- Home
- My Profile
- My Dashboard
- About
- Accepted Projects
- Events & Timeline
- Connect With Us
- Search
- FAQ
- Logout

Content of this proposal is not editable at this moment, because student application period is over. If you still need to make some changes, please send a comment to the organization. They can enable modifications for this proposal.

Interactive LyX

Sushant Raikar

View revisions

Organization: LyX – The Document Processor

Abstract: Extend LyX with interactivity capabilities allowing multiple users to work concurrently on the same LyX document. This is done by adding to LyX the capability to send/receive individual LyX actions (called LFUNs in the LyX gergo) through the network, by making use of some server allowing users to find each other.Through some extension in LyX GUI, remote users will be able to connect with each other and concurrently edit the same document.

Name: Sushant Raikar

Email: sushant.s.raikar@gmail.com

IRC nick: raikar

GitHub: HotSushi

University: National Institute Of Technology, Goa

Time zone: India (GMT +05:30)

Blog: sushantprojects.blogspot.in

Platform: Ubuntu 13.10(x64)

About Me:

I am third year B.Tech. Student at National Institute of Technology, Goa. I am passionate about technologies in general, I have worked on a variety of projects from low level programming to higher level. I have worked with most of the aspects of computer technologies including Robotics, Computer Vision and Networking. I am very fascinated by how open source works and would love to be a part of such community.

I came across LyX, an open source document preprocessor and one of the project ideas caught my eye. The project 'Interactive LyX', a new concept which will be very useful and applicable in real life scenario. As a student I know situations where collaborative editing will make tasks easier and efficient, may it be project proposals, assignments or any other formal documents. I would love to be a part of such a project which is unique, very much influential and effective and of course this will give me the right to brag about it.

I am pretty confident with my programming skills and have experience working with C++ and Qt.

Coding skills:

Here are some of the projects I did in my 3 years of engineering.

- Android controlled mouse and keyboard Link
- First Person Shooter Game Link
- Google Cloud Developer Challenge 2013 : Project Scholar

  Got selected as one of the 9 finalist in India, got 1K$ and Google Nexus Tablet Project Link
- Intel App Innovation 2013 : 3donomic Camera Link

  Got selected in Intel AIC, got Windows Lenovo ThinkPad Tablet. I learned and used Qt for this project.
  YouTube Link

There are many other mini projects like Online Coding Competitions with Java Socket , mouse controlled by hand gestures (Open CV) and many more..

I had to work heavily with Qt framework for my Intel App Innovation project. I am not only familiar with GUI features provided by Qt but also with QSocket, QProcess and QThread, which might become relevant as I progress with my project.

I am comfortable with network programming and have worked on many projects including Android controlled mouse.

I started learning LyX internals just recently and I know the basics of it.

The Project: Interactive LyX

Goal:

Through some extension in LyX GUI, remote users will be able to connect with each other and concurrently edit the same document.

Benefits:

- Remote users can connect and edit same document in real time.
- Uniqueness of this feature will attract more users to LyX.
- Collaborative editing can make tasks easier and efficient.
- More secure, as the document will not be held by the server.
- Faster, as only LFUNs (will be explained later) will be serialized and passed around.

Terminology:

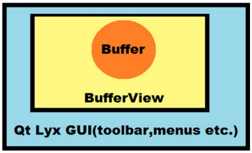


Figure 1: Structure of Buffer and BufferView

LFUN:

stands for LyX functions, every action that can be performed in LyX has a corresponding LFUN for it. We can send this LFUNs to the remote users and execute them locally.

Buffer:

holds pure document, doesn't provide functionality for editing (it doesn't have a cursor, no selection, no undo/redo stack etc.)

BufferView:

holds Buffer and provides functionality for editing such as cursor, selection etc.

Cursor:

indicates location where the BufferView is being edited currently. It is represented by

- Cell Index (row, column)
- Paragraph Offset (which paragraph)
- Position (position inside a paragraph)

Progress:

A proof-of-concept of this feature has been sketched out as a patch on the LyX mailing list (http://www.lyx.org/trac/ticket/7964), but it is a bad hack, that stands up only a few seconds then it crashes.

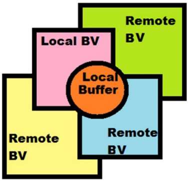


Figure 2: Local and Remote BufferViews acting on single local Buffer

Keeping separate cursors for each user in a BufferView will become complex. An ingenious solution to this is to keep separate remote BufferView for each remote user, which will act on the same local Buffer. This way, each remote user will have his separate cursor, and even undo, redo, delete will work locally.

In this patch, each client holds a remote BufferView of the server. Whereas, the server holds remote BufferViews of all the clients

Initiation:

Whenever a new client connects, it gets the current copy present at the server and simultaneously server creates a remote BufferView for it.

Propagation:

Whenever a client makes a change, LFUNs are sent to the server, which executes it on its respective remote BufferView. When the server makes any changes those changes are executed by the clients on their remote BufferView.

