University of Tromsø
Computer Science Department

INF3791
Practice Week at NST

Project: Symptom based disease surveillance in the North Norwegian Health region

Investigate methods and solutions for message polling through firewalls for jabber (XMPP) servers. Compare the solutions to the Snow Agent system polling mechanism. Write a short report and a presentation of your findings.

16-20 April 2007

Taridzo Chomutare
tch023@student.uit.no, Mobile: 91563026

Introduction

Jabber, which has changed its name to eXtensible Messaging and Presence Protocol (XMPP), is an instant messaging (IM) protocol that has emerged recently. The protocol is based on open standards and features have been added to it to include VoIP and file transfer. It is used by many popular websites and applications such as Google Talk and Gizmo Project.

XMPP now uses port 5222 to connect to servers all over the internet. An XMPP server has the ability to forward server connection requests to other servers on the network. Using the DNS address of where the targeted user resides, the server can connect with the user anywhere. As can be guessed, the user name or Jabber ID (JID) has to have a DNS address portion. The JID is of the form user@server.com/mobile :

- user = user ID
- server.com = DNS address
- mobile = user resource

XMPP can connect to any network as long as the port (normally 5222) is open on the server. In most cases, however, the port is closed, in which case the server would have to open the port and configure it for incoming connection requests. This opens security holes, and systems administrators would rather keep the port closed. Messaging under such conditions require special techniques, some of which are discussed in this paper.

Snow Agents System (SAS) Polling Mechanism

Of interest is the messaging architecture when the networks are behind firewalls and yet they need to communicate. A firewall makes it difficult to connect to the network and SAS introduced a connection manager (proxy) for messaging between networks. The proxy acts as a tunnel by forwarding all requests to their respective destinations. The diagram below illustrates this scenario:

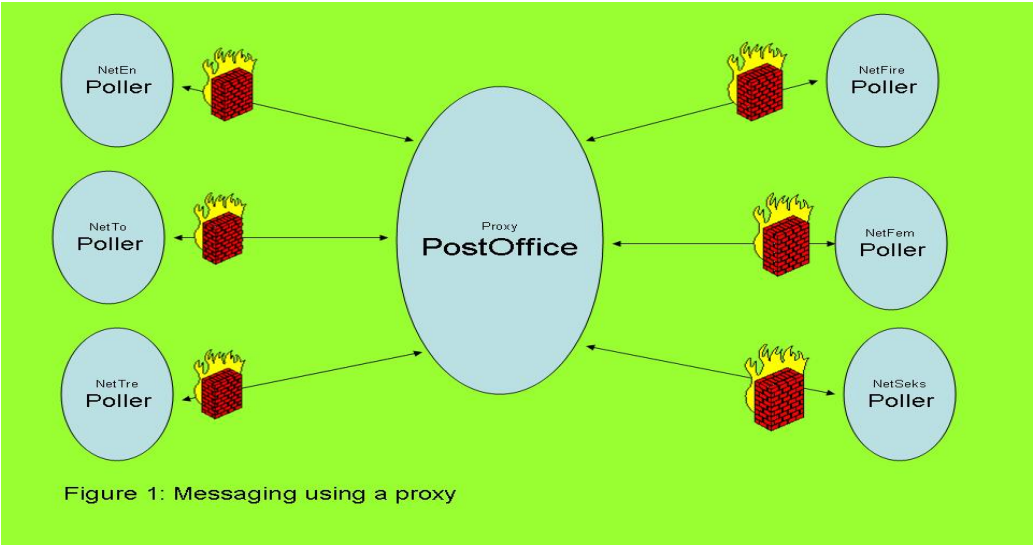

Figure 1: Messaging using a proxy

All the different servers will deposit their requests in the post office computer. The other networks keep polling the post office for messages in their respective “mail boxes”. There is no direct communication between the networks since each network is behind a firewall and does not accept incoming connection requests.

Messages are then relayed to the target as responses to the requests (that is, GET and POST) that come as polling. The networks poll the post office for messages at predefined intervals. Bandwidth is lost by polling the proxy even when there are no messages. With sufficiently high number of connections, a considerable amount of bandwidth could be wasted. The delay in message delivery could be up to `X time units`, where `X` is the polling interval. In situations that require high performance, the delay could be significant hence the need to find alternative solutions.

Alternative

XMPP has come up with an ingenious solution for connections where the firewall forbids even outgoing connections on port 52222. This came as a replacement of the extension XEP-0025: Jabber HTTP Polling. The solution also uses HTTP on both port 80 and 443, for normal and secure connections, respectively. The name of the extension is XEP-0124: Bidirectional-streams Over Synchronous HTTP (BOSH) and has been recommended since 26 July 2006 when XEP-0025 was deprecated. BOSH is based on a sustainable HTTP connection so that the recipient can get the message as soon as it is sent because the connection is kept alive for longer periods.

By using HTTP port 80/443 instead of the Jabber standard port, the clients can send and fetch messages without any restrictions. The XMPP XML data is embedded in the HTTP message and shares the connection with other HTTP requests unrelated to XMPP. This is a huge advantage considering that many firewalls allow HTTP access anyway.

