

# SPINE: Publish/Subscribe for Wireless Mesh Networks through Self-Managed Intersecting Paths

Jorge A. Briones , Boris Koldehofe , Kurt Rothermel  
Universität Stuttgart, IPVS,  
Universitätsstr. 38, 70569 Stuttgart, Germany  
{first name.last name}@ipvs.uni-stuttgart.de

## Abstract

*Application deployment on Wireless Mesh Networks (WMNs) is a challenging issue. First it requires communication abstractions that allow for interoperability with Internet applications and second the offered solution should be sensitive to the available resources in the underlying network. Loosely coupled communication abstractions, like publish/subscribe, promote interoperability, but unfortunately are typically implemented at the application layer without considering the available resources at the underlay imposing a significant degradation of application performance in the setting of Wireless Mesh Networks.*

*In this paper we present SPINE, a content-based publish/subscribe system, which considers the particular challenges of deploying application-level services in Wireless Mesh Networks. SPINE is designed to reduce the overhead which stems from both publications and reconfigurations, to cope with the inherent capacity limitations on communication links as well as with mobility of the wireless mesh-clients. We demonstrate the effectiveness of SPINE by comparison with traditional approaches in implementing content-based publish/subscribe.*

## 1. Introduction

Wireless Mesh Networks are gaining importance as a flexible and low cost wireless communication infrastructure that serve as backbone for wireless sensor networks and as community networks that provide *public Internet access* to rural and urban environments [2]. Various models of WMNs exist. While some approaches consider any device to participate in an ad-hoc manner to establish a wireless mesh network, *infrastructure-based* WMNs consist of a set of dedicated *mesh routers* which provide communication services to *mesh clients* connecting to the WMN.

In order to support the deployment of distributed appli-

cations inside WMNs event notification services can play a significant role. In particular, publish/subscribe allows the delivery of information according to the interest of subscribers and supports the decoupling of subscribers and publishers [14]. Although such services could be provided by the infrastructure of the mesh network, we expect that they are typically implemented by the mesh clients supporting a wider deployment of applications.

In spite of many benefits publish/subscribe can offer to distributed applications it remains a challenging task to realize publish/subscribe to match well the characteristics of WMNs. Mesh routers typically have limited capacities with respect to available bandwidth on their established communication links. Hence, the performance of event notification systems as well as the WMN itself depends a lot on the number of messages which need to be sent by the mesh routers in order to establish a service. Moreover, even if mesh clients disseminate information in a way that reduces the overhead of mesh routers, it is expensive to maintain such properties in the presence of mobility of the mesh clients. This holds especially for the maintenance of content-based publish/subscribe systems which provide expressive matching of events by filtering over a possibly large set of attributes.

While traditional P2P-based solutions towards implementing publish/subscribe in a decentralized manner are an appealing candidate, there are several shortcomings depending on their nature. First, most approaches perform well in terms of overlay messages, however they do not consider the underlying topology, for instance formed by the mesh routers. Rendezvous-based [27, 25] approaches have limitations when events are matched over several attributes. Finally, approaches which use subscription-forwarding [6] impose low overhead in terms of messages sent at the underlying topology, however have severe limits concerning the adaptation to changes in the underlying topology.

In this paper we present an algorithm which offers a low cost regarding:

- i) messages required for reconfigurations and the matching of events,
- ii) storage required for the matching of subscriptions

Every subscriber will be placed in a 2-dimensional grid according to the location of the wireless mesh router. Subscriptions and publications are associated with intersecting paths in the overlay. Every hop in a path involves mesh clients with physical proximity, thus providing for many topologies a low overhead concerning messages sent at the underlay. Our evaluations show that in spite of the comparably low cost for reconfiguration and required storage, the algorithm performs similar with respect to subscription forwarding algorithms which take into account the underlying topology of the WMN.

**Organization** Section 2 states the model and problem. In Section 3 we present an algorithm towards establishing content-based publish/subscribe in wireless mesh-networks as well as analyze its properties. Moreover, we discuss further methods to reduce storage cost as well as increase performance. Section 4 presents an evaluation of the performance regarding subscription management, notification forwarding as well as the robustness of the algorithm considering accuracy of location information and churn. Section 5 discusses related work, while Section 6 presents conclusions and future work.

## 2. Publish/subscribe in wireless mesh networks

We consider WMNs which follow the *infrastructure meshing* model (cf. Figure 1)[2, 15, 1]. In this model, the WMN is formed by a set of *mesh routers* and a set of *mesh clients*. Mesh routers form the *wireless backbone* which provides to mesh clients the possibility of establishing point-to-point connections to any other client which resides in the area covered by the WMN. Mesh clients do not participate in the routing of messages, nor do we assume that mesh clients have an insight on how the mesh routers manage their routing tables and forward messages. Although WMNs can also provide interconnectivity to other networks which potentially could increase the performance by connecting a pair of mesh routers via a more efficient connection medium, we focus here on the issue of establishing a service only for a network formed by direct communication of mesh routers. Moreover, we assume that mesh routers can provide information to mesh clients about their approximate location in the network, for instance having integrated a GPS device or being configured with location information. Similar assumptions have been considered by existing protocols for WMNs, like Geographically Distributed Addressing (GDA)[11], where the IP address of each device is assigned based on the location information.

