Middleware for Semantic Multicast in Spontaneous Multi-hop Networks

Paolo Bellavista and Carlo Giannelli

DISI – Università di Bologna
Viale Risorgimento 2, 40136 Bologna, Italy
{paolo.bellavista,carlo.giannelli}@unibo.it

Abstract. Spontaneous Multi-hop Networks (SMNs) are emerging as a novel networking and communication paradigm, strongly pushed by the widespread availability of smartphones equipped with heterogeneous wireless connectivity and powerful computing capabilities. SMN nodes can opportunistically exploit peer-to-peer contacts to seamlessly share resources/content in an impromptu and transient way. The paper presents a novel 3-layer modeling abstraction for multicast in SMNs, in order to characterize the different kinds of possible inter-node interaction based on different degrees of expressiveness and social-aware collaboration. In addition, we originally present the design and implementation of some novel semantic-based multicast mechanisms that efficiently target SMN nodes based on user interests and that are integrated into our SMN middleware solution. First preliminary results show the feasibility of the approach and its limited overhead.

Keywords: Spontaneous Multi-hop Networking, Multicast, Resource/Content Sharing, Smartphones, Middleware.

1 Introduction

Spontaneous networking is receiving growing attention for its promising aspects of better exploitation of available wireless connectivity, resource connectivity sharing, and immediate connectivity in regions with difficult coverage [1, 2]. The most relevant and specific property of Spontaneous Multi-hop Networks (SMNs) is that they are enabled by the willingness of social interaction and resource sharing via impromptu interconnection of people and their carried mobile personal devices, e.g., smartphones and tablets [3, 4]. In SMNs mobile devices should seamlessly discover and interact one another opportunistically and without any prior mutual knowledge, by exploiting any wireless opportunity available, e.g., Bluetooth ad-hoc links and Wi-Fi infrastructure-based ones. In particular, group-related behaviors and the ever increasing willingness to share rich user-generated contents, also pertaining to the personal sphere, call for a user-centric communication paradigm shift, where the ad-hoc interconnection of mobile devices in direct visibility plays a central role.

Sharing user-generated content (and, more in general, under-utilized resources) over SMNs requires new forms of node collaboration and communication, also

C. Borcea et al. (Eds.): MobilWare 2012, LNICST 65, pp. 45–61, 2013.

responding to new and/or extended paradigms, possibly always based on the standard substrate of universally available IP protocols for immediate deployability, but substantially enhancing their expressive power and effectiveness when applied to the novel SMN scenarios. In particular, we claim the primary importance of supporting a multiplicity of different multicast communication paradigms (e.g., based on a variety of mechanisms, from simple-to-manage and efficient syntax-based packet dispatching to more powerful and complex semantic-based discovery) at different layers of abstraction and at the same time, suitable for different application requirements.

To clarify the envisioned scenario by starting with some practical usage scenarios, let us consider the following examples of SMN collaboration, at different abstraction layers, aiming to discover and invoke a collaborative file sharing service (Figure 1):

1) in a simple and traditional deployment scenario, nodes are located in the same private IP subnet and can exchange data in a "direct" way, e.g., by exploiting UPnP to discover/advertise available services or SAMBA to expose local directories as if they were network drives;
2) in the second case, SMN nodes residing in two or more non-coordinated IP subnets (with possibly overlapping and conflicting IP addresses) are willing to collaborate by working together on dispatching packets from senders to receivers at a higher level of abstraction. For instance, node A may send discovery packets via local flooding in order to retrieve the nodes in its locality that host the file sharing service; neighbor nodes may cooperate by dispatching the request to remote nodes residing in other IP subnets; finally nodes offering the file sharing service may reply to node A, possibly by exploiting the chain of dispatchers used for service discovery. Note that in this case nodes can participate to many-to-many communications even if they are located in different private IP subnets with clashing addresses by performing packet dispatching at the application layer and by solving addressing/routing issues at this higher layer. However, if nodes have limited knowledge of their surrounding environment, discovery packets should be sent via flooding, with all the potentially connected limitations in terms of overhead and scalability;
3) in the third and most challenging/innovative scenario, we would like to have nodes (typically smartphones carried by users) enabled to maintain and share content related to their users. As a practical example, let us consider a semantically-enhanced and opportunistic file sharing service. Alice specifies that she is interested in music and skiing contents, Bob in skiing and gardening, and Cate in tennis. When discovering a file sharing service, Alice specifies that she is interested only in nodes whose users have common content interests, e.g., thus preventing from connecting to Cate's node and its offered contents. Of course, this requires mechanisms to proactively acquire additional knowledge about some SMN nodes, with the possibly associated costs in terms of overhead and scalability.

The paper presents a novel 3-layer multicast model for service discovery and content sharing in SMNs that clearly categorizes and describes the different mechanisms and opportunities available in the three scenarios rapidly described above. The three layers of the model are supported by an original middleware solution for SMNs that we

have designed and implemented. In particular, in this paper we originally focus on our solution for novel semantic-based multicast mechanisms (third layer), which represent the most novel and challenging case of multicast communication based on users' contextual metadata. The proposed solution exploits Semantic Web mechanisms to describe user characteristics and appropriately dispatch packets to most interested users. Thus, it performs packet delivery while completely decoupling senders and receivers, by focusing on users' characteristics rather than locations/addresses of their associated nodes.

