Social-aware Differentiated Visibility of Home-to-Home Shared Resources in Spontaneous Networks

Paolo Bellavista, Carlo Giannelli, Stefano Poli
CIRI-ICT & DISI – University of Bologna – Italy
{paolo.bellavista, carlo.giannelli, stefano.poli6}@unibo.it

Abstract — **The widespread adoption of online social networking to upload and share user-generated personal content rises novel issues related to the management of content ownership and privacy. To retain content ownership, we have previously designed and implemented an original solution for social-driven content sharing in home-to-home federated and spontaneous networks. This paper relevantly enhances our solution by proposing novel mechanisms to tune the visibility of shared resources based on the dynamic evaluation of social relationship tightness, inferred through social data gathered from widespread online social networks. On the one hand, the proposal autonomously evaluates the tightness of social relationships based on the primary guideline that the more users interact the tighter their relationships are. Moreover, it determines a default decision tree suitable for many application scenarios and enables its dynamic personalization based on user's feedback. On the other hand, it defines and supports a grammar to define visibility filters: runtime visibility of shared resources is automatically tuned based on both relationship tightness and defined filters. The presented prototype demonstrates how to effectively design/implement the proposal and the feasibility of our approach in terms of performance.**

Keywords: spontaneous networking, social networking, resource sharing, visibility filtering, relationship tightness.

I. INTRODUCTION

The daily digital experience of users is increasingly characterized by the exploitation of multiple Online Social Networks (OSNs) to remotely interact by exploiting a wide and heterogeneous set of mechanisms. For instance, Facebook is massively used to exchange messages among users both in a public and private manner (posting on friends' walls and sending private messages respectively), while Twitter is mostly exploited to write short messages about specific topics (hashtag mechanism) and get updates about the activity of other users (follower mechanism). Furthermore, OSNs are widely adopted also as platforms to share resources. For instance, users frequently upload personal pictures and videos on Facebook and share them with friends, eventually commenting upon their own and other's content. On the opposite, Twitter is mainly exploited to advertise the location of content, e.g., twitting the URL of a news article or of a YouTube video.

Considering the diffusion of OSNs, it is of primary importance to ensure users' privacy providing suitable policies and easy-to-use tools to actually enforce privacy management. Also pushed by governments and users [1], OSNs have introduced mechanisms to differentiate the visibility of information and resources. For instance, Facebook allows to put user's contacts in different sets, such as close friends, colleagues, and acquaintances; then, it is possible to finely tune the visibility of posts and resources by explicitly specifying which sets of users should be able to access them. However, users do not seem to widely exploit this filtering mechanism and often rely on default visibility filters, either because of the time needed to explicitly categorize their relationships and their visibility filters for shared resources, or because they do not know the availability of this mechanism at all. Moreover, Twitter has the default behavior of accepting new followers without any explicit agreement of users. Despite the adopted OSN is either Facebook or Twitter, less careful users could share posts and resources with a much wider set of people they had actually intended to, by inadvertently violating their own privacy desiderata. In addition, industrial and academic researchers have recently focused their interests on how to manage ownership of contents shared on OSNs [2-4]. In fact, users are currently used to lose (at least part of) control on their contents when they upload them on OSNs. Eventually, data are migrated to locations/countries with laws not completely ensuring users' privacy, while profiling algorithms are commonly employed to dig users' content and prompt personalized advertisements. Moreover, users cannot be sure that their data are actually removed once they delete them or close their accounts.

We have recently proposed an original middleware allowing to share content in a social-driven way while fully preserving content ownership [5]. Our solution gathers and exploits social identities and relationships to dynamically interconnect home subnets of users linked by social relationships. Thus, in another perspective, we exploit social data gathered from traditional OSNs to enable automatic content sharing in a peer-to-peer way. However, the shared content is directly sent from the device storing it to the device requesting it, without any storage support by third parties (see background in Section II).

While providing an effective and easy-to-use mechanism to share resources and to maintain content ownership, our solution inherits the visibility limitations of OSNs themselves. Since home-to-home interconnections are based on OSN social relationships, it may be difficult to suitably tune inter-home resource access, also considering that most of the times users specify only one type of relationship, e.g., Facebook friendship and Twitter follower/following. For instance, to tune the visibility of a DLNA Media Server offered by a Network-Attached Storage (NAS), users would be able to only specify that their Facebook friends can access it (while Twitter followers not, for instance), while they should be able to explicitly categorize their relationships one by one to get finer-grained visibility. Moreover, even users more concerned by privacy issues and categorizing their relationships in different sets may find difficult and time-consuming to appropriately associate suitable visibility rules to every shared resource.

The paper proposes a novel solution that dynamically evaluates and exploits the tightness of users' relationships to make easier the management of their content sharing. The access to shared resources is supervised by portable visibility filters distinguishing relationships as "tight", "regular", and "loose", independently from the type of OSN or the specified relationships. In particular, the proposed solution is original in:

a) autonomously **evaluating the tightness of social relationships** based on how much and how frequently users have socially interacted in the past. The primary simplifying guideline is that the more two users interact, the tighter their relationship is (more sophisticated strategies are out of the scope of this paper and can be easily integrated in our framework). Our solution generates a decision tree taking into consideration several parameters of OSNs, e.g., the amount of exchanged private messages and pictures where both users are tagged (Facebook) and the number of re-tweets (Twitter);

b) supporting a **default decision tree** suitable for most of the users as well as **its dynamic personalization based on user's feedback**. On the one hand, we have developed a default model based on an explicit user survey to achieve a coarse-grained ready-to-use decision tree. On the other hand, users can easily refine the default model by specifying the tightness of (part of) their relationships used to generate a highly personalized decision tree;

c) providing a **well-defined grammar to specify high-level filters**. The proposed grammar models shared resource consumption into three phases, i.e., discovery, browsing, and access, allowing to **filter with per-phase per-service granularity**. Moreover, users can define filters with **differentiated granularity levels**, by selecting the most appropriate granularity based on their requirements, skills, and willingness. On the one hand, our solution allows to specify portable filters based on relationship tightness, thus abstracting from the specific characteristics of the used OSN or type of relationships it supports. On the other hand, it enables the definition of ad-hoc filters, specifically designed for a given OSN or relationship;

d) performing **runtime visibility tuning of shared resources** based on dynamically evaluated social tightness and applicable filters, by efficiently monitoring ingress/egress packets, by modifying their content, or even by dropping them. In particular, our solution adopts a whitelist approach, hiding resources as long as there is not a specific rule allowing them.

