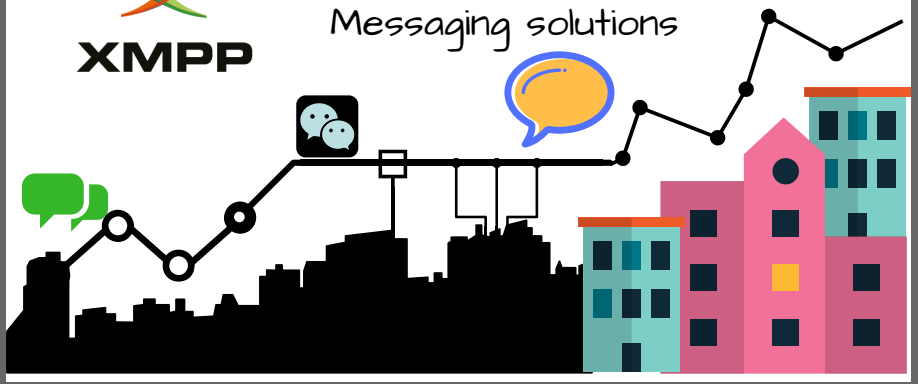
A Friendly INTRODUCTION to XMPP

By Daniel Gakwaya

Xmpp stands for eXtensible Messaging and Presence Protocol

It is an open standard protocol that is used to build real time applications and mainly Instant Messaging solutions



It uses XML as a means to exchange information

Xmpp allows you to send out chunks of XML data from one node to another

Example applications are Instant Messaging Apps, whiteboard solutions, games ,...

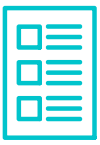
Xmpp is broken into two parts :

Xmpp Core | Xmpp Extensions

Core contains the most common & essential services like connection and sending messages

Core is made of services like :

- Data Communication & Security
- Presence and Contact Lists
- One to one messaging



Core is documented in the standard documents RFC6120 and RFC6121

The extensions, also called XEPs(Xmpp Extension Protocol),add additional and useful features to the foundation provided by core.

MUC | Http File Upload

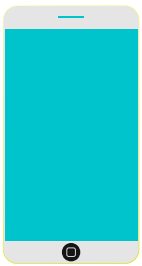
Some good examples of XEPs:

- Multi-user chat : For group chats
- Http File Upload : For sending files
- User Avatars : For profile Pictures

There is a good chance a feature you're looking for is covered in some XEP document

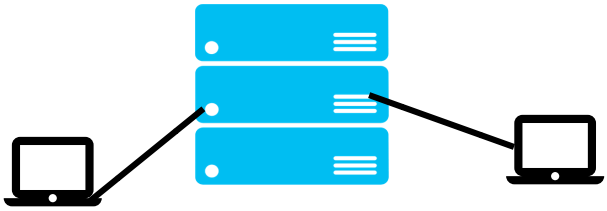
One could do something like :