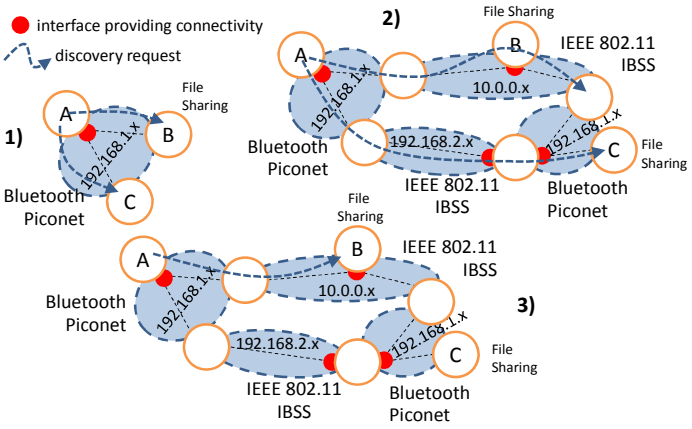

Fig. 1. Service discovery in SMNs at different layers of abstraction

The remainder of the paper is organized as follows: Section 2 details our novel and layered multicast model. Section 3 presents our original multicast mechanisms for SMNs based on semantic data, by describing the primary design/implementation choices we made in the realization of their prototype. Some preliminary performance results, followed by related work and conclusive remarks, end the paper.

2 A 3-Layer Multicast Model for SMNs

We identify three different layers of abstraction for multicast in SMNs, corresponding to three different possible communication overlays, each one characterized by a different degree of node collaboration and different definition of communication endpoints, in relation to both endpoint identifiers (how to specify the identity of a participating node) and endpoint addresses (how participating nodes can be reached by multicast communications). In particular:

1) **The Traditional IP** layer is based on IP addressing (both for endpoint identity and addressing) in private subnets where nodes interact one another directly. On the one hand, networking and local broadcasting issues are automatically solved by traditional IP-based solutions at layer3 and layer4 of the classical OSI stack.

On the other hand, there is the strong limitation that only nodes in the same private IP subnet can cooperate directly, e.g., in order to avoid address clashing between possibly overlapping IP namespaces;

2) **The Spontaneous Multi-hop** layer stems from the need of a multi-hop routing overlay in order to allow SMN nodes to dispatch packets from senders to receivers that do not reside in the same private IP subnet, by solving the associated identification and addressing issues. In this case, SMN nodes should be willing to collaborate more actively, not only in the case they are service endpoints: intermediary nodes have to offer a portion of their computing/communication resources to receive, manage, store, and forward traversing packets. Let us point out that SMNs based on users' cooperation usually originate from the opportunistic interconnection of private IP subnets in proximity [3]: users create layer2 links via multiple and possibly heterogeneous wireless interfaces; this usually leads to the configuration of IP parameters in an uncoordinated way, e.g., IP addresses are assigned autonomously by each node to its clients. As a consequence, multi-hop paths in SMNs have to exploit different single-hop IP networks, without a homogeneous address space (making unsuitable the exploitation of traditional IP-based identification and packet routing). As better detailed in the following section, the proposed spontaneous multi-hop layer solves endpoint identity and addressing issues by exploiting absolute node identifiers and subjective DSR-like multi-hop paths, respectively [5];
3) **The Semantic Dispatching** layer enables the delivery of multicast packets in a completely distributed way among loosely coupled endpoints. It does not require the sender to know its destination endpoints (neither identifiers nor addressing) when generating communication packets, because the dynamic determination of suitable endpoints is based on semantic data associated with the multicast packet and SMN nodes relationships. In other words, senders specify the characteristics of the endpoints that should receive the multicast communication rather than their identities or addresses. On the one hand, to enable this higher expressive power, node cooperation should be higher because nodes have to collaborate not only to dispatch packets, but also to agree on formats to describe users' interests and contents (and to disclose these data to participating nodes). On the other hand, this overlay potentially enables better exploitation of shared resources because it allows to multicast packets only to dynamically determined and really interested receivers. Let us notice that this layer has the notable positive side-effect of greatly facilitating the automatically filtered management of the ever increasing amount of reachable nodes (and their offered discoverable resources/services). For instance, users interested in retrieving only jazz music in a SMN could feel uncomfortable if they are forced to discover and access a large number of apparently similar instances of the same file sharing service, check all the corresponding lists of shared content, and manually identify the only jazz-related files; on the contrary, once users' contextual data are available, it is possible to prioritize the available file sharing instances, e.g., by first inquiring only the SMN nodes that belong to users who are fond of jazz music.

3 The Design of a Middleware Solution Implementing Our 3-Layer Multicast Model

We claim that content sharing in SMNs calls for the availability of middleware support with mechanisms at all the layers of our previously presented model: mechanisms at the three different layers should not be mutually exclusive but coexist, also in the same deployment environment; endpoints should dynamically adopt the layer best fitting their application requirements, by possibly benefitting from different layers even in different phases of the same interaction. For instance, the semantic dispatching layer can be exploited to discover the set of remote users sharing at least k topics of interest with the sender while, once identified a specific content, its delivery can use the traditional IP or the spontaneous multi-hop layers, e.g., depending on whether endpoints are in the same IP subnet or not.

