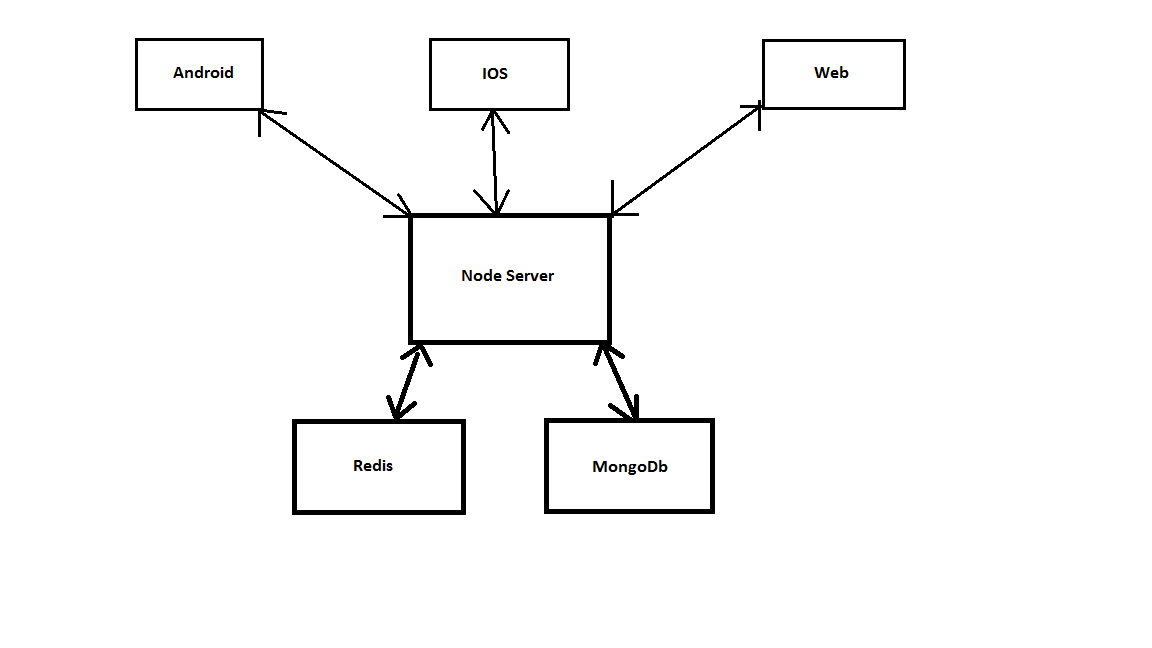
**Chat server implementation using Redis, Node and MongoDb**

Almost all the modern real-time communication apps are using redis somewhere in their systems.

Hike app is popular these days in India and is a good example, read : <https://medium.com/the-work-we-do-hikeapp/storing-hundreds-of-millions-of-simple-keys-in-282-mb-with-redis-2142b7c52cc9>

Redis pub sub is not persistent, however redis has five data types. For chat we can use combination of **hashes**, **lists** and **streams**. They are persistent, so we can use them for offline message handling.

We can also configure redis to back up its data regularly after intervals in rdb file or aof. And that is asynchronous.

**Basic Message Format:**

Message : {

“From” : “50”,

“To” : “45”,

“ChannelName“50:45“”,

“MessageType” : “1”,

“MessageDeliveryStatus” : “0”,

“IsDeleted”: “0”

