Data Manipulation Using HTML5

Keyur Prajapati[1], Patel Mayur H.[2]
[1, 2] MBICT, New V.V.Nagar – 388120, Gujarat, India

Abstract- *In this paper we evaluate the potential of the next major revision of HTML (Hy-pertext Markup Language), that is HTML5, to provide an effective platform for the transmission and visualization of vector based geographical data. Relative to the current version of HTML, HTML 4.01, HTML5 offers an improved platform to perform these tasks through greater inter-operability with existing technologies and the introduction of many new API's. Visualization of vector data can be achieved using the new methods of inline-SVG and the Canvas API. An anal-ysis of the pros and cons of each method is presented. HTML5 introduces a novel WebSocket API which defines a full-duplex communication channel between client and server. This pro-vides improved data communication both in terms of bandwidth utilization and network latency relative to existing push technologies. To demonstrate the effectiveness of HTML5 for vector data delivery a novel selective progressive transmission methodology is implemented using the WebSocket and Canvas API's.*

Keywords- HTML5, Web-Mapping, Progressive Transmission

I. INTRODUCTION

Web application development is in the middle of a paradigm shift (Kuuskeri and Mikkonen; 2009). The web browser is rapidly evolving from a renderer of simple HTML into a runtime environment capable of deliv-ering rich interactive applications across many application domains (Melamed and Clayton; 2010). From the users perspective, web-based applications are becoming more like traditional desktop applications. De-spite this progress, web-mapping applications still significantly linger behind their corresponding desktop counterparts in terms of both interface and functionality (Shi et al.; 2009). Much of this discrepancy can be attributed to a number of limitations of the current version of HTML. In this paper we focus on two limitations in particular. These are the inability to effectively visualize data or to perform efficient commu-nication between client and server. We discuss how HTML5 overcomes these issues and demonstrate this through a corresponding implementation. The layout of this paper is as follows. In section 2 we introduce the Canvas API and inline-SVG which represent the methods HTML5 provides for data visualization. Sec-tion 3 describes the webSocket API which HTML5 provides for effective communication between client and server. Section 4 describes a selective progressive transmission implementation which uses the Canvas and webSocket API's. Finally in section 5 we draw conclusions with some caveats about the use of HTML5.

II. DATA VISUALIZATION

Before the arrival of HTML5, client based data visualization could only be performed through plugins such as Adobe Flash and Scalable Vector Graphics (SVG) (Lubbers et al.; 2010). Although such plugins provide the required functionality they reduce the Web's openness and platform independence, and tend to lock users to specific technologies and vendors (Vaughan-Nichols; 2010). In fact, Ian Hickson, one of the W3C's HTML5 editors, states that "One of our goals is to move the web away from proprietary technologies". Due to the inability of the current version of HTML to visualize data, without the addition of a plugin, only pre-rendered data may be displayed. In the context of web mapping this means that the client browser cannot visualize 2-D vector map data. To overcome this problem most web-mapping applications, such as Google maps and OpenStreetMap, pre-rendered the data on the server and transmit the corresponding images to the client. The client cannot perform any spatial analysis using such an image-based representation. They also cannot personalize the map visualization process in any way - for example changing the colouring of objects. This represents a significant loss in functionality relative to desktop GIS applications. When working with large scale maps of high detail it is necessary to reduced such detail to prevent user information overload. The most common means of reducing such detail, and in turn information overload, is map generalization which is a well studied cartographical process (Corcoran et al.; 2011). Map generalization is a progressive process where details are gradually removed to produce a corresponding set of multi-scale maps. It is necessary to allow the user of any web-mapping application to freely choose and move between this set of scales. If the multi-scale maps are transmitted in their original vector format, any scale may be represented by the set of simplifications or refinements which must be applied to the previous scale (Corcoran and Mooney; 2011). The data size corresponding to these simplifications or refinements only represents a small percentage of the overall dataset size which would otherwise be transmitted. Therefore if a client requests a particular scale, using a vector representation reduces the data size which must be transmitted. On the other hand, if the

multi-scale maps are transmitted in an image-based format and a particular scale is requested, the entire data set corresponding to the scale in question must be transmitted. This generates significant network traffic and latency.

To overcome the above issues, HTML5 introduces two methods for data visualization. These are the Canvas API and inline-SVG which provide pixel and vector based visualization solutions respectively. Can-vas is a scripting based graphics environment which transforms drawing commands into a corresponding raster or image. Cartagen (Boulos et al.; 2010) is an open-source vector mapping framework developed at the MIT Media Lab which allows OpenStreeMap XML data to be downloaded and visualized using Can-vas. SVG is a language used to describe two-dimensional objects in XML is now supported as standard by most browsers. Much research on the applications of SVG to map representation has been performed (Wang and Meng; 2010). With HTML5, SVG has been natively integrated through a new <svg> element. Thus, you can now create web pages with inline-SVG graphics, where the SVG graphics are fully integrated with the rest of the page; e.g. following CSS styling, allowing JavaScript to interact with SVG objects, drawing graphics, or creating effects on hover-over of certain SVG objects (Pfeiffer; 2010). SVG represents a higher level means for visualization compared to the Canvas API. SVG provides a DOM (Document Object Model) and has an event model not available to Canvas. Thus, for applications that require graphics with interactivity, SVG represents the better platform. In cases where such functionality is not required, the Canvas provides better performance. It is worth nothing that Adobe Flash/Flex and Microsoft Silverlight have each arrived at the fact that both pixel and vector based visualization solutions are necessary. Consequently each technology has implemented corresponding solutions very similar to the Canvas API and inline-SVG.