Table 1. Concise summary of the properties of the introduced multicast layers

Layer	Scenario	Cooperation	Endpoint
Traditional IP	Basic layer, suitable by itself for static/administered/small networks	Service provisioning	IP address as both identifier and address
Spontaneous Multi-hop	Packet delivering in heterogeneous, contiguous, dynamic, and uncoordinated networks	Packet dispatching	nodeId as identifier (absolute), DSR-like IP sequence as address (relative)
Semantic Dispatching	Efficient service discovery in large spontaneous networks	Information sharing	Semantic-based: either direct (relative) or blind (absolute)

In the following, we recall very concisely the main characteristics and properties of the traditional IP layer and of the spontaneous multi-hop one (already described in the literature); on the contrary, we will go into the needed architecture and design detail about our novel middleware support for semantic dispatching multicast.

3.1 Traditional IP Layer

As well-known, the traditional IP layer identifies remote hosts via IP addresses and receiving processes via port numbers. Nodes residing in the same IP subnet easily communicate in a direct way, possibly exploiting native broadcast mechanisms. However, let us recall that for inter-subnet communications there is the need of managing routing tables, by updating them whenever nodes join, leave, or move. For this reason, the traditional IP layer is generally considered unsuitable by itself for the interconnection of SMN islands [1]. In fact, the self-organized, not explicitly administered, and volatile nature of SMNs pushes for novel solutions, not based on proactive configuration of network topology, but taking advantage of mission-oriented connectivity created among nodes that opportunistically collaborate to support their socially interacting users, e.g., to share personal pictures or transmit multimedia streams.

3.2 Spontaneous Multi-Hop Layer

One possible solution to support multicast communications at the spontaneous multi-hop layer is given by our Real Ad-hoc Multi-hop Peer-to-peer (RAMP) middleware, in particular by its RAMP Dispatcher component [6]. RAMP supports spontaneous multi-hop communication independently from how underlying (possibly heterogeneous) links/IP sub-networks have been autonomously and independently created. RAMP nodes cooperate at the middleware layer to dispatch packets, with no need to modify routing tables at the operating-system level, thus achieving the degree of dynamicity needed in SMNs [7].

On the one hand, our RAMP middleware supports a notion of endpoint different from traditional IP, by identifying remote nodes in terms of globally unique *nodeIds*. On the other hand, RAMP performs addressing in a DSR-like fashion, i.e., based on traditional IP addresses of intermediary nodes composing the path between senders and receivers. In this way RAMP distinguishes between identifiers (used to refer nodes) and addresses (used to reach nodes). In addition, while traditional IP addressing is absolute and shared among every node, RAMP addressing is relative to the sender, since different nodes may exploit different intermediaries to reach the same destination through different paths (composed by different and heterogeneous links).

3.3 Semantic Dispatching Layer

The semantic dispatching layer has the goal of completely decoupling senders and receivers, effectively supporting the abstraction of content-based multicast. In fact, it allows specifying endpoints based on shared contents and semantic similarity, i.e., by detailing receiver characteristics rather than its identifier or the path to reach it.

As a consequence, the communication semantic is inherently multicast, as relates to both destination nodes (multiple nodes may receive the same packet) and destination processes (multiple processes on the same node may receive the same packet). The idea is that the semantic dispatching middleware should be able to transparently manage packet exchange and to check whether a node should receive a packet or not, while application-level senders know neither the identities nor the addresses of their receivers. To this purpose, we propose a middleware solution that stores local user's data together with (a subset of) information about previously contacted remote users (partial local knowledge of SMN participants, opportunistically built at runtime based on launched queries). As better detailed in Section 4, to shorten the bootstrap phase and leverage the semantic-based discovery of remote users, collaborative nodes distribute partial knowledge about their spontaneous network by periodically broadcasting subsets of local user's data. In addition, it is worth noting from the beginning that the semantic dispatching layer should exploit the potential advantages of a cross-layer approach between user information and routing layers.

The rest of the section introduces our mechanisms to support semantic-based content delivery, namely, semantic multicast and semantic forward. The former is based on two novel communication primitives aiming at completely decoupling sender and receiver endpoints; the latter allows efficiently widening the scope of packet delivery

based on distributed awareness of the established SMNs. The main objective is to achieve a proper trade-off among delivery correctness (packets delivered only to interested nodes) and efficiency (in terms of both communication and processing overhead), by also considering specific requirements expressed at the sender side.

3.3.1 Semantic Multicast

Multicast senders define their sets of interested receivers, possibly with the fine granularity of the single packet, based on their criteria specification (see the following). The set of semantic multicast receivers may depend on the location where and the time when delivery criteria are checked. In fact, different SMN nodes may have very different runtime knowledge of other SMN participants, e.g., since a node may have joined the network before/after other nodes or may have interacted less/more frequently with neighbors. Also based on this observation, we have identified (and decided to support) two types of semantic multicast primitives, namely Direct Multicast and Blind Multicast, with different characteristics in terms of delivery correctness and communication overhead.

Direct Multicast is based on the concept of applying delivery criteria on senders. Very concisely, depending on node-related data collected by a sender, our middleware identifies the set of nodes the packet should be delivered to; then, packets are sent directly to destination nodes via unicast communication; finally, receiving nodes propagate the packet upward to the application layer, without performing any additional check/filtering operation.

Let us note that our Direct Multicast implements the above delivery semantics lazily, with no strict consistency. In fact, senders usually have incomplete and not up-to-date knowledge of remote nodes' data (e.g., about their preferences), it is not possible to ensure that only and all the nodes actually verifying the specified criteria will receive an associated packet. For instance, the sender could not be aware of the fact that a remote user has just added/removed "skiing" in her interest list. For this reason, Direct Multicast is unsuitable for scenarios with highly varying preferences or stringent correctness requirements. However, the associated computing/communication overhead is limited because criteria are checked only once on the sender-side and packets are directly delivered to locally-selected destination nodes.