Let us highlight that our middleware can easily apply to heterogeneous OSNs and shared services. For instance, our current prototype i) integrates with Facebook and Twitter by exploiting their data and relationships, and ii) supports filters for both legacy DLNA Media Servers and our built-in File Sharing service; as better detailed in the following, it can be easily extended to integrate with additional OSNs and with other sharing services.

The rest of the paper is structured as follows: Section II briefly describes our previous work, by detailing how our middleware enables social-driven home-to-home spontaneous networking and peer-to-peer resource sharing. Section III presents our original mechanisms to dynamically evaluate the tightness of social relationships, while Section IV describes our model of service discovery/invocation and our filter specification grammar. Sections V and VI provide details about our prototype implementation and its performance, respectively; related work and conclusions end the paper.

II. RAMP Middleware for Social-driven Home-to-home Resource Sharing

Our middleware supports home-to-home resource sharing based on the creation of User Centered Networks (UCNs) and their automatic federation [5]. We start from UCNs that are personal overlays that tightly interconnect devices owned by a unique social identity. Devices are typically located in different physical networks and are virtually interconnected to easily support full sharing of data belonging to the same user. Then, we enable UCN federations that represent the dynamic and loosely interconnection of UCNs associated with different social identities and linked by social relationships.

Figure 1 shows how UCNs and their federation can be easily exploited to share content without third-party infrastructures. Cate associates her Internet-connected devices to her Facebook profile, thus automatically generating a UCN composed of her tablet, gateway, and NAS. Now Cate's tablet can effortlessly upload pictures directly to her NAS Web server via HTTP, even if the latter resides in her private home LAN. Furthermore, since Alice and Cate are friends on Facebook, their UCNs are federated, thus supporting to browse DLNA AV Media Server content stored in Cate's NAS from Alice's smart TV as if they were in the same IP subnet, by exploiting legacy mass-market solutions based on standard UPnP. To this purpose, we have developed a UPnP Proxy supporting the transparent dispatching of UPnP messages to remote nodes located at multi-hop distance, with no need of additional hardware/software components on legacy UPnP devices. Therefore, we can overcome the traditional UPnP limitation of interconnecting devices only if located in the same subnet [6].

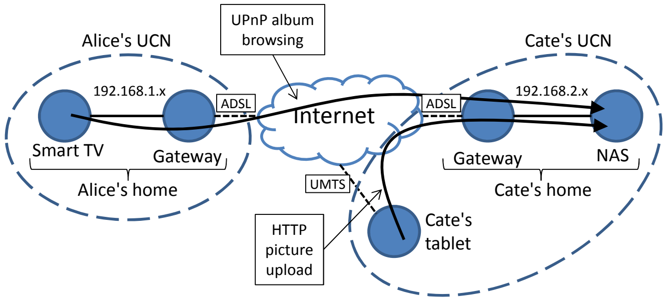

Figure 1. Example of federated UCNs.

The previous version of our middleware also supported a basic and coarse-grained filtering mechanism. For instance, it allowed specifying visibility rules with per-device granularity, allowing to hide services offered by a given node. In addition, it supported per-relationship filters, requiring users to specify rules for each social relationship of each OSN they are registered to. This paper relevantly advances our previous solution, by supporting a finer-grained filtering mechanism allowing to differentiate among discovery, browsing, and access phases, and supporting the definition of articulated rules based on social relationship tightness. Thus, the user experience is largely improved and the definition of portable cross-OSN visibility rules is made relevantly easier.

Let us stress that the target scenario is characterized by devices located in different IP subnets managed in a completely decentralized and uncoordinated way, e.g., NAS/desktops in home LANs of different users together with mobile terminals connected to the Internet via operator-managed UMTS networks. In fact, our middleware takes advantage of spontaneous networks [7] composed of different private subnets with heterogeneous and possibly conflicting IP address spaces. To this purpose, we rely on our Real Ad-hoc Multi-hop Peer-to-peer (RAMP) framework, supporting the seamless dispatching of packets among collaborative nodes (http://lia.deis.unibo.it/ Research/RAMP/). A detailed and general description of RAMP is out of the scope of the paper (see [6, 8, 9]); in this section we simply provide a short description of primary RAMP facilities/characteristics to permit the full understanding of the original extensions proposed here.

RAMP provides application developers with APIs to support fast prototyping of collaborative applications over spontaneous networks. It supports end-to-end unicast/broadcast over heterogeneous multi-hop paths, by exploiting nodes willing to share their local resources and by estimating sharing opportunities via novel context indicators, such as joint mobility [9, 10]. In addition, it supports peer-to-peer collaborative services via registration, advertising, and discovery mechanisms. For instance, on top of RAMP API we have implemented the File Sharing service. On the server side (node offering the File Sharing service), to make the new service available to RAMP nodes, there is only the need to register it locally (`registerService` method) and simply wait for requests (`receive` method). In our implementation, a File Sharing request may be for either the list of available files or the download of a given file. In both cases the File Sharing service replies directly to the requester (`sendUnicast` method), with no need to manage the underlying network heterogeneity and complexity (transparently provided by our middleware). On the client side, there is only the need for our File Sharing client to discover remote services (`findServices` method); once determined the node offering the service, the client can simply require the list of shared files or the content of a given file (`sendUnicast` and receive methods).

RAMP operates connectionless, mission-oriented, and middleware-layer routing: RAMP packets include the ordered set of IP addresses they must traverse to reach their destinations, in a Dynamic Source Routing-like way [11]. Thus, intermediary nodes can dynamically and flexibly forward data among uncoordinated IP subnets based on fast packet header evaluation (similarly to what happens, at a different abstraction layer, in MPLS-based solutions [12]). In addition, intermediary nodes can also actively monitor traversing packets, with no additional communication overhead (e.g., see the Visibility Tuner component in the following). Moreover, our middleware-layer routing better copes with the high dynamicity of spontaneous networks, e.g., by permitting to quickly discover and configure new paths in case of nodes leaving the spontaneous network and breaking a path under exploitation. Finally, working at the middleware layer simplifies portability and rapid deployment over heterogeneous operating systems and nodes, which is crucial for fast industrial deployability.