One to One chat + Multi user chat + Presence + Contact Lists + Http File Upload



To create a full blown Instant Messenger Solution like whatsApp, Wechat or Viber

Xmpp is based on the Client-server Architecture

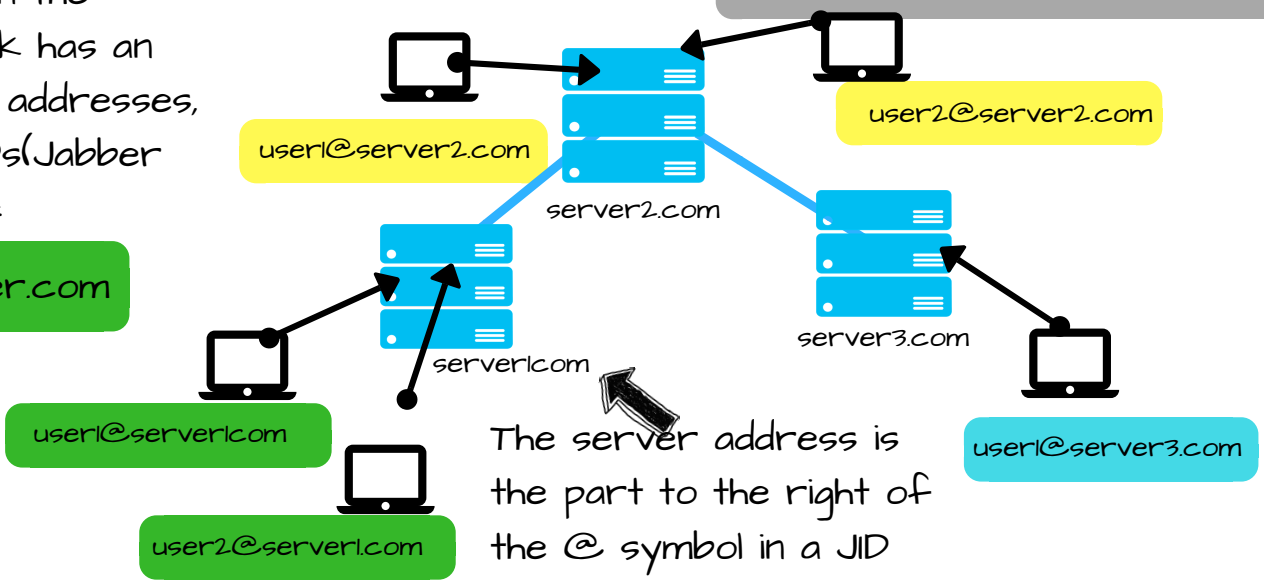


Effort is made to move complex stuff to the server. Making clients easy to develop

XMPP ARCHITECTURE

Every node on the Xmpp network has an adress. Client addresses, also called JIDs(Jabber IDs) look like :

user@server.com



The server address is the part to the right of the @ symbol in a JID

Xmpp is a federated protocol : This means that you can send messages between several completely independent servers.

FEDERATION

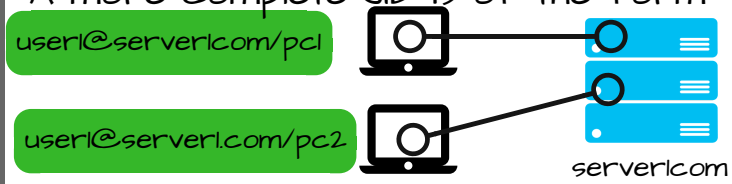
In our architecture, user1@server1.com can send a message to user1@server3.com.

server2.com

The message will go through server2.com

server2.com will be smart enough to know that the message is not his, and forward it to server3.com. server3.com will deliver the message to the connected user

A more complete JID is of the form : user@server.com/resource



The resource part in a JID identifies the device you are using to connect to your Xmpp server(pc1,pc2)

In Xmpp, you can use one account user1@server1.com to connect from multiple devices (pc1,pc2)

- A JID of the form user1@server1.com is called a bare JID.
- A JID of the form user1@server1.com/ps is called a full JID

Xmpp allows us to exchange chunks of XML data between clients and servers

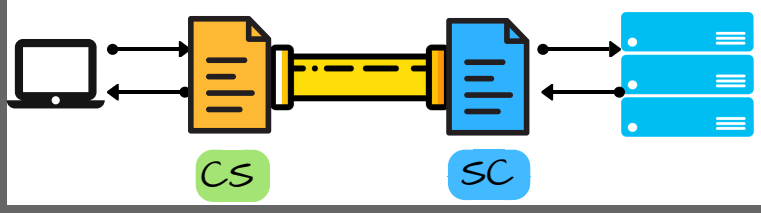
Before the exchange of XML chunks, an XMPP client has to

- Open a long lived TCP connection to the server
- Inside the TCP connection, negociate an XML stream to the server

When the server accepts,

- It also opens an XML stream to the client

We have to streams in total : a stream from client to server (CS) and from server to client (SC)



There are two stream files : CS and SC

When a client sends a message to the server, it goes throgh CS and down through the TCP connection to the server

The same happens from server to client : first through SC and off to the client

Messages go through stream files before reaching the destination

Stream files are great for debugging. Many Xmpp libraries provide a way to connect to them

COMMUNICATION PRIMITIVES

My name is stanza, and I own you in Xmpp!