Figure 2 depicts a simple and practical example of Direct Multicast. Only some nodes are semantically-enabled, i.e., manage and dispatch local and remote preference data (dashed circles). Node S sends a packet via Direct Multicast to nodes interested in "gardening" ("g" tag). The packet is delivered only to a subset of potential receivers, i.e., node X and node W; node Z does not receive the packet since it has just joined the network and not yet exchanged preference data with node S.

Blind Multicast is based on the idea of applying delivery criteria only on the receiver side. Criteria are attached to packets and delivered exploiting the RAMP Dispatcher broadcast mechanism, by flooding packets to SMN participants in a TTL-bound fashion. Nodes receiving these packets dispatch their content to local applications registered to receive multicast packets only if the specified criteria are locally verified.

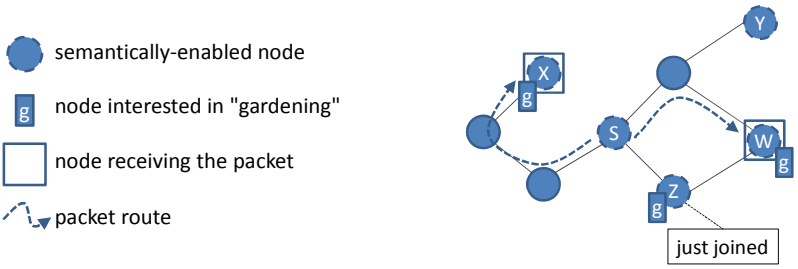

Fig. 2. Direct Multicast

Let us stress that, if compared with Direct Multicast, Blind Multicast can ensure a larger coverage of the multicast destination group. In fact, since each node has full and up-to-date knowledge of its own context (e.g., updated preference data), only and all applications running on top of nodes actually verifying the specified criteria at packet reception time will receive the packet. In addition, since packets are delivered by exploiting the RAMP Dispatcher broadcast mechanism [6], also nodes unknown by senders at packet sending time will receive the packet. However, the disadvantage is in i) packet delivery also to not interested SMN participants and ii) criteria checking needed at any node.

Figure 3 shows that Blind Multicast, if compared with Direct Multicast, also allows packet delivery to nodes the sender has not previously interacted with, e.g., node Z. Node S sends the packet to every nearby node, even if only a subset of them is actually interested in receiving it; of course, nodes outside the TTL boundaries are not interested by the blind multicast, e.g., node X.

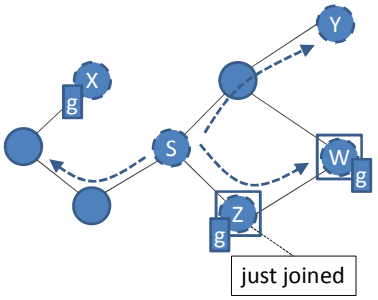

Fig. 3. Blind Multicast (with TTL=2)

3.3.2 Semantic Forward

Based on Direct and Blind Multicast primitives, our middleware also supports the capability of forwarding multicast packets in such a way to maximize the coverage of interested SMN nodes while minimizing communication overhead. The basic idea is that a subset of SMN nodes, which are particularly willing to collaborate, after receiving a multicast packet (either via Direct or Blind Multicast), not only propagate the payload to the local application layer but also re-transmit the packet to remote nodes.

Nodes receiving forwarded packets perceive them as if they were sent by original senders. Note that a node could forward a packet even if it is not interested in it, e.g., in the case of a Blind Multicast packet whose criteria are not locally verified.

To achieve a good tradeoff between coverage and limited overhead, we exploit the principle of locality: the primary assumption is that the closer a receiver node, the greater the interest of the sender that the node receives the packet. Based on this consideration, our solution forwards packets by exploiting Blind Multicast when close to senders, Direct Multicast when far from senders. As much as the distance from original senders increases, our solution adopts the following equation to decrease the probability to re-transmit packets via Blind Multicast:

$$BP = SBP \cdot ED^{\#FW} \qquad \text{if } \#FW \leq MF \qquad (1)$$

where **Blind Probability (BP)** in the [0,1] range is the probability forwarding nodes exploit the Blind Multicast mechanism to re-send packets, **Starting Blind Probability (SBP)** in the [0, 1] range is the BP value at the first forward, and **Exponential Decaying (ED)** in the]0, 1] range tunes how packets should be forwarded in the successive forwarding steps. At each forward, the middleware computes the BP value and exploits Blind Multicast if a randomly generated value in the [0, 1] range is lower than or equal to BP, Direct Multicast otherwise. Moreover, retransmissions are inhibited if the amount of forwards has reached the **Max Forwards (MF)** value.

Based on (1) and as a general consideration, our solution adopts Blind Multicast more probably in initial forwards and Direct Multicast in the following ones. Thus, it achieves the notable effect of disseminating information with a decreasing overhead and correctness gradient, i.e., the greater the distance from the original sender, the lower the imposed overhead and the lower the probability that interested nodes receive the packet. Application developers can tune the behavior of the forwarding mechanism by appropriately setting MF, SBP, and ED values, in a fine-grained and per packet way. In particular,

- the greater the SBP value, the greater the communication overhead; on the contrary, the lower the SBP value, the lower the probability to reach far nodes interested in the packet. For instance, if SBP is equal to 1, the first forward is always performed in a Blind way, while if SBP is equal to 0, forwards are always performed according to the Direct way;
- the greater the ED value, the slower our middleware switches from Blind to Direct mechanisms. ED equal to 1 means that BP is always equal to SBP, ED equal to 0.5 means that at each forward the BP value halves, ED value equal to 0.1 means that at each forward the BP value is 1/10 of the previous forward step.

