A Brief Analysis on SignalR Library Function for Real Time Applications

Ms. Santoshi Savant[1], Mr. Pavan Kumar[2], Dr. Jayarekha P [3]
[1]MTech Student, Department of Infomation Science, BMSCE, Bangalore
[2]Team Lead, Prysm India Ltd,Bangalore
[3] Associative Professor, Department of Infomation Science, BMSCE, Bangalore

Abstract

Signalr is the library function developed by Microsoft mainly for building application with real time functionality. The main examples where SignalR concepts are used in social media application, weather forecast and dashboard used mainly in company. It mains includes the Asp.net and javascript involved in having successfully communication between server and client. It is mainly used where frequently updates are needed to be sent to the client, push content operation. In SignalR, the persistent connection is established between server and client. Unlike the Http request response model, SignalR maintains the open connection.

Keywords: *SignalR, Asp.net, javascript*

I. INTRODUCTION

Real time functionality comes into picture when the user receives the update whenever there is information available at server side. In most of the previous applications the real time functionality was not been completely implemented as it was partially seen in many applications. The real time functionality helps the user to get information instantly whenever the server buffer has the information available. In case of website the real time functionality the user gets the new updates from the server without refreshing the page [1].

In figure 1.1 represents the architecture of the signalr server-client. The server end contains various layers such as:

- Hub Api
- Persistent connection API
- In transport layers:
  - Web sockets module
  - Server sent events module
  - Forever frame module
  - Long polling module

The Client end contains various layers such as:

- Hub Api
- Persistent connection API
- In transport layers:
  - Web sockets module
  - Server sent events module
  - Forever frame module
  - Long polling module

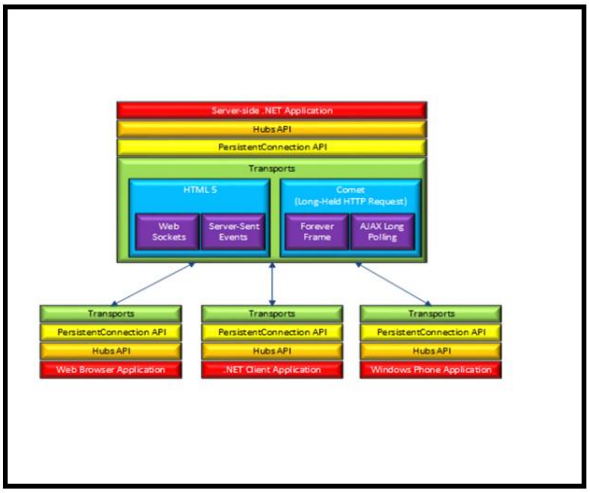


Fig 1.1: Server-client architecture of signalr

The server side module in transport layer which one of the module is supported by both and client and server.

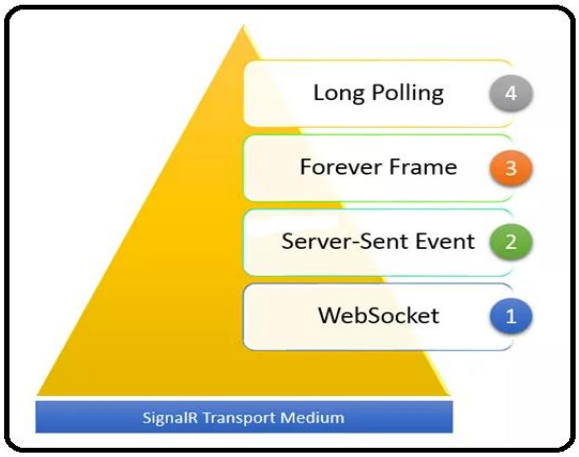


Fig 1.2: Priority of transmission mode

In chat application the real time functionality is handled by the signalr library. By extending the feature of the hub which not only broadcasts the messages to the connected users

.

The figure 1.2 depicts the priority considered while choosing the transmission mode.