}

export enum MessageTypes {

TEXT,

IMAGE,

TYPING,

STOP\_TYPING,

AUDIO,

VIDEO

}

export enum MessageDeliveryStatuses {

pending = 0,

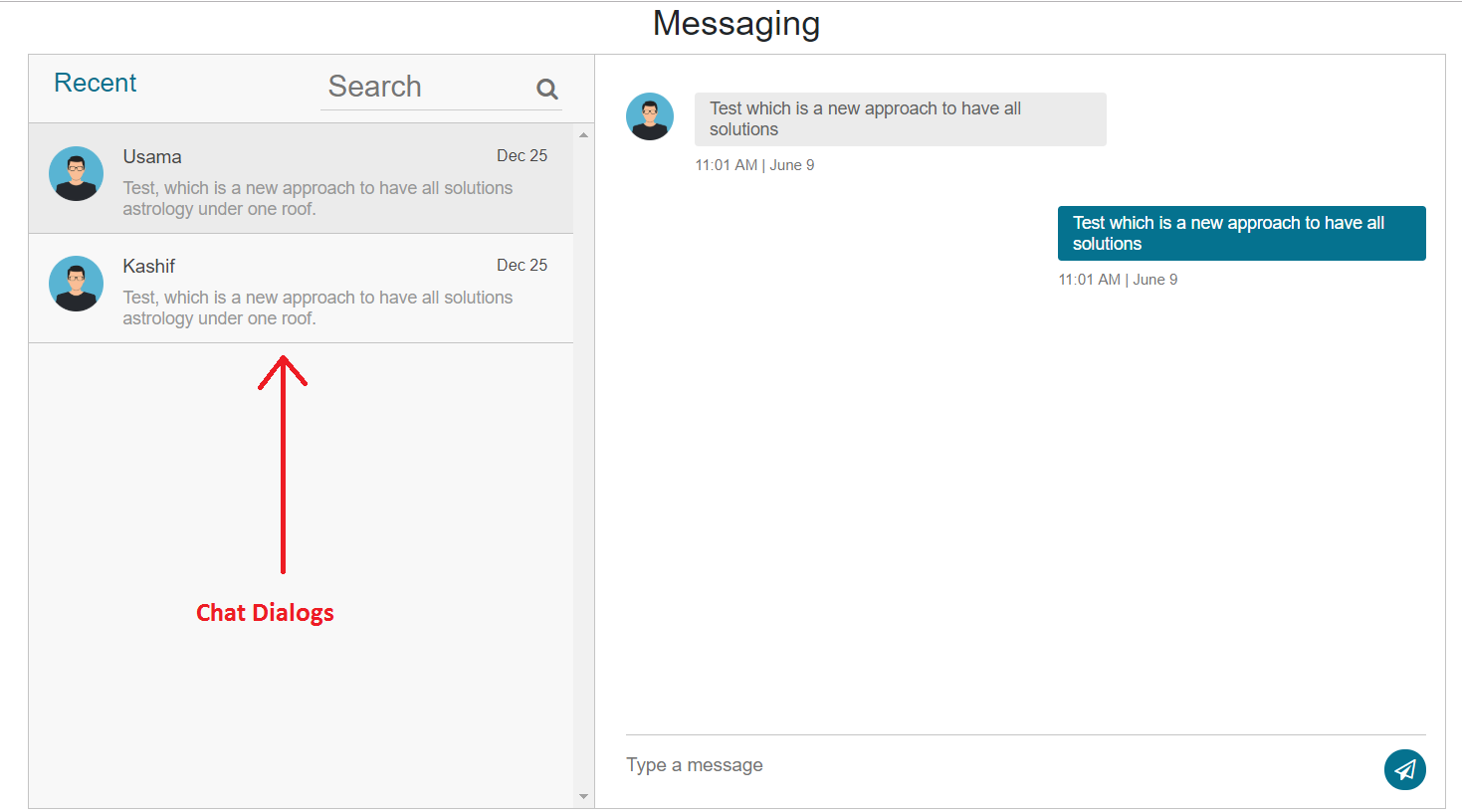
delivered = 1,

seen = 2,

}

**1 : 1 chat Dialog Format:**

“ChannelName” : “50:45”



Every User will have its list which contains its chatDialogs.

923325345126:98745632154

Push Notifications will be sent against each message asynchronously.

**Flow:**

When every push notification receives, the app user will get channel name out of it and will subscribe to it if didn’t already.

Mohsin sends a message to usama, Server creates their channel if new. If new, Usama’s app will receive push notification and will check if he has subscribed to channel, if not, subscribe to it.

**Use case:**

The receiver isn’t subscribed to channel, so he will not receive message even if he is online, but when push notification arrives, he checks if he has subscribed to this channel, if yes, ignore, if no subscribe.

Now for further messages, server will save and publish on their channel, both client will receive but sender will ignore based on it’s from id.

For message,

Client will subscribe to his chatdialog channels, for e.g

Foreach(var chatDialog in ChatDialogs)

{

Socket.subscribe(chatDialog, function MessageReceived)

}

Everyone will have hisId:Presence channel

Socket.subscribe(3:Presence)

So, for all presences there will be one functionBody and for all message received, there will be one message received function body.

Chat Dialogs Screen:

We see chat dialogs besides chat windows in almost every app. A chat dialog is either a 1-1 chat or a group chat. User gets its all chat dialogs from server. Chatdialogs will have name and channel properties. Let’s take below example.

{

“Dialogs” : [

{

“DisplayName” : “Usama”

“ChannelName” : “50:45”

},

{

“DisplayName” : “Usama”

“ChannelName” : “50:45”

},

{

“DisplayName” : “Now Soft Devs ”

“ChannelName” : “f34bsd789dfnk9lkjkdf8”

}

]

}

Every user has a list of chatdialogs saved against him in redis with key = chatDialogs:50

When new chat is created and user is offline, chatDialog is created at server and when user will come online, it will be given his list of all chatDialogs.

Server will also return last message against each chatDialogs.

**Features:**

**Online Offline presence:**

Every user will subscribe to presenceChannels of his all chatDialogs.

Every user will have 50:Presence channel subscribed.

{

“DisplayName” : “Usama”,

“UserId” : “45”,

“ChannelName”:””,

“IsOnline” : 0

}

**Offline Message Sending:**

When a user will fetch its chatDialogs, Last message will be returned against each of them + unread count of that chat dialog.

Onclick on each chat dialog user can fetch messages by paging. For e.g 20, when scrolls up for older messages, again will fetch next page size and so on.

We can have presence as a message.

Data Structures:

Lists for Messages

Lists have features like fetching based on paging/indexes, update, push pop from top or bottom

Sets for contacts

Sets cannot have duplicates in it. So we just need to push into users contact list without verifying that it exists or not.

Hashes:

For storing lastMsg and unreadCount against user’s chat dialogs.

Lag issue for concurrent connections <https://stackoverflow.com/questions/49091270/why-is-socket-io-is-very-slow-on-windows>