publish/subscribe paradigm for distributing policies. Section IV describes the enhanced request/response mechanism while section V analyzes our systems' design. Section

VI evaluates our architecture through simulation experiments, while section VII is devoted to performance evaluation via testbed experiments. Finally, section VIII concludes the paper and gives pointers to our future work.

II. Related Work

Limited work has been done in the area of management of ad-hoc networks where most approaches follow a hierarchical organization model. The first attempt was made by [3], proposing an Ad-hoc Network Management Protocol (ANMP) based on hierarchical clustering of nodes in a three level architecture. The "Guerilla" architecture [4] adopts an agent-based two-tier distributed approach where at the higher level "nomadic managers" make decisions and mobile agents exploit a utility function to decide their migration and probe deployment to fulfill management objectives. In [5] a PBM system is proposed where policy agents are deployed and manage the network through a two tier hierarchical architecture with the use of several proprietary protocols (YAP, AMPS, DRCP/DCDP). Another PBM approach is presented in [6] for QoS provisioning in Mobile Ad-hoc Networks (MANETs). They propose a k-hop clustering scheme and extensions to COPS for policy provisioning (COPS-PR) protocol. Finally in [2], a hybrid hierarchical and distributed approach is adopted, presenting a system architecture that uses the policy-based management paradigm together with context awareness for managing MANETs. While all the above approaches were designed for solving relevant problems in the area of ad-hoc network management, none of them has dealt with the critical issue of policy distribution in them. The use of a publish/subscribe architecture for distributing policies in wireless ad-hoc networks has not been considered in the past and comprises one of the innovative aspects of our approach, shortly introduced in [7].

There are several research efforts concerned with the development of an event notification service including IBM's Gryphon [8], Siena [9], Elvin [10], TIBCO [11] and Jedi [12] which implement the publish/subscribe architecture. Some of them are distributed while others are centralized. The majority of them address scalability and ease of implementation by realizing the broker tree as an overlay network. The topology is assumed to be stable, a requirement that clashes with the reality of dynamic scenarios like wireless ad-hoc networks. REDS [13] on the other hand defines a protocol to organize the nodes of a mobile ad-hoc network in a single, self-repairing tree that efficiently supports content-based routing.

III. System Architecture

We adopt a 2-tier hierarchical and distributed approach for our management organization model, where nodes of the network are categorized based on their management functionality. The simplest nodes are the ones having the PEP (Policy Enforcement Points) functionality which are usually the most lightweight nodes in terms of capabilities, such as sensors, offering only a management interface to nodes with enhanced management intelligence (PDP - Policy Decision Point) denoted as cluster managers in [4]. These nodes have the ability to retrieve and translate policies to management operations on nodes they are responsible for managing, or enforce them on the co-located PEP if the node does not have a cluster manager role. Our architecture does not follow a strict management hierarchy in the sense that although a node may not have a cluster manager role, but is capable of interpreting policies, it retrieves the related policy enforcing it on the co-located managed objects. The idea is to use a management by delegation principle reducing this way the remote operations and the possibility of failing to apply a policy due to the frequent disconnections that might occur in a wireless environment. Depending on the derived degree of distribution policy repositories (PR) can be hosted in several nodes in the network with varying capabilities and behavior (e.g. mobility) avoiding this way the retrieval from a single point in the network. Finally, in the highest level of our hierarchy are the nodes with the PMT (Policy Management Tool) functionality, enabling the administrator of the network to introduce policies that should be stored in the repositories and enforced by all the nodes with the PDP functionality in the network. It is possible that a node in the network might have a policy management functionality that may not be active, for example a policy repository that is decided not to be used to store policies due to the node's behavior and characteristics such as the mobility pattern, remaining battery etc, while PEP functionality is present to all nodes that participate in the network.

In order to incorporate a robust and efficient policy distribution mechanism in our architecture all the policy management components described above communicate using a publish/subscribe paradigm in a loosely-coupled and asynchronous manner. In order to achieve that, several nodes of the network take the role of the event broker i.e. a publish/subscribe (pub/sub) router as it is shown in figure 1 forming an overlay publish/subscribe network handling the distribution of policies from the point they are introduced to the repositories and the PEPs responsible to enforce them. All the policy components present in our system are the clients of our pub/sub network namely the publishers and the subscribers. Due to the nature of ad-hoc networks clients and pub/sub routers may be physically co-located in the same node. Our system architecture is depicted in figure 1 where all nodes with a PDP functionality subscribe to the pub/sub router they are connected to, waiting for a relevant policy to be introduced/published by the administrator (PMT).

In our work we assume the following well adopted representation of a policy as defined by IETF [14].