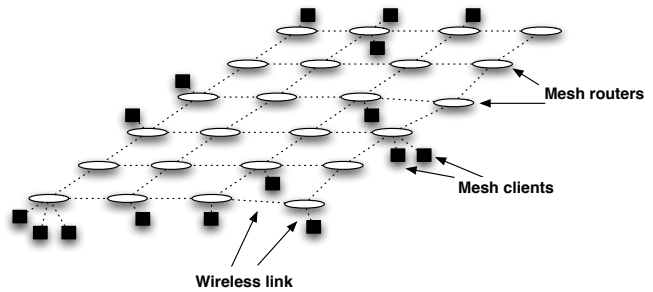


Figure 1. Infrastructure meshing model

In SPINE the location information is shared to the mesh clients, so mesh clients can also obtain a rough estimate on their own location in the WMN. Similar to the bootstrap problem in P2P systems we consider that even for services implemented by the mesh clients, there is some initial support by the wireless infrastructure in finding one client which can offer the corresponding service.

Our focus is on establishing a decentralized content-based publish/subscribe service which runs entirely on mesh clients and that matches the characteristics of WMNs. This allows the execution of the system on current deployments of WMNs and avoids the manageability issues that arise when the publish/subscribe system runs on mesh routers. The system will provide the common operations of the publish/subscribe model, denoted by *pub()*, *sub()* and *unsub()*. Processes express their interest in events by issuing subscriptions and unsubscriptions, by calling the *sub()* or *unsub()* operation respectively. Correspondingly subscribers will be asynchronously notified about all published events submitted by publishers by calling the *pub()* operation.

Our data model follows traditional content-based systems [6, 14], where events consist of an arbitrary number of attributes, and each attribute corresponds to a name/value pair. A subscriber can express its interest in events by placing filters which allow conjunction and disjunction over arbitrary ranges of values. Whenever a specific filter evaluates for a disseminated event to true we say the event is *matched* and the publish/subscribe service will deliver the event to the respective subscriber. Typically, publish/subscribe guarantees space, time and synchronization decoupling. However, for similar reasons as applied in general in the design of decentralized publish/subscribe systems we do not address time decoupling, i.e. a subscriber only receives events while being up.

In order to use the available bandwidth of mesh routers efficiently it is important to reduce the overhead which comes from:

- i) the cost of forwarding notification messages to subscribers inside the WMN,
- ii) the cost of reconfiguration because of new subscriptions and unsubscriptions or mobile mesh clients

Moreover, from the perspective of a mesh client it is important to allow the maintenance of the data structure at reasonable storage cost.

Note there is no solution which can satisfy all properties in an optimal manner, but require a good compromise between all these properties. For instance, consider an algorithm which aims to achieve the optimal number of underlay messages. In fact, each publication can match any subset of all subscribers. Hence, the algorithm would need to maintain in the worst case a minimum spanning tree for each subset of subscribers in the system. Clearly, this solution is impractical since a huge amount of storage is required and the maintenance of corresponding routing information would lead to a large overhead for every reconfiguration. Therefore, many approaches towards content-based publish/subscribe (e.g., [6, 7]) only maintain a single spanning tree and perform inside the tree content-based routing. *Subscription Forwarding*, for example, performs well in terms of messages needed for each notification dissemination step, however the cost for storage and reconfiguration is still high.

Further, improvements can be achieved if publishers advertise their messages beforehand [6], establishing multiple trees according to the source of the event. However, this knowledge is not available to applications in general. An alternative is the establishment of dedicated rendezvous-points on which all events are matched and forwarded [27, 25]. This technique is typically applied in the context of topic-based publish/subscribe, i.e. events consist only of a single attribute. However, if events need to be matched over multiple attributes performance degrades significantly since during a subscription each attribute needs to be mapped to a dedicated rendezvous point.

The solution we propose aims to achieve similar performance for the delivery of notifications as can be achieved by traditional subscription forwarding by significantly reducing the expected cost for reconfiguration and storage. Each operation only involves a subset of mesh clients in the system. The cost in forwarding notifications does not depend on the number of attributes, nor do we make any assumptions on knowledge what kind of events will be disseminated by publishers. Also further optimizations as typically applied to other schemes such as covering of filters can be applied to further reduce the storage cost and cost for messages in notification forwarding.

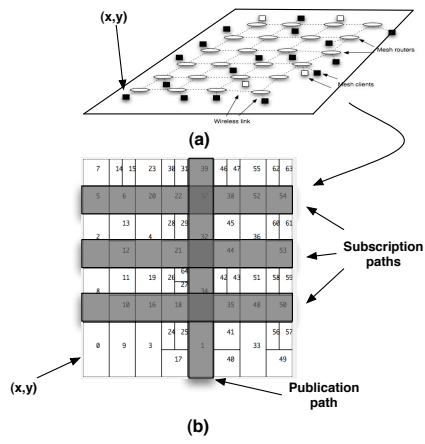


Figure 2. General idea of the algorithm

### 3. SPINE

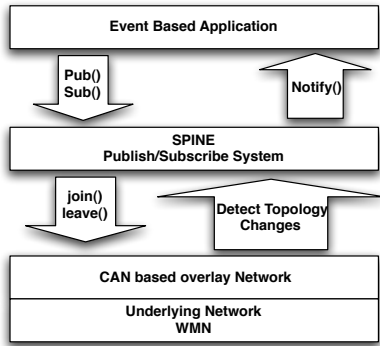
SPINE implements the operations of the publish/subscribe system by establishing vertical and horizontal paths within a 2-dimensional space partitioned into rectangular zones. While subscriptions and unsubscriptions are propagated along a horizontal path through the overlay, publications will be forwarded along a vertical path (cf. Figure 2(b)). The paths are maintained such that any vertical and horizontal path will intersect. Thus it is ensured that any forwarded event will meet any matching subscription. The management of paths relies on each peer managing a directional *neighborhood table* denoted by  $N_i$ , which provides information about neighbors to the right, left, up, down as well as peers which share the same zone. Changes to the *neighborhood table* are triggered by a neighborhood detection scheme that works similar to a 2-dimensional CAN [26] and will trigger reconfigurations for the actual path maintenance (cf. Figure 3).