Termination:

remote BufferView of the client is deleted.

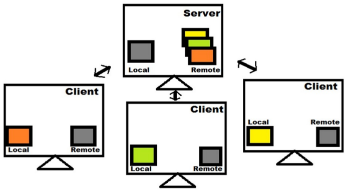


Figure 3: Client holds remote BufferView of Server, Server holds remote BufferViews of all the clients

Problems:

changes made by 1 of the client will not be executed correctly on another client, as it doesn't hold its remote BufferView.

Approach:

I will stick with the previous approach only with some minor changes.

Changes:

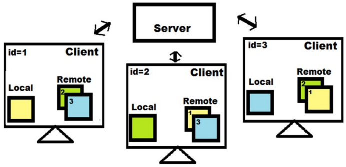


Figure 4: Each client holds remote BufferViews of every other client, ids are used to identify in which BufferView the incoming LFUNs should be executed.

- Every client will hold remote BufferView for every other client.
- Server will not hold the document and will do the task of broadcasting data and assigning identifiers.
- Changes made by a client will be sent to the server which will forward it to every other client.
- A new client will contact server and get its identifier, simultaneously server broadcasts and every other client makes a new BufferView for that identifier.
- Client sends LFUN to server, server forwards (id, LFUN) to other clients.

Advantages:

- ❄♒♓⬧ ♋◻◻❒◻♋♍♒ ♒♋■♎●♏⬧ ♍◻❍❍◆■♓♍♋⧫♓◻■ ♌♏⧫⬥♏♏■ ❍◻❒♏ ⧫♒♋■ 🗀 ♍●♓♏■⧫⬧✍
- Will be easy to port to chat based protocols like xmpp.
- Server doesn't hold the document

Challenges:

Version Control System

When a LFUN hits the remote BufferView and it is dispatched, it acts on a version of the document that might be slightly different, as modified by the local user concurrently. It is important to clarify what revision of the document each LFUN applies to. So the collaborative editing should be seen making an analogy to a VCS like 'git', as continuously committing local changes, pulling and pushing changes across Buffers of the users participating in the collaborative session.

When edits are very close, it should be important to consider what revision each LFUN applies to. A failure scenario can be "User1's and User2's cursors are at the end of a para. Now if U1 starts typing characters, and U2 types 'right' before receiving those LFUNs , U2's cursor will be in next paragraph. where as his remote BufferView's Cursor will still stay in the previous paragraph"

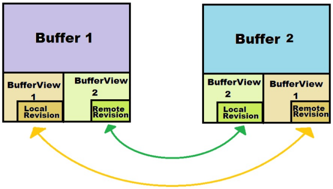


Figure: A local Buffer shared by multiple BufferViews. Each local BufferView will hold current revision the document is in. Each remote BufferView will hold latest revision it thinks its respective Buffer is in.

To handle this, each BufferView should maintain a version number and also simultaneously keep track of relative Cursor change, across different version. So the remote BufferView 2 (acting on Buffer 1), will keep track of cursor changes across different versions of Buffer 1. Similarly , remote BV1 will keep track of cursor changes across different versions of B2.

Whenever an LFUN is sent it should be packed with target version and current version it's buffer is in.

Send{ LFUN, target_version, cur_version}

the receiving client will than adjust his cursor according to target_version. Sending cur_version will help in clearing out the unnecessary history up-to cur_version .

Cursor Validity

As discussed previously, a cursor is represented by index, paragraph offset and position. Care should be taken to maintain cursor validity, simple example would be if local cursor is on 4th paragraph, but remote user deletes 2nd paragraph, the local cursor is no longer valid because its paragraph offset is incorrect. This can be handled by tracking cursor after every edit LFUN i.e. Delete, copy, undo etc.

Whenever a user makes changes in his local Buffer, all the remote BufferViews should track this change and keep a history of how the cursor position was affected with each version change. A remote cursor will be effected, only if its position is after the editing cursor.

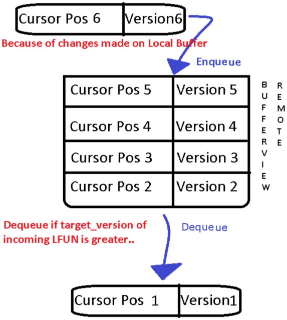


Figure: History Queue to track cursor positions across different revisions

If a user makes a change in his Buffer, the change in the cursor position in his local BufferView should be tracked to get the difference(for example if he typed a character, then difference in cursor->pos_type will be 1). This information can be used by remote BufferViews to update their cursor (only if their cursor is ahead of the editing cursor)and then update their history queue, by enqueueing (cursor_postion, version). This way, history of revisions can be maintained.

Another important point is to remove the history once it becomes useless.It will only become useless , when it is no longer being referenced by the remote BufferView. Or in other words, user 2's remote BufferView knows, in which latest revision User 1's Buffer is. So, whenever an incoming LFUN reaches remote BufferView , it checks the target_version, and sees that versions less than target_version are no longer referenced and hence they can be de-queued.

