Lisp in Summer Projects Submission

Submission Date	2013-10-23 18:30:41
Full Name	Lars Tveito
Country	Norway
Project Name	Shared buffer
Type of software	other
General category	development tool
LISP dialect	Emacs Lisp
GitHub URL	https://github.com/larstvei/shared-buffer
Did you start this project?	Yes, all the code is written by me
Project Description	I want to upload a free-form 3-4 page PDF composition.
Upload 3-4 page detailed PDF	shared-buffer-description.pdf
Build Instructions	(This is also covered in README.md) Client: To install the Emacs extension just download the shared-buffer.el and store it in your load-path. Server: You will need a common lisp interpreter and quicklisp installed. SBCL is the only implementation that has been tested, and therefor also recommended. Download the shared-buffer-server.lisp and store it anywhere you like.
Test Instructions	No tests provided.
Execution Instructions	(This is also covered in README.md) Client: Once shared-buffer.el is loaded you can start sharing a buffer by interactively running the command: M-x sb-share-this-buffer RET

	Host: virvel.de RET Key: this-is-a-key RET To connect to a shared buffer, run the following command: M-x sb-connect-to-shared-buffer RET Host: virvel.de RET Key: this-is-a-key RET Server: Running SBCL type CL-USER> (load "/path/to/shared-buffer-server.lisp")
Describe any bugs or caveats	If shared buffer goes out of sync it is recomended to use M-x sb-disconnect RET or just kill the buffer (C-x k). For further information see the project description.
Screen shots	Screen Shot 2013-10-24 at 12.30.08 AM.png
Official	I have read rules and have abided by them. I am 18 years of age or older. I am not living in Brazil, Quebec, Saudi Arabia, Cuba, Iran, Myanmar (Burma), North Korea, Sudan, or Syria.

Shared Buffer

Lars Tveito

October 23, 2013

1 Purpose

Shared buffer is a project that enables real-time collaborative editing in Emacs. It is split up in two parts, client and server. The client is an Emacs extension entirely written in Emacs Lisp. The server is a small Common Lisp program; there is currently a server running on 'virvel.de'.

2 Function

In Emacs one is simply able to share a buffer and connect to a buffer that is already shared. This is done by requesting a connection to a shared buffer server. Once a connection is established all changes in your buffer is sent to the server. The server simply redirects these messages to all Emacs clients connected to that shared buffer.

3 Motivation

Working on a small scale project with friends, fellow students and coworkers was not simple enough to do with Emacs. Having recently started learning Lisp, it seemed like a fun and ambitious project.

4 Audience

Initially me, and whomever I wanted to work with. After realizing I'd might actually make it work, I think it can be useful for anyone using Emacs in collaboration with others. It is written with small scale software projects in mind, but can be used for all kinds of text editing.

5 Methodology

The project is divided into two parts, a client and a server. The server is written in Common Lisp, and it's main job is to allow the clients to communicate. The client is an Emacs extension written in Emacs Lisp which mainly send changes to the server, or receives changes from the server.

The client may ask to establish a new session or connect to an existing one. If a new session is required, the client provides a key. This key is used by the server as key in a hash table, containing lists of clients. A client asking to connect to a shared buffer is simply added to the list of clients that corresponds to the given key.

When a new client connects to an already established session, a single client is asked by the server to send it's entire buffer content. This package is marked as being for new clients only. From that point on they should keep synced. The session is kept alive as long as there are clients connected to it.

The main challenge in this project was to figure out how to keep several separate Emacs buffers mirrored. This is resolved by sending a message for every command a user invokes (this is done by adding functions to after-change-hook and post-command-hook, both built-in variables in Emacs). These messages will dictate a change that happened in a buffer. Assuming the shared buffers are identical to the one sending the message prior of that change, we can safely apply that change to any client that receives this message.

A problem arises if our assumption is wrong. The most common situation is that a client has made changes in a buffer between the time the message was sent and received. The point where the change should be applied is then calculated by using the difference in the size of the buffer the message was sent from, and the size of the buffer receiving the message. This works in most cases.