A. Web socket module:

This module is used as the transport medium if the browser at the client side is higher version of chrome and Mozilla browser. The figure 1.3, depicts the websocket module.

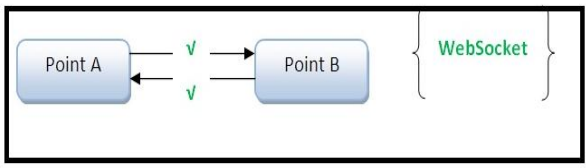


Fig 1.3: Web socket module

It provides server and client to communicate in both direction. This is the first choice taken by the server [1]. This transport medium module provides complete real time functionality.

B. Server sent event module

This module is the next choice done by the server while choosing transport medium. It supports all the client version browser but expect with an exception internet explorer [1]. The figure 1.4 depicts the server sent module.

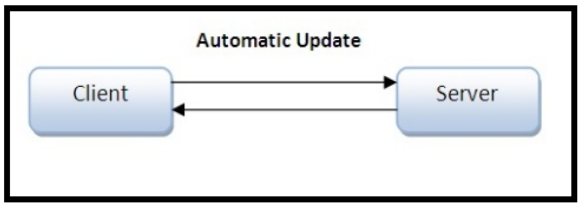


Fig 1.4: Server sent module

Here the server sends continuously request to client to keep the connection between client and server open.

C. Forever frame module

This is third choice or priority given while choosing the transport medium. It is supported only if the client side browser is Internet explorer [1]. The Fig 1.5 depicts the forever frame.

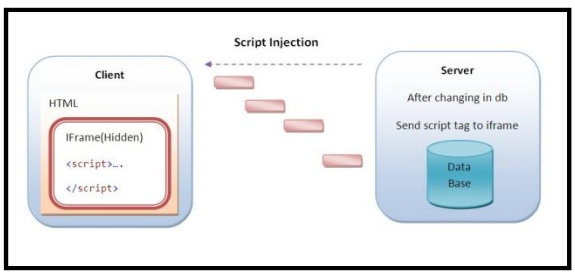


Fig 1.5 Forever frame module

Here the client module continuously sends the frame which are almost hidden to server, server in response sends certain scripts to keep the connection alive. The server and client do not use same connection for sending and receiving.

D. Long polling module

This last option or priority given while choosing the transport medium. Here the client keeps on sending request to the server inorder to keep connection alive. The figure 1.6 depicts the polling module

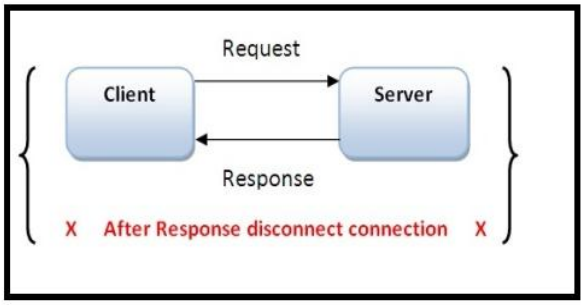


Fig 1.6: Long polling module

This module does not provide the real time functionality. Many requests results in traffic congestion in the network and thus results in delay in the response.

The figure 1.7 depicts the websocket transmission mode used in chat application using signalr. The transmission mode used can be found in developing tool in visual studio console.

Fig 1.7: Websocket transmission mode

The figure 1.8 depicts the console with polling method been used when client does not support the websocket as the transmission mode.

