Real Time Apps Using SignalR

Anurag Choudhry
Solution Architect,Banking Technology Group,Tata Consultancy Services,New Delhi, India

Anshu Premchand
Lead, Presales & Solutions,Banking Technology Group,Tata Consultancy Services,Chennai, India

***Abstract*— Real-time applications can be defined as applications that respond within a defined timeframe that can be considered by user as immediate or current. Real-time software applications are different from other real-time systems. Real-time applications deliver information instantly, and increase interaction & engagement between system and users. We have been using real-time technologies for more than a decade, but in the last couple of years, we have seen them in applications that we use frequently. Real-time technologies have matured and users are demanding more of such applications that provide excellent user experience and can engage users for a long duration as well. Therefore, real time technologies and applications have become the integral part of the organizations IT eco system across industry verticals.**

In this paper, we shall strive to discuss real-time technologies, use cases, challenges with the existing real-time communication mechanism, how SignalR can be used to resolve these issues and in the last section we will look at SignalR security, design considerations and performance improvement as well.

***Keywords*— Real-time applications, Real-time technologies, SignalR.**

I. INTRODUCTION

In real-time applications data is delivered to the subscriber as soon as it is published by the publisher/author. Real-time applications are used across several industry domains such as Banking (Fraud detection, transactions management, trading etc.), transport (Air traffic control systems), collaborations (Google docs), social networking (Facebook, twitter, Google+). We have been using various technologies such as HTTP long pooling, Forever Frame, Server Sent Events, and WebSockets etc. for developing real-time applications. All these technologies have some limitations. Microsoft introduced SignalR, which is part of ASP.NET libraries. It uses existing transport techniques to provide real-time communication and resolves the issues with existing real-time technologies.

II. USE CASES OF REAL TIME APPS

With technology gaining importance in all aspects of our existence around the world, real-time technology on mobile and web is one the fastest emerging areas of computing. Real-time web will be the foundation of next web (say Web 3.0). There is various use cases of real-time apps, key ones are as follows:

A. Banking and Financial Services (BFS)

In BFS industry, real-time technologies are used in various areas mainly in capital markets where several push-based messaging patterns are used. For instance, clients subscribe to services that publish stock prices, broadcast messages &real-time charts etc.

B. E-Commerce

This is an emerging area where real-time technologies are being used to engage and interact with customers. Real-time technologies allow retailers to watch as customer shop on their sites and accordingly offer discounts and customer support to convert web visits into sales. Showing the related products or pushing online hot deals to all connected customers is the real-time apps features for e-commerce segment.

C. Real-time Analytics

Nowadays many organizations are working on real-time analytics, the biggest push is coming from the technology giant Google. With real-time technologies we have the option to capture interactive user data such as mouse over, mouse click and other user interactions.

D. Publishing

Key goal of online publishers is to keep customers engaged. Real-time data can be used to generate visuals of complex data which helps users in analysis.

E. Online Games

Multiplayer online games depend on low latency communication between players and this is one of the key use cases for real-time technology.

F. Collaboration

Large organizations are increasingly emphasizing on collaboration between teams as this impacts productivity of team members positively and can cut cost significantly. There are various tools in the market that are used for collaboration such as on-line meetings, web-based trainings, in-browser IDE and so on.

G. Chat

Chat is the de-facto example of real-time technology as it requires bi-directional real-time communication between chat users. With the evolution of real-time technologies, there are various opportunities to the modernize chat applications.

H. Monitoring Services

Organizations use central monitoring services where remote devices and servers are connected to the central monitoring services. This service helps to watch endpoints without logging into the remote machines and allows sending alerts. Bi-directional real-time communication is used for this kind of monitoring services.

I. Social Networking Apps

Gigantic social applications such as Facebook, Twitter and Google+ use real-time web technologies for instant messaging and interactive experience.

Hence, it's clear from the above use cases that real-time technologies are generating huge opportunities for the IT industry. Real time technologies & apps are going to be the key driver of all our web interaction in near future; they are already driving most of our online footprint.