III. Building a Portable Cross-OSN Model of Inter-user Social Relationship Tightness

As anticipated in Section I, the wide spread of OSNs and their ubiquitous pervasive exploitation by final users push for addressing the open issues related to privacy management, in particular to regulate access to shared user-generated content. Let us point out that in our target scenario the proper, easy, and fine-grained management of users' privacy is even more important. In fact, the automatic federation of UCNs related to different users, only because they are linked by social relationships, potentially enables large amounts of remote users' accessible contents. Our solution guideline is to categorize social relationships based on the dynamic evaluation of their tightness. The main idea is that if two users interact very frequently, it is likely that their relationship is tighter and thus they are willing to share a wider set of their user-generated contents. The rationale is that a malicious attacker could even successfully obtain a friendship relationship confirmation impersonating a user's friend, e.g., based on a fake Facebook account. Therefore, our middleware solution takes in account not only the identity, but also the observation of the kind of runtime interaction among identified entities. In other words, to get access to one user's data, the attacker should be able to increase the tightness of the associated social relationship, thus requiring much more sophisticated social attacks.

In particular, our solution tunes the runtime enforced visibility of shared resources based on dynamically evaluated tightness based on inter-user communication patterns (see Section IV). Let us note that it achieves a trade-off among the capability of finely-tuning visibility with per-user granularity and easily filtering resources in an automatic manner. On the one hand, users are typically unwilling to perform time-consuming manual configuration processes: solutions requiring users to explicitly specify per-user visibility rules are very rarely successful in terms of wide adoption and usage. On the other hand, our solution allows to automatically categorize the tightness of relationships and apply filtering rules based on this information, thus leveraging final users from the burden of manually tuning visibility. In addition, as better detailed in Section V, we allow users tuning the algorithm to categorize relationships, e.g., to manually modify the automatically determined tightness of (part of) their relationships.

Moreover, since different OSNs typically adopt very different communication mechanisms and tools, we support the generation of per-OSN models taking into consideration the specific peculiarities of the integrated OSNs. Currently our prototype integrates (and has been validated) with Facebook and Twitter, but in this way it can be easily extended to other OSNs. However, it is not trivial to identify which communication tool(s) should be monitored to automatically infer relationship tightness. On the one hand, OSNs provide several and heterogeneous mechanisms to interact, e.g., Facebook public posts/private messages and Twitter re-tweet/citations. In addition, the tightness of relationships could be evaluated also by considering other social metadata and data, e.g., the number of pictures two users are tagged in together or the amount of shared follower/following users. On the other hand, users may have dramatically different OSN usage patterns, ranging from users writing posts on friends' walls on a daily basis to users writing posts only occasionally but uploading and tagging pictures every week.

To better understand typical OSN usage patterns and map them to the tightness of social relationships, we have submitted a survey to 20 Facebook users (9/11 females/males, about 150 friends per participant) and 10 Twitter users (3/7 females/males, about 30 followers/followings per participant) ranging from 18 and 30 years old. Our survey asked participants to specify the tightness of at least 15/9 of their Facebook/Twitter relationships choosing among tight, regular, and loose. Then, we asked their permission to gather data about their usage of Facebook and Twitter and compared their answers with per-relationship statistical data.

To evaluate which parameters can better enable inferences on relationship tightness, we have followed two main steps:

1. for each tagged friend of each participant, we have gathered information to understand her interactions, normalized in relation to the participant's usage pattern;
2. we have fed Weka [13] (a well-known machine learning software suite) with normalized multi-participant aggregated data and triggered J48/C4.5 [14] (a widely adopted algorithm generating decision trees based on training sets) to determine a model for the evaluation of social tightness.

In particular, we have collected social data associated with each participant and each friend the participant has specified the relationship tightness in the survey. For instance, data gathered from Facebook are related to the direct messages from the participant to her friends (and vice versa), the pictures of the participant where her friends are tagged in (and vice versa), shared links, mutual friends, and common groups; data gathered from Twitter range from followers and followings shared between the participant and her friends to re-tweets and direct participant-friend messages. Note that we have not collected the uploaded data, e.g., pictures and the content of posts, but only their numbers, e.g., how many pictures and posts the participants have uploaded.

As already pointed out, each parameter has been normalized to take into consideration the typical usage patterns of the participant and her friends. Only to mention an example, the number of Facebook common groups has been normalized by considering the ratio between common groups and the total amount of groups joined by the participant not including common ones. By analyzing such normalized values we observed that there is not a single parameter allowing to clearly evaluate the tightness of relationships. For instance, users exchanging many messages are likely to have a tight relationship; however, exchanging few messages (or no messages at all) does not mean that the relationship is loose. Based on this consideration, we have applied the J48/C4.5 algorithm to normalized data (aggregating information gathered from every participant) to identify the parameters that permit to better evaluate relationships tightness. In particular, we have identified the following parameters (in order of relevance):

- Facebook:
  - direct messages from participants to friends;
  - direct messages from friends to participants;
  - pictures of participants where their friends are tagged in;
  - posts of participants where their friends are tagged in;
  - posts sent to participants by their friends;
  - posts of participants where their friends are tagged in;
  - posts of friends where participants are tagged in;
  - comments of friends to participants' status;
  - comments of friends to participants' pictures;
  - pictures of friends where participants are tagged in;
- Twitter:
  - direct messages from participants to friends;
  - tweets of participants to friends;
  - tweets to participants sent by friends;
  - direct messages to participants from friends;
  - tweets of participants re-tweeted by friends;
  - tweets of friends where participants are mentioned;
  - tweets of participants mentioning friends;
  - tweets of friends re-tweeted by participants.

Let us stress that in both cases direct messages have very high priority, reflecting the fact that two users exchanging many direct messages are likely to have a tight relationship. Then, other parameters at lower priority depend on specific interaction mechanisms supported by different OSNs. Starting from the default decision tree based on our survey, RAMP users can provide their own feedback re-tagging some of their relationships. Then, Weka re-runs the J48/C4.5 algorithm to get a refined decision model better fitting the usage pattern of considered users (additional details in Section V).