In the following we present how SPINE implements the operations of the publish/subscribe system based on the basic topology.

#### 3.1. Basic data structures

In addition to the *neighborhood table*  $N_i$ , SPINE maintains:

- i) a *local subscription table*  $L_t$  which consists of the set of subscriptions that a peer has submitted, and
- ii) a *routing table*  $R_t$  which is used by the peers to forward notifications based on their contents to the corresponding subscribers



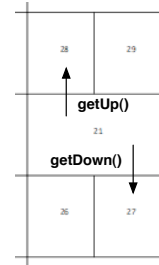
**Figure 3. Decoupling the Publish/Subscribe System from the P2P overlay management**

The routing table will contain information about subscriptions, identifiers of corresponding subscribers, and a direction information which allows to determine the next peer in forwarding a notification to a subscriber.

**Management of zones.** The management of zones follows closely the way zones are managed inside CAN. A new peer  $p$  that joins can contact an arbitrary peer and will be guided to a peer, say  $q$ , which manages a zone containing the location of  $p$ .  $q$  will assist  $p$  in updating the neighborhood table, but also ensure the maintenance of the corresponding routing tables in neighboring zones. A peer managing a zone performs a split of zone if it contains more than one peer which are connected to different mesh routers. Moreover, peers managing a zone may need to initiate a merge operation if an adjacent zone has no coordinator anymore, for instance because of a failure.

**Management of identities.** An identity  $(x, y, r)$  of SPINE consists of a 2-dimensional location information determined by  $x$  and  $y$  and an additional parameter  $r$  which is selected according to the physical identity over a consistent hash function. The peer with the smallest value  $r$  is responsible to manage dynamic changes to a zone, i.e. splitting a zone or merging with an adjacent zone. SPINE initiates a split of zones if a zone contains peers with different location information.

**Establishing paths in the overlay.** SPINE establishes *horizontal* and *vertical* paths inside the overlay by associating with a message a specific *direction*  $\in \{left, right, up, down\}$ . For a neighbor set  $Nt$  the functions  $getLeft()$ ,  $getRight()$ ,  $getUp()$  and  $getDown()$  will return one neighbor located in an adjacent zone in the respective direction (cf. Figure 4). A horizontal path is formed by the



(a)



(b)

**Figure 4. Directional functions**

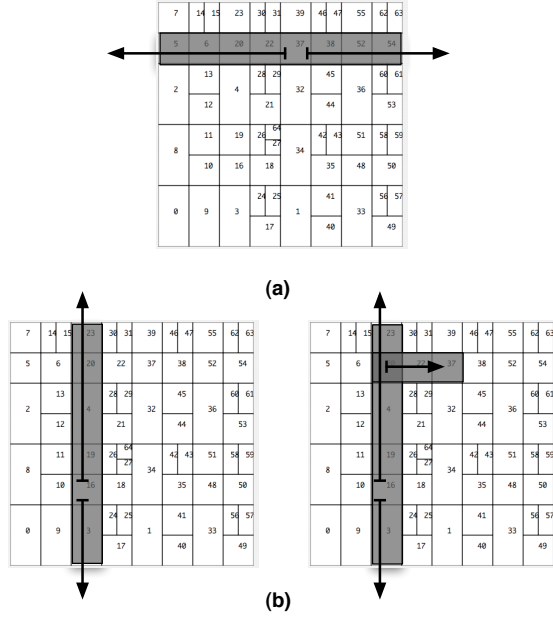
transitive closure over the right/left relation while a vertical path is formed over the transitive closure of the up/down relation. Note that a zone may have several neighbors in the same direction. In this case an arbitrary metric can be used in order to decide on the peer which should be selected. For instance, a peer could select the peer with smallest latency in order to improve end-to-end latency of the publish/subscribe system. The selection can also take into account load balancing issues by establishing for each message a different path by relying on a random choice.

### 3.2. Publish/subscribe operations

Figure 6 and Figure 7 describe the basic algorithm SPINE uses in order to manage the basic operations of the publish/subscribe system.

**Subscriptions and unsubscriptions.** On subscription, any peer  $p$  will store its subscription in  $Lt$ .  $p$  also creates two messages of type  $SUB$ , one directed to the *right* and the other directed to the *left*. Each message is forwarded along its direction to a neighbor  $q$  in an adjacent zone and  $q$  is stored along the subscription in  $Rt$ . Finally,  $p$  will also forward to each peer of the same zone a non-directed  $SUB$  message.

A peer  $p$  which receives a  $SUB$  message from  $q$  selects a successor  $r$  following the direction of the  $SUB$  message.  $p$  will store the subscription in  $Rt$  including information on  $q$  and  $r$  as well as the direction of the message. Finally,  $p$  sends a directed  $SUB$  message to  $r$  as well as non-directed  $SUB$  messages to all peers of the same zone (cf. Figure 5 (a)).



**Figure 5. (a) Subscription dissemination (b) Notification Dissemination and Routing**

A peer  $p$  which unsubscribes will forward directed messages of type *UNSUB* by following the path formed by the directed *SUB* messages. Moreover  $p$  will inform all peers of its zone about the unsubscription. Also every peer receiving a directed *UNSUB* message will forward the message according to its information of the routing table and inform all peers in its zone. Every peer which has obtained information about an unsubscription will remove all related information from its routing table.