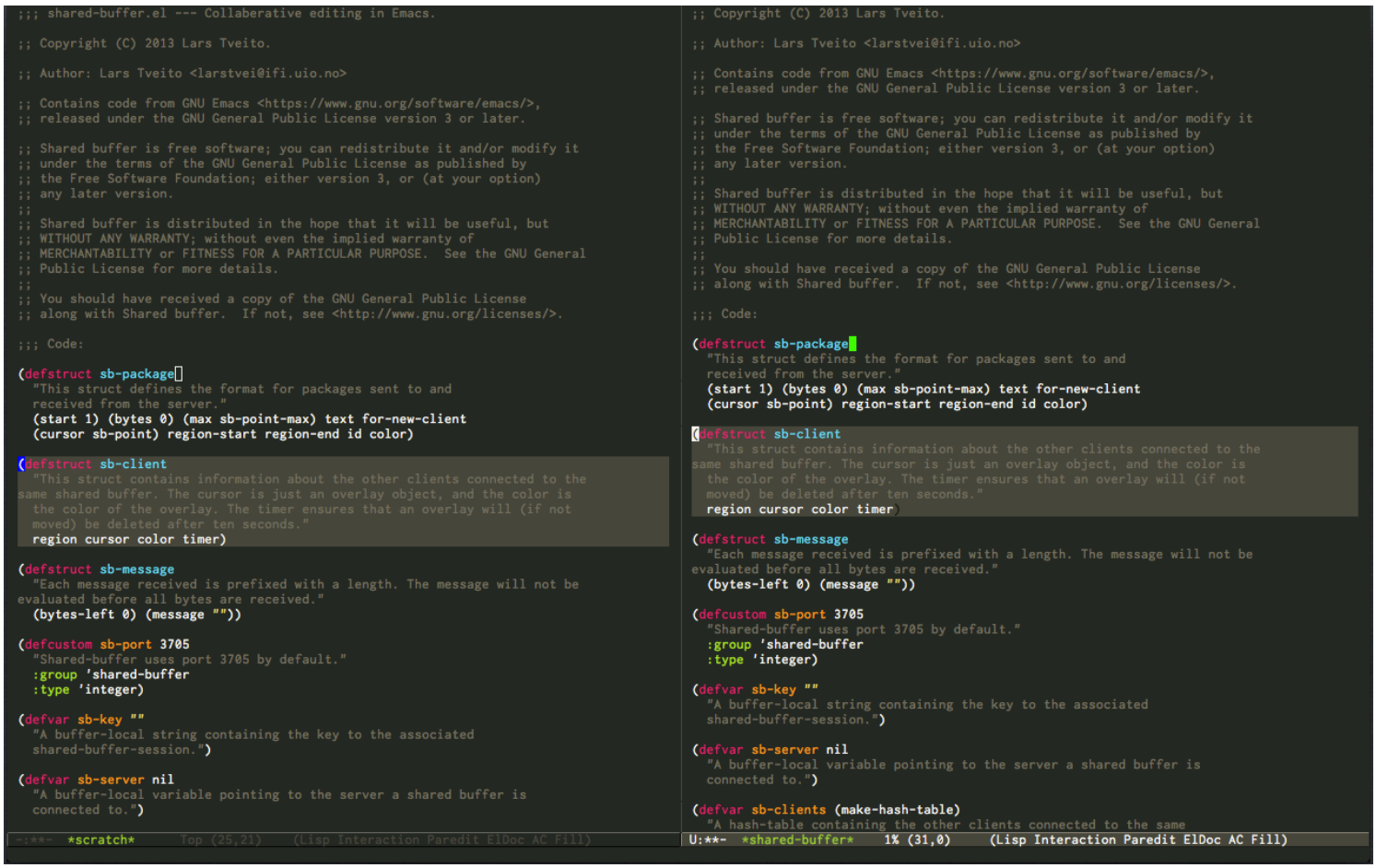
6 Conclusion

After a summers worth of coding I am glad to say that the core functionality is up and running. It is fast and lightweight. A lot of time has gone into finding the *right* solution to the big problems, and finding good workarounds for Emacs's many idiosyncrasies. I believe the project has great potential.