III. TECHNOLOGIES TO DEVELOP REAL-TIME APPS

In order to build real-time apps there needs to be a bi-directional communication between client and server. Using persistent connection between client and server, information can be delivered instantly. Real-time communication is easy in thick client applications where queuing infrastructure or message oriented middleware is used and client is connected over TCP. Real-time communication in web based applications is harder to implement. Traditionally, HTTP is used for the communication between client (browser) and server which is uni-directional in nature and built on request/response model. Real-time communication can be achieved by using various techniques. Key ones are as follows:

A. Java Applets

In earlier days, bi-directional communication was achieved by using Java applets. Applets can make persistent connection to server and communicate with JavaScript in the page. But this concept could not be accepted widely due to dependency on browsers plugins.

B. Techniques based on HTTP

To remove the dependency of plugins, HTTP Polling, HTTP Long Polling and HTTP Streaming are used for real-time communication between client and server. In all three techniques polling, long polling and streaming, client sends request to server for data and server send the information to client but main difference amongst these three techniques is as follows.

1) *HTTP Polling*: In HTTP polling, client makes a request and waits for the server to send information. If there is no information available, server sends an empty response.
2) *HTTP Long Polling*: In HTTP long polling, server hold the connection until new information is available and sends the same to client and closes the connection.
3) *HTTP Streaming*: In HTTP streaming, the connection is not closed after sending the new information to client.

C. Forever Frame

Forever Frame uses hidden IFrame to push data from server to client. In this technique, a set of JavaScript commands is sent to IFrame and scripts are executed as events occur for real-time communication from server to client. From client to server communication, new connection opens for the data that needs to be sent to server. Forever Frame works only in Internet Explorer.

D. Server-Sent Events

Serer-Sent Events (SSE) is a mechanism for getting data from server via HTTP connection using a JavaScript API called EventSource. SSE is uni-directional HTTP streaming connection for publishing a stream of data. This can be useful in some cases but when rich interaction is required, bi-directional communication becomes mandatory.

E. WebSocket

WebSockets allow bi-directional, persistent, full duplex connection between client and server. EventSource and WebSockets are part of HTML5 standards. WebSockets have become the standard solution for real-time interactive application development and have many advantages over the above mentioned techniques. WebSockets are natively built into browsers hence do not depend on browser plugins like Applet, Flash and Silverlight. Similarly, WebSockets have advantage over HTTP streaming, extra effort for parsing of data received is not required while using WebSockets.

In our experience, all HTTP based techniques are inefficient as client and server resources usage causes overhead. Also, implementation of these techniques is different across different browsers. SSE is not supported by Internet Explorer and WebSockets are only supported by new browsers and servers that support HTML5. Also, WebSockets need to be supported by intermediate proxy layers.

So, in order to use these technologies, there are two options, first is to design the solution which is supported by the existing infrastructure. The second option is to implement "if-else" kind of logic to detect the type of client, intermediaries and server and then choose the right technology. Therefore, in implementations of these technologies it is difficult to achieve real-time communication and this is where SignalR comes into the picture.

IV. SIGNALR

SignalR is a new ASP.NET library, which uses existing transport technologies based on the infrastructure. SignalR has the capability of real-time communication with wide range of clients such as Web Applications, Windows Applications, and Windows Phone Apps etc. Now, developers don't need to worry about transport mechanism and do not need to decide on fallbacks if infrastructure does not support a particular technology.

Below given diagram describes how SignalR decides the transport mechanism based on the infrastructure available.

```
If (browser <= IE8)
        {
        Use Long Polling;
        }
Else
        {
        If (JSONP = Configured)
                {
                Use Long Polling;
                }
        Else
                {
                If (connection is cross-domain)
                        {
                        If (client supports CORS & Server supports WebSockets & Client supports WebSockets)
                                {
                                Use WebSockets;
                                }
                        Else
                                {
                                Use Long Polling;
                                }
                        }
                Else
                        {
                        If (Server supports WebSockets & Client supports WebSockets)
                                {
                                Use WebSockets;
                                }
                        Else
                                {
                                If (Client supports Server Sent Events)
                                        {
                                        Use Server Sent Events;
                                        }
                                Else
                                        {
                                        If (Browser >= IE9)
                                                {
                                                Use ForeverFrame;
                                                }
                                        Else
                                                {
                                                Use Long Polling;
                                                }
                                        }
                                }
                        }
                }
```

Fig. 1 SignalR Transport Sequence

A. SignalR Architecture

SignalR provides high level APIs to make Remote Procedure Calls. Using these APIs, server side code can call client functions and vice versa. These APIs also provide functions for connection management such as connect/disconnect, connection grouping etc. SignalR architecture is depicted in the following diagram.