A stanza is the most basic unit of communicatio n in Xmpp

Three types of stanzas in Xmpp:

- MESSAGE
- PRESENCE
- IQ

Each of these stanzas is handled differently by Xmpp clients and servers

Message stanzas are used to send data between Xmpp entities

They are not acknowledged. When you get no error after sending, you can assume that they reached the destination.

Different types of messages : chat, group chat, error.

Chat message stanzas are used to send one-to-one messages

Group-chat message stanzas, used for sending group messages (MUC)

Error message stanzas are used to report back that something went south(wrong)

Here is what a message stanza looks like in XML :

```
<message
    from='juliet@example.com/balcony'
    id='ktx72v49'
    to='romeo@example.net'
    type='chat'>
  <body>Art thou not Romeo, and a Montague?
</body>
</message>
```

It has a tag name : message

It has attributes : from , id , to, type

It also has a nested child body XML tag, that contains the meat of the message we want to send

The attributes specify the source JID, the id(tracking number), destination JID and the type of message stanza

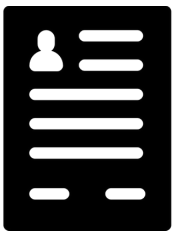
Xmpp entities look at all this info, and know what to do with the stanza

online

offline

The presence stanza is used to advertise online status information

When you log in your Xmpp client and go online, your Xmpp client uses Presence stanzas to notify your contacts. That you are online



Presence(online status advertisement) follows the subscription model : to see your contact's presence(online status) you need to explicitly ask for that

Subscription (friendship) is also controlled using presence stanzas

Different types of presence stanza :

- Online status : chat, away ,xa , dnd
- Subscription : subscribe unsubscribe,subscribed, unsubscribed

Presence is a double duty stanza : It can do online status and subscription

Here is what a presence stanza looks like in XML:

```
<presence
from='juliet@example.com/balcony'
id='pres2'>
  <show>away</show>
  <status>stepped away</status>
</presence>
```

It has most of the attributes we saw on the message stanza and it can contain child elements to carry additional information : show,status

IQ in IQ stanza stands for Info Query

IQ STANZA

IQ stanzas are used to :

- get one of pieces of data from the Xmpp server
- apply some settings to the server

Similar to GET and PUT requests in Http

Two types of IQ stanzas : set and get,result and error

If you wanted to get your contact list from the server, your Xmpp client would send an IQ-get stanza :

```
<iq
from='juliet@example.com/balcony'
      id='bv1bs71f'
      type='get'>
  <query xmlns='jabber:iq:roster'/>
</iq>
```

The <query xmlns='jabber:iq:roster' /> child element is what denotes that you want to get your contact list. type='get' simply specifies that you want to get something.

The server would respond with an IQ-result stanza

```
<iq id='bv1bs71f'
      to='juliet@example.com/chamber'
      type='result'>
  <query xmlns='jabber:iq:roster' ver='ver7'>
    <item jid='nurse@example.com'/>
    <item jid='romeo@example.net'/>
  </query>
</iq>
```

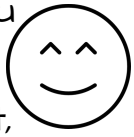
If is request had failed, the server would have responded with and IQ-error stanza

But we were successful and we got our contact list of two people(JIDs)!

Now, do I need to go down to the metal and build my own Xmpp infrastructure if I wanted to write an Xmpp client or server ?

XMPP SOFTWARE

You don't have to, there are tons of Xmpp software projects you can base your work upon whether you are writing a client, server or library



You can learn more by visiting www.xmpp.org

Xmpp is a really fun and powerful protocol. This booklet is intended to break down the basics into easy to digest pieces

Hoping you liked it so far

Who writes these booklets ?

Hey, I'm Daniel, I'm a software engineer and I write & make videos about various tech topics

You can catch me on various social media channels (follow& subscribe for updates) :

https://bit.ly/2JJTdJP

https://bit.ly/2LJMxJ4

https://bit.ly/2Jx40Yz

I also have some detailed video courses about Xmpp on our online school:

Build an Android XMPP Chat App using XMPP and Smack

https://bit.ly/2TNm06J

Android XMPP Chat App : Sending and Receiving Files

https://bit.ly/2X5c3i7

Would like to see more booklets like this ? Please let me know on twitter.

ROCKS