Packet forwarding is performed either if packets are sent via Direct Multicast and the local nodes are receivers or if packets are sent via Blind Multicast and TTL is equal to 0. In other words, in case of Direct Multicast only actual receivers can forward packets, while in case of Blind Multicast only last receivers can forward them.

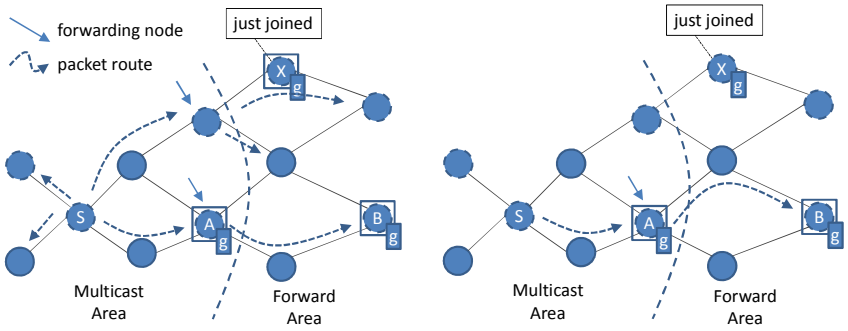

Fig. 4. Packet Forward based on Blind (left) and Direct (right) Multicast

Figure 4 provides two examples at the two extremes in the range of possible cases. On the left, node S performs a Blind Multicast with TTL = 2 and SBP = 1; on the right, node S performs a Direct Multicast with SBP = 0 (nodes dispatch local interests to neighbors at 2-hop distance); MF and ED are ignored in the depicted scenario for the sake of clarity. In the former case, the packet is correctly delivered to every node with interest in it, but at the cost of transmitting the packet also to additional nodes; in the latter case, traffic overhead is lower, but node X does not receive the packet.

3.3.3 Semantic Dispatching Information Management

To support our solution for interest/content matching, senders and receivers must adopt a common vocabulary. To this purpose, we adopt simple and reasonably lightweight Semantic Web mechanisms to store and manage data: user preference data are stored as Resource Description Framework (RDF) graphs, while packet delivery criteria are implemented as SPARQL Protocol and RDF Query Language (SPARQL) queries. We have implemented query and application examples based on the Friend Of A Friend (FOAF) vocabulary. By adopting a specific ontology, on the one hand, we demonstrate the feasibility of our approach and provide final users with ready-to-use examples (see below); on the other hand, we encourage middleware extension and refinement by providing developers with a template on how to, for instance, include a wider set of queries and adopt additional ontologies. Let us note that anyway the proposed multicast and forward mechanisms are independent from the selected Semantic Web solutions for preference management and query representation.

For the sake of clearness, consider the FOAF document below about Alice, identified by her email (`foaf:maker`), specifying that she is interested in skiing&music (`foaf:topic_interest`) and knows Bob (`foaf:knows`).

```
<?xml version = "1.0"?>
<rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#" [...]
    <foaf:PersonalProfileDocument rdf:about = "" >
        <foaf:maker rdf:resource = "mailto:alice@example.org" />
        <foaf:primaryTopic rdf:resource = "mailto:alice@example.org" />
        <foaf:name>Profile of Alice</foaf:name>
```

```
    </foaf:PersonalProfileDocument>
    <foaf:Person rdf:about = "mailto:alice@example.org" >
        <foaf:name>Alice</foaf:name>
        <foaf:topic_interest>Skiing</foaf:topic_interest>
        <foaf:topic_interest>Music</foaf:topic_interest>
        <foaf:knows>
           <foaf:Person rdf:about= "mailto:bob@example.org" >
              <foaf:name>Bob</foaf:name>
           </foaf:Person>
        </foaf:knows>
        ...
    </foaf:Person>
</rdf:RDF>
```

Each SMN node can store multiple graphs, one graph for the FOAF document of each local user (typically one only graph and user, especially for smartphones), other graphs for FOAF documents of remote users (one different graph for each user). As better detailed in Section 4, when nodes opportunistically interact one another, they exchange a subset of their graphs based on relationships among users. Note that graphs related to remote users provide only a partial (and possibly not up-to-date) view of the remote user information, since they contain only RDF triples exchanged depending on previous node interactions. Triples of the local graph are tagged in order to specify different visibility rules; we currently support three sets:

1) *known people (KP)*, available to the set of users directly known by the local users;
2) *known people plus people known by known people (KP+)*, available to the previous set plus the social contacts of people known by the local user;
3) *public*, available to everyone.

Each node contains specific rules to define KP and KP+ sets in relation to the local user. For instance, considering the previous FOAF document, possible rules are:

```
[ knownPersonRule: ( mailto:alice@example.org foaf:knows ?p ),
       notEqual( mailto:alice@example.org, ?p )
       -> ( ?p rdf:type ramp:KnownPerson )]
[ knownByKnownPersonRule: ( mailto:alice@example.org foaf:knows ?p ),
       notEqual( mailto:alice@example.org, ?p ),
       ( ?y foaf:knows ?z ), notEqual( ?y, ?z ),
       notEqual( mailto:alice@example.org, ?z )
       -> ( ?z rdf:type ramp:KnownByKnownPerson ) ]
```

where `KnownPerson` and `KnownByKnownPerson` are additional Ontology Web Language (OWL) classes defined to create the `KnownPersonAndKnownByKnownPerson` as union of type collection of `KnownPerson` and `KnownByKnownPerson` classes.