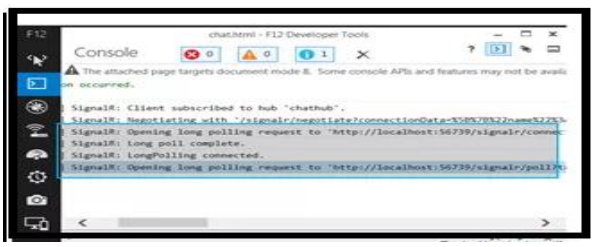


Fig 1.8: Long polling transmission mode

The figure 1.9 depicts the flow diagram of signalr transmission module

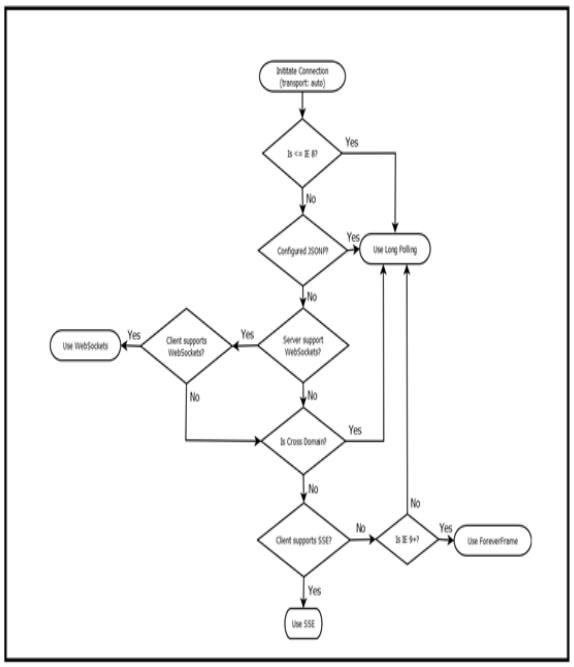


Fig 1.9: Flow diagram of signalr transmission module

Signalr utilizes any one model listed below for the mode of communication shown in the figure 1.11:

E. Hub mode of communication

As shown in the figure 1.1, the server end sends a packet to the client. This packet is send across the transport medium. This procedure is mainly done to poll the client method. This packet is encapsulated in json format. The client method verifies the packet whether this packet matches the method name. If this method name matches the name in the packet the method is executed. There is tool called fiddler used for verifying such method execution [1].

```
10:38:03:1298 Session846.WebSocket'WebSocket #846' - Pushing 104 bytes from server WebSocket
31 66 7B 22 43 22 3A 22 42 2C 31 35 7C 43 2C 30     f{"C":"B,15|C,0
7C 44 2C 30 7C 45 2C 30 22 2C 22 4D 22 3A 5B 7B  |D,0|E,0","M":[{
22 48 22 3A 22 4D 6F 76 65 53 68 61 70 65 48 75  "H":"MoveShapeHu
62 22 2C 22 4D 22 3A 22 75 70 64 61 74 65 53 68  b","M":"updateSh
61 70 65 22 2C 22 41 22 3A 5B 7B 22 6C 65 66 74  ape","A":[{"left
22 3A 35 30 31 2E 30 2C 22 74 6F 70 22 3A 33 30  ":501.0,"top":30
32 2E 30 7D 5D 7D 5D 7D                          2.0}]}]}
```

Fig 1.10: Packet format

In the above figure 1.10

- The 'H' -mentioned in packet represents the hub name
- 'M' represents the method name
- 'A' represents piece of data sent for the method execution [1].

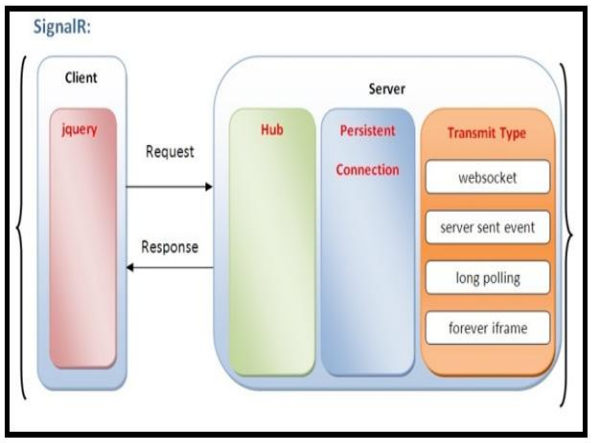


Fig 1.11: Connection model of signalr at the server end

F. Persistent connection module

The hub connection model used the json format, here the packet is sent in the raw format from server to client. It mainly deals with lower level layer of the application. In most cases the hub model is used as it is implemented over persistent connection model which includes internally the lower level specification in persistent connection.