IV. Filtering Mechanisms with Differentiated Granularity

Our filtering mechanisms tune the differentiated visibility of services/contents i) based on relationship tightness and ii) taking advantage of a general-purpose access model to easily map filters to heterogeneous services. By delving into finer details, we exploit a service access model consisting of three main phases: discovery of services, browsing of shared resources, and actual access to content. Based on this model we provide three types of filter:

- Discovery Filter (DF), to tune the retrievable set of services. This filter allows hiding services to remote users looking for them, e.g., by dropping RAMP Discovery Service packets sent to search the location of RAMP-based File Sharing or by discarding UPnP/GENA packets searching for DLNA Media Servers. In this case, if a participant node already knows the location of its needed service, it can directly interact with it despite DF rules;

- Browsing Filter (BF), to tune the set of shared contents, e.g., by dropping UPnP/SOAP "browse" packets. In this case, similarly to DF, if a node already knows the URL of a given file, it can directly access it despite BF rules;
- Action Filter (AF), to tune the accessibility of a single shared resource. For instance, it denies/allows to get a given file via a DLNA Media Server or files with given extensions via a RAMP-based File Sharing service. Moreover, AF also allows tuning the access to other services, e.g., by denying/allowing to invoke UPnP actions such as "powerOn" of remotely controlled UPnP lights.

Let us note that DF and BF filtering mechanisms can be applied to either ingress or egress packets, namely packets entering or leaving remote UCNs respectively, by achieving differentiated granularity of visibility filtering (AF applies to ingress packets only, to filter the execution of actions). When DF applies to ingress discovery packets, it allows completely hiding a type of service, e.g., by dropping discovery packets looking for File Sharing, or independently from the searched service. On the contrary, when applied to egress packets, DF allows hiding specific service replicas offered by given nodes. Considering the example in Figure 2, Cate's gateway (GW$_c$) can hide every DLNA Media Server by dropping every service discovery packet related to this service (Figure 2-a, left-to-right service discovery packet); otherwise, it can admit discovery packets and then selectively forward only egress replies coming from given devices (Figure 2-b, right-to-left service response from NAS$_1$/NAS$_2$ dropped/allowed).

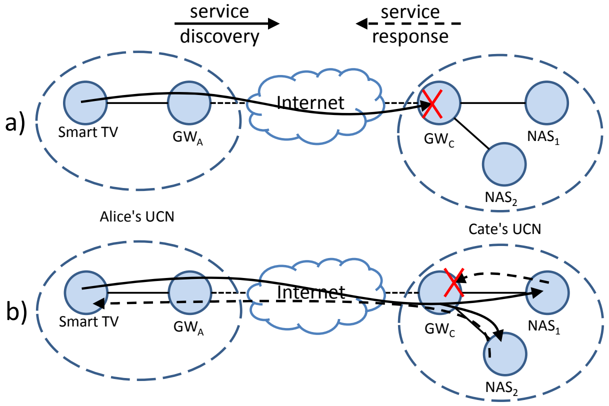

Figure 2. Examples of differentiated discovery granularity.

Similarly to DF, when BF applies to ingress packets, it offers the coarse-grained capability of dropping browsing requests sent to a given service, e.g., by preventing a remote user from getting the list of contents shared by a UPnP DLNA Media Server. Instead, when applied to egress packets, DF allows modifying the set of shared contents, e.g., by modifying the list of shared files sent by a File Sharing service by removing every file but the ones ending with ".mp3".

DF, BF, and AF are defined as `<filterType> = <filter>` based on the EBNF in Figure 3, allowing the specification of filtering rules with the granularity that users deem most appropriate for their own requirements. In particular, our filtering rules are based on a whitelist approach, dropping every packet not specifically allowed by defined rules. While the majority of users are willing to specify rules easily, e.g., allowing tight friends browsing a given DLNA Media Server despite the exploited OSN, some users may desire to define finer-grained rules, e.g., allowing only a given Facebook friend to access a File Sharing service. In any case, note that while expert users can directly specify rules based on the proposed EBNF, most mass-market users are expected to use a GUI to easily map relationships with permissions simply by selecting them from drop-down menus.

```
<filterType> ::= <packetFilter>_<socialRel>
<packetFilter> ::= DF | BF | AF
<socialRel> ::= FacebookFriend_<rel> | TwitterFriend_<rel> |
GenericSocial_<rel> | Generic
<rel> ::= <relationshipLevel> | <friendId>
<relationshipLevel> ::= tight | regular | loose

<filter> ::= <ruleDiscoveryList> | <ruleBrowseList> | <rule-
ActionList>

<ruleDiscoveryList> ::= <scope> (, <device>)*
<device> ::= <deviceId> [<scope> (, <service>)*]
<service> ::= <visibility><serviceId>
<visibility> ::= + | -
<scope> ::= * | /

<ruleBrowseList> ::= <scope> (, <deviceService>)*
<deviceService> ::= <deviceId>[<scope> (, <serviceBrowse>)*]
<serviceBrowse> ::= <serviceId> <regExp>

<ruleActionList> ::= <scope> (, <deviceAction>)*
<deviceAction> ::= <deviceId>[<scope> (, <serviceAction>)*]
<serviceAction> ::= <serviceId>[<scope> (, <action>)*]
<action> ::= <visibility><actionId>
```

Figure 3. The EBNF grammar adopted for our filtering rules.

To clearly and practically present our filtering grammar, we provide some simple filter examples with differentiated granularities. Figure 4 provides the most permissive filter exploiting the "Generic" relationship. In particular, it specifies that remote users can freely discover services, browse content, and perform actions ('*' parameter). However, there are some exceptions specifically defined for the service File Sharing (FS) offered by the node with identifier "1051": in this case, it is possible to browse only csv and txt files and only to get files, while it is not allowed to put files on the server.

```
DF_Generic = *
BF_Generic = *, 1051 [/, FS .*?\.(csv|txt)$]
AF_Generic = *, 1051 [/,FS [/, +get]]
```

Figure 4. Example of generic rules.

To enable permissions based on relationship tightness (independently of the integrated OSN), it is possible to modify rules adding tight, regular, or loose at the end of the rule name (see Figure 5).

```
DF_GenericSocial_loose = *
BF_GenericSocial_tight = *, 1051 [/, FS .*?\.(csv|txt)$]
AF_GenericSocial_regular = *, 1051 [/, FS[/, +get]]
```

Figure 5. Example of rules related to relationship tightness.