SPARQL queries of Construct type (providing sub-graphs as results) are exploited to create a view of the local graph that fits the visibility rules that are dynamically considered suitable for a given remote user. For instance, the SPARQL query "Privacy Filter" below creates a graph including Alice's data by considering Bob's visibility rules.

```
CONSTRUCT {
   mailto:alice@example.org ?prop ?obj.
   WHERE {
      <mailto:alice@example.org> ?prop ?obj.
      ?privacyRule ramp:onPerson <mailto:alice@example.org>.
      ?privacyRule ramp:onProperty ?prop.
      ?privacyRule ramp:permittedRole ?class.
      <mailto:bob@example.org> a ?class.
   }
}
```

Finally, Blind and Direct Multicast are performed by exploiting SPARQL queries of Ask type (providing true/false values as results), the former on senders, the latter on receivers. For instance, the query below is attached to a Blind Multicast packet to specify that the payload should be propagated at the application layer only if the local user is interested in Music.

```
ASK {
    ?ppd a foaf:PersonalProfileDocument.
    ?ppd foaf:primaryTopic ?user.
    ?ppd foaf:maker ?user.
    ?user foaf:topic_interest Music.
}
```

4 Design/Implementation Insights and Preliminary Experimental Evaluation

Based on our multicast model and the design guidelines presented in the previous section, we have implemented a middleware prototype based on two primary layers: the Communication layer and the Semantic layer (see Figure 5). The **Communication layer** exploits the "traditional" RAMP solution to send/receive unicast and broadcast packets in SMNs and to advertise/discover the set of locally/remotely available services [7]. The **Semantic layer** includes novel middleware components to support the dynamic management and dispatching of user preferences and to provide application developers with API i) to receive semantically-enabled packets, ii) to perform Direct/Blind Multicast, and iii) to enable/disable our Semantic Forward mechanism.

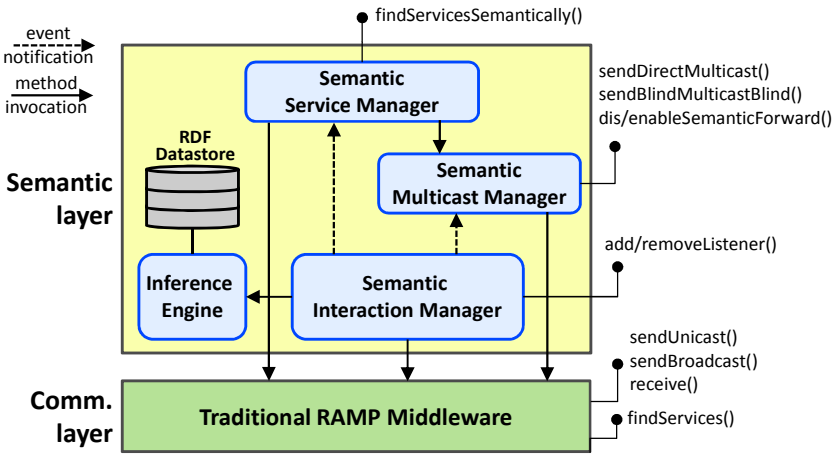

Fig. 5. The component architecture of our semantic multicast prototype

RDF Datastore and **Inference Engine** exploit the Jena framework to store and manage RDF triples [8]. The RDF datastore manages multiple named graphs, each one related to a remote user, and a default graph, related to the local user. The Inference Engine is in charge of applying rules to infer KP and KP+ sets based on local information, creating sub-graphs via Construct SPARQL queries and verifying Ask SPARQL queries.

Semantic Interaction Manager is in charge of interacting with remote users to spread preference data related to the local user and gather analogous data by remote users. In particular, each node periodically sends a so-called *social beam* message as a RAMP broadcast packet (default TTL and period values of 3 hops and 30s respectively), containing the local node unique identifier; for instance, in the case of FOAF vocabulary, it is possible to send the `foaf:maker` value. Then, nodes receiving the social beam message reply via unicast providing the portion of the graph of the local node, based on the original sender node visibility and by exploiting the Privacy Filter SPARQL query. Moreover, Semantic Interaction Manager provides the capability of receiving packets in an event-based way, via the registration of packet listeners implementing the `ISemanticListener` interface below.

```
void addListener(ISemanticListener listener) ;
void removeListener(ISemanticListener listener) ;

interface ISemanticListener {
   onEvent(LocalProfileUpdateEvent evt);
   onEvent(RemoteProfileUpdateEvent evt);
   onEvent(RemoteProfileRemoveEvent evt);
   onEvent(MulticastMessageReceivedEvent evt);
   onEvent(UnicastMessageReceivedEvent evt);
}
```

Finally, it is worth noting that Semantic Interaction Manager represents the basic mechanism to share social information (such as social-aware preferences) among nodes; however, SMN nodes can also exchange this kind of data with other mechanisms, e.g., applications running on top of the Semantic layer, which can populate and enrich RDF stores with additional metadata.