The main issue that needs fixing is how to detect and resolve problems with synchronization. As of now, once buffers go out of sync, there is really no other solution than to disconnect and reconnect. There are also quite a few bugs triggered by Emacs's many features and extensions, and I'm hoping to resolve these after the competition is over.

I plan to make Shared buffer more user friendly, by supplying a Emacs minor mode accompanied by a chat feature. When these things are in order it will be released in melpa, and will hopefully be found useful.

Screenshot:


;;; shared-buffer.el --- Collaborative editing in Emacs.
;; Copyright (C) 2013 Lars Tveito.
;; Author: Lars Tveito <larstvei@ifi.uio.no>
;; Contains code from GNU Emacs <https://www.gnu.org/software/emacs/>,
;; released under the GNU General Public License version 3 or later.
;; Shared buffer is free software; you can redistribute it and/or modify it
;; under the terms of the GNU General Public License as published by
;; the Free Software Foundation; either version 3, or (at your option)
;; any later version.
;;
;; Shared buffer is distributed in the hope that it will be useful, but
;; WITHOUT ANY WARRANTY; without even the implied warranty of
;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
;; Public License for more details.
;;
;; You should have received a copy of the GNU General Public License
;; along with Shared buffer. If not, see <http://www.gnu.org/licenses/>.
;;; Code:
(defstruct sb-package
"This struct defines the format for packages sent to and
received from the server."
(start 1) (bytes 0) (max sb-point-max) text for-new-client
(cursor sb-point) region-start region-end id color)
(defstruct sb-client
"This struct contains information about the other clients connected to the
same shared buffer. The cursor is just an overlay object, and the color is
the color of the overlay. The timer ensures that an overlay will (if not
moved) be deleted after ten seconds."
region cursor color timer)
(defstruct sb-message
"Each message received is prefixed with a length. The message will not be
evaluated before all bytes are received."
(bytes-left 0) (message ""))
(defcustom sb-port 3705
"Shared-buffer uses port 3705 by default."
:group 'shared-buffer
:type 'integer)
(defvar sb-key ""
"A buffer-local string containing the key to the associated
shared-buffer-session.")
(defvar sb-server nil
"A buffer-local variable pointing to the server a shared buffer is
connected to.")
-:**- *scratch* Top (25,21) (Lisp Interaction Paredit ElDoc AC Fill)
;; Copyright (C) 2013 Lars Tveito.
;; Author: Lars Tveito <larstvei@ifi.uio.no>
;; Contains code from GNU Emacs <https://www.gnu.org/software/emacs/>,
;; released under the GNU General Public License version 3 or later.
;; Shared buffer is free software; you can redistribute it and/or modify it
;; under the terms of the GNU General Public License as published by
;; the Free Software Foundation; either version 3, or (at your option)
;; any later version.
;;
;; Shared buffer is distributed in the hope that it will be useful, but
;; WITHOUT ANY WARRANTY; without even the implied warranty of
;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
;; Public License for more details.
;;
;; You should have received a copy of the GNU General Public License
;; along with Shared buffer. If not, see <http://www.gnu.org/licenses/>.
;;; Code:
(defstruct sb-package
"This struct defines the format for packages sent to and
received from the server."
(start 1) (bytes 0) (max sb-point-max) text for-new-client
(cursor sb-point) region-start region-end id color)
(defstruct sb-client
"This struct contains information about the other clients connected to the
same shared buffer. The cursor is just an overlay object, and the color is
the color of the overlay. The timer ensures that an overlay will (if not
moved) be deleted after ten seconds."
region cursor color timer)
(defstruct sb-message
"Each message received is prefixed with a length. The message will not be
evaluated before all bytes are received."
(bytes-left 0) (message ""))
(defcustom sb-port 3705
"Shared-buffer uses port 3705 by default."
:group 'shared-buffer
:type 'integer)
(defvar sb-key ""
"A buffer-local string containing the key to the associated
shared-buffer-session.")
(defvar sb-server nil
"A buffer-local variable pointing to the server a shared buffer is
connected to.")
(defvar sb-clients (make-hash-table)
"A hash-table containing the other clients connected to the same
U:**- *shared-buffer* 1% (31,0) (Lisp Interaction Paredit ElDoc AC Fill)