To achieve finer-grained visibility, it is possible to specify different rules for specific OSN relationships, e.g., specializing the previous rules (see Figure 6).

```
DF_FacebookFriend_loose = *
BF_FacebookFriend_tight = *, 1051 [/, FS .*?\.(csv|txt)$]
AF_FacebookFriend_regular = *, 1051 [/, FS[/, +get]]
```

Figure 6. Example of rules related to relationship tightness and OSN.

Furthermore, it is possible to specify rules for a given remote user: the example of Figure 7 specifies that the Facebook friend with id 5592 can discover only the UPnP Switch Power service offered by the UPnP device with id 62872bd2 and invoke every supported action, while she cannot find, browse, or access any other service.

```
DF_FacebookFriend_5592 = /, 62872bd2 [/, +SwitchPower]
BF_FacebookFriend_5592 = /
AF_FacebookFriend_5592 = /, 62872bd2 [/, SwitchPower[*]]
```

Figure 7. Example of rules related to OSN identity.

Finally, let us stress that users can activate the two previous examples of rules at the same time. Our filtering solution applies rules from the most specific to the most general one, thus allowing to exploit coarse-grained rules to define the default behavior and finer-grained rules as exceptions specifically written for given OSNs, relationships, or friends.

V. Design and Implementation Insights

Our middleware solution is based on the two main components depicted in Figure 8:

- **Tightness Evaluator**, in charge of gathering social data and metadata from OSNs and of evaluating the tightness of social relationships;
- **Visibility Tuner**, in charge of managing and applying filter rules at runtime to differentiate resource visibility by actively monitoring RAMP traversing packets.

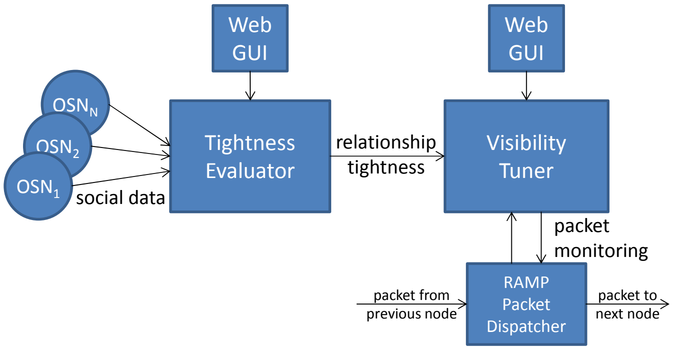

Figure 8. The proposed architecture for social-aware differentiated visibility.

Let us note that both Tightness Evaluator and Visibility Tuner provide a Web GUI allowing users to customize the middleware behavior to better fit their specific requirements. For instance, users can set the tightness of some of their friends to adapt the tightness model based on their specific OSN usage patterns rather than exploiting our default model. In addition, they can activate already available rules (and eventually define new rules) to show/hide resources to specific OSN friends or categories. It is also worth noting that Visibility Tuner actively monitors traversing packets at provisioning time while Tightness Evaluator runs in background when activated by users.

A. *Tightness Evaluator*

Tightness Evaluator is based on three primary modules: Data Collector, gathering and normalizing information from OSNs, Decision Tree Generator, exploiting collected data to create a decision tree based on our default model for social relationships, and Model Customizer, allowing users to improve our default model by specifying the tightness of (part of) their relationships.

By delving into finer details, **Data Collector** interacts with OSNs to gather users' data and stores them to support Decision Tree Generator with suitable information. Given that OSNs typically offer heterogeneous mechanisms to access their data, we have implemented different instances of Data Collector for Facebook and Twitter based on their specific sets of APIs. For instance, to gather the list of Facebook friends we exploit the Facebook Query Language (FQL) query

```
String query =   "SELECT uid2 " +
                 "FROM friend " +
                 "WHERE uid1 = me()";
```

and then invoke

```
List<FacebookUser> friends = fbClient.executeQuery(query,
FacebookUser.class);
```

while, to gather the list of Twitter followers, we exploit the Twitter APIs

```
IDs ids = twitter.getFollowersIDs(userId, -1);
```

by specifying the id of the user we are interested in and the pagination criterion to split the retrieved list.

Decision Tree Generator takes advantage of data gathered by Data Collector to build a decision tree based on J48/C4.5. Since this classification algorithm is based on automatically generated decision trees, it could be affected by the well-known underfitting and overfitting problems. To avoid these issues, we have tuned J48/C4.5 attributes to prune the decision tree to achieve a general model, not specifically tailored on the given training set, while providing sufficient accuracy. In particular, we have set the J48/C4.5 algorithm to prune the tree via a post-pruning method that discards leaves with confidence factor lower than 0.35. Moreover, considering that some categories may include only a few friends, we have set the per-leave minimum number of friends to 2; in this way, there are always at least two friends for each path (online-pruning). Tuning these parameters, we have achieved a model accuracy of 83% for the experimental data collected so far.

Figure 9 shows the Twitter decision tree model based on the survey presented in Section III (for the sake of clarity we omit the Facebook decision tree). To better explain how the decision tree works, consider a friendship with `sentMessagesRate` greater than 0.045 and `receivedMessagesRate` equal to 0 (see Figure 9). In this case the relationship is evaluated as regular, reflecting the intuitive rule that if two users exchange private messages their relationship is not loose.

```
sentMessagesRate <= 0.045
|   sentTweetsOnReceivedRate <= 0.007
|   |   sentTweetsOnTotalSentRate <= 0.047
|   |   |   commonFollowingRate <= 0.144
|   |   |   |   mentionTweetsRate <= 0.083
|   |   |   |   |   commonFollowersRate <= 0.098: LOOSE
|   |   |   |   |   commonFollowersRate > 0.098: REGULAR
|   |   |   |   mentionTweetsRate > 0.083: REGULAR
|   |   |   commonFollowingRate > 0.144: REGULAR
|   |   sentTweetsOnTotalSentRate > 0.047: REGULAR
|   sentTweetsOnReceivedRate > 0.007: REGULAR
sentMessagesRate > 0.045
|   receivedMessagesRate <= 0: REGULAR
|   receivedMessagesRate > 0: TIGHT
```

Figure 9. Default Twitter decision tree.