Semantic Multicast Manager is the component actually supporting Direct/Blind Multicast and Semantic Forward mechanisms. In particular, Semantic Multicast Manager offers the following methods

```
sendDirectMulticast(pattern, forwardParameters, payload);
sendBlindMulticast(ttl, pattern, forwardParameters, payload);
enableSemanticForward();
disableSemanticForward();
```

to support Direct/Blind Multicast and enable/disable Semantic Forward mechanisms respectively. Both Direct and Blind Multicast methods require a SPARQL query of Ask type, a byte array payload, and (optionally) Semantic Forward parameters, i.e., SBP, ED, and MF. In the case of Direct Multicast, the SPARQL query is run locally and then the payload sent via multiple unicast packets, one for each locally stored named graph verifying the query. In the Blind case, the SPARQL query is sent within the packet payload, via RAMP broadcast at `ttl` maximum distance.

Semantic Service Manager exploits Semantic Multicast Manager to support the discovery of services hosted on remote nodes. Similarly to Semantic Multicast Manager, Semantic Service Manager provides two different methods for service discovery exploiting either Direct or Blind Multicast. In the former case, the middleware sends discovery requests via unicast to nodes only if their locally stored named graph verifies the SPARQL query. In the latter case, the SPARQL query is sent via broadcast together with the service name and receivers check if the required service is available; only in the positive case, they reply to the sender.

```
findServicesDirect(pattern, forwardPar, serviceAmount, serviceName);
findServicesBlind(ttl, pattern, forwardPar, serviceAmount, serviceName);
```

We have performed some first tests over real testbeds and measured first quantitative performance results of our middleware implementation, with the main aim of validating our Semantic layer and of comparing the performance of "traditional" RAMP communications with semantically-enabled ones. For the sake of briefness, here we focus on the results that show how the adoption of semantic multicast can improve the quality of the discovery process perceived by final users, while imposing very little overhead. First of all, our semantic multicast reduces the set of nodes involved in packet exchange because it allows the dynamic retrieval of only the data that final users are interested in. To provide a quantitative example, consider the case of a user who is fond of jazz music and is looking for a related File Sharing service. The target SMN consists of N nodes (plus the sender), FS (< N) nodes providing a generic File Sharing service, J (< N) users interested in jazz; JFS is the intersection of J and FS nodes. In case of no semantic multicast, in RAMP we would be forced to have:

- 1 flooding-based service discovery to retrieve FS nodes, involving N nodes;
- FS responses sent by nodes hosting the service;
- FS requests from the client to service replicas to gather the list of shared files;
- FS responses with file list to the client.

Instead, in case of Direct/Blind Multicast there are (for simplicity, suppose that the client's local list of nodes interested in jazz music is complete):

- J unicast/1 flooding-based service discovery, involving J/N nodes;
- JFS responses;
- JFS requests of shared file list;
- JFS responses with the file list.

In short, our semantic-based solution can take advantage of (partial) knowledge of interest heterogeneity of socially interacting users to filter out useless discovery traffic and to limit the associated overhead in a probabilistic way.

Secondly, we have collected results to quantitatively show that the usage of carefully selected and lightweight Semantic Web techniques does not affect too much the multicast overhead and the time required to retrieve data, at least in SMNs. Table 2 reports about the time required to discover a remote service in case of "traditional" RAMP-based service discovery and our novel semantic-enabled discovery based on Blind Multicast. The reported results are obtained while varying path length and RDF dataset size; the employed small/medium/large datasets contain 10/250/750 `foaf:knows` and 10/50/200 `foaf:topic_interest` relationships respectively. The table shows that our semantic-enabled discovery increases latency but only linearly in relation to both path length (IEEE 802.11b ad-hoc links) and RDF dataset size (the greater the dataset, the more time required to run SPARQL queries). It is important to note that, even in the challenging case of three wireless hops and large RDF datasets composed of hundreds of entries, our implementation of the semantic-based discovery gets a response in less than 0.3s, thus demonstrating the practical applicability of the approach in all the application domains of interest for SMNs and the good efficiency of our prototype implementation.

Table 2. Latency of our semantic-enabled multicast discovery

		Path length (#hops)		
Service Discovery		1	2	3
Traditional		0.03 s	0.06 s	0.07 s
Semantically enabled	small dataset	0.06 s	0.13 s	0.18 s
	medium dataset	0.06 s	0.14 s	0.21 s
	large dataset	0.07 s	0.15 s	0.28 s

5 Related Work

Consolidated literature about context-aware middleware include some interesting and relevant solutions to support interest-based group communication primitives [9].

More recently, the use of semantic information to improve final user satisfaction has gained growing attention, also pushed by increased availability of shared user generated content associated with semantic tags. For instance, [10] combines ontology-based solutions with information gathered by tagging mechanisms typical of social networks, in order to provide a semantically enabled recommendation system. Instead, [11] supports a distributed social network based on recommendation structures implemented as RDF graphs. In particular, it supports the spread of data and resources based on semantically rich information stored in FOAF documents. Even the exploitation of semantic information in mobile environments is receiving growing attention. For instance, the Yarta middleware considers the heterogeneity of mobile nodes and data adopting RDF triples to store and spread semantic information [12]. In this manner application developers can easily share information and create/delete semantic-based inter-user social relationships.

