A Real-Time Group Communication Architecture Based on WebSocket

Yan Zhangling and Dai Mao

Abstract—**As the World Wide Web has succeed in distributed information publication system which is an unidirectional communication system , there is an increasing demand for other network services, such as real-time data feeds, group communication and teleconferencing, which requires a full-duplex connection between client and server. Ajax and Comet has been introduced to archive such real-time requirements on the web, which have several limitations such as poor performance and high resource consumption [1]. This paper proposes a real-time group communication software architecture called Komm based on WebSocket, which represents the next evolutionary step in web communication compared to Comet and Ajax. Comparing with Comet and Ajax implementation, Komm shows better usability, higher performance and lower resource consumption.**

Index Terms—**Full-duplex web communication, real-time web, server push, web socket.**

I. Introduction

People believe that communicating is an important activity in today's information society. We spend a significant portion of our life, either in work or leisure, communicating with other people, either face to face directly or via phone, email or micro-blog remotely. Therefore, finding more efficient ways for communication with others is an important area to research [2]. We believe that the most significant problem that keeps people from communicating with one another via traditional teleconferencing system is the limitation of accessibility.

The Web has become a commonly available in the current society, either on the computer or the mobile phone. The Web has been the most acceptable way for access information in the internet in both business and personal life. Developing group communication service on the web is an excellent way to improve accessibility of such service. This should be done in a way that does not require specific proprietary features of web browsers or plug-ins.

Group communication is the exchange of information between groups of participants in a session. This can roughly be divided into two categories: synchronous and asynchronous. Asynchronous communication can easily be implemented by using HTML forms and back-end system including database and CGI-like application. Synchronous communication system also known as real-time communication system always includes a large number of passive recipients [3].

Ajax provides a mild salve to the HTTP communication model model by enabling web clients to asynchronously poll for server-side events. Comet introduced an even greater departure from the HTTP communications model by enabling "push"-style of communication over HTTP. HTML 5 WebSocket represents the next evolution of Comet and Ajax in an attempt to stand HTTP communications on its head. The HTML5 WebSocket specification defines a single-socket full-duplex (or bi-directional) connection for pushing and pulling information between the browser and server. Thus, it avoids the connection and portability issues of Comet and provides a more efficient solution than Ajax polling [4].

In this paper, we focus on real-time, synchronous group communication, design the system architecture of such system and implemented one called Komm based WebSocket, message queue and NoSQL database[5][6].

Manuscript received July 15, 2012; revised September 1, 2012.

Yan Zhangling is with the Department of Computer Science and Software Engineering, Sichuan University Jincheng College, Chengdu 610000 China (e-mail: yanzhangling@scujcc.cn).

Dai Mao is now with the Department of Information Management, Sichuan University Distance Education College, Chengdu 610000 China (e-mail: daimao@scude.cc).

II. Design Aspects for Group Communication

A. The Key Features of Group Communication

According to the common scene of group communication, we can see that real-time group communication system should include at least several features:

- Two or more participants could be involved.
- Messages should be broadcast in a very short time, which is typical less than 1 second between sender and receiver.
- Messages could be sent to specific receiver or group of receivers.

B. High Level Architecture on B/S Mode

Certainly, the first important problem needs solving is what technical framework will be involved in this system in the first stage. What impact the whole progress decisively is choosing the right development toolkit, which will affect the programming development, testing and bug fixing. After evaluating, we choose JavaScript language and Node.js, framework and HTML Web Socket as the technical foundation.

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices. Node.js consists of Google's V8 JavaScript engine, libUV, and several built-in libraries. Node.js original goal was to create the ability to make web sites with push capabilities as

seen in web applications like Gmail. Similar environments written in other programming languages include Twisted for Python, Perl Object Environment for Perl, libevent for C and Event Machine for Ruby. Unlike most JavaScript programs, it is not execute in a web browser, but is instead a server-side JavaScript application. Node.js implements some Common JS specifications. It provides a REPL environment for interactive testing.

Node.js boasts of high concurrency, is a JavaScript framework and hence functional or event based programming. Socket.io works well with Web Socket, so it would be easy to manage. There are the node modules we used:

- Socket.io, wraps Web Socket.
- Express, easy building web interface.

Web Socket is designed to be implemented in web browsers and web servers, but it can be used by any client or server application. The Web Socket protocol makes possible more interaction between a browser and a web site, facilitating live content and the creation of real-time applications. In Web Socket, a two-way(bi-directional) ongoing conversation can take place between a browser and the server. A similar effect has been archived in non-standardized ways using stop-gap technologies such as Comet. However, a comet is not trivial to implement reliably, and due to the TCP handshake and HTTP header overhead, it may inefficient for small messages. The WebSocket protocol aims to solve these problems without compromising security assumptions of the web.

