Dynamic Shared Groups within XMPP: An Investigation of the XMPP Group Model

Leigh Griffin, Eamonn de Leastar and Dmitri Botvich
Telecommunication Software and Systems Group
Waterford Institute of Technology
Waterford, Ireland
{lgriffin, edeleastar, dbotvich}@tssg.org

Abstract— **Group communication offers a means for resource sharing and collaboration, often delivered through diverse technologies. One of the technologies, Instant Messaging, traditionally took the role of a facilitating service within such communities. Driven by a flexible XML based protocol, XMPP, instant messaging has developed functionality to a point where it can be considered a standalone group communication medium. Harnessing the extensible nature of the protocol continues to be a challenge. Alternative usage scenarios not envisioned by its core group management models have since emerged. This paper examines the current XMPP group models and how they can fulfill the requirements of a modern scenario centered on dynamic shared groups,**

***Keywords-** Dynamic Group; Group Management; Shared Group; XMPP*

I. Introduction

Group Communication has developed significantly since the turn of the century. What emerged from fragmented technologies, used by a minority of internet users, has evolved into consolidated applications. Innovations in design, usability and functionality, have led designers to examine the role that the group itself plays. This paper focuses on one particular group communication medium, Instant Messaging (IM), as characterised by the XMPP protocol, in an emerging context of interest. With the move towards e-health, the notion of a care group has emerged as an environment for collaboration, information exchange and instantaneous communication within a hospital environment. A care group capable of forming around each individual patient, with the membership of that group restricted to the current healthcare professionals involved with the care and rehabilitation plan of the patient, has many benefits. As a patient moves through various hospital departments this group is constantly evolving with new staff and medical team members taking over the care of the patient. It is important that a unified view of the group is presented to all health facilitators who are members, allowing them to monitor the progress of patients as well as communicate effectively with peers in their group. With the nature of some emergencies within a hospital, the establishment of a care group should not be hindered by administration burdens. As such, the care group can be described as dynamic, as per the following definition: *Dynamic Groups are groups with a membership base that is prone to fluctuation. The membership levels change rapidly and often without notice. The membership turnover is high with the groups generally being barrier free, with users joining and leaving at will.* This definition is based on related work in [1] which examined the usage of Dynamic groups and the scenario is based on previous work by the authors in [2] and [3]

This paper will profile the group management structures available to XMPP to realise such a scenario underpinned by theoretical analysis. This paper is divided up into five sections. The first is this introduction. Section two examines the structure and workings of XMPP. Section three outlines the group management mechanism available within XMPP. Section four presents dynamic shared groups within XMPP. The fifth, and final section, represents the future work and the concluding remarks.

II. XMPP

A. *Instant Messaging with XMPP*

Instant Messaging has inherent advantages over other text based delivery platforms. It is almost instantaneous, usually includes a built-in subscription mechanism and provides an indication of a users availability and context via presence. Using Instant Messaging as the communication medium of a compelling use case was a logical decision, with the eXtensible Messaging and Presence Protocol (XMPP) the protocol of choice. XMPP is an open, XML based protocol tailored specifically to provide extensible instant messaging and presence information [4][5]. XMPP assumes a client-server architecture [6], with multiple clients able to connect to an XMPP server. A client is an entity that establishes an XML stream with a server, passing along user credentials and when successful, binds itself to the connecting resource. The stream, as denoted by the </stream> tag, acts as a container for XML stanzas which provision for both upstream and downstream communication. An XML stanza is a discrete semantic unit of structured information that is sent from one entity to another over an XML stream. The three stanzas defined are:

- </message> : A stanza to facilitate the transmission of a message from a sender to a recipient.
- </presence> : Used to broadcast the availability, and thus willingness to be contacted, of a user to anyone subscribed.

- <iq> : The Info/Query tag is a request/response mechanism used for setting and retrieving information.

The streams allow for the delivery of stanzas from client to client via the server. The servers' responsibilities include storing and managing XML data used by the clients and managing the delivery of XML streams to local clients. The routing and delivery of streams to foreign clients is possible through local service policies allowing server to server federation. Adhering to the key tags and XML semantics outlined, the entire stream can be viewed as one valid XML document. This highly structured means of communication allows for core extensions to be integrated without breaking the design rules and functionality of the protocol.