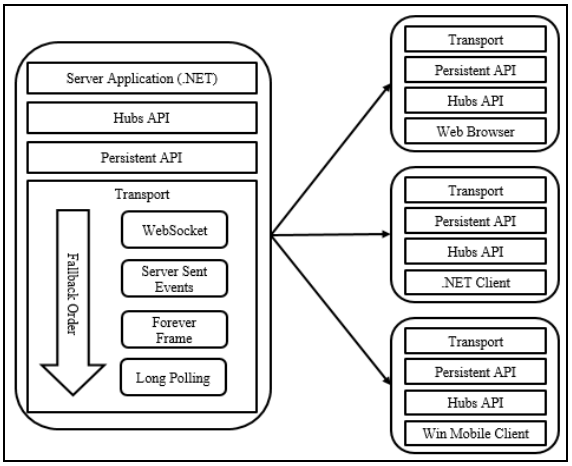

Fig. 2 SignalR Architecture

As shown in the diagram above, there are two types of APIs for communication in SignalR technology i.e., Hubs API and Persistent API.

Hubs API provides high level of abstraction and can be used where developer wants SignalR to handle maximum tasks automatically. A hubs API also provides the functionality to directly call server and client methods. We recommend use of Hubs API.

Persistent Connection API is a low level and message oriented programming interface which provides developer a lot of control over connection. Next layer in the architecture is Transport layer which is described in section 3. .NET application developed using SignalR can communicate with web client, desktop client and Windows phone client.

B. Connection Security in SignalR

SignalR maintains the connection security by validating the identity of the requestor. When any client requests server for a new connection, server generates a connection ID randomly and associates it with the response along with authenticated user name. Connection token contains connection ID and user name. Connection ID is unique for each connected client and persists for the duration of the connection. Digital signature and encryption is used by SignalR to safeguard the connection token. For each subsequent request, server validates the token data (against the data that persists in connection token store) to make sure that the request is coming from the identified user. SignalR validates the user name and connection ID to prevent the malicious attacks. If connection token is invalidated then request terminates. Connection ID of one user should not be shared with another user and also should not be saved on the client machine such as in cookies. Following diagram depicts how connection security works in SignalR.

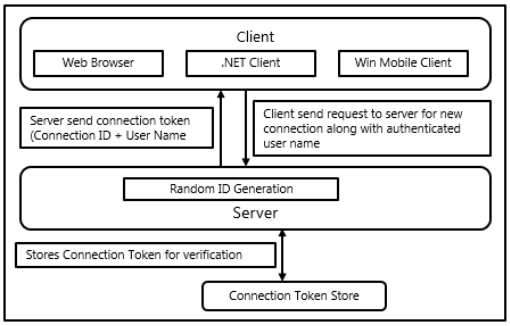

Fig. 3 Connection Security in SignalR

C. Infrastructure Scalability in SignalR Applications

There are two options when we talk about infrastructure scaling of any application i.e. scale up and scale out.

Scale up is to replace existing server with a larger server having more RAM, CPU etc. In scale out, more servers are added to distribute the load. Scaling up option is not recommended in most cases as we hit the limit on size of the server and then need to procure upwards again leading to severe cost pressure. When we scale out by adding more servers; load is distributed by the load balancer but in case of real-time applications clients connected to one server will not receive messages sent by another server. This issue can be resolved if we send the messages from all servers to a component and that component sends the messages to all other application instances. There is the concept of Backplane in SignalR which resolves this issue. Below given figure-4 shows how to scale out SignalR application infrastructure using backplane. When an application instance broadcasts [step 1 in the following figure] a message, it sends the message to backplane and backplane forwards it to all applications instances [step 2] and further applications send this message to all connected clients [step 3].