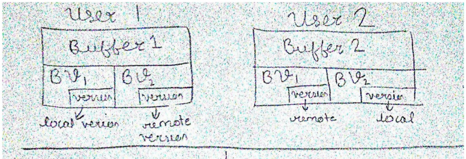
Heavy Processing

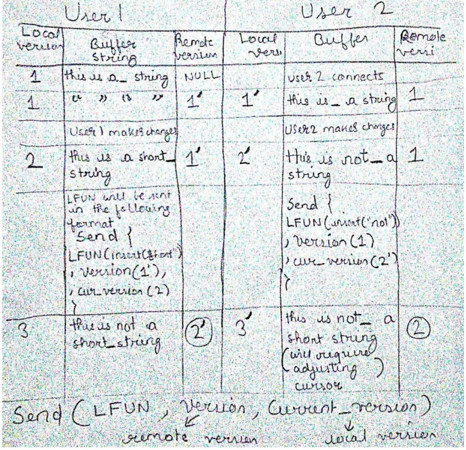
Suppose 1 of the client puts a higher resolution image, it might create problems as other clients will have to download that image separately. Can be handled if the image is not transferred rather just its dimensions.

There will be many other unforeseen scenarios that will have to be handled correctly.

Example

A simple example to handle use case, when edits become close..





User 1			User 2		
Local version	Buffer String	Remote version	Local versi	Buffer	Remote verri
1	this is a_ string	NULL		user 2 connects	
1	" " " "	1'	1'	this is_ a string	1
	User1 makes changes			User2 makes changes	
2	this is a short_ string	1'	2'	This is not_a string	1

	LFUN will be sent in the following format
	Send {
	LFUN(insert("short")
	, version(1'),
	, cur_version (2)
	}
				Send {
				LFUN(insert("not"))
				, version(1)
				, cur_version(2')
				}	

3	this is not a short_string	②'	3'	this is not_ a short string (will require adjusting cursor)	②

Send (LFUN, version, current_version)

remote version

local version

Timeline

- 22 April – 28 April

  I have my exams during this period and may not be able to start with community bonding right away.

- 30 April – 18 May (Community Bonding Period)

  Read LyX source code. In depth understanding of LyX Internals. Build LyX and resolve any configuration/ compiler issues. Discuss failure scenarios with the mentor.

  Port LyX Patch to the current Trunk.

  Clean LyX Chat branch, get familiar with xmpp.

  Modify the patch to illustrate the project approach.

- 20 May (Official Coding Phase Begins)

- 20 May - 5 June (Server and LyX client)

  Implement multiple clients, remote BufferViews. At the end of this phase connection initiation, termination and sending LFUNs will be handled.

- 5 June -14 June (Cursor Validity)

  Track cursor change made by edit LFUNs (inserting, deleting, undo, redo, copy etc.) modify other cursors accordingly. At the end of this phase cursor validity will be handled.

- 15 June -23 June (Testing)

  Make note of all abnormal behavior and reason behind it.

- 23 June (Submit patch for Google's survey of midterm evaluation.)

- 23 June – 10 July (Version Control)

  BufferView should be able to track changes in the cursor, store history,act on a specific version , update version etc.

- 10 July – 14 July (Testing)

- 15 July – (Handle unforeseen cases)

  There will be many unpredictable cases, therefore during this phase a cycle of testing and handling will be repeated continuously.

Note: The timeline for this project cannot be predicted definitely, therefore the time requirement of each step may vary.

Commitments

I have my exams during the start of community bonding period, besides that, no other commitments. This project will be my sole focus this summer.

2 COMMENTS

Sushant Raikar March 22, 2014, 4:21 a.m.

A simple working example...

http://i.imgur.com/3E2dX3x.jpg?1

Tommaso Cucinotta March 22, 2014, 11:23 p.m.

Hi,

I've unlocked the proposal, if you want to apply any further last-minute refinements.
There's actually a couple of minor things:
1. the initial community bonding period in your schedule, with ~3 weeks just to compile LyX, seems a bit long; either you consider this a "code reading" period, or otherwise this seems too long;
2. a possibility might be the one to stick, during such an acquaintance period, a couple of things:
 - port the old Interactive LyX patch to the latest trunk, clean up from lyx-chat that now is available as separate branch;
 - as a proof-of-concept, modify that path to realize the totally-ordered-server-based interaction we discussed (each LFUN goes to server, it dispatches to all views a common and consistent view of all edits);

The above may be 2 points that allow us to get quickly to a simple early prototype of how things might look like when under nice connectivity, but especially might be good for you to get oriented into the LyX source code.

3. please, detail whether you have any other commitments during the project period, including exams, lectures, vacations, unavailability periods.

Thanks,

Tommaso.

LEAVE A COMMENT

SUBMIT