**Publications.** Similar to the subscription management a publisher will create directed messages of type *PUB* to neighbors in direction *up* and *down*. Every peer receiving a *PUB* message will further forward the message by preserving its direction. Every peer involved in forwarding an event will also try to match the event against all its subscriptions. If an event  $e$  matches one or multiple subscription, the peer will create for each distinct subscription path in its a routing table a message of type *CBR* and forwards it along every path. Finally, each peer receiving an event in a *PUB* or *CBR* and the event matches the local subscription table  $Lt$  will deliver the event to the application (cf. Figure 5 (b)).

**Coping with dynamics and failures.** The CAN-based management of zones ensures that zones are maintained in a dynamic and fault-tolerant manner. Similar to traditional failure detectors in distributed computing, changes to the

**proc** *Sub*( $s$ )  
**Input:** Subscription  $s$   
 (\* Subscribe function \*)  
 1.  $l \leftarrow getLeft(Nt)$ ;  
 2.  $r \leftarrow getRight(Nt)$ ;  
 3.  $S \leftarrow getSameZonePeers(Nt)$ ;  
 4. **if**  $l$  exists  
 5.     **then**  $send(SUB, s, true, l)$ ;  
 6. **if**  $r$  exists  
 7.     **then**  $send(SUB, s, true, r)$ ;  
 8. **if**  $S \neq \emptyset$   
 9.     **then**  $\forall q \in S \ send(SUB, s, false, q)$ ;  
 10. **return**

**proc** *OnReceiveOfSub*( $s, pr, forward$ )

**Input:** Subscription  $s$   
**Input:** Peer  $pr$   
**Input:** Boolean  $forward$   
 (\* On receive of a subscription \*)  
 1.  $dir \leftarrow getPeerDirection(pr)$ ;  
 2.  $odir \leftarrow getOppositeDirection(dir)$ ;  
 3.  $r \leftarrow getPeerWithDirection(odir)$ ;  
 4.  $storeSub(s, s.ID, dir, pr.ID, r)$ ;  
 5. **if**  $r$  exists  $\wedge forward = true$   
 6.     **then**  $send(SUB, s, r)$ ;  
 7. **return**

**Figure 6. Algorithm for establishing subscription paths**

topology are reflected by changes inside the neighbor table  $Nt$ . The maintenance of paths relies on reacting to changes in  $Nt$ . If a peer  $p$  encounters that for one of its directed subscription paths in  $Rt$  the successor is not in  $Nt$  then  $p$  will reinitiate the forwarding of subscriptions in direction of the original subscription message. Similar if the predecessor of a subscription path maintained in  $Rt$  has left  $Nt$ , SPINE initiates after a delay an unsubscription along the subscription path. Moreover, a new peer added to  $Nt$  is informed by the coordinator about subscriptions maintained in the zone.

**Mobility of mesh clients.** When a peer  $p$  connects to a new mesh router,  $p$  computes the distance of its current location to the location of its identity. If the threshold diverges more than a predefined threshold (for instance an error location of 10% has shown to yield still very good performance results), a peer will trigger a reconfiguration, i.e.  $p$  will determine a new identity based on its current location and join the overlay. Moreover, it will reinitiate all of its subscriptions. After an additional delay it will initiate a leave with respect to its old location.

```

proc Pub(p)
Input: Publication p
(* Publish function *)
1.  $u \leftarrow getUp(LocalNt)$ ;
2.  $d \leftarrow getDown(LocalNt)$ ;
3. if u exists
4.   then send(PUB, p, u);
5. if d exists
6.   then send(PUB, p, d);
7.  $mt \leftarrow GetSetOfPeersOfMatchedSubs(Rt, p)$ ;
8. if  $mt \neq \emptyset$ 
9.   then  $\forall q \in mt$  send(CBR, p, q);
10. return

proc OnReceiveOfPub(p, pr)
Input: Publication p
Input: Peer pr
(* On receive of a Publication message *)
1. match(p)
2.  $dir \leftarrow getPeerDirection(pr)$ ;
3.  $odir \leftarrow getOppositeDirection(dir)$ ;
4.  $r \leftarrow getPeerWithDirection(odir)$ ;
5. if r exists
6.   then send(PUB, p, r);
7.  $mt \leftarrow GetSetOfPeersOfMatchedSubs(Rt, p)$ ;
8. if  $mt \neq \emptyset$ 
9.   then  $\forall q \in mt$  send(CBR, p, q);
10. return

proc OnReceiveOfCBR(p, pr)
Input: Publication p
Input: Peer pr
(* On receive of a CBR message *)
1. match(p)
2.  $dir \leftarrow getPeerDirection(pr)$ ;
3.  $mt \leftarrow GetSetOfPeersOfMatchedSubs(Rt, p)$ ;
4. if  $mt \neq \emptyset$ 
5.   then  $\forall q \in mt$  AND  $q.dir \neq dir$ 
6.     send(CBR, p, q);
7. return

```

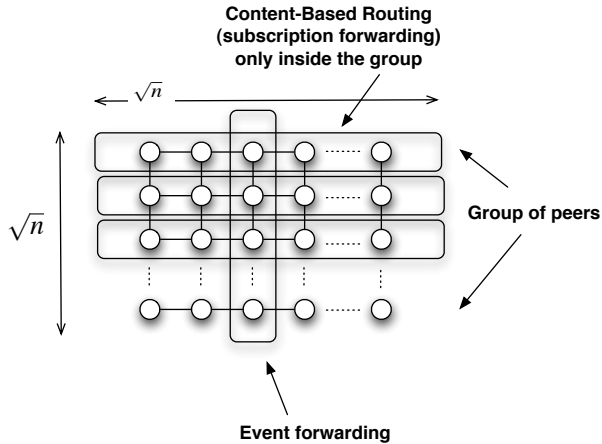
**Figure 7. Algorithm for establishing publications paths**

