Using a Local Chat Server In Your Library

John Fink, McMaster University
Using a Local Chat Server in your Library
John Fink

The librarian-staffed academic reference desk is an endangered species. As little as five years ago the desk was the near-exclusive domain of librarians; now, in the face of declining reference desk statistics, the reference desk is increasingly captained by staff members, freeing up librarians for more intensive tasks. But for whatever gains in librarian productivity that arise from being unshackled from reference, there is something that is lost from not being in contact with the everyday pulse of the activity at the desk. Our skills, unpracticed, atrophy. Having all manner of questions from all subject areas keeps us limber, forces us to learn new skills and new methods.

Coupled with this is the notion that although the new people at the desk are undoubtedly as eager to server the public as any librarian, there are questions that they will run into that they can’t answer. Just like when, say, a business librarian is confronted with a question about the early history of Buddhism or an English subject specialist is consulted about quantum physics, anyone who is at the reference desk will need a scheme for referrals. Enough of those exist today, of course; folks on the desk can call up an appropriate authority, or make a referral for a patron for a later time. However, nothing quite beats the immediacy of having access to a group of peers, something like a backroom, or a news bullpen. A room filled with knowledgeable coworkers who can be consulted on short notice.

The backroom model has a distinct advantage over the phone call to a colleague or the direct referral in that it seeks to leverage the advantage of the group mind. Single source referral can work if there's a known authority to contact, but all too often either the known authority is unavailable or the query is too nebulous and we want to get the question out to as many people as possible so we can get it answered. But experts can be anywhere -- in the same room or across the hall maybe, but all too often they're on a different floor or in another library, or maybe even in another city.

The modern answer for this problem would seem to be to use a commercial internet instant messaging (IM) service; something like AOL, MSN, or Google Talk. Most staff are undoubtedly familiar with at least one of these. But these traditional instant messenger services are almost all bound by the same limitations as the plain old telephone; they're meant to be a one-to-one conversation method. Person A has Person B on their buddy list, person A initiates contact, person A and person B have a conversation. Virtual chat rooms can be set up, but they aren't persistent -- if person A starts a chatroom and invites person B and person C, the chat room stays up as long as A, B and C are connected. Once they go, the room goes with them.

Compounding this difficulty of group chat is that some institutions may block connections to remote IM services, or some staff might not feel comfortable having accounts on public IM services. But the chat protocol known as Jabber or XMPP is freely available in the same way that the W3C web standard is freely available, and the popular free software indexing site Freshmeat (http://freshmeat.net) is rife with XMPP implementations some of which pop up in surprising places, such as the Evergreen ILS's use of the XMPP server ejabberd or Facebook's internal chat mechanism.

XMPP has an additional advantage in that not only is it well supported on the server side, but it enjoys wide support in popular instant messenger client software as well, with just about any client that supports multiple protocols (like, say, Pidgin for Windows and Linux, Adium for OSX).

So that said, how do we implement a local chat service that is as easily usable as possible, cross platform, and -- important in these times -- free or cheap to install for both client and server? For our implementation at McMaster University, we went with a Java program called Openfire (http://www.igniterealtime.org/projects/openfire/). Because Openfire is Java, it will work on any computer with a Java implementation, which just about covers every modern computer and operating
system anyone is likely to find. We run it on Linux, but it probably could as easily be done with a Mac or a Windows PC. Upon installation, Openfire requires very little customization and most settings can be left at their defaults. When it comes to setting up users, fortunately Openfire makes the process relatively painless, and once all users have been set up, you can simply set up a chat room to allow all users to access a single chat space.

Once everything is set up satisfactorily on the server side, it’s time to move to the client. Although any client that speaks the XMPP protocol should be fine, to minimize administrative hassle it’s probably best to standardize on one or two clients. At McMaster, we’ve settled on largely using Pidgin on Windows (http://pidgin.im). After staff members login to Pidgin, they can join the backroom, which is usually amply populated with happy staff! Although the chief reason for our server is to facilitate group conversation, there’s nothing to prevent people from using this local server as something similar to traditional offsite IM with a one-to-one conversation.

Our own Openfire installation has been up for about a year now, and it’s been a success. Both for internal communication and for group chat, it’s been heavily used by staff and it’s a valuable tool both to facilitate internal communication and to route tough reference questions more effectively. From server to client, it’s free software, so it didn't cost us anything. Try it in your library, and see what happens!