C. Event-Driven Model

In a traditional thread-based system, when a person get to the receptionist he stand at the counter for as long as it takes him to complete the transaction. If he have to fill out 3 forms, he would do so right there at the counter while the receptionist just sites there waiting for him. This way is blocking the counter man from servicing any other customers. The only real way to scale a thread-based system is to add more receptionists. This, however, has financial implications in the more resources would be consumed. So we will try another new model to solve this problem, which is called event-based model.

In an event-based system, when a person get to the counter window and find out he had to complete additional forms, the receptionist gives him the forms, a clipboard and a pen and tells him to come back when he have completed the forms. The person go sit down in the waiting and the receptionist helps the next person in line. This way is not blocking the counter man from servicing others. The system is already highly scalable. If the waring queue starts getting too long, we could certainly add an additional receptionist, but we don't need to do so at quite the rate of a thread-based system.

D. The Consideration of Persistence

Although the RDBMS is widely accepted as a persistence solution, it's not suitable for object-oriented programming. So the NoSQL comes out. NoSQL database systems don't use SQL as its query language. They are often highly optimized for retrieve and append operations and often offer little functionality beyond record storage.NoSQL database systems is particularly useful for statistical or real-time analyses for growing list of elements. To build the architecture on NoSQL, we consider following five aspects:

1) Elastic scaling[11]. For years, as transaction rates and availability requirements increase, and as databases move onto virtualized environments, the economic advantages of scaling out on commodity hardware become irresistible. RDBMS might not scale out easily on commodity clusters, but the new breed of NoSQL databases are designed to expand transparently to take advantage of new nodes, and they're usally designed with low-cost commodity hardware in mind.
2) Big data. Just as transaction rates have grown out of recognition over the last decade, the volumes of data that are being stored also have increased massively. Today, the volumes of "big data" that can be handled by NoSQL systems, such as Hadoop and MongoDB.
3) Low management cost. NoSQL databases are generally designed from the ground up to require less management: automatic repair, data distribution, and simpler data models lead to lower administration and tuning requirements.
4) Economics. NoSQL databases typically use clusters of cheap commodity servers to manage the exploding data and transaction volumes, while RDBMS tends to rely on expensive proprietary servers and storage systems. The result is that the cost per gigabyte or transaction/second for NoSQL can be many times less than the cost for RDBMS, allowing people to store and process more data at a much lower price point.
5) Flexible data models. Change management is a big headache for large production RDBMS. Even minor changes to the data model of an RDBMS have to be carefully managed and may necessitate downtime or reduced service levels. NoSQL databases have far more relaxed data model restrictions. NoSQL key-value model stores and document databases allow the application to store virtually any structure it wants in a data element. The result is that application changes and database schema changes do not have to be managed as one complicated change unit. In theory, this will allow applications to iterate faster, though, and clearly without managing data integrity.

Mongo DB is an open source document-oriented No SQL database system. It's part of the No SQL family of database systems. Instead of storing data in tables as is done in a classical relational database, Mongo DB stores structured data as JSON-like documents with dynamic schemas, making the integration of data in certain types of applications easier and faster.

Mongo DB maintains many of the great features of a relational database -- like indexed and dynamic queries. But by changing the data model from relational to document-oriented, we gain many advantages, including greater agility through flexible schemas and easier horizontal scalability.

III. System Architecture Analysis

A. Overview of the System Architecture

The system should be based on Browser/Server mode. The browser client sends the message to the server, and the server

process and forward the message to other clients. The client is built by HTML, which make it easy to update.

The key concept of system design is that, it should not limit the user in a specific network. For the actual information to be communicated, it should be possible to use other network services based on the requirements of the particular type of communication [7].

The client is divided into four sub modules: authentication module, information transmission module, communication module and management module. The client is built upon a JavaScript framework called jQuery, which ease the development of JavaScript application.

B. Architecture Design

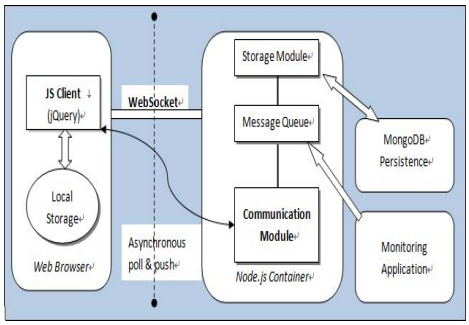

Fig. 1. System architecture

Depicted as Fig. 1, we could see that WebSocket and JavaScript play an important role in the whole architecture.

