XEP-0289: Federated MUC for Constrained Environments

Kevin Smith
mailto:kevin.smith@isode.com
xmpp:kevin.smith@isode.com

2012-05-29
Version 0.2

Status	Type	Short Name
Deferred	Standards Track	FMUC

This document provides a protocol for federating MUC rooms together in order to reduce the effects of constrained network (e.g. unreliability, severely limited bandwidth) on the room occupants.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

MUC's design generally assumes a highly reliable network providing plenty of bandwidth, and it functions well in Internet settings. It is sometimes the case that server to server traffic is heavily constrained, with typical problems for constrained links being high latency, tiny amounts of available bandwidth and unreliability (including, potentially, long-term failure of S2S links). This document provides methods for allowing experiences close to those of standard MUC use while operating across such constrained links by allowing rooms to federate with remote counterparts and for users to connect to the federated MUC node nearest to them on the network for a given FMUC room. It requires no setup in advance, and needs no bandwidth for remote rooms without local occupants. The premise is that a proxy room joins another room and receives stanzas from the MUC just as another occupant would; this is analogous to the client to server model, whereby a client would connect to their local server and the server deals with connections elsewhere - the client joins a local room and the room deals with connections to other federated rooms.

1.1 Terminology

As MUCs are generally self-contained entities with a single address, federating them requires the introduction of some new terminology:

- FMUC set - the union of all MUC rooms that federate together
- FMUC node - a single MUC room that is a member of an FMUC set (abbreviated to "node")
- FMUC room - a room represented by an FMUC set
- Master-Master mode - two FMUC nodes operating such that both will continue to work when a network fails between them. This mode has the properties of reduced network traffic and of not having a guarantee of consistent message ordering between nodes
- Master-Slave mode - two FMUC nodes operating where one, the master, will continue working during a network outage while the slave will cease to work while it cannot communicate with the master. This mode has increased network traffic and a consistent message delivery order across both nodes.
- Joining FMUC node - a node that initiates an FMUC connection to another node (abbreviated to "joining node").
- Joined FMUC node - a node that accepts an FMUC connection from another node (abbreviated to "joined node"). Note that when nodes are connected in a tree-like structure a node in one of the middle layers will be both a joining node in the context of its connection to the room above it and also a joined node in the context of those nodes below that join to it.

For illustration: if room1@rooms.server1.lit and room2@rooms.server2.lit federate with each other, then room1@rooms.server1.lit is an FMUC node, as is room2@rooms.server2.lit. Both nodes are in the FMUC set (along with any other node rooms that mutually federate) while the conceptual single room created by joining the FMUC set together is the FMUC room (and this FMUC room does not have a single definitive identifier).

2 Requirements

- If appropriately configured, avoid bandwidth use that isn't strictly necessary for message exchange.
- Allow conversation in a federated MUC to continue when one of the federated nodes is unavailable (e.g. due to network failure preventing S2S links forming), such that the nodes operate in a 'peer to peer' or 'multi-master' mode.
- If configured, allow a master/slave configuration such that a disconnected node is no longer usable for local chat
- *Acceptable compromise* When operating in multi-master mode the message ordering may not be consistent between FMUC nodes.

3 Addressing

In Federated MUC an FMUC room does not have a single logical address; when joining the FMUC room a user's client can join any of the nodes in the FMUC set for that room, and all addressing will appear to that client as if this was the single canonical representation of the room's address - while other users in the room may see different addresses dependent upon the node they joined.
It is possible, although not required, for an implementation and deployment to use JID Escaping (XEP-0106) [1] to make naming schemes easy to manage, but this is a matter of deployment policy and not of the protocol defined herein.

4 Actors

The following JIDs are used in this document.

- wonderland.lit - service
- rooms.wonderland.lit - MUC service on wonderland.lit.

[1] XEP-0106: JID Escaping <https://xmpp.org/extensions/xep-0106.html>.

- alice@wonderland.lit - User on wonderland.lit
- hatter@wonderland.lit - User on wonderland.lit
- rabbithole@rooms.wonderland.lit - MUC room / FMUC node.
- denmark.lit - service, likely connected to wonderland.lit over constrained link
- talk.denmark.lit - MUC service on denmark.lit.
- hamlet@denmark.lit - User on denmark.lit
- ophelia@denmark.lit - User on denmark.lit
- elsinore@talk.denmark.lit - MUC room / FMUC node.