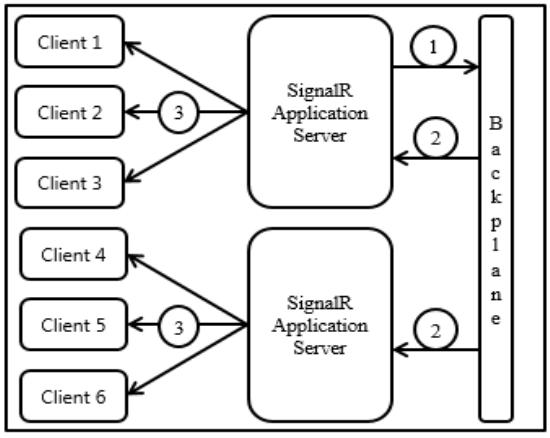

Fig. 4 Scaling in SignalR

SignalR provides three types of backplanes:

1. Windows Azure Service Bus (WASB) – This is a messaging based infrastructure which is used to send the messages.
2. Redis – Redis is an open source advanced key-value in-memory store which provides pub-sub type of implementation for sending messages.
3. SQL Server – In this approach, messages are saved in SQL table by backplane component.

When SignalR solution is hosted on Window Azure then WASB backplane is recommended to use and if hosting is on premises then either SQL Server or Redis backplane can be used.

There are some limitations in using backplane. The maximum throughput is lower than if client directly talks to server, because backplane sends every message to all application instances. But this depends on application type. If backplane is used for server broadcast type of applications such as stock ticker, then no throughput difference is observed. For client-to-client type of applications such as chat apps sometimes it creates performance issues with the growth in number of users. Backplane is not recommended to use for high frequency real-time applications (e.g. games).

D. Performance

SignalR application performance can be improved significantly by considering the following points while designing and developing the applications:

1) Reduce Network Traffic: While designing SignalR applications, the frequency of message delivery should be considered. In high frequency real-time applications such as games, not more than a few messages per second are required, so to reduce the network traffic that each client generates, a message loop can be implemented where messages queue and get sent out at a fixed rate.

2) Reduce Message Size: Message size can be reduced by reducing the size of serialized objects. Also, object properties which are not required need not be sent.

3) Reduce Default Buffer Size: SignalR keeps 1000 messages by default in memory per hub per connection which can create memory issues if large messages are used. This can be resolved by reducing this value in Application_Start event in ASP.NET application.

4) Increase Max Concurrent Requests in IIS: By increasing the value of concurrent requests in IIS, more server resources are available for serving more requests. By default, the value of concurrent requests is 5000 and this can be set by using the following command.

```
cd %windir%\System32\inetsrv\ appcmd.exe set config
/section:system.webserver/serverRuntime
/appConcurrentRequestLimit:10000
```

5) Increase maximum concurrent requests per CPU: By increasing the maximum concurrent requests per CPU, performance can be improved significantly.

V. CONCLUSION

SignalR technology is very promising; it allows developers to build real-time applications without writing complex code for pushing data from the server. SignalR applications take advantage of the best possible communication mechanism provided by the application infrastructure. In this paper, we have tried to put forth real-time technologies, use cases, challenges and SignalR architecture, communication strategy and design considerations for SignalR applications. In the next paper, we shall strive to show how to design and develop Real-Time Applications using SignalR by taking a real world example.

REFERENCES

[1] Bozdag, E., Mesbah, A., van Deursen, A., A Comparison of Push and Pull Techniques for AJAX, IEEE, 2007

[2] Wenlan Guo, Hong Liu, The analysis of push technology based on iphone operating system, ICMIC, 2013

[3] Min Huang, Jingyang Wang, Huiyong Wang, Research for real time information transfer scheme based on HTTP persistent connection, ICCET, 2010

[4] Puranik, D.G., Feiock, D.C., Hill, J.H., Real-Time Monitoring using AJAX and WebSockets, IEEE, 2013

[5] Jason Lengstorf, Phil Leggetter, Realtime Web Apps: With HTML5 WebSocket, PHP, and jQuery, Apress, 2013