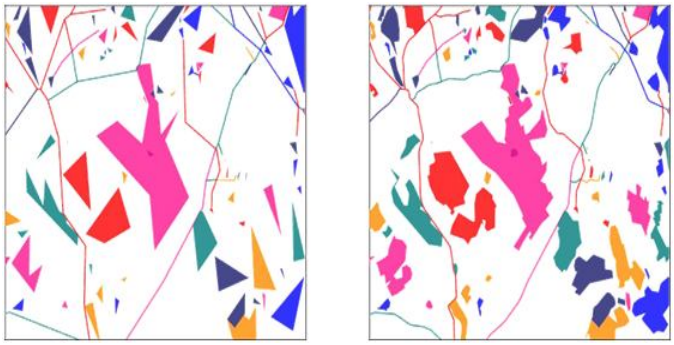
III. CLIENT-SERVER COMMUNICATION

Before discussing client-server communication we introduce the concepts of network bandwidth and latency which are fundamental attributes of network performance. The bandwidth of a network is equal to the number of bits that can be transmitted over the network in a certain period of time. Network latency equals the time it takes a message to travel from one end of a network to the other. Typically communication between a client and server operates as follows. The client generates an HTTP request; this is sent to the server which acknowledges this request and sends back the response. In certain cases the information contained in the response may become outdated over time. This would be the case if the information requested was current stock prices. To receive real time information the client may continuously refresh that page manually but this is not an elegant solution. Current attempts to provide real-time web information generally use polling and other push technologies, the most notable of which is Comet, which delays the completion of an HTTP response. Push technologies exhibit a number of disadvantages. All communication involves HTTP request and response headers, which contain additional unnecessary header data and introduce latency. To simulate full-duplex communication over half-duplex HTTP, two connections are used and this is very computationally inefficient. Lubbers et al. (2010) states that ”simply put, HTTP wasnt designed for real-time, full-duplex communication”. This has obvious implications when attempting to transmit vector data over the internet. To overcome these limitations HTML5 introduces the WebSocket API which provides a full-duplex communication channel that operates through a single socket over the web. This approach to client server communication provides a significant improvement in network performance, both in terms of network bandwidth and latency, when compared to existing push technologies (Lubbers et al.; 2010).

IV. SELECTIVE PROGRESSIVE TRANSMISSION

In order to demonstrate the potential of HTML5 to provide effective web-mapping solutions a progressive transmission strategy for the transmission and visualization of vector data was implemented using HTML5. This work represents an implementation of some concepts presented in Corcoran and Mooney (2011). The data used in this work was taken from OpenStreetMap (OSM). The data is downloaded initially in OSM XML format. It is then processed, generalized and stored on the server. Client interaction with the map, in the form of zooming and panning, is continuously tracked to allow the current viewing window to be determined; this information is in turn continuously transmitted to the server. Upon processing the server determines what data must be added or removed from the clients local data set and this is subsequently transmitted. This step uses an R-tree data structure to perform spatial queries (Samet; 2006). If the client performs a map zooming, detail is added only to those objects in the view window such that they are represented at a finer scale. This implements the concept of selective progressive transmission. Figure 1(a) displays the Canvas element of the proposed web-mapping system for a given view. Following a zooming by the client the map is progressively refined to increase detail; the corresponding result is displayed in Figure 1(b). All communication between server and client is performed using the WebSocket API. The vector data is visualized using the HTML5 Canvas API. The Canvas was chosen over inline-SVG because this application does not required the user to select individual objects in the map. The

proposed system differs from the Cartagen project (Boulos et al.; 2010), which also uses the Canvas API, by the fact that it is an implementation of selective progressive transmission and the Cartagen project is not. Network bandwidth performance of the proposed system was determined using the Linux tool iptraf. When compared to OSM, which is a tile-based mapping system, the proposed system exhibited significant superior bandwidth performance.



(a) (b)

Figure 1: Progressive transmission of the map is performed when the client performs map zooming.

V. CONCLUSIONS

In this paper we demonstrate that HTML5 represents an effective platform for the effective transmission and visualization of vector data. Despite this it is advised that developers should proceed with caution. HTML5 is still in development and many of its features are not widely supported by different browsers. It is estimated that it will be 2013 at the earliest until HTML5 achieves a sufficient level of browser support to be considered reliable enough for deployment (Vaughan-Nichols; 2010). Also many people have raised security concerns regarding HTML5 which must be considered (Mansfield-Devine; 2010).

BIBLIOGRAPHY

[1] Boulos, M., Warren, J., Gong, J. and Yue, P. (2010). Web GIS in practice VIII: HTML5 and the canvas element for interactive online mapping, International Journal of Health Geographics 9(1): 14.

[2] Corcoran, P. and Mooney, P. (2011). Topologically Consistent Selective Progressive Transmission, 14th AGILE International Conference on Geographic Information Science

[3] Lubbers, P., Albers, B. and Salim, F. (2010). Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development, Apress.

[4] Melamed, T. and Clayton, B. (2010). Mobile Computing, Applications, and Services, Springer, chapter A Comparative Evaluation of HTML5 as a Pervasive Media Platform, pp. 307–325.

[5] Pfeiffer, S. (2010). The Definitive Guide to HTML5 Video, Apress.

[6] Samet, H. (2006). Foundations of Multidimensional and Metric Data Structures, Morgan Kaufmann.