The whole system, either the client or the server side, is both built by a same language--JavaScript. This kind of design lower the complication of the whole system a lot, and ease the long-time maintenance work. On the client side, we use j Query framework to simplify the development work. And the server side is divied into three module: communication module, message queue module and persistence module. There is a lot of JavaScript , then they must be organized well. The preferred way to do this is utilizing Model-View-Controller (MVC) programming model. MVC has been adapted as an architecture for most web applications. It divides the application code into three kinds fo components: a controller, a model and a view part. With the responsibilities of each component thus defined, MVC allows different views and controllers to be developed for the same model.

Based on Web Socket, the communication module communicates asynchronously with the client in a bi-directional style, which improves the responsiveness of system. This module manages all the WebSocket connection.

Many programming models for real-time collaboration have been suggested. The simplest and most basic mechanism is message passing between the objects associated with the participants. A convenient API can offer asynchronous multicast for remote method invocation on all or a subset of participants. This model is very suitable and intuitive to use for volatile communication-oriented tasks, such as chatting. It can be refined by distinguishing certain characteristics for the messages.

The message queue module is important to avoid message congestion and reliability. A standalone monitoring application is designed, to monitor and capture messages in the queue instead of a thread inside the main application. This way, we can make the monitoring application in the easy to use application form, such as windows client software. Moreover, it can make the architecture more flexible to extend, since the messages can to captured in different servers or different environments, and can be displayed in different web applications. In our architecture, monitoring applications can run in different servers or difference environment, the messages they monitored can be easily collected from the queue. In the same way, there can be more than one web application for displaying the monitored messages.

The persistence play the role of communicating with NoSQL system MongoDB, to persistence huge message data that fetched from the message queue. The actual MongoDB is installed in multiple physical machine, by utilizing MongoDB's automated sharding/partitioning feature. The MongoDB sharded cluster automatically manage fail-over and balancing of nodes, with few or no changes to the original application code. These data will be the source of analysis in future.

IV. Komm: A Proof-of-Concept Implementation

We have implemented a system, called Komm, providing group communication functionality on the web, and it's easy to be integrated into any existing web applications.

Komm consists of a server program and client scripts. The most valuable feature is the common language between server and client. It lowers the technical complication of the system, and improves the reliability of it.

When a document that references Komm client scripts is opened by a user, a new bi-directional connection is created. The connection will pull data from the server and push client data to the other side asynchronously. In the other side, the server program will maintain all these connections, to copy or exchange data between them based on the busyness logic Besides this, this data is also persistent into MongoDB for future analysis. The huge number of data is a big challenge to the disk input/output capability. This could be solved by disk array.

A. The Server

The server side is responsible for message receiving, message forwarding and data indexing. Komm built all this fictionality based on Node.js. Since Node.js is a script language, the applications build upon it could be run platform-independent easily [8].

The server side includes a management module to organize and coordinate the connections between different groups, and providing the active connection information. It support access control feature, which is useful to control one-to-one and one-to-many connections. These configures could be set by some clients if they are authorized correctly.

Transaction Name	SLA Status	Minimum	Average	Maximum	Std. Deviation	90 Percent	Pass	Fail	Stop
Action_Transaction		6.218	21.887	36.991	7.421	32.244	319	0	0
vuser_end_Transaction		0	0	0.001	0	0	319	0	0
vuser_init_Transaction		0	0.001	0.092	0.006	0.001	319	0	0

Fig. 2. The result of stress testing

The multi-thread architecture makes the server side good performance. We give Komm a stress on a server that has a Intel Xeon E5540 CPU, 1GB RAM. We continuously

sending a 100 bytes message to it in 1 second, what we expect are the time between sending and receiving.

Depicted from Fig. 2, tmin, the fasted response time, is 6.218 millisecond. And tmax, the slowest one, is 37.084 millisecond. The good news is that the entire 319 message we sent is processed successfully, which shows that the server side has good concurrent process capability.

This result also shows that Komm could be run easily on a low cheap hardware. It could also expand on multiple nodes (physical computers) to increase the capacity of the group communication system.

B. The Implemented Communication Mode

Komm implemented two communication modes: group communication and one-to-one communication.