### 3.3. Properties

**Performance.** For a subscription the cost in the number of overlay messages is bounded by the number of zones visited on a horizontal path and the total number of peers which reside in the respective zones. Similar the cost for a notification is bounded by the spanning tree formed by sending an event along a vertical path and the length of each subscription path that matches the respective event. In a WMN which consists of  $n$  mesh routers embedded in a regular grid topology as illustrated in Figure 8, the depth of each path is of  $O(\sqrt{n})$ . In spite of the comparably large number of messages (i.e.  $O(\log n)$  for many P2P systems), SPINE is expected to perform well in terms of underlay messages. Since SPINE always communicates with peers located in proximity, communication cost of a path would approximately yield the same cost then sending an overlay message from a zone located at left boundary of the grid to the right boundary of the grid. Although SPINE cannot guarantee good performance for arbitrary topologies we expect SPINE to perform at low cost for many wireless network topologies – especially those which provide a full coverage of an area.

To further reduce the message overhead SPINE can be used in combination with covering of filters. By using the successor relation to cover its own subscriptions and the subscriptions of its successors/predecessors a peer can decide whether an event over a horizontal publication path has any potential subscribers. This is especially useful if there exists no interested subscribers for a particular event. By covering over the routing table *Rt* we can further reduce storage requirements, which otherwise grow in the order of subscriptions placed along a subscription path.

**Reliability.** Reliability in most decentralized systems is coupled to the dynamics and the number of reconfigurations which need to be tolerated. Also in SPINE, changes in the basic overlay, caused by failures or subscriptions/unsubscriptions, can lead to temporally inconsistencies in the subscription paths and hence lead to events which cannot be delivered during the reconfiguration. SPINE’s mechanisms to reorganize the overlay, however, ensure that at most peers along a subscription path need to be involved in a reconfiguration step. In many cases, the reconfiguration only involves a pair of adjacent nodes. Hence, the effect of a single reconfigurations on reliability is very limited compared to traditional algorithms. For instance, reconfigurations in the context of *Subscription Forwarding* (SF) [8] (cf. Figure 9) would require in the worst case a reconfiguration cost in the order of the whole overlay structure [6]. Nevertheless, reliability of SPINE can be enhanced by buffering published messages and allow for retransmission if changes in *Rt* or *Nt* occur.



**Figure 8. Performance in a regular grid topology**

Once the overlay paths are stable, SPINE guarantees by the geometric properties of the overlay that all published events are delivered by corresponding subscribers. Note that the set of zones form a rectangular area, and no pair of zones is overlapping, hence each vertical path and horizontal path embedded in the overlay will have one zone at which paths are intersected if routing tables are stable. Since a subscription will be propagated to all peers of a zone, any vertical publication path is guaranteed to involve one peer which has stored information about the subscription. In particular, if an event matches a particular subscription the routing table ensures that the event can be forwarded to the subscriber.

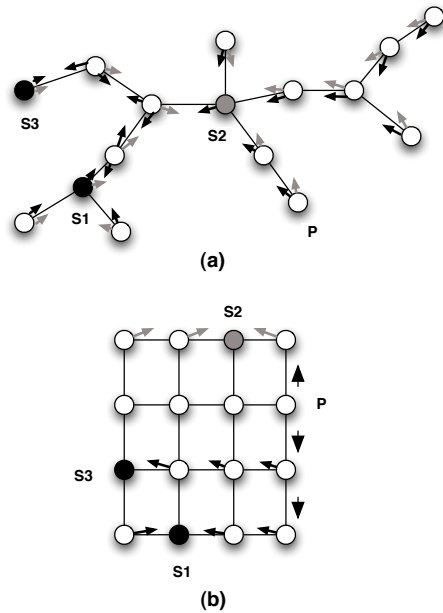
## 4. Evaluation

### 4.1. Evaluation of the algorithm on WMNs

We have implemented SPINE in the network simulator NS-2. To ensure a realistic simulation we modified the current NS-2 implementation of the DSDV protocol to fit the *infrastructure meshing model*.

#### 4.1.1 Simulation settings

The underlying topology is composed of sets of 64 mesh routers and 64 mesh clients. In the evaluation we consider a *random* topology, in which a pair of mesh routers and clients are placed uniformly at random within a 700x700m area. We use a random topology because it captures more realistically real life deployments of WMNs compared to typically evaluated grid topologies. The propagation model



**Figure 9. Subscription forwarding vs SPINE**

used is *Two-Ray Ground Reflection*, the transmission range is 250m and the data rate is 11MB.

In the simulation, events and subscriptions are represented as randomly-generated  $n$  character strings. Each one of the generated strings have any of the typically 96 printable characters, like in [23]. We perform a comparison two other standard approaches: *Content-Based Subscription Forwarding* and *Rendezvous-Based Routing*. The first one is a traditional content-based routing approach which is commonly used as a baseline for comparison [8]. The second uses Chord, a structured P2P overlay, to route subscriptions and notifications to *Rendezvous Nodes* [21]. The three approaches were tested under the same physical topology and the same set of publishers and subscribers. In the case of *Content-Based Subscription Forwarding* the overlay network was pre-computed by calculating a spanning tree considering the topology of the WMN. For *Rendezvous-Based Routing* and SPINE the self-management properties were used to handle the creation of the overlay network. The metrics employed for the comparison of the algorithms are: *subscription forwarding overhead* and *notification forwarding overhead*. The first one measures the message complexity for handling subscriptions and unsubscriptions while the latter measures the message complexity in disseminating and matching events.