Roles [TimePeriod] if $\{$conditions$\}$ then $\{$actions$\}$

The concept of roles is critical in our architecture since it defines the scope of the policy i.e. to which nodes it applies and comprises the subscription filter (or part of it) in the pub/sub network. Nodes with only the PEP functionality supply their role to their cluster manager node, which in turn creates an aggregated subscription with all the roles of the

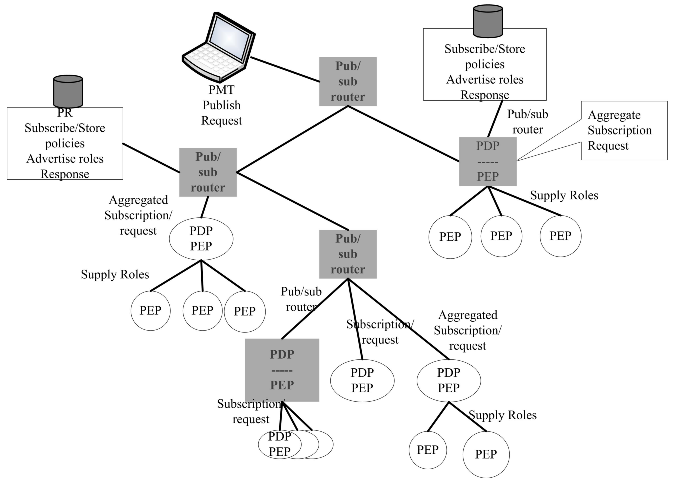


Fig. 1. System Architecture.

managed nodes in order to receive all the relevant policies. Nodes with the PDP functionality present and not in a cluster manager role subscribe only to policies related to their own operation.

The operator of the network defines the policies in a node with the PMT functionality, which publishes the policy to the pub/sub router it is attached to. The pub/sub network is responsible for delivering this policy to all the relevant components based on the Roles attribute and the subscription tables of the pub/sub routers, that were created by the client subscriptions. The benefits from using a publish/subscribe system for the communication of policies are evident since it enables an asynchronous and loosely coupled communication between the publishers and subscribers. With the recent advances in the pub/sub systems and their enhancements for mobile environments handling both the mobility or disconnections of clients (i.e. both the manager and nodes with PDP functionality can move or disconnect) as well as the event brokers (i.e. nodes with the pub/sub router functionality), the pub/sub overlay network takes all the reconfigurations actions needed to assure a minimum amount of lost messages.

Finally, nodes with the Policy Repository component also subscribe to receive policies and are responsible to store them for later retrieval by new subscribers or for validation and conflict detection checking before a new policy is introduced. Both replication and distribution of the PR component is supported and is realized by subscribing to roles of policies that every node hosting a PR desires to store. For example, if full replication is decided, every PR node subscribes to all roles of policies supported and when a manager defines and publishes a policy, this will be received and stored by all PRs present in the network. In the case of a distributed policy repository, PR nodes subscribe to different roles, possibly depending on the PEPs present in its cluster (subtree). Finally, a combination of replication and distribution can easily be supported to the power of expressiveness of the subscription filters provided by publish/subscribe systems.

In order to support this kind of operation, we enhance the publish/subscribe paradigm with a request/response mechanism so that nodes that have just joined the network to be able to request all the previously defined policies, relevant to their role, from the PRs. Moreover manager nodes that do not have a local repository or view of all the policies present in the system (e.g. due to the presence of other manager defining policies from different nodes) are able to check for any conflict with previously installed policies using the proposed mechanism.

An advertisement mechanism is also proposed (used by the PR nodes) so that requests for policies coming from clients are routed to the nodes hosting a PR. When a manager desires to publish a new policy, a request is first initiated for all the policies that could lead to a potential conflict and responses with all the relevant policies already stored in the PRs are sent. When the PMT receives all the responses with these policies, its conflict detection mechanism checks if there is any conflict and if not, the PMT publishes the policy that will be received by all the relevant nodes (subscribers) in the network.

The proposed mechanisms are crucial for enabling storing in pub/sub systems and can also be used in the case of wireless networks, since clients that are disconnected from the network due to mobility or bad connectivity can request and retrieve content that has been published the time that were disconnected. In the section below, we describe in detail the proposed enhancements to the publish/subscribe framework.

IV. Enhancing Publish/Subscribe With Request/Response

In our pub/sub system we use the subscription forwarding routing strategy [9] where the routing paths for policies are set by subscriptions, which are propagated throughout the network so as to form a tree that connects the subscribers to all the brokers in the network. This scheme is optimized to avoid forwarding the same subscriptions in the same direction by exploiting “coverage” relations among filters. This means that a subscription is forwarded to a neighboring broker only if it is not being covered by a subscription already forwarded to the same neighbor. Particularly we say that a subscription filter f_1 covers another subscription filter f_2, denoted by $f_1 \geq f_2$, iff any event matching f_2 also matches f_1 [15]. When a client (PMT) publishes an event (policy) that matches a subscription, the event is routed towards the subscriber following the reverse path put in place by the subscription.

