International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 02 | Feb-2018 www.irjet.net p-ISSN: 2395-0072

Real Time Code Collaboration with Source Control

Aditya Kurhade[1], Arun Nair[1], Omkar Dubas[1], Suyog Gadhave[1], Prof. Aruna Kamble[2]

[1]Student, Department of Computer Engineering, BVCOE, Navi Mumbai, MH, India
[2]Professor, Department of Computer Engineering, BVCOE, Navi Mumbai, MH, India

---***---

Abstract – *Real-Time Collaboration of code between the members of the team/s is the gist of our project. First, let's give the part collaborative meeting meaning, your teammate sees the area of the workspace in their person in control of the paper. This means your teammate can read the code you shared without having to copy a repository or put in a position of authority any dependent relations your code is dependent on. The development of a software using the concept of operational transformation to manage the versions created by the team members in a single piece of code. The collaborative part lets you give part the area of the code, so you get a short time, as a two-way working mechanism. Each of you can use a plugin for making or putting the right things that you have made for a person so that you have the freedom to create the development environment. You cannot dependently make an observation of a question under discussion without stepping on each other. There is no need for handing-off control or amount with latency. Do the work together with the shared environment only when you need to.*

1. INTRODUCTION

Consider a scenario where a software development has to be made done and the whole team collaborates to make it a success. Here the advent of GitHub, a Version Control System (VCS), came into being for helping developers for such a collaboration.In a VCS, a single branch is created which can be further divided into more branches and then these branches are merged into a single branch, but this all requires the code being committed to the repository and after which the code is being accessible to every other team member.But When multiple developers are collaborating on the same project, it happens a lot that the modules on which they are working are often interdependent and require a real-time sync to debug and run the respective modules. Also, the Version Control System is quite precise and hard for new users, which often results in conflicts and the files become inconsistent from the view of project.

1.1. CHALLENGES

The idea itself at the beginning of 2016 was challenging itself. But as being a part of the development team made us realize the effort that can be saved if a common ground for sharing is created where instant sharing is possible.

1.1.1. Firstly

We have attempted to make a system where the computer is the chief and the persons for whom one does work are persons as property. This system is where the clients fight for the right of being taken first and be given agreement the right to a chief over other persons for whom one does work.

But this system majorly plays a part on the base of facts rate get moved from one position to another which can (make, become, be) different from place to place and this leads to loss of sensitive knowledge for computers in this process.

1.1.2. Secondly

Now, we have attempted to make come into existence a true time net structure, where applications lead to produce events, which have changed state, as well as on which particular operator does its operation furthermore. PUSHER, made in this example, uses narrow ways in which to make distribution events, has a need to make one log in order for ones doing when such instances are fired. For almost any user-enacted event which makes use of a keyboard, selections, syntax-changes, copy-pastes, saves, and so on we will let one easily Register 1 a call-back in order to Execute it.

So similarly, for instances a web socket puts it into motion (when a person working in a group sorts, shares a teaching book record and so on.), pusher makes it simple, not to put it one's hands on a single event such that to transfer their control to the owner.

1.1.3. Thirdly

This was the case when the comparing the two sets of data. This was the case when we were trying to insert a piece of code a particular position was utilized but when we were trying to replace the text with a modified text version from a different user, the range of the values from the edit turned out to falsely reverted back.

The use of delete function also had a similar issue of returning a false range of the text edited. There were instances when the code was in an infinite loop due to this range value for the edited text which would crash the entire window.

2. ENVIRONMENT

The plug-in we have developed is intended to preserve the original sanctity of a developer and feed the required necessities. The plug-in is an open source plug-in which allows no restriction of what so ever manner in matters of coding in any language.

A common area of a folder showcases that folder is open. But when is open is cannot be termed as common area. The common area provides support for listening to events and discovering files.

The common area under which the folder is shared can be accessed by the members of the team by inputting the team id as well as the teammates name. On every edit, the common environment is being updated with the text being typed.

3. INTERACTION WITH ENVIRONMENT

The integration with environment is an important aspect as it gives the very first idea that why real-time file or folder sharing is completely different from real-time file editing.

Sharing of files or syncing them across multiple personal computers, servers, hosts, etc. requires to directly write the chunks of data (mostly bytes) to a file-system, if it's a personal computer or workstation; server, path if it's a server; IP address, if it's a remote host. Whereas in real-time file editing the file data is not only shared but it is continuously accessed by the user.