5 Use Cases

5.1 Initial Federation

Here hamlet@denmark.lit is going to join the (currently empty) elsinor@talk.denmark.lit room. This room is configured as an FMUC node, federating with the rabbithole@rooms.wonderland.lit node, which current has one occupant - alice@wonderland.lit. The method of configuration that elsinor should federate with rabbithole is considered out of scope for this document - it is suggested that it be including in the standard MUC room configuration form. Note that this configuration only needs to be one way (that is: there is no protocol reason why rabbithole needs to know that elsinor will be federating with it in advance) - this allows for the ad-hoc addition of additional nodes to the FMUC room.
First hamlet@denmark.lit issues a normal MUC join request to elsinor@talk.denmark.lit

Listing 1: User joins FMUC node

```
<presence from='hamlet@denmark.lit/priam-ubuntu-vm' to="elsinore@talk.
    denmark.lit/Hamlet">
     <x xmlns="http://jabber.org/protocol/muc"/>
</presence>
```

Elsinor then attempts to join with the FMUC node rabbithole@rooms.wonderland.lit.

Listing 2: FMUC Node begins initiates federation

```
<presence from='elsinore@talk.denmark.lit/Hamlet' to='rabbithole@rooms
    .wonderland.lit/Hamlet'>
     <fmuc xmlns='http://isode.com/protocol/fmuc' from='hamlet@denmark.
         lit/priam-ubuntu-vm'/>
     <x xmlns="http://jabber.org/protocol/muc"/>
     <x xmlns="http://jabber.org/protocol/muc#user">
```

```
      <item affiliation="none" role="participant" jid="hamlet@denmark.
         lit/priam-ubuntu-vm"/>
    </x>
</presence>
```

Now rabbithole may reject the join, if elsinor is not permitted to federate. To do this it sends a 'presence' reply from its bare JID to the bare JID of the joining node with an 'fmuc' payload containing a 'reject' element and MAY include a human-readable explanation as the text content of the 'reject' element.

Listing 3: Second FMUC Node rejects federation

```
<presence from="rabbithole@rooms.wonderland.lit" to="elsinore@talk.
   denmark.lit">
  <fmuc xmlns="http://isode.com/protocol/fmuc">
    <reject>No cross-domain federation.</reject>
  </fmuc>
</presence>
```

Or it may accept the federation request and reply with the list of current occupants and message context in the same order as specified in XEP-0045. Note that the fmuc element is always added containing the JID of the user (possibly passed down from other FMUC nodes, or indeed from the joining node for the presence of the user used for the initial join), while the XEP-0045 rules apply for whether to include the jid in the muc#user element.
As part of the initial join of one node to another, the node being joined will send the current topic to the node doing the joining. The node receiving this (the joining node) SHOULD replace its own subject with the received one.
The joining node may add an element to the initial presence to the node being joined limiting the amount of history to be sent in the normal manner, as in XEP-0045.

Listing 4: Second FMUC Node accepts federation and sends occupants and history

```
<presence from="rabbithole@rooms.wonderland.lit/Alice" to="
   elsinore@talk.denmark.lit">
  <fmuc xmlns="http://isode.com/protocol/fmuc" from="alice@wonderland.
     lit/priam-ubuntu-vm"/>
  <x xmlns="http://jabber.org/protocol/muc#user">
    <item affiliation="owner" role="moderator" jid="alice@wonderland.
       lit/priam-ubuntu-vm"/>
  </x>
</presence>

<presence from="rabbithole@rooms.wonderland.lit/Hamlet" to="
   elsinore@talk.denmark.lit">
  <fmuc xmlns="http://isode.com/protocol/fmuc" from="hamlet@denmark.
     lit/priam-ubuntu-vm"/>
  <x xmlns="http://jabber.org/protocol/muc#user">
```

```
    <item affiliation="none" role="participant" jid="hamlet@denmark.
       lit/priam-ubuntu-vm"/>
  </x>
</presence>

<message from="rabbithole@rooms.wonderland.lit/Alice" to="
    elsinore@talk.denmark.lit" type="groupchat">
  <fmuc xmlns="http://isode.com/protocol/fmuc" from="alice@wonderland.
     lit/priam-ubuntu-vm"/>
  <body>This is an old message from the history</body>
  <x xmlns="urn:xmpp:delay" from="rabbithole@rooms.wonderland.lit"
     stamp="20120419T16:00:44"/>
</message>

<message from="rabbithole@rooms.wonderland.lit/Hatter" to="
    elsinore@talk.denmark.lit" type="groupchat">
  <fmuc xmlns="http://isode.com/protocol/fmuc" from="hatter@wonderland
     .lit/priam-ubuntu-vm"/>
  <body>This is another old message from the history</body>
  <x xmlns="urn:xmpp:delay" from="rabbithole@rooms.wonderland.lit"
     stamp="20120501T10:03:24"/>
</message>

<message from="rabbithole@rooms.wonderland.lit/Alice" to="
    elsinore@talk.denmark.lit" type="groupchat">
  <fmuc xmlns="http://isode.com/protocol/fmuc" from="hatter@wonderland
     .lit/priam-ubuntu-vm"/>
  <subject>This is the subject</subject>
  <x xmlns="urn:xmpp:delay" from="rabbithole@rooms.wonderland.lit"
     stamp="20120528T14:39:34"/>
</message>
```