The subscription forwarding routing strategy and all its known implementations ([9] and [13]) does not provide the capability of retrieving a published event (policy) at a time later than the time of its publication. In order to achieve this we enhance the pub/sub system with advertisement messages. Each broker maintains a set ST “Subscription Table” containing the identifiers of the brokers to which the broker is connected and the subscriptions that those neighbors had sent to the broker. Particularly each neighbor n_i in ST has an associated set $filters\,(n_i)$ containing the subscriptions sent by n_i through subscription messages. In our case we add to each broker another set AT ”Advertisement Table” which contains the identifiers of the brokers to which the broker is connected and the advertisements that those neighbors had sent to the broker. Similarly each neighbor n_j in AT has an associated set $advertisements\,(n_j)$ containing the advertisements sent by n_j through advertisement messages.

Advertisement messages are messages sent by the PR nodes containing the Roles to which the stored policies are referred to and are treated similarly to subscription messages so as to form a tree that connects the PRs to all the brokers in the network. Coverage also occurs with advertisements and as in subscriptions is used to avoid forwarding the same advertisements to the same direction. We also add to the system two additional types of messages besides Subscribe(), Publish() and Advertise(). We call those two new messages Request() and Response(). As shown in figure 2, when a client node S_2 (PDP) subscribes to a broker node n_a (step 3), sending a $Subscribe\,(f_a)$ message also sends a $Request\,(f_a, S_2)$ message(if she/he wants to retrieve previously introduced messages). The Request() message is similar to the Subscribe() message but apart from the filter it also carries the sequence of the nodes that passes from. Node n_a upon receiving the Request() message checks in AT_a for advertisements matching filter f_a. In other words checks if the role of events published in the past match the role of the new subscriber. If such an advertisement occurs (adv_a) the broker forwards the Request() message to the identifier for which the adv_a was in the table otherwise the Request() message is dropped. Particularly the syntax of the message looks like $Request\,(f_a, n_a \rightarrow S_2)$ and is sent to broker n_b (n_b the identifier for which the adv_a was in AT_a or in other words the broker from which a matching advertisement was sent in the past).

When a broker n_b receives a Request() checks in AT_b for matching advertisements and according to the findings it forward the Request() appending its identifier ($Request\,(f_a, n_b \rightarrow n_a \rightarrow S_2)$). In this case there will be an entrance in a brokers' AT (AT_c) which points in a PR node. Upon receiving the Request() the PR node checks for the matching event and sends back a Response() message (step 4) for each matching event/policy. The Response() message is similar to the Publish() one but apart from the event it also carries the sequence of nodes carried by the Request(). For instance the syntax in our example for the Request() sent by broker n_c would be $Response\,(msg, n_b \rightarrow n_a \rightarrow S_2)$ where msg the matching event found in the client PR_1. When a broker receives a Response() message pops off its identifier from the sequence and forwards it to the first broker of the remaining sequence. In the end client S_2 will receive the event. With the above procedure every new subscriber and only that one will receive every old event (policy) matching its role.

If the network has more than one replica repositories, there will be cases where a request message will meet more than one entries in a brokers' AT. In all those cases the corresponding broker forwards the request to only one of them. The selection of the broker to whom the request is to be forwarded is random in this paper but we can supply the brokers with more sufficient selection mechanisms (based on traffic statistics or topological knowledge).