### 4.1.2 Simulation results

Figure 10 presents the *Subscription Forwarding Overhead* measured at the overlay layer. The experiment involved 45 randomly generated subscribers each one of them issuing the same number of subscriptions in the range of 2 to 18. Each subscription was composed of only a single attribute. As expected *Content-Based Subscription Forwarding* performs at the largest overhead while *Rendezvous-Based Routing* and SPINE provides significantly lower overhead since both approaches involve only a subset of nodes. In this case, the *Rendezvous-Based Routing* using Chord as P2P substrate provide the more efficient routing scheme in terms of overlay messages.

The experiments illustrated in Figure 11 considers the overhead in underlay messages and the overhead which stems from the expressiveness of the event notification system by varying the numbers of attributes from 1 to 7. Again the experiment involved 45 randomly generated subscribers, where each of the subscribers issued 4 subscriptions. One can see that SPINE already provides lower overhead for two attributes. Moreover, the overhead for subscriptions does not increase with the expressiveness of events disseminated.

Figure 12 and Figure 13 show that SPINE can achieve similar performance to *Content-Based Subscription Forwarding* with respect to the notification forwarding overhead, while the overhead for *Rendezvous-Based Routing* is significantly larger. Both experiments considered 45 subscribers, each issuing 10 subscriptions. In the setting of Figure 12 the experiment used a varying number of publications (10,50,90,130) for a fixed number of 9 attributes, while in the setting of Figure 13 the number of attributes varied from 1 to 9 for a fixed number of 100 publications.

Figures 14 and 15 evaluate the impact of inaccurate location information on the number of underlay message in SPINE for notifications and subscriptions, respectively. The experiments consider 5 sets varying from absolutely accurate location information up to 100% of peers having inaccurate location information (a location selected uniformly at random inside the area covered by the WMN) considering each time a randomly generated set of 64 subscribers. We can see that on the one hand accurate location information significantly reduces the message overhead, but on the other hand also tolerates location failures up to 10% with only small additional overhead. This information is of importance in the presence of mobility to trigger a reconfiguration step.

The experiment in Figure 16 shows the effect of churn on the event delivery rate. In the setting a publisher disseminated an event every 0.5 sec over an interval of 40sec which needs to be matched at 66 different subscribers. 10 seconds after the publisher has disseminated its first event the system needed to cope with a burst at which up to 25% concurrent

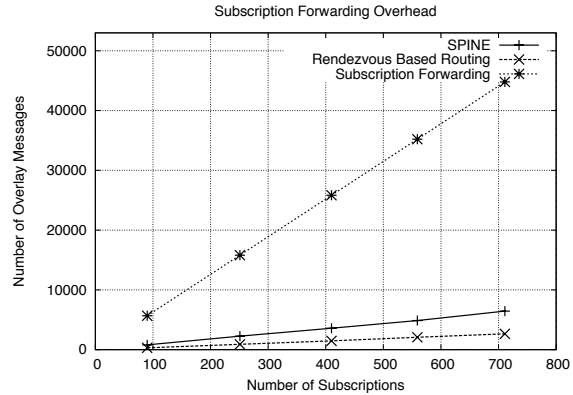


Figure 10. Subscription forwarding overhead depending on the number of subscriptions

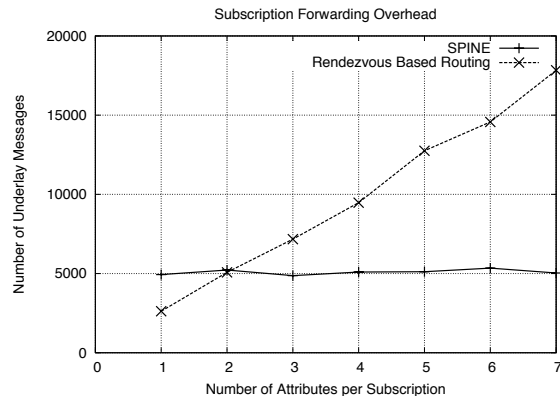


Figure 11. Subscription forwarding overhead depending on the number of attributes

