has to be done in two steps because fetching the result feed doesn't always contain all of the contacts, you need the second query, with the count, to fetch them all. The query is called in the feedURL, which is the URL of the contacts feed, which returns a ContactsFeed object. This can be used with an iterator to process the results, in this case (though removed from the above snippet) the results, if they contain an email address, are used to create an html form containing a dropdown box which is returned to the chat.jsp page to be displayed, and is used to select a contact to chat with.
In the scope of this demo application, there is currently no way to know if a contact is online or using the application. This is not provided by the contacts API or the Channels API (as described below). It would have taken too much time to manually produce this feature for a prototype and make it fit with the App Engine's strict quotas, but if more time was available then it could be implemented. Also, the users both have to open channels to each other for them to receive messages.

4.3.2 Channels API

To facilitate the chat communications the original plan was to use the App Engine XMPP library to provide the communication. There were two main issues with this. Firstly, the XMPP addresses had to be of the form "username@stegchat.appspotchat.com" and an existing XMPP account, such as Facebook chat or Google talk could not be used due to the restriction on outgoing tcp connections. Secondly, all messages have to be received by a specific handler, no matter what address they are sent to, and then somehow passed down to the appropriate users. The first attempt to do this was to use the memcache (as is used with the contacts authentication). The problems mentioned before could cause problems for this (messages being dropped before they are retrieved). The datastore was not an option as to write the messages to the datastore for any more than a few users would run over quota very quickly.
Fortunately, Google offers the Channels API for providing a method of communication between instances of the application, for example for sending data to a user from a backend process. They are essentially a local version of UDP sockets, as in a channel is opened using a token generated from a string (the address). The channel itself is opened via JavaScript, and it is up to the user to decide what happens when certain event occur, such as receiving a message:

```
1 <script type="text/javascript" src="/_ah/channel/jsapi"></script>
  </head>
3 <body>
  <script>
5   var channel = new goog.appengine.Channel('<%=token%>');
    var socket = channel.open({
7         onopen : function() {
```