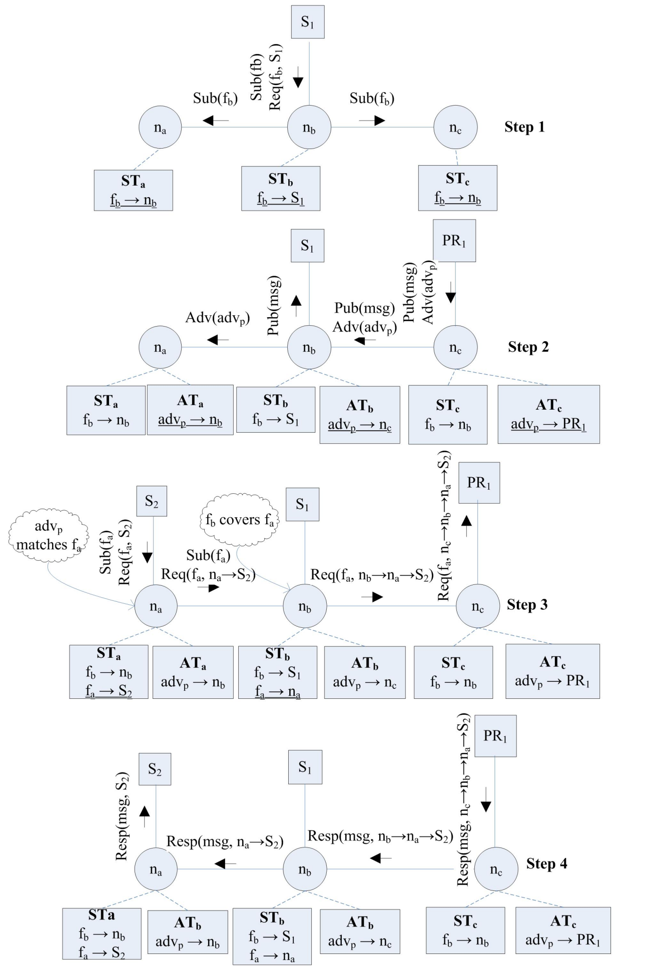


Fig. 2. Enhanced publish/subscribe paradigm. Step 3 and step 4 illustrates the novel introduced Request() and Response() messages.

V. System Design

For our system we use REDS [13] since it was designed to tolerate dynamic reconfigurations of the dispatching infrastructure.

The reconfiguration problem can be broken down into the problem of repairing the overlay dispatching network, to retain connectivity among brokers, and the problem of reconciling the subscription and advertisement information held by each broker and used for routing messages, to keep it consistent with the topological changes, without interfering with the normal processing of subscriptions, and advertisements.

There are two types of overlay network failures: link and broker. Link failure creates two trees with exactly the same nodes as before the link break. Repair, therefore, involves adding a link with endpoints in each of the two trees. Failure of a node with n neighbors results in n partitions, which require the addition of $n-1$ new links, so a broker loss can be addressed as a combination of several link losses. The main challenge to address in repair of the overlay network is the selection of these links to repair the tree. REDS inspired by MAODV (Multicast Ad-hoc On-Demand Distance Vector) to create COMAN [16] (COntent based routing for Mobile Ad-hoc Networks), a protocol to organize the nodes of a wireless ad-hoc network in a tree-shaped network able to a) self repair to tolerate the frequent topological reconfigurations typical of ad-hoc networks and b) achieve this goal through repair strategies that minimize the changes that may impact the content based routing layer exploiting the tree.

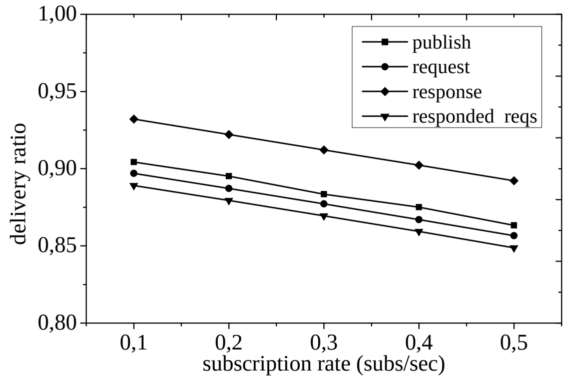
After ensuring the maintenance of the overlay tree, the next step is maintaining the subscription and advertisement tables to allow messages to continue to reach the subscribers. The very first solution (Strawman) is that when a link disappears, a broker behaves as if it received unsubscription and unadvertisement messages from the former neighbor, updating its subscription and advertisement table and propagating the unsubscription and unadvertisement message if necessary. When the new link is added, its endpoints send subscriptions and advertisements to one another for all entries in their subscription and advertisement table, allowing events to flow across the new link. This approach successfully reconfigures the subscription and advertisement tables but cause unnecessary overhead. REDS uses a reversal technique called Deferred Unsubscription to delay the unsubscription and unadvertisement process until the subscription and advertisement process is complete, reducing in this way the overhead of reconfiguration.

VI. Performance Evaluation

In this section we evaluate the proposed mechanism using a discrete event simulator. $N = 15$ brokers are organized in a balanced binary tree and clients are dynamically generated on each broker according to a birth process with birth rate λ. We are looking at the following interesting metrics.

- The *Publish delivery ratio*, the ratio of the published messages actually delivered to a client to the overall number of published messages (policies) matching at least one subscription at that client.
- The *Request delivery ratio*, the ratio of the request messages actually delivered to a repository to the overall number of request messages sent to the network by the clients.
- The *Response delivery ratio*, the ratio of the response messages actually delivered to a client to the overall number of response messages sent by the repository to that client.
- The *Responded requests delivery ratio*, the ratio of the successful request-response transactions made in the network to the overall request messages sent to the network.
- The *Minimum hop distance*, is measured for each successful response and corresponds to the minimum number of hops between the responding repository and the broker where the client making the request is attached to. This metric is indicative of the delay of responses as a function of hops in the network.

