XMPP Services

Matthew J. Graham (Caltech, NVO)

The problem

- SOAP and REST:
  - work fine for synchronous services
  - utilize polling for asynchronous services
- For high volume, frequent, low latency services (LSST-era):
  - polling kills most HTTP servers, e.g. Twitter API during 2007 MacWorld keynote
    - does not scale
    - not real time
  - SOAP is overly complicated

XMPP (aka Jabber)

- Extensible Messaging and Presence Protocol:
  - Persistent connections
  - Stateful
  - Designed to be an event stream protocol
  - Natively federated and asynchronous
  - Identity, security and presence built in
- Jabber servers designed to handle at least 10^5 concurrent connections
- JEP (Jabber Enhancement Proposal):
  - 0060: PubSub
  - 0114: Jabber Component Protocol

Sample service – I

- Sample message:

```
<message id="q9MC9-4" to="conesearch.brighid">
 <body>
  <ConeSearch>
   <RA>120</RA>
   <Dec>35</Dec>
   <Radius>0.2</Radius>
  </ConeSearch>
 </body>
</message>
```

- Sample response:

```
<message to="mjg@brighid/Smack" from="conesearch.brighid">
 <body>
  <VOTABLE>…
 </body>
</message>
```

Sample service – II

- Implement using Whack and OpenFire:

```
public class ConeSearchService implements Component {
    public String getName() {}
    public String getDescription() {…}
     public void processPacket(Packet packet) {…}
    public void initialize(JID jid, ComponentManager cm) {}
    public void start() {}
    public void shutdown() {}
}

public class ExternalConeSearchService {
    final ExternalComponentManager man = new
     ExternalComponentManager("localhost,5275);
    man.setSecretKey("conesearch", "foobar");
    man.setAllowMultiple(true);
    man.addComponent("conesearch", new ConeSearchService());
}
```