II. LITERATURE SURVEY

Detailed study was conducted to understand server client concepts, Signalr, Hubs, Real time concepts, Synchronization and owin and signalr packages to implement the system. The main conclusions from study are noted below.

A. Microsoft module package for the application framework

Server module is responsible and keeps listening to the request coming from client in order to serve the client module.

In other hand, Client keeps listening to the response back from the server for the request sent

.

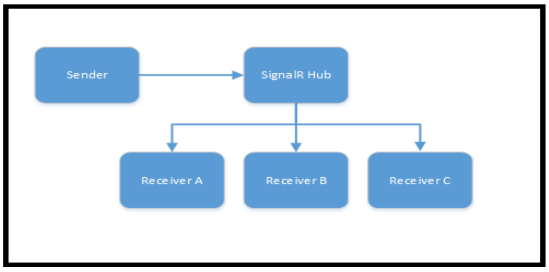
If the client that is the browser, if it older version the framework has the automatic features to handle it out.

The server module plays prominent role in serving the client module, internally helping in maintaining the resources factor. The server module is designed in such way that it provides feasible way for client to request the required information.

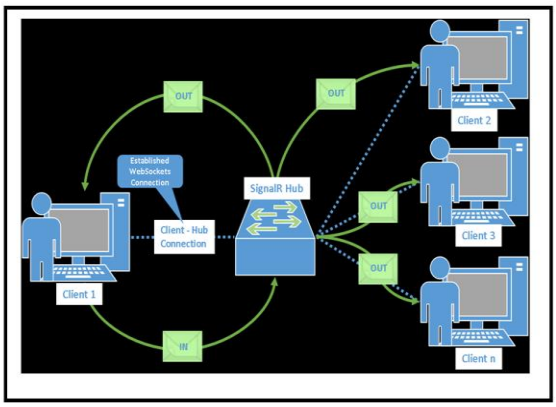
B. Pipeline module between client module and server module

It's predominately called as hub module, it provides the pipeline feature between client and

server module, which provides the client and server module to communicate within this open connection.

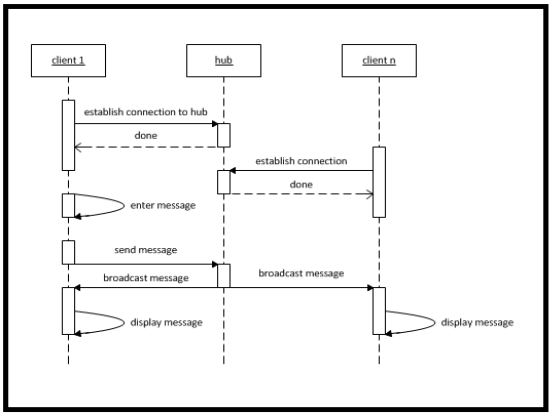


Internally helps in making procedure calls within the client and server module. Once the pipelining module is build the client and interaction begins, initially the server polls the client module method by sending a packet. This packet is sent from server to client module, across the transport layer. The packet mainly contains the information packed in json format. The parameters such as name of the client and related parameters are been listed on the packet.



The above figure depicts how the message from the signalr pipeline hub module is been broadcasted to all the connected client. The packet is being broadcasted to all the connected clients, each client takes the packet and validation process starts, the packet contains the method name. If the method of packets matches the client module method, then client module that method part is been executed. This allows the desired functionality to be built on the server module.

The figure depicts how the connection is established between the client module and hub module. Once the connection is established the client creates the message and then sends it to hub module. The hub module that later broadcast message to all the connected clients. This message is displayed on all the connected clients.



C. The building block of the chat application

The building block for the chat application is the library function readily in Microsoft framework module, signalr helps building the application which is exceptional and incorporates the features of real time functionality. It has the packages which provide feasible and simplest way of creating web application module that incorporates the real time functionality. It has the functionality of the broadcasting as built in, as it server has asp.net and javascript that helps in creating real time environment for the web applications.