B. Roster Management

The XMPP Communication mechanism as specified in the XMPP IM document, RFC 3921, [7] outlines the process in which groups of contacts are managed. A contact list, or roster, is used to manage a set of users, termed buddies, and to optionally group them together. The roster is specified in the roster schema, [8]. Figure 1, shows the schematic view of the XMPP roster.

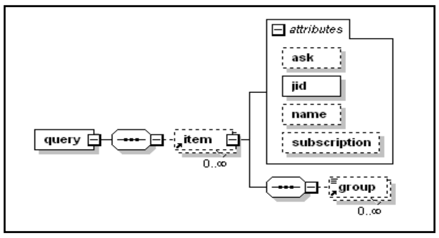

Figure 1. XML Schema Design View of the Roster

The "item" element in figure one is a representation of a roster entry. The attributes within the item store important information about this contact. The Jabber Identifier (JID), is a unique means to identify an individual. The syntax is based on the structure of an email address, with a username associated to a domain name, which represents their home server. An optional resource mechanic is associated with a JID, specified by a slash suffix, allowing multiple simultaneous logins by the user with the XMPP server able to route the messages appropriately.

The subscription attribute of the request is used to establish the type of presence subscription that will exist between the two entities, the sender and the recipient, or user and contact respectively. The allowable values for this attribute are:

- None – the user does not want to subscribe to the contacts presence information and does not wish for the contact to have a subscription to the users updates
- To – The user wishes to have a subscription to the updates of the contact but does not offer a reverse subscription.
- From – the contact will have a subscription to the users presence updates but the user will not subscribe to the contacts presence updates
- Both - both the user and the contact are subscribed to each others presence information

The optional attribute, "ask", can have the value "subscribe", which means that an acknowledgement from the recipients' server must be received to verify the connection. When the ask attribute is present, the presence subscription is then set to "none", by default, until the response is received. The name attribute is an optional nickname to associate with the roster entry. Presence is an important mechanism for XMPP and desirable within the scope of the scenario to advertise the availability of participants as well as provide a mechanism for service advertisement.

The schema allows for the formation or population of groups during a subscription request. The group attribute is used to store the text based name of the group associated with the roster entry. This group attribute is optional and if it is not included in a request the contact will still be stored and rendered in the user client. Users can add new contacts to their roster through a roster set message. The user can send a request to another user requesting a friendship link be established. A sample XML roster set request, to add a new contact to a users roster and allow it to be rendered in the clients user interface is shown in figure 2 below.