The above metrics are random variables and we estimate their mean by simulating thousands of observations. We set four experiments, one varying the dynamics of the clients



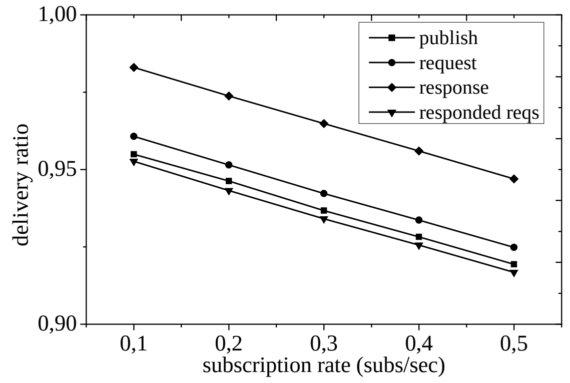


Fig. 3. Delivery ratio (publish, request, response, responded requests) vs subscription rate when the system has 1 (upper graph) and 2 (lower graph) (replicas) repositories respectively.

(λ), one varying the interval time between link failures, one varying the recovery time from a link failure and one varying the number of brokers in the network. We execute those experiments twice, once with one repository in the network and once with two replica repositories. For this first approach we supposed that each time only one link is broken, leaving for our future work the cases where more than one links are "down" each time.

Figures 3, 4, 5 and 6 depict the delivery ratios for each one of the four experiments when the system is equipped with one and two PRs (policy repositories) respectively. The charts show that our system is only marginally influenced by the traffic; an increase in the subscription rate brings less than $5\% - 10\%$ reduction in every delivery metric for both scenarios. We can also identify the exponential impact to the delivery ratio of the link break interval time (in other words the link failure rate) and also verify that the system which is equipped with two repositories offers at least 10% better delivery ratio than the system with the one repository.

Figure 5 also shows that delivery ratios are linear to the time that the overlay tree is broken in two pieces, while figure 6 shows that with only one link failure at a time the delivery ratio increases with respect to the number of brokers in the network. This occurs because the ratio of the brokers, (consequently the clients) that are connected to the part of the tree which includes

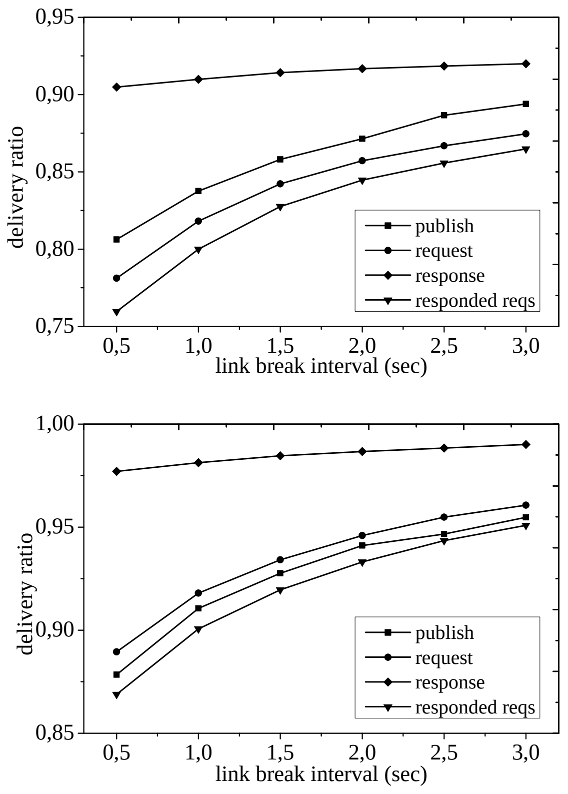


Fig. 4. Delivery ratio (publish, request, response, responded requests) vs link failure interval time when the system has 1 (upper graph) and 2 (lower graph) (replicas) repositories respectively.