Upon receiving the occupants, history and subject from the other node the joining FMUC node will process these in the normal way, treating the received presence as joins, adding the history to the room's history (re-ordering by delay?) and changing the subject. The joining FMUC node MUST NOT send the traffic generated by these data back to the joined room, but only deliver them to local participants (and in the case of chained FMUC nodes, any nodes joined to it). It also MUST NOT pass the fmuc payloads through to local clients.

Listing 5: First FMUC Node relays the state to the joined client

```
<presence from="elsinore@talk.denmark.lit/Alice" to="hamlet@denmark.
    lit/priam-ubuntu-vm">
  <x xmlns="http://jabber.org/protocol/muc#user">
    <item affiliation="owner" role="moderator" jid="alice@wonderland.
       lit/priam-ubuntu-vm"/>
  </x>
</presence>
```

```
<presence from="elsinore@talk.denmark.lit/Hamlet" to="hamlet@denmark.
    lit/priam-ubuntu-vm">
  <x xmlns="http://jabber.org/protocol/muc#user">
    <item affiliation="none" role="participant" jid="hamlet@denmark.
        lit/priam-ubuntu-vm"/>
  </x>
</presence>

<message from="elsinore@talk.denmark.lit/Alice" to="hamlet@denmark.lit
    /priam-ubuntu-vm" type="groupchat">
  <body>This is an old message from the history</body>
  <x xmlns="urn:xmpp:delay" from="rabbithole@rooms.wonderland.lit"
      stamp="20120419T16:00:44"/>
</message>

<message from="elsinore@talk.denmark.lit/Hatter" to="hamlet@denmark.
    lit/priam-ubuntu-vm" type="groupchat">
  <body>This is another old message from the history</body>
  <x xmlns="urn:xmpp:delay" from="rabbithole@rooms.wonderland.lit"
      stamp="20120501T10:03:24"/>
</message>

<message from="elsinore@talk.denmark.lit/Hatter" to="hamlet@denmark.
    lit/priam-ubuntu-vm" type="groupchat">
  <subject>This is the subject</subject>
  <x xmlns="urn:xmpp:delay" from="rabbithole@rooms.wonderland.lit"
      stamp="20120528T14:39:34"/>
</message>
```

5.2 Sending messages

Then hamlet@denmark.lit sends a message to the FMUC room, which is sent from the elsinor node to the rabbithole node and then broadcast to the local occupants of each room according to the standard XEP-0045 rules (rabbithole distributes to alice, elsinor distributes to hamlet). This example is for master-master mode, so rabbithole does not echo the message back to elsinore and elsinore does not need to wait for receipt of this stanza from rabbithole before distributing the stanza locally.

Listing 6: User on FMUC Node sends a message

```
<message from='hamlet@denmark.lit/priam-ubuntu-vm' to='elsinore@talk.
    denmark.lit' type='groupchat'>
    <body>Hi Alice</body>
</message>

<message from='elsinore@talk.denmark.lit/Hamlet' to='hamlet@denmark.
    lit/priam-ubuntu-vm' type='groupchat'>
```

```
    <body>Hi Alice</body>
</message>

<message from='elsinore@talk.denmark.lit/Hamlet' to='rabbithole@rooms.
    wonderland.lit' type='groupchat'>
    <body>Hi Alice</body>
    <fmuc xmlns="http://isode.com/protocol/fmuc" from="hamlet@denmark.
        lit/priam-ubuntu-vm"/>
</message>

<message from='rabbithole@rooms.wonderland.lit/Hamlet' to='
    alice@wonderland.lit/priam-ubuntu-vm' type='groupchat'>
    <body>Hi Alice</body>
</message>
```

alice@wonderland.lit then replies to this message, causing a similar distribution.