Focusing on semantic multicast, OntoNet supports flexible and scalable packet delivery in emergency scenarios on top of mobile ad-hoc networks [13]. To efficiently propagate messages, OntoNet adopts tree-shaped topologies and perform multi-query aggregation. OntSum aims at discovering desirable resources based on semantically rich information, exploiting heterogeneous ontologies [14]. To maximize scalability, inter-node topology is dynamically reconfigured to make nodes with similar ontologies close one another, thus creating different ontology-based clusters. In this manner OntSum provides a concise index to efficiently route queries towards the right location, i.e., close to nodes satisfying query constraints. Instead, MobiSN adopts ontologies to spread information along participants of mobile social networks [15]. In particular, the proposed solution forwards discovery packets based on semantic information among one-hop distant nodes: the main goal is to select the best node towards the packet should be forwarded to.

Finally, it is worth noting that our definition of multicast is similar to multi-hop content-based pub-sub communication [16], but without a sharp distinction among subscribers, brokers and broker network. In fact, the Semantic Dispatching layer efficiently supports advanced forms of service discovery specifically designed for innovative and challenging SMN environments.

6 Conclusions

The originally proposed 3-layer multicast model points out the opportunities opened by different forms of node collaboration, at different levels of abstraction, in SMN environments in order to enhance advanced forms of communications, e.g., by completely decoupling packet senders and receivers. Designing and implementing middleware solutions that follow the proposed multicast model can also permit to improve the quality of experience and satisfaction of mass-market final users, e.g., by focusing user attention only on discoverable resources that provide content of most probable interest. First performance considerations and achieved results confirm that the proposed solution can decrease the number of SMN participants uselessly

involved in discovery thanks to semantic-based filtering, thus increasing overall scalability, at the same time while imposing very limited overhead.

The encouraging results achieved up to now are stimulating our further research activities, on the one hand, on the integration with widespread social networking applications via emerging standard APIs, on the other hand, on building trust estimations based on past interactions (stability of collaborations, previous mobility patterns, willingness to offer local resources, etc.).

References

1. Ferreira, L.S., De Amorim, M.D., Iannone, L., Berlemann, L., Correia, L.M.: Opportunistic Management of Spontaneous and Heterogeneous Wireless Mesh Networks. IEEE Wireless Comm. 17(2), 41–46 (2010)
2. de Amorim, M.D., Ziviani, A., Viniotis, Y., Tassiulas, L.: Special Issue on Practical Aspects of Mobility in Wireless Self-organizing Networks. IEEE Wireless Comm. 15(6) (2008)
3. Feeney, L.M., Ahlgren, B., Westerlund, A.: Spontaneous Networking: an Application Oriented Approach to Ad Hoc Networking. IEEE Comm. Mag. 39(6), 176–181 (2001)
4. Latvakoski, J., Pakkala, D., Paakkonen, P.: A Communication Architecture for Spontaneous Systems. IEEE Wireless Comm. 11(3), 36–42 (2004)
5. Johnson, D.B., Maltz, D.A., Broch, J.: DSR: The Dynamic Source Routing protocol for multi-hop wireless ad hoc networks. In: Perkins, C.E. (ed.) Ad Hoc Networking, ch. 5, pp. 139–172. Addison-Wesley (2001)
6. Bellavista, P., Corradi, A., Giannelli, C.: The Real Ad-hoc Multi-hop Peer-to-peer (RAMP) Middleware: an Easy-to-use Support for Spontaneous Networking. In: 15th IEEE Symp. on Computers and Communications (ISCC 2010), Rimini, Italy (2010)
7. Bellavista, P., Corradi, A., Giannelli, C.: Application-Driven Management Middleware for Differentiated Service Provisioning in Spontaneous Networks. IEEE Pervasive Computing (in press), doi:10.1109/MPRV.2011.59
8. McBride, B.: Jena: a Semantic Web toolkit. IEEE Internet Computing 6(6), 55–59 (2002)
9. Yau, S.S., Karim, F., Wang, Y., Wang, B., Gupta, S.K.S.: Reconfigurable Context-sensitive Middleware for Pervasive Computing. IEEE Pervasive Computing 1(3), 33–40 (2002)
10. Passant, R., Raimond, Y.: Combining Social Music and Semantic Web for music-related recommender systems. Social Data on the Web (2008)
11. Ghita, S., Nejdl, W., Paiu, R.: Semantically Rich Recommendations in Social Networks for Sharing, Exchanging and Ranking Semantic Context. In: Proc. of the 4th International Conference on The Semantic Web Pages, pp. 293–307 (2005)
12. Toninelli, A., Pathak, A., Issarny, V.: Yarta: A Middleware for Managing Mobile Social Ecosystems. In: Riekki, J., Ylianttila, M., Guo, M. (eds.) GPC 2011. LNCS, vol. 6646, pp. 209–220. Springer, Heidelberg (2011)
13. Kopena, J.B.: Boon Thau Loo: OntoNet: Scalable Knowledge-based Networking. In: IEEE 24th Int. Conf. on Data Engineering Workshop, pp. 170–175 (2008)
14. Li, J., Vuong, S.: OntSum: A Semantic Query Routing Scheme in P2P Networks Based on Concise Ontology Indexing. In: 21st Int. Conf. on Advanced Information Networking and Applications, pp. 94–101 (2007)
15. Li, J., Khan, S.U.: MobiSN: Semantics-Based Mobile Ad Hoc Social Network Framework. In: IEEE Global Telecommunications Conference, pp. 1–6 (2009)
16. Martins, J.L., Duarte, S.: Routing Algorithms for Content-based Publish/Subscribe Systems. IEEE Communications Surveys & Tutorials 12(1), 39–58 (2010)