This continuous access integration requires environmental privileges and direct subscription with whom user is interacting with.

To achieve this type of ease of access the file on the file-system cannot be accessed directly. It needs to be accessed via the environment, the file is being edited with. Most of the editors use their own buffer memory to store changes made in the file by the user. These changes are kept in buffer until the file is not being saved by the user.

The save operation simply replaces the current buffer with the actual data in the file on the file-system. But before this save operation completes, when the editor environment accesses the file-system and checks if the version of file on the file-system is exactly the same as it was before the user started making changes in the file.

If the versions are different the user needs to take action, if the file edited by him/her should be replaced or the new version that has just occurred on the file-system is to be preserved.

This self-version of the file-system requires an extra effort to the user which cannot be neglected easily. To avoid this the files need not to be directly synced with the file system itself. They should be synced with the environment editor buffer.

Other than the management of buffer of editor and the file on file system, the consistency of all paths and their hierarchy with respect to the root of file system is to be maintained. The file and directory paths are managed differently on different Operating Systems. On Unix and Linux systems the paths convention is same. But on Windows or NT operating systems, the convention is totally different. This problem is well handled by the URI, i.e. Uniform Resource Identifier. It not only handles the diversity between different Operating Systems but also provides solution for Servers, Hosts, etc.

4. SOURCE CONTROL

In software development, version control systems (VCS) provide branching and merging support tools. Such tools are popular among developers to at the same time/together change a code base in separate lines and (cause agreement with) their changes automatically later/after that. However, two changes that are correct independently can introduce bugs when merged together. We call (related to the meaning of words) merge conflicts this kind of bugs.

Without source control, a user is eager to keep versions of the same file. This is not beneficial as one may modify the wrong replica of the file and thus leading to loss of work.
So basically, source control's work flow manages the chaos created above in their own development process.

With every version created, a bug is fixed or a new update is provided. This thus synchronizes the versions created and manages the conflicts.

Now that data is being synced between multiple clients, the concern raises, what is the contribution of each user to the shared file. The source control system is integrated with each users' system and works independently. The task of source control is to check for the file changes solely on the current system only and not the remote systems whatsoever.

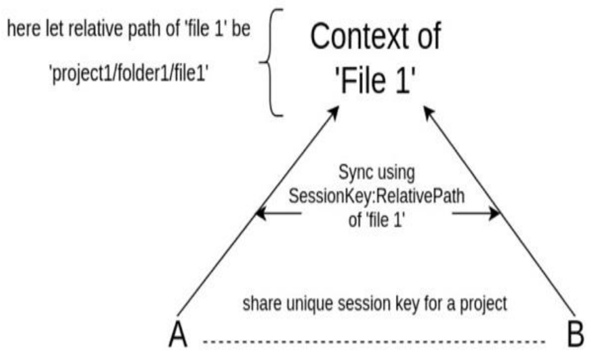
Since the data change occurred on the file on the host computer (host computer is the computer which initiated the file sharing channel) may have edits from remote users too. Similarly, the remote users who are making these changes will also get changes reflected on their system too, which will, in turn, make the source control system detect changes which are not completely made by that user.

So, there is need to acknowledge these changes with respect to each user. This can be done by analysing the data coming from each socket packets containing the changes with the users' identity who has made changes.
The above task will definitely take more work as it will account for calculations at each host. For example, if 4 users are in the collaboration session, but are not contributing anything at all. Still, the calculations will take place at each host for no reason. This can avoid by simply accounting for the changes for current host only. The host is responsible for keeping track of all changes for the respective host only.

The source control can be then submitted with the changes which the current user has made and committed as soon as the session is completed. Finally, the commits from each user need to be shared with each user which then can be reflected the source control at each host.