Listing 7: User on other FMUC Node replies to message

```
<message from='alice@wonderland.lit/priam-ubuntu-vm' to='
    rabbithole@rooms.wonderland.lit' type='groupchat'>
    <body>Hi Hamlet</body>
</message>

<message from='rabbithole@rooms.wonderland.lit/Alice' to='
    alice@wonderland.lit/priam-ubuntu-vm' type='groupchat'>
    <body>Hi Hamlet</body>
</message>

<message from='rabbithole@rooms.wonderland.lit/Alice' to='
    elsinore@talk.denmark.lit' type='groupchat'>
    <body>Hi Hamlet</body>
    <fmuc xmlns="http://isode.com/protocol/fmuc" from="
        alice@wonderland.lit/priam-ubuntu-vm"/>
</message>

<message from='elsinore@talk.denmark.lit/Hamlet' to='hamlet@denmark.
    lit/priam-ubuntu-vm' type='groupchat'>
    <body>Hi Hamlet</body>
</message>
```

5.3 Subsequent activity

Another user joining or parting a room will be "fanned-out" in much the same way - the node to which they're joined will send out their presence to all the locally joined users and to the other FMUC nodes to which it's connected, and those nodes will then do the same - noting that in master-master mode they won't distribute the stanza back to the node from which

they received it.

Listing 8: Additional user joins FMUC room

```
<presence from="hatter@wonderland.lit/priam-ubuntu-vm" to="
    rabbithole@rooms.wonderland.lit/Hatter">
  <x xmlns="http://jabber.org/protocol/muc"/>
</presence>

<presence from="rabbithole@rooms.wonderland.lit/Hatter" to="
    alice@wonderland.lit/priam-ubuntu-vm">
    <x xmlns="http://jabber.org/protocol/muc#user">
        <item affiliation="none" role="participant" jid="
            hatter@wonderland.lit/priam-ubuntu-vm"/>
    </x>
</presence>

<presence from="rabbithole@rooms.wonderland.lit/Alice" to="
    hatter@wonderland.lit/priam-ubuntu-vm">
  <x xmlns="http://jabber.org/protocol/muc#user">
    <item affiliation="owner" role="moderator" jid="alice@wonderland.
        lit/priam-ubuntu-vm"/>
  </x>
</presence>

<presence from="rabbithole@rooms.wonderland.lit/Hamlet" to="
    hatter@wonderland.lit/priam-ubuntu-vm">
  <x xmlns="http://jabber.org/protocol/muc#user">
    <item affiliation="none" role="participant" jid="hamlet@denmark.
        lit/priam-ubuntu-vm"/>
  </x>
</presence>

<presence from="rabbithole@rooms.wonderland.lit/Hatter" to="
    hatter@wonderland.lit/priam-ubuntu-vm">
    <x xmlns="http://jabber.org/protocol/muc#user">
        <item affiliation="none" role="participant" jid="
            hatter@wonderland.lit/priam-ubuntu-vm"/>
        <status code="110"/>
        <status code="100"/>
    </x>
</presence>

<!-{}-Message history sent to hatter elided for brevity -{}->
<message from="rabbithole@rooms.wonderland.lit/Hatter" to="
    hatter@wonderland.lit/priam-ubuntu-vm" type="groupchat">
  <subject>This is the subject</subject>
  <x xmlns="urn:xmpp:delay" from="rabbithole@rooms.wonderland.lit"
     stamp="20120528T14:39:34"/>
```

```
</message>

<presence from='rabbithole@rooms.wonderland.lit/Hatter' to='
    elsinore@talk.denmark.lit/Hatter'>
  <fmuc xmlns='http://isode.com/protocol/fmuc' from='hatter@wonderland
      .lit/priam-ubuntu-vm'/>
  <x xmlns="http://jabber.org/protocol/muc#user">
    <item affiliation="none" role="participant" jid="hatter@wonderland
        .lit/priam-ubuntu-vm"/>
  </x>
</presence>

<presence from="elsinore@talk.denmark.lit/Hatter" to="hamlet@denmark.
    lit/priam-ubuntu-vm">
    <x xmlns="http://jabber.org/protocol/muc#user">
        <item affiliation="none" role="participant" jid="
            hatter@wonderland.lit/priam-ubuntu-vm"/>
    </x>
</presence>
```

5.4 Leaving a room

When a user leaves a room the presence is distributed in the same way.

