Part I
Background

1 Networks

2 Visualization

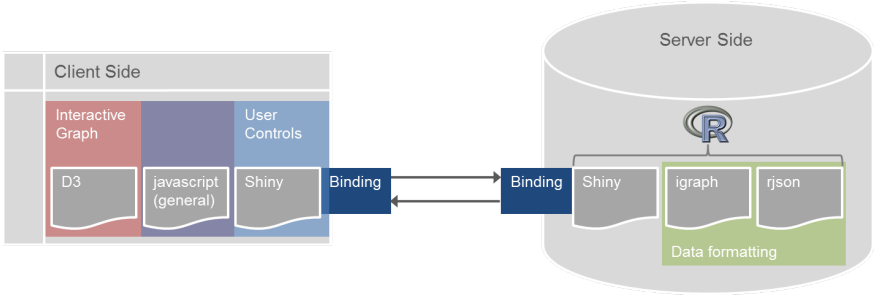
3 Layout Algorithms

Part II
User Interface

4 Technical Aspects

4.1 Software

Gravicom utilizes three main pieces of software to establish interactive user control of a random graph. The three pieces used are Shiny, D3, and igraph. They are used to manage server/client interaction, user interface and graph layout, and data formatting, respectively.



4.1.1 Shiny

Shiny is an R package created by RStudio that enables R users to create an interactive web application that utilizes R as the background engine.[4] Through default methods to build user interface elements in HTML and a handle to the server side code, Shiny is a very simple way to turn R code into a website.

Gravicom uses the Shiny functionality to create user controls, pass correctly formatted data to the client, and as a means to display summary information regarding the user's interactions with a graph at any point in time. In this context, Shiny serves as the translator between the formatted data and what the user sees and interacts with on their screen.

4.1.2 D3

D3 is a javascript library written by Mike Bostock with the main purpose of visualizing and interacting with data and can be found at `http://www.d3js.org`.[1] The title D3 stands for "Data Driven Documents. This library facilitates manipulation of HTML elements, SVG (scalable vector graphics), and CSS (cascading style sheets) with the end goal of animations and interactions that are tied to underlying data. The key idea behind the library is that DOM elements are completely determined by the data, so rather than adding elements to a web page to be viewed by users, D3 allows users to see and interact with graphical representaions of their data in a web framework.

Gravicom uses D3 to handle all graphical displays and user interactions with the graph. The data is passed to the client and able to be used through Shiny's input bindings. It is crucial that the data has been formatted correctly at this point for the javascript to properly function. For this reason, we limit the file types being passed in to the tool to a robust graph-specific type.

At this point in the page lifecycle, the graph's nodes are tied to circles and the edges are tied to paths on the page. User manipulations such as selecting, dragging, and grouping are handled by D3 and then data is passed back to the server via Shiny's output bindings to allow for communication between user and the R engine underneath. What this means is that all visualization and user interaction with the graph are accomplished using javascript, more specifically the library D3. Shiny and R serve as the framework on which the data sits, but when the user touches the data they are doing so through the javascript pieces.

4.1.3 igraph

igraph is a software package used for creating and manipulating undirected and directed graphs.[3] It is a cross-language package available for C, R, python, and Ruby. igraph also supports multiple graph file formats and visualization of graph structures.

Gravicom utilizes two parts of igraph, first is the conversion from a gml file to an XML file. The gml file format, short for Graph Modelling Language, is a hierarchical ASCII-based file format for describing graphs. Here is an example gml file of an undirected graph with two nodes and one edge. The important points to note are the the node ids are numeric, the edge has only source and target attributes of the node ids to be connected, and the nodes can have other attributes, as typified by "value" in the example.

```
## graph
## [
##   directed 0
##   node
##   [
##     id 0
##     label "Node 1"
```

```
##     value 100
##   ]
##   node
##   [
##     id 1
##     label "Node 2"
##     value 200
##   ]
##   edge
##   [
##     source 1
##     target 0
##   ]
## ]
```

Then, once we have an XML file we can convert to a JSON file using the R package rjson.[2] Getting the graphs in a JSON file formats makes working with them in the D3 library incredibly straightforward. Here is our example in the finalized JSON format.

```
## {
##   "nodes":
##   [{"id":"n0","v_id":"0","v_label":"Node 1","v_value":"100"},
##    {"id":"n1","v_id":"1","v_label":"Node 2","v_value":"200"}],
##  "edges":
##   [{"source":0, "target":1}]
## }
```

igraph is also used to compute initial x and y coordinates for the graph using a force-driven layout prior to being passed to the force layout in D3. This reduces the initial work that must be done by the javascript library and helps minimize unnecessary movement by the nodes. This is critical as the extra movement at the loading of the pages creates an unnecessarily chaotic start to the user's experience.

4.2 Data Management

5 Design and Functionality

Part III
Further Work

References

[1] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. "D3: Data-Driven Documents". In: *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011). URL: `http://vis.stanford.edu/files/2011-D3-InfoVis.pdf`.

[2] Alex Couture-Beil. *rjson: JSON for R*. R package version 0.2.12. 2013. URL: `http://CRAN.R-project.org/package=rjson`.

[3] Gabor Csardi and Tamas Nepusz. "The igraph software package for complex network research". In: *InterJournal* Complex Systems (2006), p. 1695. URL: `http://igraph.sf.net`.

[4] RStudio and Inc. *shiny: Web Application Framework for R*. R package version 0.4.0. 2013. URL: `http://CRAN.R-project.org/package=shiny`.