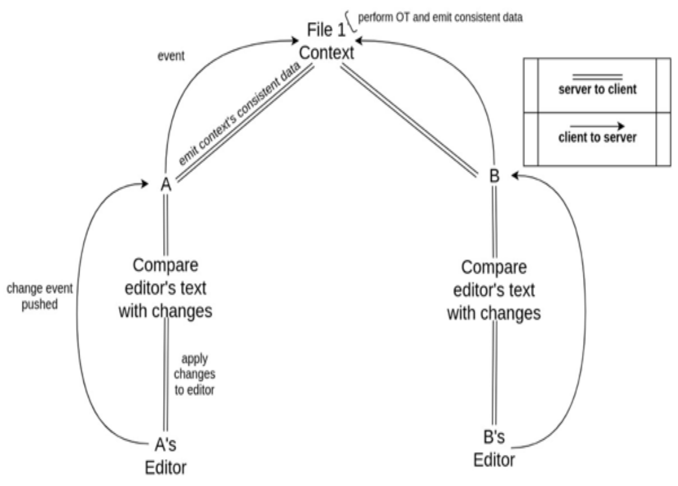
5. DATA FLOW

Connection: Connections between two or more collaborators are done by using unique session key (this session key is unique for every project or file on which team of collaborator is going to work). Users have to store this key to access respective document.


here let relative path of 'file 1' be
'project1/folder1/file1'
Context of 'File 1'
Sync using SessionKey:RelativePath of 'file 1'
share unique session key for a project
A
B

How files Are stored? As every project will have unique session key every file in the project is stored under 'SessionKey: RelativePath'. RelativePath gives the path of the file from project folder (e.g. *project/file* or *project/folder/file*). In this way context of each file is stored on the server. This is how the connection is established, as we can see now 'A' and 'B' are in sync with file 1 changes now let us see how changes from each client.

How changes flow? when changes are created in the file by any client the change event is created and these changes are passed on to the server, here one thing needs to be noted down is that the context of a file is the consistent version and at the end, all the clients will have this version maintained in their editors.


perform OT and emit consistent data
File 1 Context
event
emit context's consistent data
server to client
client to server
A
B
change event pushed
Compare editor's text with changes
apply changes to editor
Compare editor's text with changes
A's Editor
B's Editor

As all the clients are in sync with the context, when the context changes the respective changes occur in all the synced clients. When changes are pushed by any client through his/her editor these changes are examined by the server following that server performs Operational Transformation between changed text and context previous changes which results in the consistent data and this data is overwritten in the context and the same thing is synced to all other connected clients through their editor.

6. DATA CONSISTENCY

The main concern that we have to solve when it comes to real-time code collaboration is the consistency of the data. there have to be specific protocols which need to be followed:

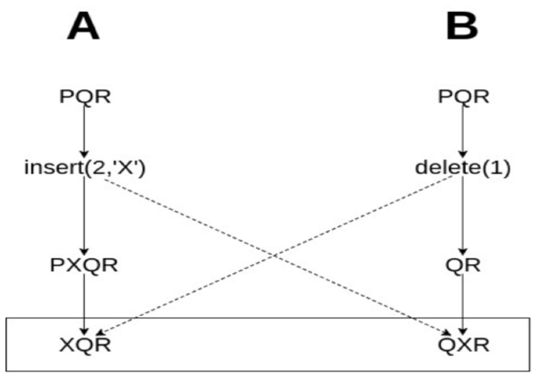
6.1. First Approach (Using real-time databases and emit changes using real-time database protocol)

When we used a real-time database to control concurrency major problem was we didn't have any access to database servers which leads to very limited operations and controlling concurrency with just those provided operations was a troublesome job to do which also affected our speed of collaboration in some way, so designing protocol on real-time database services was not a great choice.

6.2. Second Approach (Using Just sockets)

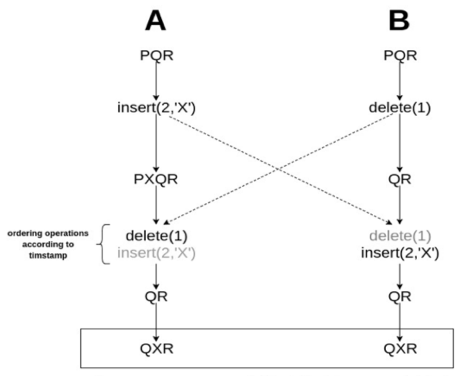
Using sockets made one benefit here that then we had better control of servers as events are emitted and they are solely handled by server this approach did solve most of the equations for us but writing protocol on our own without having any reference to theoretical research would have taken ages to create what we want. Avoiding loops of events, maintaining versions of code to merge them into forming converged code these problems have to be solved with proper protocol our main aim here is not to create concurrency control algorithm but to use this algorithm in some software or editors. So the best algorithm that we could find was Operational Transformation(one which Google uses) Now to explain how we used this protocol in our case consider two clients (they need not be clients always one of them can be server or both can be servers)'A' and 'B', also consider there is data 'PQR' written on both clients' side. Here Client A insert 'X' at position 2 and client B delete letter at position 1. These are the methods we used before using Operation Transformation.