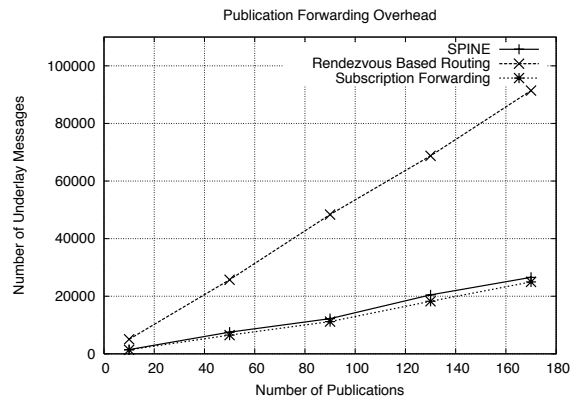
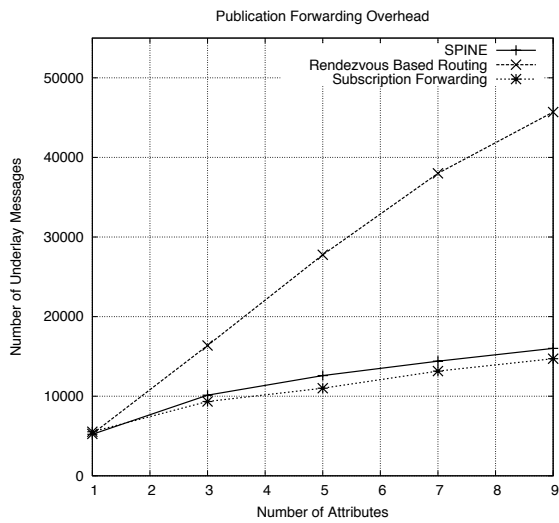
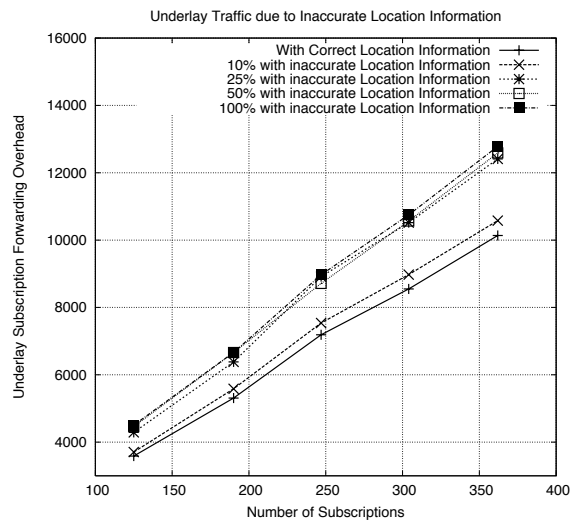


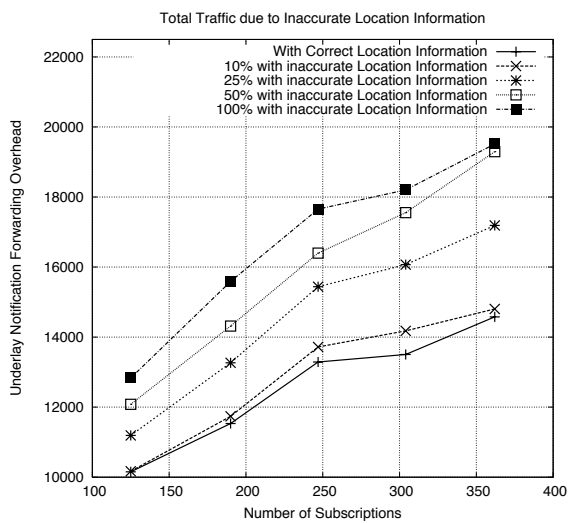
Figure 12. Notification forwarding overhead depending on the number of publications



**Figure 13. Notification Forwarding Overhead depending on the number of attributes**



**Figure 15. Subscription overhead because of inaccurate location information**



**Figure 14. Notification overhead when peers use inaccurate location information**



**Figure 16. Event delivery in the presence of sudden changes in the overlay**

join and leave operations occurred. We can see that the path management scheme converges quickly to a stable configuration allowing only for a small message loss even in the presence of a large number of reconfigurations.

## 5 Related Work

Intersecting paths have been proposed in the context of Sensor Networks as a technique to store and retrieve data without flooding the network. Similar, to SPINE *Comb-Needle*[19] replicates an information stored on a directed path in the sensor and allows to pull information on a corresponding intersecting path. Double Rulings [28] is an extension to this approach, which considers intersection between curves instead. However, the dynamic management has not been an issue in the setting of sensor networks. HyperCBR [9] recently adapted intersecting-paths for wide area network applications. While HyperCBR yields good performance in terms of overlay messages, our evaluation results have shown that SPINE can provide significant performance gains in WMNs by considering location informations and taking non-uniform distributions of peer identities into account. Moreover, SPINE also considers churn which results from mobility of the mesh clients by providing a lightweight hand-over mechanism.

In the context of WMNs, most approaches towards event notification services [16] have considered issues which are of importance to establish notification services provided by the wireless mesh router themselves. The possibility to adopt such approaches towards publish/subscribe services maintained by mesh clients have severe limits since mesh clients typically have no insight on how mesh routers establish connectivity for the mesh clients. The same is true for the application of approaches in Mobile Ad-Hoc Networks (MANET) (e.g., [29, 3, 20]).

In fact most relevant in our setting are the classical approaches towards publish/subscribe in Wide Area Networks [6, 13, 24, 21]. However, the requirement to allow for low cost reconfigurations and low message overhead for notification of messages is mainly dealt with by topic-based publish/subscribe systems [27, 4, 5, 17]. The deployment of expressive content-based solutions [23, 12, 22] typically rely on structures which have shortcomings considering dynamic changes, which are of particular importance in the context of mobility or imply a significant overhead on underlay messages. In our evaluation we have focused on those approaches which are most promising to reduce the notification overhead or reduce the reconfiguration overhead for dynamic changes.

Approaches which rely on semantic clustering [10] or covering of filters [21] can be seen as composite techniques which would also allow SPINE to further reduce the message load.

## 6. Conclusions

We presented SPINE, an expressive Publish/Subscribe System, which matches the particular characteristics of Wireless Mesh Network. In particular, we have proposed an algorithm which maintains intersecting subscriptions paths in a decentralized and self-organizing manner. Our evaluation shows that SPINE performs well in terms of dynamic changes of the overlay caused by mobility, unsubscriptions and subscriptions. SPINE introduces a small overhead regarding the number of underlay messages, i.e. the algorithm achieves similar performance compared to approaches which assume knowledge about the underlying topology. At the same time storage requirements and reconfiguration overhead are significantly lower.