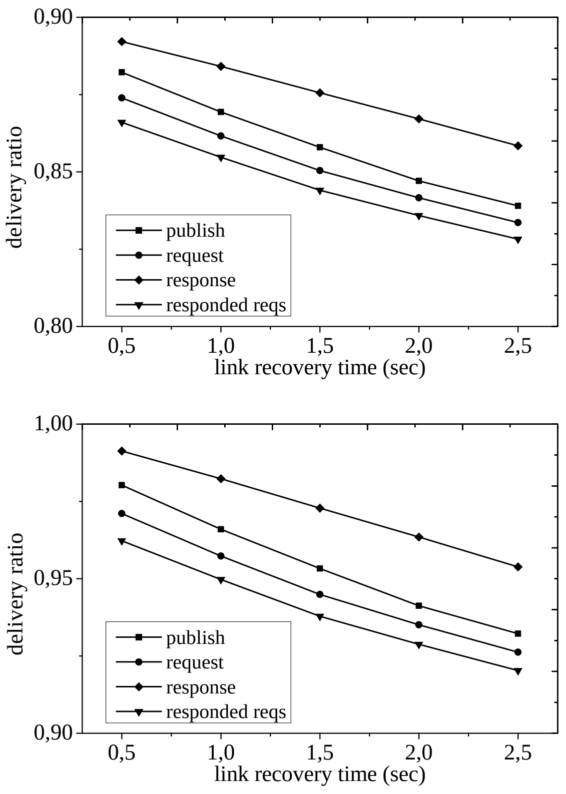


Fig. 5. Delivery ratio (publish, request, response, responded requests) vs recovery time from a link failure when the system has 1 (upper graph) and 2 (lower graph) (replicas) repositories respectively.

the repository(ies), to the brokers that are not connected to that part increases.

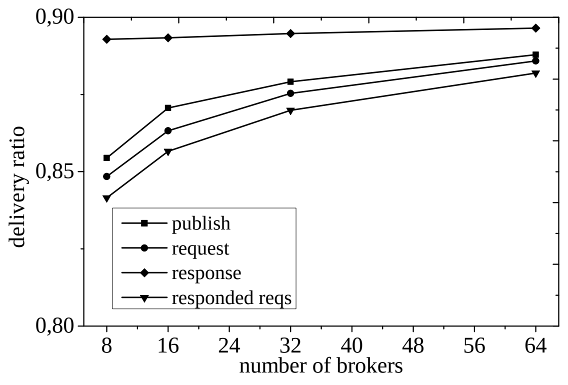
Finally figure 7 shows the average minimum hop distance between the responding repository and the broker where the client making the request is attached to. This metric is indicative of the delay of responses as a function of hops in the network and is obvious that the system with two repositories outperforms the system with only one repository since now messages (policies) can be retrieved from two different points in the network which implies that a clients' request could find a closer repository.

VII. Testbed Experimentation

We modified the REDS system in order to implement the newly introduced messages and support the storage of the advertisements (Advertisement table). Particularly we added as functions in the Routing Strategy interface and its SubscriptionForwardingRoutingStrategy implementation the new messages Advertise(), Request() and Response() and we also added a new interface called AdvertisementTable and implemented it in GenericAdvTable so that it can store the advertisements. The GenericAdvTable and the AdvertisementTable is similar to the GenericTable and the SubscriptionTable already implemented in REDS. Apart from those fundamental changes we also modified from the Overlay layer the Transport interface and its implementations TCPTransport and UDPTransport in such a way that can transfer the new type of messages. Of course minor changes had been made to other modules in order to make the system compatible. Changes has also been done to the client API and particular in functions that generate the Advertise, the Request and the Response message. The main changes were done in the DispatchingService interface and its implementations (TCPDispatchingService and UDPDispatchingService).

For our evaluation we used 6 laptops equipped with a 1,6 GHz Intel Celeron M CPU, 512 MB of RAM and a wireless card configured in ad-hoc mode. In each laptop we installed a broker.

Set 1: In the first set of our evaluation at each laptop clients were subscribing with a rate ranging from 5 - 20 subscriptions per minute (0,09 - 0,33 subs/sec) while the system was equipped firstly with one repository and then with two replica repositories. PMT was introducing new policies with a rate of 6 policies per minute (0,1 policies/sec) in the first scenario and 12 policies per minute in the second, while in every scenario the repository initially has ten stored policies (one for each role) so that every subscriber can retrieve at least one policy from the network. We also assumed no link disconnections. We measure the average time that takes a client to retrieve all the previously introduced policies that matches his/her subscription, we call this time as "stability time".



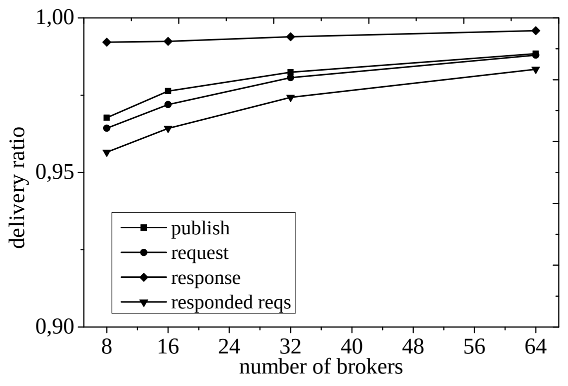


Fig. 6. Delivery ratio (publish, request, response, responded requests) vs number of brokers when the system has 1 (upper graph) and 2 (lower graph) (replicas) repositories respectively.

