Contents

- Background and History of IM
- Jabber System Architecture
- Jabber Protocol Basics
- Conclusion and Future
- References
Background and History of IM

Instant Messaging (IM) Systems

AOL AIM (began in 1985): 130 Million Users
Yahoo Messenger: 10 Million Users
MSN Messenger: 10 Million Users
Typical IM Implementation

The Future of IM: Integration and Interoperability

- AOL and the FCC
  “interoperability progress reports”
Jabber

- Open Source:
  - Jabber Open Source License
- Jeremie Miller began 1998
- jabber.org and jabber.com

Jabber and IETF

- 1999-2001: IMPP fizzled
- 2002: new effort XMPP Working Group
Jabber System Architecture

- "email + Presence"
- Client/Server
- Server Peer Network
- XML based
- Transports
Thin Client

- Connects to Server using XML Stream
- All messages go through server
- TCP/IP Socket or SSL
Fat Server

- Server Has Most of the Functionality
  - Routing
  - Maintains State Info
  - Database

Jabber Server

- Server Peer Network
- Routing Between Peers
- Use DNS to Forward and Authenticate
- Store and Forward: Fault Tolerant
- Dialbacks
Jabber Server

• State Info
  – Presence
  – Session
  – Offline Message Storage
  – Subscriber List Maintenance

Transports

• Modular Structure of Server
• Provides Protocol Translation
Server Implementation

Modular Structure: Components

EtherX Component (deliver)

Jabber Session Manager

Basic Message Flow
Jabber Protocol Basics

XML Streams

- Stream Establishment
- Protocol Namespaces
- XML Chunks
- "Superdocument“ Approach:
  - disadvantages
XML Stream Protocol

- XML Chunks
- XML Stream
- TCP/IP Socket

Example XML Stream

- NODE:
  <stream:stream
to='host'
    xmlns='jabber:client'
    xmlns:stream='http://etherx.jabber.org/
    /streams'>
• HOST:
<br/>&lt;stream:stream
<br/>from='host'
<br/>id='id_123456789'
<br/>xmlns='jabber:client'
<br/>
<br/>xmlns:stream='http://etherx.jabber.org/streams'
<br/>&gt;

• NODE:
<br/>&lt;message from='node@host' to='receiving-ID'&gt;
<br/> &lt;body&gt;Watson come here, I need you!&lt;/body&gt;
<br/>&lt;/message&gt;
• HOST:
  <message from='receiving-ID'
    to='node@host'><body>I'm on
    my way!</body>
  </message>

• NODE:
  </stream:stream>

• HOST:
  </stream:stream>

---

**The Node’s XML Document**

```xml
<stream:stream
to='host'
    xmlns='jabber:client'
    xmlns:stream='http://etherx.jabber.org/streams'>
  <message from='node@host' to='receiving-ID'>
    <body>Watson come here, I need you!
      </body>
  </message>
</stream:stream>
```
Jabber Identifiers (JIDs)

- Identifies Any Jabber Entity
- user@server/resource
- dorothy@over.the.rainbow.org/somewhere

Protocol Namespaces

- Stream namespaces:
  - jabber:server  server/server communications
  - jabber:client  client/server communications
Common Data Types

- Exist in all above stream namespaces
- `<message/>`
- `<presence/>`
- `<iq/>`

`<message/>`

- Exchange Data Between Two Jabber IDs
<message/> Attributes

- type = "normal" | "chat" | "groupchat" | "headline" | "error"
- to
- from
- id (a message id, not a JID)

<message/> Subelements:

- <subject/>
- <body/>
- <error/>
- <error/> really XHTML
- <thread/>
- <x/>
<message/> Element

Example Use Case

A simple threaded conversation

<message
to="romeo@montague.net/orchard"
from="juliet@capulet.com/balcony"
type="chat">
<body>Art thou not Romeo, and a Montague?</body>
<thread>283461923759234</thread>
</message>
A simple threaded conversation continued

<message
    to="juliet@capulet.com/balcony"
    from="romeo@montague.net/orchard"
    type="chat">
    <body>Neither, fair saint, if either thee dislike.</body>
    <thread>283461923759234</thread>
</message>

A simple threaded conversation continued

<message
    to="romeo@montague.net/orchard"
    from="juliet@capulet.com/balcony"
    type="chat">
    <body>How cam'st thou hither, tell me, andwherefore?</body>
    <thread>283461923759234</thread>
</message>
<presence/>