Listing 9: User leaves FMUC node

```
<presence from="hatter@wonderland.lit/priam-ubuntu-vm" type="
    unavailable" to="rabbithole@rooms.wonderland.lit/Hatter"/>


<presence from="rabbithole@rooms.wonderland.lit/Hatter" type="
    unavailable" to="hatter@wonderland.lit/priam-ubuntu-vm">
    <x xmlns="http://jabber.org/protocol/muc#user">
        <item affiliation="none" role="none" jid="hatter@wonderland.
            lit/priam-ubuntu-vm"/>
        <status code="110"/>
    </x>
</presence>

<presence from="rabbithole@rooms.wonderland.lit/Hatter" type="
    unavailable" to="alice@wonderland.lit/priam-ubuntu-vm">
    <x xmlns="http://jabber.org/protocol/muc#user">
        <item affiliation="none" role="none" jid="hatter@wonderland.
            lit/priam-ubuntu-vm"/>
    </x>
</presence>
```

```
<presence from='rabbithole@rooms.wonderland.lit/Hatter' to='
    elsinore@talk.denmark.lit/Hamlet' type='unavailable'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <item affiliation='none' role='none' jid='hatter@wonderland.lit/
        priam-ubuntu-vm'/>
  </x>
  <fmuc xmlns='http://isode.com/protocol/fmuc' from='hatter@wonderland
      .lit/priam-ubuntu-vm'/>
</presence>

<presence from="elsinore@talk.denmark.lit/Hatter" type="unavailable"
    to="hamlet@denmark.lit/priam-ubuntu-vm">
    <x xmlns="http://jabber.org/protocol/muc#user">
        <item affiliation="none" role="none" jid="hatter@wonderland.
            lit/priam-ubuntu-vm"/>
    </x>
</presence>
```

When the last user on a joining FMUC node leaves the room the joining node has no more users in the joined node and the joining node will be considered to have left the FMUC set. Further activity in the FMUC set not be sent to the joining node (unless it subsequently rejoins the set).
The joined room confirms that the joining room has left the set by sending a presence stanza from the bare JID of the joined room to the bare JID of the joining room with an FMUC payload containing an element 'left'.

Listing 10: Joined FMUC node alerts the joining node that it is no longer in the FMUC set

```
<presence from="rabbithole@rooms.wonderland.lit" to="elsinore@talk.
    denmark.lit">
  <fmuc xmlns="http://isode.com/protocol/fmuc">
    <left/>
  </fmuc>
</presence>
```

5.5 Administration

When an FMUC node receives notice that one of their users has been kicked by a moderator on another node it SHOULD kick the user and fan-out the consequent presence stanzas to other nodes.
Role change should be distributed across nodes and fanned out to users, but only cosmetically (e.g. an owner on another node cannot effect changes to the affiliation lists on this node).

5.6 Private Messages

When sending a private message to an occupant of another node, each node that receives it is responsible for routing it to the appropriate node (or the occupant, if local).

Listing 11: User sends private message to occupant of other node

```
<message to="elsinore@talk.denmark.lit/Alice" from="hamlet@denmark.lit
    /priam-ubuntu-vim" type="chat">
  <body>Hi, I want to say something in private</body>
</message>

<message to="rabbithole@rooms.wonderland.lit/Alice" from="
    elsinore@talk.denmark.lit/Hamlet" type="chat">
  <fmuc xmlns="http://isode.com/protocol/fmuc" from="hamlet@denmark.
      lit/priam-ubuntu-vm"/>
  <body>Hi, I want to say something in private</body>
</message>

<message to="alice@wonderland.lit/priam-ubuntu-vim" from="
    rabbithole@rooms.wonderland.lit/Hamlet" type="chat">
  <body>Hi, I want to say something in private</body>
</message>
```

6 Business Rules

7 Security Considerations

This allows an FMUC node to proxy for another JID, so should only be deployed in scenarios where either the FMUC nodes are trusted, or it is known that the users of an FMUC node are in the same security domain as the FMUC node itself.

8 TBD

How to get the join-target to tell the joining room how much history to send during a resync.
How to perform a resync (Part and then full rejoin)
Illustrate master-slave mode - this is simply that the sending room waits for the echo back from the room to which it's joined before distributing messages locally.
Describe collisions. Send fmuc payload saying there's a collision back to the node, Node with local user can then send an error message about the collision and kick them.

9 IANA Considerations

None.

10 XMPP Registrar Considerations

Needs a namespace.

11 XML Schema

When advanced.