**Turn your Pi in a private Internet chat service using XMPP**

As a matter of fact, Raspberry Pi is probably the best device to build an home server with low costs and low power consumption. An home server is able to do a lot of useful things: media center, home repository, power or temperature monitoring and much more.

One of the cleverer use of Raspberry Pi is an XMPP server. For those who don’t know what XMPP is: it is a standard protocol to Instant Messaging, used by GTalk for example, that provides infrastructure for an XMPP server to send messages and two or more XMPP clients to exchange messages each other.



I was searching the best XMPP server to put on Raspberry Pi when I came across Prosody (https://prosody.im/).

Prosody is a lightweight XMPP server written in LUA (a fast programming language based on C and used for game programming – so, it has to be fast!) with all basic capabilities of an XMPP server, plus some extra-modules you can add as you wish (like admin console, secure communication, etc).



With a XMPP server installed on your Raspberry Pi you can create a chat server to be used in your intranet Wi-Fi network (between you and your relatives, for example) or like a real internet chat server (like GTalk or WhatsApp) exposed through internet and under your control.

In this article we’ll see how to create an internet chat server with a private and restricted access.

Prosody has another big advantage: is in the Raspbian repository, so you can install it simply by typing

$ sudo apt-get install prosody

Once your download and installation are completed, you need to do two things: create a valid configuration file for Prosody and create users you want to be in your chat list. Obviously, if you want to expose your chat service through internet, you need to setup port-forwarding of your router, by pointing the selected port for Prosody server. We’ll see it in a minute.

You can locate the config file in /etc/prosody/prosody.cfg.lua.

Create a backup of any eventual existing file and override content with this one (you can find an example at https://prosody.im/doc/example\_config):

modules\_enabled = {

-- Generally required

"roster"; -- Allow users to have a roster. Recommended ;)

"saslauth"; -- Authentication for clients and servers.

"tls"; -- Add support for secure TLS on c2s/s2s connections

"dialback"; -- s2s dialback support

"disco"; -- Service discovery

-- Not essential, but recommended

"private"; -- Private XML storage (for room bookmarks, etc.)

"vcard"; -- Allow users to set vCards

-- Nice to have

"legacyauth"; -- Legacy authentication. Only used by some old clients and bots.

"version"; -- Replies to server version requests

"uptime"; -- Report how long server has been running

"time"; -- Let others know the time here on this server

"ping"; -- Replies to XMPP pings with pongs

"register"; -- Allow users to register on this server using a client and change passwords

-- Other specific functionality

"posix"; -- POSIX functionality, sends server to background, enables syslog, etc.

--"console"; -- telnet to port 5582 (needs console\_enabled = true)

--"bosh"; -- Enable BOSH clients, aka "Jabber over HTTP"

--"httpserver"; -- Serve static files from a directory over HTTP

};

-- Disable account creation by default, for security

-- For more information see http://prosody.im/doc/creating\_accounts

allow\_registration = false;

-- Create a pidfile for nohup launch

pidfile = "/var/run/prosody.pid"

-- Your personal domain

VirtualHost "mypersonaldomain"

Let’s go through. The “modules\_enabled” section defines which modules Prosody loads on startup. Some are required, others are optional. Feel free to select the modules you like: for example, the “posix” module is necessary for UNIX system (so, also for Raspberry Pi), while the “register” module is useful only if you want to allow external users to register to your chat service (see below).

The next parameter is “allow\_registration”: it allows the external registration. I set it to false because I want to create manually my chat service users – because I want to create a private and hidden service.

The last is “VirtualHost”: Prosody allows to create various hosts and each host can have one or more users. So, with “VirtualHost” parameter you can create many chatrooms or chat services, each one independent from others. For my purpose, I create a single domain named “mypersonaldomain”.

Prosody installation comes with an utility called “prosodyctl” that allow you to create users and manage service. The next step is to create users by using that tool: the command will ask you to provide a password for user.

$ prosodyctl adduser user1@mypersonaldomain

$ prosodyctl adduser user2@mypersonaldomain

We have created two users bounded to “mypersonaldomain” domain.

Now, restart the server:

$ prosodyctl restart

Congratulations! You got your chat service up and running.

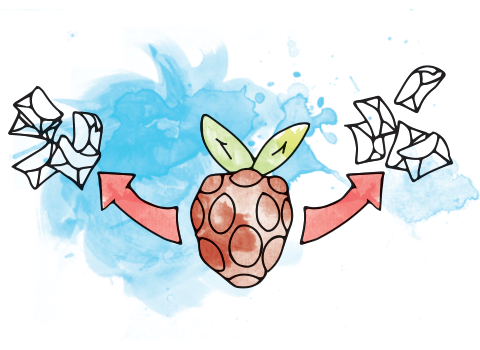
Now, to test it, let’s install some XMPP clients on your preferred devices. I have and Android phone, on which I installed “Xabber” (from Play Store) and a MacBook with “Adium”.

Both Xabber and Adium have a simple wizard to configure your account. With Xabber, you create a new account with username “user1@mypersonaldomain” and password the one you specified before.

Other important parameters you need to modify are **host** and **port**: by default Prosody runs on 5222 port (the standard XMPP port), while for the host, simply type your Raspberry Pi current IP address on network (mine is 192.168.1.10). You can find these setting in *Settings* panel > *XMPP accounts* panel.

Do the same things with Adium or with another Android device with Xabber (or with CrossTalk for iPhone – feel free to use your favourite). Once your devices are connected to chat service, simply add each other by adding a new contact; the procedure is the same for all IM services: user1 asks to add user2 and user2 has to confirm the friendship.

You should be able to send and receive messages from user1 and user2, through your XMPP clients and via your Raspberry Pi.



The last step, optional, is to publish your service on internet: by exposing port 5222 through your router and changing host configuration on your clients (move the previous 192.168.1.10 to your current router IP) you should be able to exchange messages outside your local network. Try it!

Written by Gianluca Serra



Disclaimer:

Me, Gianluca Serra, I hereby give permission to MagPi Ltd to use my name, my photograph and my article to publish them on MagPi magazine.

All trademarks are the property of their respective owners.