The protocol works by emulating bi-directional connections using synchronous HTTP request-response pairs. There is no polling or any asynchronous chunking. The trick is to keep the connection alive for as long as it takes. However, HTTP naturally does not allow keeping alive a connection for too long since this opens up security holes.

The client encourages the proxy not to send replies unless it actually has some data for the client. This is in contrast to where the proxy would just send empty responses, thereby wasting bandwidth. The proxy will only send a response when there is some data, after which the client immediately sends another request, which is not replied to until some data comes again. The vicious circle goes on and it is easy to see how this saves bandwidth.

Discussion: Polling vs. Binding

Binding uses a push-model as opposed to the pull-model used by polling. Theoretically, the push model is better and is the basis of many applications today. The push-model is the architecture of the future as we see application tending towards alerts and reminders rather than waiting for the users/clients to make enquiries.

Polling results in wastage of bandwidth or we could save bandwidth by increasing the polling intervals, but this only sacrifices application performance. Latency can only be reduced by making the polling interval shorter so that there tends to be a trade-off between bandwidth and latency. An HTTP binding and polling situation is illustrated below:

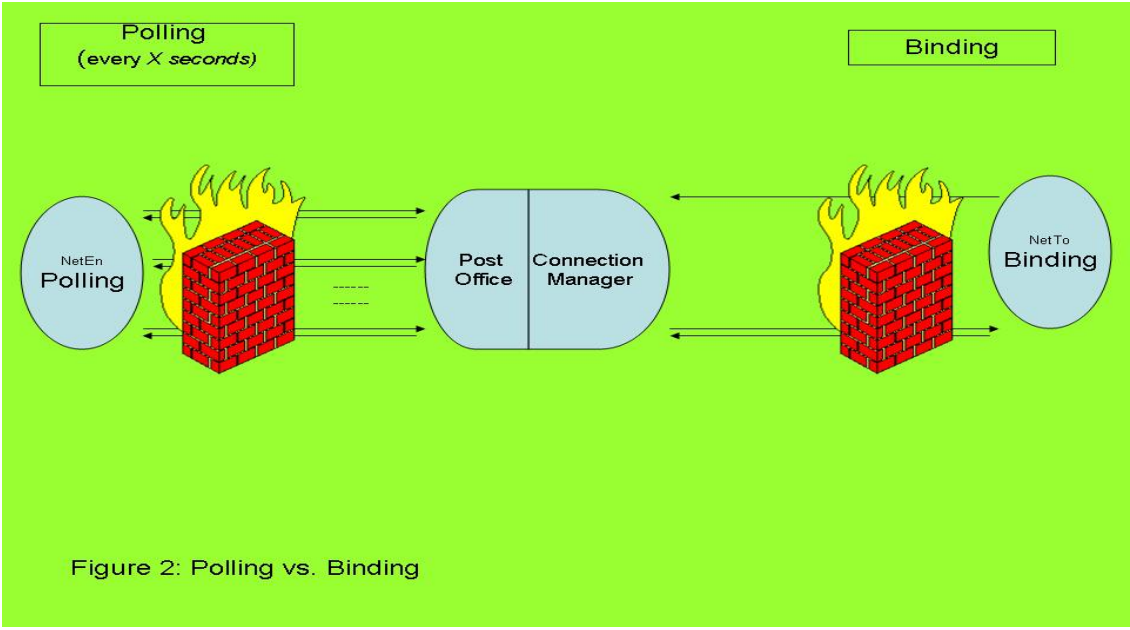

Figure 2: Polling vs. Binding

In the above diagram, we can clearly see how polling every `X seconds` can result in wastage of bandwidth by requiring a request/response pair each time. Assuming a message is available at the bottom-most arrow pairs after `10X seconds` then it is easy to see that approximately `9*message_size` is the wasted bandwidth. The previous request/response pairs were unnecessary. In contrast, binding results in the proxy responding only when there is a message, all the while keeping the connection open. However, it should be noted that the connection manager will need to send an empty response if there is no activity for too. The empty response is met with an immediate request, thus keeping the connection alive.

Interesting enough, there is no asynchronous chunking of data, for example, a post office. All the messages are forwarded by the connection manager as soon as they arrive, nullifying the need for any mail box. The dynamics of the coding itself is similar in both cases and the details are left out at this point.

Conclusion

While both methods are similar in many respects, binding is a more recent technique that works better in most situations. Binding has advantages of saving bandwidth and delivery time of messages. By keeping the connection alive, binding reduces latency. Polling works just as well, only having significant short comings with respect to bandwidth. If we consider the diminishing cost of bandwidth but increasing bandwidth available, the disadvantage tend to disappear.

Another disadvantage is that polling introduces delays in the application. If a message arrives in the post office just after a poll, then the application will have to wait for `X seconds` before it polls again.

References

1. Ian Paterson, et. Al. "*XEP-0124: Bidirectional-streams Over Synchronous HTTP (BOSH)"*, XMMP Standards Foundation, 2006. http://www.xmpp.org/extensions/xep-0124.html
2. Joe Hilderbrand, et. Al "XEP-0025: Jabber HTTP Polling" , XMPP Standards Foundation, 2002. http://www.xmpp.org/extensions/xep-0025.html