Finally, **Model Customizer** allows modifying the decision tree by taking into consideration the OSN usage pattern of each user. In other words, users can tune the decision tree model in order to better fit their behavior on OSNs, instead of relying on the default model based on average usage. It is worth noting that in this way not only reclassified relationships change, but also the model, thus actually personalizing it in relation to the specific usage pattern of users. In this manner novel relationships are evaluated based on the new customized model instead of the default one. For instance, Figure 10 shows a slightly personalized decision tree: in this case to be a regular friend `sentMessagesRate` should be greater than 0.053 and `receivedMessagesRate` lower than or equal to 0.012.

```
sentMessagesRate <= 0.053
|   sentTweetsOnReceivedRate <= 0.021
|   |   sentTweetsOnTotalSentRate <= 0.031
|   |   |   commonFollowingRate <= 0.389
|   |   |   |   mentionTweetsRate <= 0.025
|   |   |   |   |   commonFollowersRate <= 0.361: LOOSE
|   |   |   |   |   commonFollowersRate > 0.3618: REGULAR
|   |   |   |   mentionTweetsRate > 0.025: REGULAR
|   |   |   commonFollowingRate > 0.389: REGULAR
|   |   sentTweetsOnTotalSentRate > 0.031: REGULAR
|   sentTweetsOnReceivedRate > 0.021: REGULAR
sentMessagesRate > 0.053
|   receivedMessagesRate <= 0.012: REGULAR
|   receivedMessagesRate > 0.012: TIGHT
```

Figure 10. Customized Twitter decision tree.

B. *Visibility Tuner*

Visibility Tuner is based on three primary components: Filter Repository, Filter Manager, and Filter Enforcer. **Filter Repository** stores Java objects implementing actual filters, configured by users based on the rules described in Section IV. For each RAMP-based service, it is possible to provide a different filter, by specifying its behavior for the discovery, browsing, and action phases. **Filter Manager** allows adding/removing filters and de/activating them, even at provisioning time. To this purpose, it reads a configuration file to retrieve the Java classes implementing the filters, without any specific user intervention. **Filter Enforcer** performs filtering actions at runtime: it monitors ingress/egress packets and applies filters loaded and activated by Filter Manager. If filtering rules allow packets to enter/leaving a user's UCN, Filter Enforcer regularly dispatches them, otherwise it appropriately modifies their content or drops them.

To better understand how Filter Enforcer works, consider the following example. The packet

```
FS GET nodeId=1051 op=filelist
```

coming from a Facebook regular friend requires the file list provided by the RAMP-based File Sharing service offered by the RAMP node 1051. The Filter Enforcer is configured with the rules

```
DF_FacebookFriend_regular = *
BF_FacebookFriend_regular = *, 1051 [/, FS .*?\.(csv|txt)$]
AF_FacebookFriend_regular = *, 1051 [/, FS[/, +get]]
```

and the File Sharing service provides the files below:

```
notes.txt
track01.mp3
accounting.csv
```

Filter Enforcer applies the BF rule for regular Facebook friends, enabling the visibility of only csv/txt files while hiding mp3 ones. Moreover, it allows the dynamic discovery of the File Sharing service and getting any file, but not adding new files.

VI. Performance Evaluation

We have validated our prototype performing several tests to quantitatively evaluate the performance of the tightness evaluator and of our filtering mechanisms. In particular, the section reports about the performance results achieved on top of a Win 7 Pro desktop PC with a Dual-Core processor 2.0 GHz and 4 GB RAM for the primary middleware features originally presented in this paper, i.e., i) to collect social data from OSNs, ii) to evaluate the tightness of social relationships, and iii) to apply our filtering solution to ingress/egress packets.

A. *Retrieving and updating social data*

To evaluate the tightness of relationships based on up-to-date user's behavior, our solution gathers and maintains social data only within a given time window. We consider a default time window of three months (tunable at runtime). The data updating process can be split into three different phases:

- User, collecting and updating user's social data;
- Friend List, collecting and updating user's friend list;
- Friend Data, collecting and updating social data and metadata of user's friends.

For the sake of briefness, here we report about results related to Facebook only; note that Twitter maintains and provides fewer social information and thus it is much less time-consuming to collect its data. Figure 11 shows, in semi-logarithmic scale, the time required to process User, Friend List, and Friend Data while varying the amount of friends.

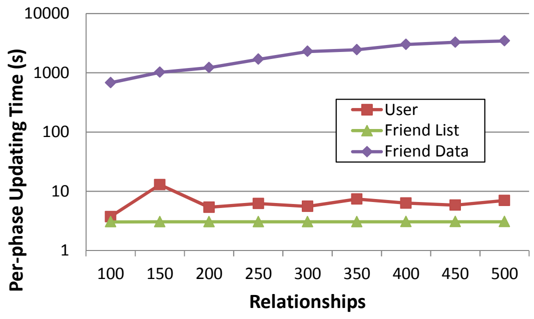

Figure 11. Performance of the OSN data gathering process.

Of course the required time also depends on how and how much the user and her friends employ Facebook, e.g., the frequency with which they upload new posts and pictures. In any case, Figure 11 demonstrates that the Friend Data phase greatly influences the total time and grows almost linearly with the amount of Facebook friends, e.g., by requiring more than 17/54 minutes to retrieve social data of 150/450 friends.

B. *Classifying the tightness of social relationships*

We have also evaluated the time required to generate decision trees based on the above collected data. On the one hand, we have run Weka while varying the size of the training set, i.e., the amount of social relationships manually classified to generate the decision tree. Table I shows a slight dependency on the training set size due to the increased amount of data J48/C4.5 has to consider, ranging from 851ms with a training set of 10 relationships to 861ms for a training set of 50 relationships. In any case, the achieved performance demonstrates that the size of the training set does not considerably lower final performance indicators. On the other hand, Figure 12 shows that the time needed to classify relationships linearly grows with the amount of friends, since the classification system has to separately evaluate each relationship.

TABLE I. PERFORMANCE OF THE BUILDING MODEL PROCESS.