In the future, we will examine techniques to further reduce the message overhead by applying covering of filters, but also take into account cross layer optimizations which can give SPINE further inside on the underlying topology.

## Acknowledgements

This work was supported by the German Academic Exchange Service (DAAD) (Grant A/05/12370).

## References

- [1] Google-Wifi, <http://wifi.google.com/>.
- [2] I. H. Akyldiz and X. Wang. A survey on wireless mesh networks. *IEEE Radio Communications*, September 2005.
- [3] R. Baldoni, R. Beraldi, G. Cugola, M. Migliavacca, and L. Querzoni. Structure-less content-based routing in mobile ad-hoc networks. In *Proceedings of the IEEE Int. Conf. on Pervasive Services*, July 2005.
- [4] R. Baldoni, R. Beraldi, V. Quema, L. Querzoni, and S. Tucci-Piergiovanni. TERA: Topic-based Event Routing for peer-to-peer Architectures. In Jacobsen et al. [18], pages 2–13.
- [5] R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg. Content-based publish-subscribe over structured overlay networks. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pages 437–446, Washington, DC, USA, 2005. IEEE Computer Society.
- [6] A. Carzaniga, D.S. Roseblum, and A. Wolf. Design and evaluation of a wide area event notification service. *ACM Transactions on Computer Systems*, 19, August 2001.
- [7] A. Carzaniga and A. L. Wolf. Forwarding in a content-based network. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 163–174, New York, NY, USA, 2003. ACM.
- [8] S. Castelli, P. Costa, and G. P. Picco. Modeling the Communication Costs of Content-based Routing: The Case of Subscription Forwarding. In Jacobsen et al. [18], pages 38–49.

- [9] S. Castelli, P. Costa, and G. P. Picco. Hypercbr: Large-scale content-based routing in a multidimensional space. In *Proceedings of the 27th Conference on Computer Communications (INFOCOM'08)*, 2008.
- [10] R. Chand and P. Felber. Semantic peer-to-peer overlays for publish/subscribe networks. In *Proceedings of International Euro-Par 2005 Parallel processing*, 2005.
- [11] B.-N. Cheng, S. Kalyanaraman, and M. Klein. A geography-aware scalable community wireless network testbed. In *Proceedings of the first international conference on Testbeds and Research Infrastructures for the development of networks and communities (TRIDENTCOM'05)*, 2005.
- [12] G. Cugola, D. Frey, A. L. Murphy, and G. P. Picco. Minimizing the reconfiguration overhead in content-based publish-subscribe. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 1134–1140, New York, NY, USA, 2004. ACM Press.
- [13] G. Cugola, E. D. Nitto, and A. Fuggetta. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. *IEEE Transactions on Software Engineering*, 27(9):827–850, 2001.
- [14] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.
- [15] S. M. Faccin, C. Wijting, J. Knecht, and A. Damle. Mesh WLAN networks: Concept and system design. In *Euro-Par 2005 Parallel Processing, LNCS*, April 2005.
- [16] D. Gavidia, S. Voulgaris, and M. van Steen. A gossip-based distributed news services for wireless mesh networks. In *Proceedings of Wireless On Demand Network Systems, WONS'06*, January 2006.
- [17] A. Gupta, O. D. Sahin, D. Agrawal, and A. E. Abbadi. Meghdoot: content-based publish/subscribe over p2p networks. In *Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 254–273, New York, NY, USA, 2004. Springer-Verlag New York, Inc.
- [18] H.-A. Jacobsen, G. Mühl, and M. A. Jaeger, editors. *Proceedings of the Inaugural Conference on Distributed Event-Based Systems*, New York, NY, USA, June 2007. ACM Press.
- [19] X. Liu, Q. Huang, and Y. Zhang. Combs, needles, haystacks: balancing push and pull for discovery in large-scale sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 122–133, New York, NY, USA, 2004. ACM Press.
- [20] R. Meier and V. Cahill. Steam: Event-based middleware for wireless ad hoc network. In *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 639–644, Washington, DC, USA, 2002. IEEE Computer Society.
- [21] G. Mühl, L. Fiege, and P. R. Pietzuch. *Distributed Event-Based Systems*. Springer, Aug. 2006.
- [22] H. Parzyjegl, G. Mühl, and M. Jaeger. Reconfiguring publish/subscribe overlay topologies. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'06)*, 2006.
- [23] G. P. Picco, G. Cugola, and A. L. Murphy. Efficient content-based dispatching in the presence of topological reconfiguration. In *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 234, Washington, DC, USA, 2003. IEEE Computer Society.
- [24] P. R. Pietzuch and J. Bacon. Hermes: A distributed event-based middleware architecture. In *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 611–618, Washington, DC, USA, 2002. IEEE Computer Society.
- [25] P. R. Pietzuch and J. Bacon. Peer-to-peer overlay broker networks in an event-based middleware. In *DEBS '03: Proceedings of the 2nd international workshop on Distributed event-based systems*, pages 1–8, New York, NY, USA, 2003. ACM.
- [26] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM.
- [27] A. I. T. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The design of a large-scale event notification infrastructure. In *NGC '01: Proceedings of the Third International COST264 Workshop on Networked Group Communication*, pages 30–43, London, UK, 2001. Springer-Verlag.
- [28] R. Sarkar, X. Zhu, and J. Gao. Double rulings for information brokerage in sensor networks. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 286–297, New York, NY, USA, 2006. ACM.
- [29] E. Yoneki and J. Bacon. Content-based routing with on-demand multicast. In *Proceedings of the 24th Conference on Distributed Computing Systems Workshop (ICDCSW'04)*, 2004.