Method 1: Merely transforming change events to another client without any operation on it here server was just listening to events and pushing it to other clients which are connected in this case, you can clearly see convergence is completely violated both the clients are not in a consistent state.



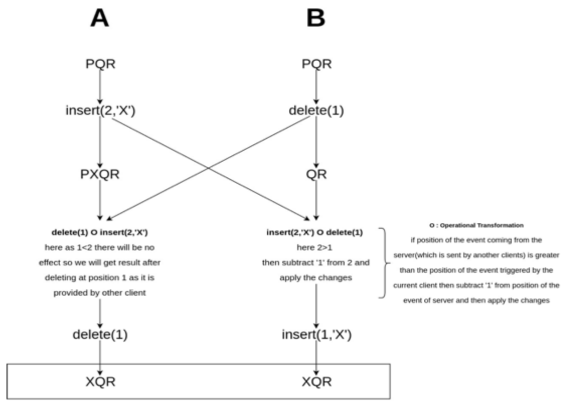
Convergence Violated

Method 2: Putting order and clock in it, using this we thought that consistency will be maintained so we tried to bind every event with time-stamp and made sure that they execute in that sequence only which is handled by server, following were the results. We were able to make data consistent that is why convergence is successful but wait, here client A's decision of putting 'X' after 'P' and before 'Q' is completely violated data is consistent but violating decision was never a plan, every decision made by clients has to be fulfilled.



Convergence successfull but decisions are violated

Method 3: Finally, we got to answer to each and every hard question using OT, following way we did it. So by using above concurrency signed agreement between nations persons of representative and decision is said (thing is true).


A
B
PQR
insert(2,'X')
PXQR
delete(1) O insert(2,'X')
here as 1<2 there will be no effect so we will get result after deleting at position 1 as it is provided by other client
delete(1)
XQR
PQR
delete(1)
QR
insert(2,'X') O delete(1)
here 2>1 then subtract '1' from 2 and apply the changes
insert(1,'X')
XQR
O : Operational Transformation
if position of the event coming from the server(which is sent by another clients) is greater than the position of the event triggered by the current client then subtract '1' from position of the event of server and then apply the changes

Convergence Successful and Decisions are also maintained

7. CONCLUSION

The Implementation of the real-time application for collaboration i.e. RCE is a web application that helps programmers to create and see the result of the executed source code by terminal, collaborate in real-time with other programmers by chat or invite to join the same project and manage the project such as import, export, shared projects. RCE has the main features: provide workspace to make, execute and build the source code, real-time collaboration, chat, and build the terminal.

8. REFERENCES

[1] Delta Impact Finder: Assessing Semantic Merge Conflicts with Dependency Analysis.

[2] Version Control Systems Stefan Otte Computer Systems and Telematics Institute of Computer Science Freie Universitat Berlin, Germany.

[3] The Semantics of Version Control Wouter Swierstra1 and Andres Loh 2 1 Universiteit Utrecht.

[4] VeCVL: A Visual Language for Version Control Nathan W. Eloe, Denise M. Case School of Computer Science and Information Systems Northwest Missouri State University.

[5] EVOLUTION OF VERSION CONTROL SYSTEMS Comparing CENTRALIZED against DISTRIBUTED Version Control models CARL FREDRIK MALMSTEN Department of Applied Information Technology IT-University of Gothenburg malmstec@ituniv.se Supervisor BILL SULLIVAN Department of Applied Information Technology IT-University of Gothenburg.

[6] An O(ND) Difference Algorithm and Its Variations, Department of Computer Science, University of Arizona, Tucson, AZ 85721, U.S.A, EUGENE W. MYERS.

[7] JavaScript diff library with support for visual, HTML-formatted output, [https://www.npmjs.com/package/text-diff].

[8] Pusher Docs [https://pusher.com/docs/].

[9] Visual Studio Code Extension API Documentation, Microsoft Corporate [https://code.visualstudio.com/docs/extensionAPI/].

[10] Firebase Documentation [https://firebase.google.com/docs/].

[11] Node JS Documentation and NPM Library [https://nodejs.org/en/docs/]