Training set size	10	20	30	40	50
Building time (ms)	851	857	856	858	861

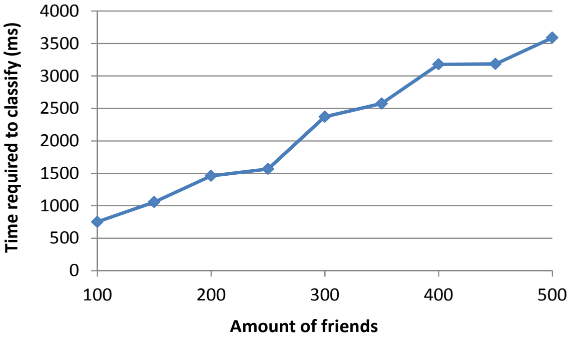

Figure 12. Performance of the classification process.

C. Packet Filters

To quantitatively evaluate the efficiency of our packet filter mechanisms, we have emulated a federation of 200 Facebook users, in particular, for the sake of evaluating the time required to filter packets in relation to the complexity of rules, the amount of rules, and the rate of packets to be checked. Based on our evaluation, we have verified that the amount and complexity of active rules do not affect the efficiency of our mechanisms. On the opposite, we have identified that the efficiency of our solution depends on the rate of arrival packets to be filtered. In particular, we have stress-tested our middleware prototype by focusing on the browse phase, considered the most challenging one since it imposes to parse the payload of packets and to check shared contents visible by other users.

Overall, the achieved results demonstrate that our implementation can efficiently filter packets, by exhibiting acceptable latencies up to the notable rate of 1000 packets per second. By delving into finer details, Figure 13 shows the average response time imposed by the filtering mechanism for ingress browse request packets (53 bytes for File Sharing browse packets, 52 bytes for UPnP ones). The achieved performance exponentially degrades in relation to ingress packet arrival rate. Packet filtering related to the File Sharing service imposes more time because there is the need to parse the whole packet content; instead, UPnP packet filtering has only to check a single field corresponding to the requested object id (already pre-parsed by our UPnP RAMP proxy).

Finally, Figure 14 shows the overhead for processing "browse" egress packets (responses to "browse" requests) of the RAMP-based File Sharing service, packet size of 60 bytes with a list of four files. The measured times are higher in comparison with ingress "browse" packets, since in this case the packet filter has to parse and modify the whole response content to eventually delete the name of resources a user may want to hide. In any case, the requested time grows almost linearly in relation to the packet rate, even if at higher rates the standard deviation grows considerably.

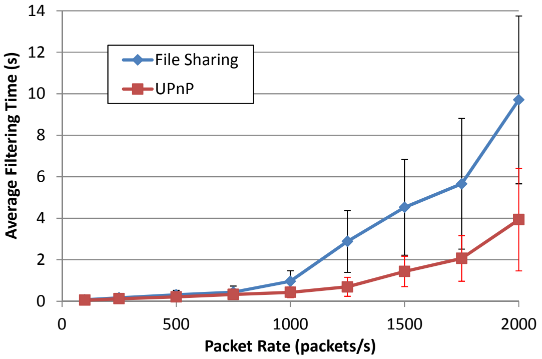

Figure 13. Average response time for ingress "browse" packet filtering.

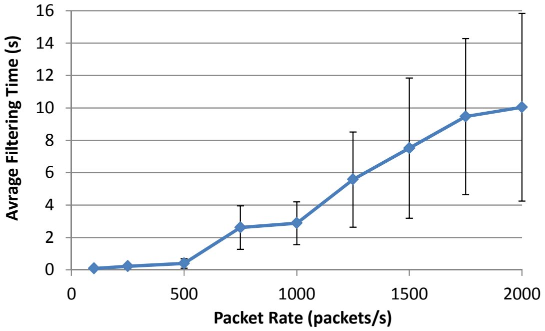

Figure 14. Average response time for the filtering of File Sharing egress "browse" packets.

VII. RELATED WORK

Collecting data of inter-user interactions to infer their relationships has been an active research field even before the wide spread and commercial success of OSNs. For instance, [15] proposes to exploit information available on Web home pages to get inter-user similarities based on interests and to gather relationships based on available hyperlinks toward other home pages; similarly, [16] exploits data available on blogs, by taking advantage of their content-intensive nature. Instead, [17] exploits the C4.5 algorithm to retrieve relationship strength among conference participants based on coauthoring.

More recently, the availability of huge amounts of social data and metadata voluntarily uploaded on OSNs has allowed more articulated solutions [18-20]. [21] proposes a latent variable model applied to LinkedIn data to infer the strength of social relationships mainly based on profile similarities, based on the primary guidelines that the more similar two people are, the stronger their relationship tends to be. [22] exploits the results of a survey similar to ours to identify predictive variables, i.e., variables depending on relationship strength, and then models social tightness by linearly combining them.

OSNs have been also exploited to improve resource provisioning with specific objectives. For instance, [23] exploits inter-user social relationships gathered on OSNs to optimize content delivery in peer-to-peer networks, e.g., by taking into consideration social contacts/interests when allocating flows, managing network topology, and scheduling packet transmission. Instead, [24] exploits trusted social relationships to improve privacy and security in peer-to-peer networks, e.g., by prioritizing paths with nodes by users who are socially tied. Moreover, [25] proposes to exploit high-level sharing policies and the content of uploaded resources to automatically suggest potentially safe friends. Finally, to the best of our knowledge, our middleware is the first to originally apply dynamically evaluated social tightness to automatically filter user-generated content visibility in home-to-home scenarios, with the primary goal of enabling simple and rapid decisions on resource visibility (eventually tunable with full expressive power), typically suitable for mass-market users.

VIII. Conclusions

The paper presents the design, implementation, and evaluation of our novel middleware prototype to filter visibility of home-to-home shared resources based on dynamically evaluated tightness of social relationships. A key aspect of our approach is the capability of defining visibility rules in a portable cross-OSN way, independently of the actual type of social relationships linking two users. The achieved performance results demonstrate the feasibility of our approach, showing that our per-packet filtering solution can process up to 1000 legacy UPnP browse packets per second while our mechanisms to gather OSN data and build decision trees well fit the target application scenario.