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
      jid='contact@example.org'
      subscription='none'
      ask='subscribe'
      name='MyContact'>
    <group>MyBuddies</group>
    </item>
  </query>
</iq>
```

Figure 2. XML Syntax for a Roster Addition

III. XMPP GROUP MODEL ANALYSIS

Groups within XMPP follow one of two basic models for the creation and management of groups, with a viable third developed as a community extension. The approaches could be described as weakly modeled, as groups within XMPP are used to logically divide up entities within a user roster and do not serve any further purpose. Their usage simplifies the roster, allowing for greater organisation and readability.

A. User Generated Groups: The default model

A user generated group is created by a user from within their own client. The group is created through a simple interface on the client device and populated by the creator. This action prompts a roster set message being sent to the server, as the group attribute has been updated. This modification occurs so the client, on future logins, understands what groups to place roster items in. Some observations about user generated groups will now be discussed:

- Membership is anonymous:

Users placed into a group are passive participants, completely unaware they have possessed membership of this group. The group is thus private and serves no purpose other then the logical placement of buddies within an end users client.

- Membership is not enforced or shared

Once a user has authorised a friend request and presence subscription, they have bi-directional visibility on their IM clients. Any groups created by either user are not enforced across the buddy lists and no membership notification occurs.

- A 1:1 relationship exists between users and groups

With user generated groups a buddy can only exist once and once only. Thus, a buddy can only have membership of one user generated group at a time on a users roster. Moving a buddy from one group to another causes them to lose their existing membership in order to be associated with the new group.

B. Pub-Sub Generated Groups

The second means of managing and creating groups within XMPP is a variation of the publish-subscribe (pub-sub) model as described in XEP-0060 [9]. This extension provides a framework for subscription nodes and event notification that is compatible with XMPP. A variety of applications dependent on event notifications, such as network management systems, can then benefit from the integration of XMPP. An adapted version of this model can be implemented server side, allowing an administrator the capability of creating groups and subscribing contacts to them. These pre-populated groups can then be published to end user rosters, effectively bypassing the process described in section two. Entities, groups and presence subscriptions can be forced onto end users rosters. This is an effective way of subscribing users to default groups, with all editing attributes removed to ensure the group structure remains intact. Some of the features and results of creating groups in this manner will now be examined:

- Membership is enforced completely

The end user has no say in their participation of a pub-sub group and do not have the choice of declining the invitation or leaving the group at will. The membership is completely enforced and the group cannot be modified by members who do not possess server administrator access.

- Overloaded Rosters

The creation of a pub-sub group which has roster sharing enabled causes all members of that group to replicate the groups structure on their roster. This means the addition of the groups population onto the end users roster. The complications arising from such a scenario are the enforced subscriptions, potentially generating a large amount of additional presence updates. As it stands, the authors of [4] identify presence as accounting for "90% of XMPP traffic, with the majority of it being redundant broadcasts". Generating additional presence broadcasts is thus an expensive side effect of enforcing memberships.

- Administrative Interaction required

To create a pub-sub group administrative access to the XMPP server is required. The creation and management of the groups needs to be performed by an administrator due to the modifications required to end users rosters

C. Community Extension: Roster Item Exchange

A third means to distribute groups within XMPP, developed by a community inspired extension called Roster Item Exchange (RIE), is outlined in XEP-0144 [10]. This work came directly from the communities recognition that shared groups should have a place within XMPP [11]. This extension provides a mechanism for a user to share elements of their roster with another user, recommending additions, deletions and modification for use primarily in shared groups. The roster items sent can range from individual entries, whereby a single contact is shared to sharing a roster group or indeed an entire roster. The extension allows a recommendation of which group the roster item should be placed.

IV. DYNAMIC SHARED GROUPS WITHIN XMPP

It is clear that user generated groups are not a viable model for distributing groups in a dynamic nature. Similarly the pub-sub generated groups require administrative interaction on the server side to guarantee the groups are distributed. Additionally the pub-sub method results in severely overloaded rosters, creating an N squared scalability problem as more users are added. Roster Item Exchange is the only practical means for distributing dynamic membership updates in a shared manner within XMPP, but RIE as a means of distributing dynamic shared groups is flawed. The extension remains in draft format and its authors acknowledge that the requirements set forth by the community for shared groups and synchornisation of rosters are not provisioned for completely within this extension but will be addressed as future work. The extension is currently not optimized for group management and group member distribution. For groups of size N, an RIE request must be sent to N-1 accounts, containing recommendations for N-1 changes to be made. The changes sent are purely recommendations, which are free to be rejected by the recipient of the RIE request. If the recommendation is accepted, a standard friendship request is issued by the recipient to the recommended entity. This request in turn can be denied. The requests sent are potentially blind, as no notification is returned to the originator of the request if the Info Query mechanism is not utilised. With two possible points of failure, the extension is not stable enough to guarantee a shared and unified roster view across group participants. Additionally, from a usability point of view the acceptance of RIE requests can be cumbersome on client devices if a batch mode option for accept / deny is not implemented.