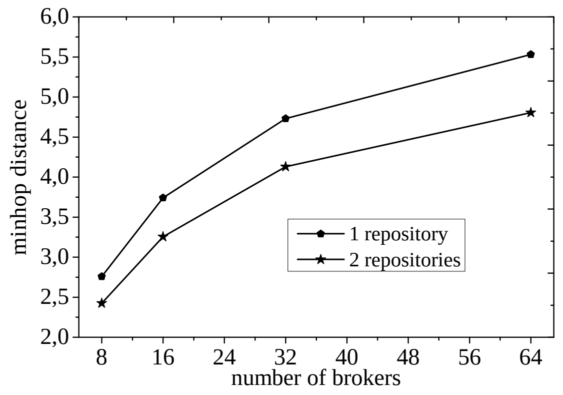


Fig. 7. Minimum hop distance vs number of brokers when the system has 1 and 2 (replicas) repositories respectively.

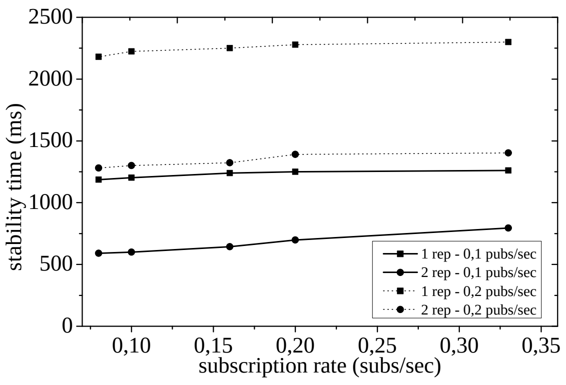


Fig. 8. Stability time vs subscription rate when the system has 1 and 2 (replicas) repositories respectively for 2 different publication rates (0,1 and 0,2 policies/sec).

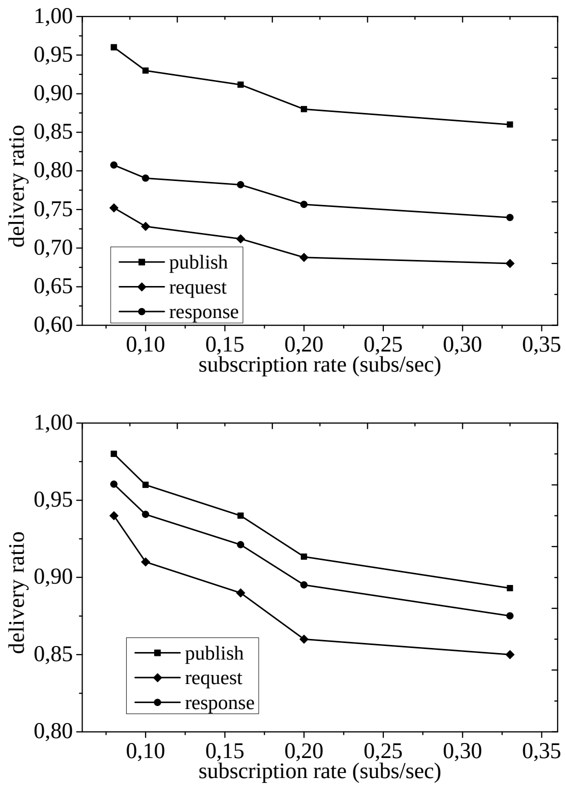


Fig. 9. Delivery ratio (publish, request, response, responded requests) vs subscription rate when the system has 1 (upper graph) and 2 (lower graph) (replicas) repositories respectively.

Figure 8 depicts the stability time. It is obvious that the stability time increases when the publication rate increases since for each request there are more matching messages that have to be delivered. Moreover when we use two repositories requests and responses travel through shorter paths leading to a significant improvement in the stability time. Finally, the increase in subscription rate has a marginal effect on stability time since this metric depends only on the number of stored policies and the distance from the repository(ies).

Set 2: In the second set of the evaluation, we assumed that brokers and clients can move in space, changing position in the network or even disconnect from it. Since our intention is not to examine the efficiency of COMAN but to check the delivery ratio of the newly introduced messages (request, response) when the system has one or two repositories we assumed a low disconnection frequency of four nodes (as node we refer to the laptops) failures-disconnections per minute (here we suppose node failures while in the simulation part we supposed link failures). In order to examine the dynamicity of the system we assumed a subscription rate per broker ranging from 5 to 20 subscriptions per minute (0,09 - 0,33 subs/sec) and a stable publication rate at PMT of 6 policies per minute (0,1 policies/sec).