The encouraging results achieved so far are stimulating our further research activities along two main directions. On the one hand, we are developing more sophisticated analytical tools based on principal components analysis to better investigate how the tightness of social relationships can be inferred from and are influenced by OSN usage patterns. On the other hand, we are developing a GUI to facilitate the retrieval of shared resources and the definition of ad-hoc filtering rules.

Acknowledgments

Partially supported by POR FESR Emilia-Romagna 2007-2013.

References

[1] G. Davis, “Facebook Ireland Ltd, Report of Re-Audit”, http://dataprotection.ie/documents/press/Facebook_Ireland_Audit_Review_Report_21_Sept_2012.pdf, September 2012 (last visited on July 26th, 2013)

[2] I. Ion, N. Sachdeva, P. Kumaraguru, S. Čapkun, “Home is Safer than the Cloud! Privacy Concerns for Consumer Cloud Storage”, 7th Symposium on Usable Privacy and Security (SOUPS '11), Pittsburgh, PA, USA, July 2011.

[3] S. Buchegger, D. Schiöberg, Le Hung Vu, A. Datta, “PeerSoN: P2P Social Networking - Early Experiences and Insights”, in Proceedings of SocialNets 2009, The 2nd Workshop on Social Network Systems, Nuernberg, Germany, March 2009.

[4] Lo-Yao Yeh, Yu-Lun Huang, A.D. Joseph, S.W. Shieh, Woei-Jiunn Tsaur, "A Batch-Authenticated and Key Agreement Framework for P2P-Based Online Social Networks", IEEE Trans on Vehicular Technology, vol.61, no.4, pp.1907-1924, May 2012.

[5] P. Bellavista, C. Giannelli, L. Iannario, L.-W. Goix, C. Venezia, “Peer-to-peer Content Sharing Based on Social Identities and Relationships”, submitted for publication, available as technical report at http://lia.deis.unibo.it/Staff/CarloGiannelli/socialsharing.pdf

[6] P. Bellavista, P. Gallo, C. Giannelli, G. Toniolo, A. Zoccola, “Discovering and Accessing Peer-to-peer Services in UPnP-based Federated Domotic Islands”, IEEE Trans. on Consumer Electronics, vol. 58, mo. 3, pp. 810-818, August 2012.

[7] L.S. Ferreira, M.D. De Amorim, L. Iannone, L. Berlemann, L.M. Correia, "Opportunistic Management of Spontaneous and Heterogeneous Wireless Mesh Networks", IEEE Wireless Comm., vol.17, no.2, pp.41-46, April 2010.

[8] P. Bellavista, A. Corradi, C. Giannelli, “Middleware for Differentiated Quality in Spontaneous Networks”, IEEE Pervasive Computing, vol. 11, no. 3, pp. 64-75, March 2012.

[9] P. Bellavista, A. Corradi, C. Giannelli, "The Real Ad-hoc Multi-hop Peer-to-peer (RAMP) Middleware: an Easy-to-use Support for Spontaneous Networking", 15th IEEE Symp. on Computers and Communications (ISCC'10), Riccione, Italy, June 2010.

[10] P. Bellavista, A. Corradi, C. Giannelli, “Differentiated Management Strategies for Multi-hop Multi-Path Heterogeneous Connectivity in Mobile Environments”, IEEE Trans. on Network and Service Management, Vol. 8, No. 3, pp. 190-204, Sept. 2011.

[11] D. Johnson, Y. Hu, D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4", http://tools.ietf.org/html/rfc4728, Feb. 2007.

[12] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", IETF, RFC 3031, Jan. 2001.

[13] “Weka 3: Data Mining Software in Java”, http://www.cs.waikato.ac.nz/ml/weka/ (last accesed on July 26th, 2013).

[14] J.R. Quinlan, “C4.5: Programs for Machine Learning”, Machine Learning, 1993.

[15] L.A. Adamic, E. Adar, “Friends and neighbors on the Web”, Social Networks, vol. 25, no. 3, pp. 211-230, July 2003.

[16] L. Fan, B. Li, "Blog-based online social relationship extraction", 8th Int. Conf. on Cognitive Informatics (ICCI '09), pp. 457-463, Hong Kong, June 2009.

[17] Y. Matsuo, J. Mori, M. Hamasaki, T. Nishimura, H. T., K. Hasida, M. Ishizuka, “POLYPHONET: An advanced social network extraction system from the Web”, Web Semantics: Science, Services and Agents on the World Wide Web, vol. 5, no.4, pp. 262-278, December 2007.

[18] A. Mislove, B. Viswanath, K.P. Gummadi, P. Druschel, “You are who you know: inferring user profiles in online social networks”, third ACM international conference on Web search and data mining (WSDM ’10), pp. 251-260, New York, USA, February 2010.

[19] M. Forestier, J. Velcin, D. Zighed, "Extracting Social Networks to Understand Interaction", Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM), Kaohsiung, Taiwan, pp. 213,219, July 2011.

[20] I. Kahanda, J. Neville, “Using Transactional Information to Predict Link Strength in Online Social Networks”, third Int. AAAI Conf. on Weblogs and Social Media (ICWSM ’09), pp. 74-81, San Jose, USA, May 2009.

[21] R. Xiang, J. Neville, M. Rogati, “Modeling relationship strength in online social networks”, 19th Int. Conf. on World Wide Web, pp. 981-990, Raleigh, USA, April 2010.

[22] E. Gilbert, K. Karahalios, “Predicting tie strength with social media”, SIGCHI Conf. on Human Factors in Computing Systems (CHI ’09), pp. 211-220, Boston, USA, April 2009.

[23] J. Chakareski, P. Frossard, "Context-Adaptive Information Flow Allocation and Media Delivery in Online Social Networks", IEEE J. Selected Topics in Signal Processing, vol.4, no.4, pp.732-745, Aug. 2010.

[24] L.A. Cutillo, R. Molva, T. Strufe, "Safebook: A privacy-preserving online social network leveraging on real-life trust", IEEE Comm. Magazine, vol.47, no.12, pp.94-101, Dec. 2009

[25] M. Jakob, Z. Moler, M. Pechoucek, R. Vaculin, "Content-Based Privacy Management on the Social Web", Int. Conf. on Web Intelligence and Intelligent Agent Technology (WI-IAT), vol. 3, pp. 277,280, Lyon, France, August 2011.