- Provides Availability Information
- Handled by JSM Component

<presence/> Attributes

- <presence/> Attributes:

  - type = "[un]available" | "probe" | "[un]subscribe[d]"
<presence/> Subelements

- <show/>
  - Values: “away”, “xa”, “dnd”, “chat”, “normal”
- <status/>
  - Detailed status information
- <x/>

<presence/> Element

Example Use Cases
• Initial presence sent to host upon login to express default availability:

  <presence/>

• Receiving detailed presence from another node:

  <presence to="romeo@montague.net"
           from="juliet@capulet.com">
    <show>xa</show>
    <status>sleeping</status>
    <priority>1</priority>
  </presence>
• Presence sent to host upon logging off to express unavailable state:

<presence type="unavailable"/>

Jabber Subscription Model
<presence/> Element

Example Use Cases

• Sending a subscription request:

```xml
<presence
to="juliet@capulet.com"
from="romeo@montague.net"
type="subscribe"
/>```

• Handling a Subscription Request
• Accepting a subscription request:

```xml
<presence
to="romeo@montague.net"
from="juliet@capulet.com"
type="subscribed"
/>  
```

• Denying a presence subscription request:

```xml
<presence
to="romeo@montague.net"
from="juliet@capulet.com"
type="unsubscribed"
/>  
```
<iq/>

- Info/Query
- Any Kind of Structured Conversation

<iq> Attributes:

- type = "get" | "set" | "result" | "error"
<iq/> Subelements:

• <query/>

Standard Extended Namespaces

• jabber:x: Extend <message/> and <presence/>
• jabber:iq: Extend <iq/>
jabber:x:

- jabber:x:delay
- jabber:x:oob
- jabber:x:roster

jabber:iq:

- jabber:iq:auth
- jabber:iq:register
- jabber:iq:roster
- jabber:iq:oob
- jabber:iq:time
- jabber:iq:conference
- + others
<iq/> Element

Example Use Case

Registration Data Flow

• Step 1: Node queries host regarding information that is required in order to register:

```xml
<iq type="get"
    to="capulet.com"
    id="reg_1">
    <query xmlns="jabber:iq:register"/>
</iq>
```
Step 2: Host responds with the required registration fields:

```xml
<iq type="result"
    from="capulet.com"
    id="reg_1">
    <query xmlns="jabber:iq:register">
        <instructions>
            Choose a username and password to register with this service.
        </instructions>
        <email/>
        <password/>
        <username/>
    </query>
</iq>
```

Step 3: Node provides required information:

```xml
<iq type="set"
    to="capulet.com"
    id="reg_2">
    <query xmlns="jabber:iq:register">
        <email>juliet@capulet.com</email>
        <password>R0m30</password>
        <username>juliet</username>
    </query>
</iq>
```
Step 4: Host informs node of successful registration:

```xml
<iq
    type="result"
    from="capulet.com"
    id="reg_2"
/>`

Step 4 (alt): Host informs node of failed registration:

```xml
<iq
    type="error"
    from="capulet.com"
    id="reg_2"/>

<error code="406">
    Not Acceptable
</error>
```

</iq>
Conclusion and Future

Three Parts of Jabber Protocol

- XML Streams
- Common Data Types
- Standard Extended Namespaces
The Future of Jabber

• Adding "presence" to Web Services
• Jabber-as-Middleware Project
• "vehicle for discovering services, authenticating sessions, reliably delivering XML packets containing service requests and responses"

(from Jabber FAQ, 3.6)

The End
References
Overviews of IM and Jabber:

2. Steele Arbeeny, *Messaging In Peer-To-Peer Networks* {arbeeny@caip.rutgers.edu}

Originally from www.jabber.org:

- 5. Peter Saint-Andre, *Jabber FAQ*
- 6. Peter Saint-Andre, *Jabber Protocol Overview*
- 7. Peter Saint-Andre, *Jabber Technology Overview*
- 8. Jeremie Miller, *The Jabber Architecture*
IETF Drafts (in state of flux)

10. draft-miller-jabber-00.txt,
11. draft-miller-xmpp-core-00.html
12. draft-miller-cpim-core-00.html
13. draft-miller-im-core-00.html
14. draft-miller-xmpp-im-02.txt