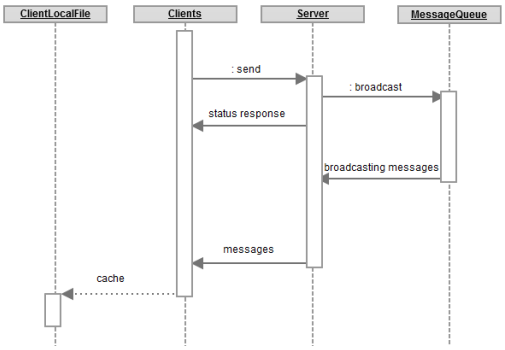

Fig. 3. Sequence diagram of group communication

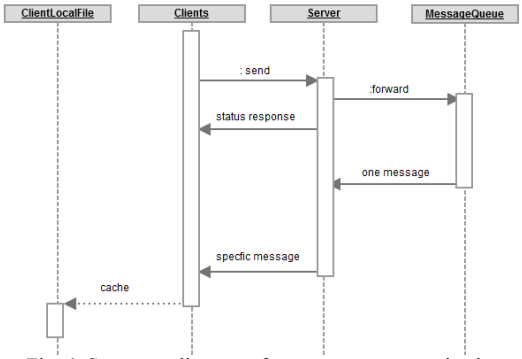

Fig. 4. Sequence diagram of one-to-one communication

The user enters message in the client interface first, which will be sent to the server through a WebSocket connection when the user press appropriate button or hotkey. The server then broadcast this message to other online users by utilizing the message queue. The message queue is a reliable way to keep messages from conflicting and losing. At the same time, all the messages will be stored in the serve's local disk through the persistence layer, and finally to stored on the hard disk using MongoDB cluster nodes. The server will send responsive message to client immediately, whether the whole broadcasting is finished or not, by using the event-driven model explained in chapter II C. So the user can get a responsive interface as soon as possible, which will improve the acceptance of user.

Similar to the group communication mode, we also implemented a one-to-one communication mode, which is useful for some private situation, such as custom service on e-commerce site. One-to-one communication mode could be treat as a special form of group communication, which the group contains only two participants. It share the same foundation setted up for group communication mode, while will reduce the complexity and requires less programming efforts.

The main sequence of one-to-one communication is similar with that of group one. The biggest difference is that when the server receives a message from client, it will forward to a specific user instead of broadcasting.

C. The Management Session Type

A management session type provides information about how many sessions are running and which online uses are available. It provides two roles: user and administrator. A user can select sessions and join them, which will collect the session into the group. The administrator can change the properties and configuration of the sessions and block some users from a session group.

V. Conclusion and Future Work

We discussed some of the issues involved in the designing an architecture for a group communication system for the web. We also proposed a system architecture and implemented a application called Komm based on the design. We believe that building such an infrastructure can help people to communicate more efficiently, since it could make the application more accessible.

There is still a lot of work to be done. In particular, Web Socket is not fully supported by current browser, and different browser support different portion of the Web Socket specification. For the purpose of making Komm able to run anywhere , the client side should include some Web Socket simulation framework such as Kaazing Gateway[9], which provides a JavaScript library that can enable any modern web browser to take advantage of Web Socket.

References

[1] S. Guoqing, Z. Wei, L. Huajun, and Z. Peng, “Survey on Server-Push Technology of Web Applications,” *Computer Systems and Applications*, vol. 18, 2008.

[2] Z. Xuan and W. L. fang, “Design and Implementation of AJAX-based Instant Messaging System,” *Science and Technology and Engineering*, vol. 9, no. 2, Jan, 2009.

[3] P. Victoria and G. N. Bradford, “Communicating and Displaying Real-time Data with WebSocket,” *Internet Computing*, vol. 16, no. 4, pp. 45-53, July. 2012.

[4] R. M. Lerner, “At the forge: communication in HTML5,” *Linux Journal*, vol. 2011, no. 7, F 2011.

[5] H. Zhang, Y. Wang, and J. Han, “Middleware design for integrating relational database and NOSQL based on data dictionary,” in *Proc. 2011 International Conf, Transportation, Mechanical and Electrical Engineering*, Changchun, 2011, pp.1469-1472

[6] S. M. N. Kuan, H. Chin, and D. Hossein, “Design Patterns to Enable Data Portability between Clouds’ Databases,” in *Proc. 2012 International Conf, Computational Science and Its Applications*, Salvador, 2012, pp.117-120

[7] A. Wessels, M. Purvis, J. Jackson, and S. Rahman, “Remote Data Visualization through WebSockets,” in *Proc. 8th Annu. International Conf. Information Technology: New Generations*, Las Vegas, 2011, pp. 1050-1051.

[8] U. Gall and F. J. Hauck, “Promondia: a java-based framework for real-time group communication in the web,” *Computer Networks and ISDN Journal*, Elsevier, April, 1997.

[9] P. Lubbers and F. Greco, “HTML5 Web Sockets: A Quantum Leap in Scalability for the Web,” *SOA World Magazine*, June, 201

[10] B. Chen, “A framework for browser-based multiplayer online games using WebGL and WebSocket,” *Multimedia Technology(ICMT)*, Jul, 2011, pp. 471-474.

[11] R. Cattell, "Scalable SQL ad NoSQL data stores," *ACM SIGMOD Record*, vol. 39, pp.12-27, Dec, 2010.