Figure 9 shows the delivery ratio when the system is equipped with one and two PRs (policy repositories) respectively. Given the fact that no additional measure is taken to recover any lost message the charts are good, and by complementing our system with a protocol which provides such message recovery (epidemic algorithms) we expect to easily obtain a full delivery of 100%. The charts show that our system is only marginally influenced, as in simulations, by the traffic: an increase from 0,09 to 0,33 subs/sec brings less than 10% reduction in every delivery metric for both scenarios, a measure of good scalability retained from COMAN [16]. The differences in the delivery ratio of each type of message is due to the different amount of messages being sent in the network. In every case the system with the two repositories has at least 10% better delivery ratio than the system with the one repository since now policies can be retrieved from two different points in the network making the system more stable in disconnections.

VIII. Conclusion And Future Work

In this paper, we described an architecture for policy based management of wireless ad-hoc networks focusing on the critical problem of policy distribution. We showed how we can incorporate a publish/subscribe communication paradigm, providing a robust and efficient policy distribution mechanism in such a volatile network environment.

Simulations and testbed experiments showed very promising results and our future work focuses on performing experiments in different mobility scenarios. Of course we intend to provide our system with a message recovery mechanism and incorporate opportunistic caching techniques to reduce the volume of the messages in the system and their delivery time.

Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful comments which helped us to improve the technical quality of this paper. This work has been supported by the European Commission through the OneLab2 program.

References

[1] Strassner J., "Policy-Based Network Management, Solutions for the next generation," Morgan Kaufmann, 2003.

[2] Hadjiantonis M., Malatras A., Pavlou G., "A Context-aware Policy-based Framework for the Management of MANETs," 7th IEEE International Workshop on Policies for Distributed Systems and Networks, pp. 23–32, Canada, 2006.

[3] Chen W., Jain N., Singh, S., "ANMP Ad hoc network management protocol," IEEE Journal on Selected Areas in Communications, vol. 17, 1999.

[4] Shen C., Srisathapornphat C., Jaikaeo C., "An adaptive management architecture for ad hoc networks," IEEE Communication Magazine, vol. 41, 2003.

[5] Chadha R., Cheng H., Cheng Y.H., Chiang J., Ghetie A., Levin G., Tanna H., "Policy Based Mobile Ad hoc Network Management," 5th IEEE International Workshop on Policies for Distributed Systems and Networks, 2004.

[6] Phanse K.S., DaSilva L.A., "Protocol support for policy-based management of mobile ad hoc networks," IEEE/IFIP Network Operations and Management Symposium (NOMS), 2004.

[7] Sourlas V., Flegkas P., Tassiulas L., "Policy Distribution using the Publish-Subscribe Paradigm for Managing MANETs," in proc. of 11th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS 2008) held as part of Manweek 2008, Samos, Greece, Volume 5274/2008, pp 14-19, 2008.

[8] Aguilera M. K., Strom R. E., Sturman D. C., Astley M., Chandra T. D., "Matching events in a content-based subscription system," 18th ACM Symposium on Principles of Distributed Computing (PODC '99) Atlanta, GA, May 4-6, pp. 53–61, 1999.

[9] Carzaniga A., Rosenblum D., Wolf A., "Design and evaluation of a wide-area event notification service," ACM Transaction On Computer Systems, vol. 19, pp. 332–383, 2001.

[10] Segall B., Arnold D., "Elvin has left the building: A publish/subscribe notification service with quenching," Proceedings of AUUG97, Brisbane, Australia, Sept. 3-5, pp. 243–255, 1997.

[11] "TIB/Rendezvous," White paper, TIBCO, Palo Alto, CA, 1999.

[12] Cugola G., Nitto E.D., Fugetta A., "The Jedi event-based infrastructure and its application to the development of the opss wfms," IEEE Trans. Softw. Eng. 27, 9 (Sept.), pp. 827–850, 2001.

[13] Cugola G., Picco G., "REDS, A Reconfigurable Dispatching System," 6th International workshop on Software Engineering and Middleware, pp. 9–16, Oregon, 2006.

[14] Moore B., Ellesson E., Strassner J., Westerinen A., "Policy Core Information Model," RFC 3060, IETF 2001.

[15] Chand R., Felber A., "A scalable protocol for content-based routing in overlay networks," 2nd IEEE International Symposium on Network Computing and Applications, pp. 123–130, 2003.

[16] Mottola L., Cugola G., Picco G. "A Self-repairing Tree Topology Enabling Content-Based Routing in Mobile Ad Hoc Networks," IEEE Transaction on Mobile Computing, vol 7, pp. 946–960, 2008.