D. Interface module

For any application to communicate an interface in required, in real time environment for the signalr to be implemented predominately interface module called owin is used. This specifies the application that using the environment which uses the package of the building block of the application signalr. With help of this interface module we can provide configuration required to the application.

E. Model use for transparency

The chat application uses the structured model, MVC [6] that provides transparency. This facilitates the application to viewed in three components. Basically, the model provides the features of the user interact with the application and how internally it is controlled and updated in the database internally.

Unit test structures in signalr2 [2,5] to produce unit tests for the signalr2 related applications. Signalr fuses the IHub Caller Connection Context interface module, which is utilized to make a reenacted protest mimic the center point strategies for testing reason.

III. WORKING OF SIGNALR

SignalR provides the API mainly for creating the RPC, remote procedure calls that

internally calls the client methods through javascripts. SignalR provides API for the connection management such as when the connection is established and for what particular section.

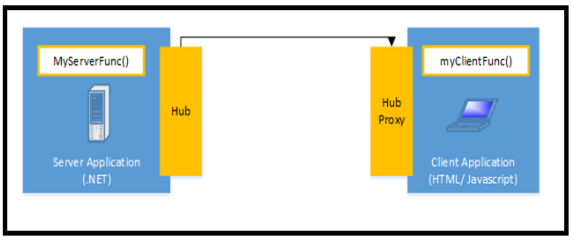


Fig: Server hub polling the proxy client hub

SignalR usally handles connection automatically, and help in broadcasting message to all the connected users as shown in the Fig. SignalR supports the push code functionality, that is whenever the server has any information it calls out thee client method
RPC.

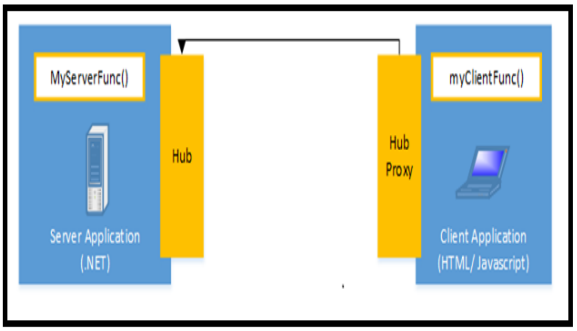


Fig : Client proxy hub calling the server hub

SignalR applications can scaleout using the database such as Redis, MSSQL.

IV. TESTING AND ANALYSIS

A. Dynamic mode of testing steps

1. Install the Xunit Runner extension package for Visual Studio 2013.
2. Download the source code of completed application readily available from MSDN Code Gallery.
3. Once the Download is completed, open package manager console in visual studio and add the signalr package to the project by clicking the restore option.
4. Next step once the signalr is added to the project is adding the unit test into the project solution. Right click under solution explorer and add the unit test module.
5. In the C# node module select the windows node module under that select the class library module and rename it.
6. Reference are to be added to the newly created module. Under the projects node which is sub section of the solution node under the given project.
7. Next step is adding the relevant packages to the project. Add the Xunit and Moq package into the project.
8. Run the below commands in console window
   - Install-Package Microsoft.A spNet.SignalR
   - Install-Package Moq
   - Install-Package Xunit
9. Next step is to create the test file. The test file is added to the project.

Add the contents of Tests.cs with the subsequent code.

B. Signalr Performance counters module

The utilities package installs the below performance counters. The total securities measure parameters that the number of procedures since the last request pool or after server restarts.

C. Connection module

The below metrics parameters measure the connection lifetime events that occur. These are Handling Connection Lifetime Events.

- Connections Connected
- Connections Reconnected
- Connections Disconnected
- Connections Current

D. *Message module*

The below are the metrics that measure the message traffic generated by Signalr.