A number of guarantees are also required to ensure that the groups would be distributed accordingly through RIE. Simultaneous RIE requests should not be allowed in order to preserve the integrity of the group structure. Two RIE requests from different sources have the potential to create different interpretations of the group structure. RIE, by design, does not include a presence subscription when adding a new user to a group. When adding a new user it could not be assumed that the intended account to be recommended to other users already had a presence relationship with the RIE recipients. As such, an additional subscription packet, would have to be sent, as presence is one of the desirable features of using IM for such a scenario. This additional packet greatly increases the traffic profile of RIE as the presence packet will need to be sent from the originating client. Server side processing will drop any presence subscription packets for a designated recipient if a presence relationship already exists which limits the cascading effect somewhat. Using this model, rosters would also require an auto accept enabled for presence and RIE subscriptions, something which the RIE specification strongly advises against due to legitimate security concerns. It would be possible for a Denial of Service attack to occur by pushing through a large volume of RIE requests that conflict in a short amount of time. Security concerns aside, if this feature was turned off it would be possible for individual recipients to simply reject the RIE request and therefore not have a shared group view. RIE deletion recommendations are also a legitimate concern, particularly if the auto accept is enabled server side. It would be possible to wipe someone's roster through RIE requests if a user with malicious intentions so desired.

V. FUTURE WORK AND CONCLUSION

The deployment of dynamic shared groups within XMPP, while possible, has too many assumptions associated with it and no formal management provisioned. A working implementation is possible with clever programming and a community willing to stick rigidly to the guidelines, but a long term, scalable management solution would allow XMPP evolve group based applications in a controlled manner. The management of groups within XMPP currently does not provision for modern scenarios, such as the example in section 1 of this paper. The group management structure is dated but serves faithfully the original purpose of XMPP. It is our belief that the management of groups is important enough to be abstracted away from XMPP, to provide more control for group management and formation. Our future work proposes an experimental architecture to manage groups external to XMPP, facilitated by the extensible nature of the protocol and managed through network based policies. The use of policies to control and manage the formation of the groups will be an important milestone. With the Policy Engine already developed as previous work [12], the focus can shift to the usability and functionality of the management policies rather then their implementation. A proposed route would take the notion of a JID, and evolve that concept to a Group ID, or GID for short. A GID would be used to hold a reference to a Group which would reside on a group server and would structurally take the same format as a JID address: group_name@group_server_domain. From a scalability and management point of view, an approach such as this would allow XMPP evolve into a group management and service platform. A complementing group management extension, would transition the medium into the realm of service group management, allowing semantically rich services tailored at different group styles to evolve. Other group styles, such as Ad-hoc [13], are also under consideration for investigation, as group formation profiles would need to be developed from existing classifications to understand the impact that they would have on the underlying infrastructure.

This paper outlined a use case involving the novel use of existing group communication architectures to better provision for the care of patients in a hospital scenario. XMPP, a viable communication protocol, and group communication medium in its own right, had strong credentials for realising this scenario and was presented for consideration. An investigation into the group management structures offered by XMPP was performed with a formal investigation carried out on the capabilities of the existing group management structures. The results were presented showing that XMPP is desirable and indeed viable for this scenario but the successful sharing and management of a dynamic group environment is beyond the scope and capabilities which the protocol was designed for. A compelling plan for abstracting the group management responsibilities from XMPP and entrusting them to a separate entity was also proposed as future work.

ACKNOWLEDGMENT

The authors would like to acknowledge funding support from the Irish HEA PRTLI Cycle 4 FutureComm (http://futurecomm.tssg.org) programme.

REFERENCES

[1] Hallberg, J., Norberg, M., Kristiansson, J., Synnes, K., Nugent, C. Creating Dynamic Groups using Context-awareness. 6th International Conference on Mobile and Ubiquitous multimedia, 2007.

[2] Smedberg, A.. Enabling Cross-Usage of Public Health Systems – A Holistic Approach to Ask the Expert Sstems and Online Communities, Proceedings of the Second Annual Conference on Digital Society, 2008.

[3] Storni, C., Bannon, L. Reassembling HealthCare: toward a patient-centric approach, Proceedings of the Healthcare Informatics Society of Ireland, 14th Annual Conference, 2009.

[4] Saint-André, Peter, Kevin Smith, and Remko Tronçon. XMPP: the Definitive Guide : Building Real-time Applications with Jabber Technologies. Farnham: O'Reilly, 2009.

[5] XMPP Standards Foundation [online]. Available from http://xmpp.org Accessed on 10-SEP-2010

[6] Fielding, R.. 2000. Architectural Styles and the Design of Network-based Software Architectures. Thesis(PhD). Univeristy of California.

[7] Saint-Andre, P., 2004. RFC 3921 XMPP: IM [online] Available from http://www.ietf.org/rfc/rfc3920.txt Accessed on 10-SEP-2010

[8] XMPP Roster Schema Document [online] Available from http://xmpp.org/schemas/roster.xsd Accessed on 10-SEP-2010

[9] Millard, P., Saint-Andre, P., Meijer, R., 2010. XEP-0060: Publish Subscribe [online] Available from http://xmpp.org/extensions/xep-0060.html Accessed on 10-SEP-2010

[10] Saint-Andre, P., 2005. XEP-0144: Roster Item Exchange [online]. Available from http://xmpp.org/extensions/xep-0144.html Accessed on 10-SEP-2010

[11] Saint-Andre, P., 2004. XEP-0140: Shared Groups [online]. Available from http://xmpp.org/extensions/xep-0140.html Accessed on 10-SEP-2010.

[12] Foley et al, "Service Group Management facilitated by DSL driven Policies in embedded Middleware" International Symposium on Computers and Communications, 2010.

[13] Minder, D., Grau, A., Marron, P. On group formation for self-adaptation n pervasive systems. 1st international conference on Autonomic computing and communication systems, 2007.