- Connection Messages Received Total
- Connection Messages Sent Total
- Connection Messages Received per sec
- Connection Messages Sent per sec

E. Message bus module

The beneath measurements parameters measure the quantity of movement related through within Signalr message transport module, the line in inside all approaching and active Signalr messages are lined in this module. At whatever point the message is accessible it is sent or communicate. A customer in the setting is a membership on the message transport which for this situation should level with the quantity of customers including the server itself. An Allocated Worker module is a segment that transmits information to dynamic associations module. The Busy Worker module work is to effectively communicate something specific.

- Message Bus Messages Received Total
- Message Bus Messages Received per sec
- Message Bus Messages Published Total
- Message Bus Messages Published per sec
- Message Bus Subscribers Current
- Message Bus Subscribers Total
- Message Bus Subscribers per sec
- Message Bus Allocated Workers
- Message Bus Busy Workers
- Message Bus Topics Current

F. Error module

The below metrics parameters measure the number of errors generated by Signalr message traffic. The **Hub Resolution module** related errors occur when a hub or hub module when it cannot be resolved within the module. **Hub invocation Request** errors are exceptions thrown during hub method is invoked. **Transport** errors are connection errors thrown during an HTTP request or response session.

- Errors: All total
- Errors: All per sec
- Errors: Hub Resolution total
- Errors: Hub Resolution per sec
- Errors: Hub Invocation Total
- Errors: Hub Invocation per sec
- Errors: Transport total
- Errors: Transport per sec

G. Scaleout module

The below metrics parameters measure the traffic and errors which are generated by the scaleout provider module. A **Stream** module in this framework is a scale unit used by the scaleout provider, this is a table if MSSQL Server is been used along with a Topic if Service Bus is been used, and a Subscription if Redis database is used for scaling purpose. Each stream ensures ordered read and write operations; a single stream is a potential scale bottleneck, so the number of streams can be increased to help reduce that bottleneck. If multiple streams are used, SignalR will automatically distribute or broadcast messages across the related streams in a way such that it ensures messages sent from any given connection are in delivered in the order they have been sent.

The MaxQueueLength module parameter controls the length factor of the scaleout send queue maintained by Signalr. Changing it to a value greater than 0 will place all posts in a send queue to be sent one by one to the organized messaging backplane. If the scope of the queue goes above the configured length, subsequent configured length, subsequent calls to send will fail immediately with an InvalidOperationException error message until the number of messages in the queue is less than the length setting again. Queueing is disabled by default because the implemented backplanes generally have their own queuing or flow-control in place. In the case of MSSQL Server, linking pooling successfully limits the number of sends going on at any one time.

By default, any one stream method can be used for MSSQL Server and Redis, total of five streams are used for Service Bus, and queueing setting is disabled but these settings can be altered through configuration on MSSQL Server and Service Bus whenever required depending upon the request.

.NET Server module for configuring table keep the count and queue length for SQL Server as the backplane

```
var connectionString = "(your connection string)";
var config = new SqlScaleoutConfiguration(connectionString) {
TableCount = 3,
MaxQueueLength = 50
};
GlobalHost.DependencyResolver.UseSqlServer(config);
```

.NET Server module for configuring to keep the count and queue length for Service Bus as the backplane.

.

V. CONCLUSION

SignalR provides the real time functionality to the web based application in the simplest way. SignalR concept is widely used where the updates from the server are frequent and through signalR synchronization among the users connected server can be well achieved.

ACKOWLEDGMENT

I would like to express my profound gratitude to Prysm India Ltd, Bangalore and BMSCE, Bangalore for the guidance and support in all my endeavors.

REFERENCES

[1] https://docs.microsoft.com/en us/aspnet/signalr/overview/getting started/introduction-to-signalr

[2] https://docs.microsoft.com/en-us/aspnet/signalr/overview/testing-and-debugging/

[3] https://stackoverflow.com/questions/22881493/system-requirements-for-running-a-signalr-chat-application

[4] https://doc.akka.io/docs/akka-http/current/common/http-model.html

[5] https://github.com/SignalR/SignalR/wiki

[6] https://developer.mozilla.org/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture