An Architecture for Adaptive Content Extraction in Wireless Networks

P. West, G. Foster, and P. Clayton, *Rhodes University*.

Abstract— Content extraction is used as a method of reformatting web pages to display more appropriately on handheld devices and to remove unwanted content.

A framework is presented that facilitates content extraction, which is extended to provide personalization of extraction settings and adaptation to network conditions. The system is implemented as a proxy server for a wireless network and handles HTML documents. Once a document has been requested by a user, the HTML is retrieved and parsed, creating a Document Object Model tree representation. It is then altered according to the user's personal settings or predefined settings, based on current network usage and the network resources available. Two extraction techniques were implemented; spatial representation, which generates an image map of the document, and tag extraction, which replaces specific tags with links.

Testing was performed by accessing sample web pages through the content extraction proxy server. Tag extraction works correctly for all HTML and XML document structures, whereas spatial representation is limited to a controlled subset. Results indicate that the adaptive system has the ability to reduce average bandwidth usage, by decreasing the amount of data on the network, thereby allowing a greater number of users access to content. This suggests that adaptive content extraction has a positive influence on network performance metrics.

Index Terms—Content extraction, Handheld devices, Quality of Service, Network performance metrics.

I. Introduction

RECENT computing trends indicate a global increase in the use of handheld devices [1]. However, there are two major barriers preventing these devices being used extensively for web browsing; the lack of access to high-speed wireless networks and the relatively small screen size [2]. The emergence of high-speed 3G networks has gone a long way to solving the first issue, where mobile users desire updated information in real-time [3]. Content extraction attempts to address the second issue by manipulating documents to be more suited for display on handheld devices. The premise is that the majority of web documents contain irrelevant content, such as advertisements and link lists. By removing this "clutter", the size of the document is reduced, while retaining the relevant content. Numerous content extraction approaches have been developed [2] [4] [5] [6] [7].

A. Text Summarization

A technique referred to as text summarization improves the network access times of handheld device users and reduces the number of required pen actions while browsing the internet [4]. This approach displays web pages in text, and allows sections of the pages to be hidden, partially displayed, summarised or displayed in full.

Text summarization is extended to include a solution for handling graphics [5]. The web page is arranged into a two-tier hierarchy. The first tier provides an overview of the web page in the form of a graphical table of contents, which is displayed as a thumbnail image (Fig 1a). Content is grouped according to semantic relation, and these groupings are then displayed on the thumbnail in different colours. In order to view content the user selects a colour block, which has been formatted for display on a small screen (Fig 1b). There are two main processes; page analysis and page splitting or auto positioning. The web page is first parsed into a Document Object Model (DOM) tree to allow for manipulation. Page analysis attempts to extract the semantic structure of the web page, grouping content by inferring the content layout. This is derived from high-level content blocks, such as headers, footers and sidebars, and visual separators such as white space.

a b

Fig. 1. Text summarization extended to include graphics [5]. (a) A web page represented as a thumbnail image. (b) The content corresponding to the block selected in (a).

Single-subject splitting divides the page into several smaller pages which are connected by hyperlinks and is used for pages with a particular topic. Multi-subject splitting

This work was undertaken in the Distributed Multimedia Centre or Excellence at Rhodes University, with financial support from Telkom SA, Business Connexion, Verso Technologies, THRIP, the National Research Foundation and Microsoft.

All authors are with the Computer Science Department, P O Box 94, Rhodes University, Grahamstown 6140, South Africa.

(P. West. g00w4340@campus.ru.ac.za; 046 603-8642)

divides the page into smaller pages, and generates an index page which is used for navigation. This approach is used to split an entire web site. Auto-positioning is suggested as an alternative to page splitting. The process is similar in that a thumbnail view of the page is generated, displaying the content blocks in different colours. However, when a particular block is selected, instead of splitting the page it is automatically positioned correctly to display the selected content. Certain implementations have extended this approach to split pages and deliver the sections to different devices [6].

B. Crunch

Content extraction has been implemented as a means of removing "clutter" from web pages [8]. By avoiding methods which remove images, adjust font sizes and disable JavaScript, the inherent "look and feel" of documents has been maintained. This approach has been adapted for the benefit of visually impaired users [9]. Screen readers for the blind do not automatically convert HTML into plain text. In this case, content extraction benefits these readers by having the ability to produce plain text as output. The system was implemented as a proxy server, called "Crunch". An HTML parser stores the requested document in a hierarchical DOM tree, which can then be manipulated by the filter algorithms. The set of filters consists of a tag omitting filter, an advertisement remover, a link list remover and an empty table remover. The advertisement remover utilises the conventional technique of determining the servers to which links on the table refer. Server addresses obtained from the links are compared to a table of known advertisement servers and the tags containing these fields are omitted if a match is found. The link list remover identifies and filters large link lists, which often occur on the sides of web pages, leaving the main body of relevant content intact. Partially or completely empty tables are identified and removed.

Crunch exhibits various limitations. Filtering is restricted to HTML content. Non-HTML content can either be allowed or filtered; it is not possible to filter within the content. Dynamically generated pages, scripts such as JavaScript, do not filter well and therefore must also be either completely omitted or allowed. The system does not permit personalisation as the settings can only be changed on the proxy, affecting all users.

Networks, such as GPRS, provide a billing module where users are charged according to the amount of data they download. Content extraction facilitates this approach by exposing content in pages, allowing users to only download desired content. Modern networks make ever increasing bandwidth available to users. In these instances, extraction will have little or no effect on network performance. However, many legacy networks with limited bandwidth still exist, and could benefit from such an approach.

This paper presents a mark-up language, content extraction system, based on the techniques of [5] and [8], for use with wireless networks. Our system is designed to provide users with the ability to select only the content they desire. The architecture is extended to provide an algorithm for automatically adapting extraction settings according to varying network conditions, and functionality is provided to allow users to personalize their extraction settings.

II. A Framework for Adaptive Content Extraction

Our system is implemented in four main sections (Fig 2). A proxy server listens for requests from mobile clients. Once a request is received, the server obtains the corresponding web document from the internet or a local intranet site. The document is then manipulated by the extraction plug-in, using spatial representation and/or tag extraction. This altered document is then returned to the client.

A browser plug-in allows mobile clients to define their own extraction settings. These preferences are appended to requests sent to the proxy server, which stores the settings in a server-side database. A network monitor tracks the amount of traffic on the network (in bytes), and alters the extraction settings to adapt to these conditions.

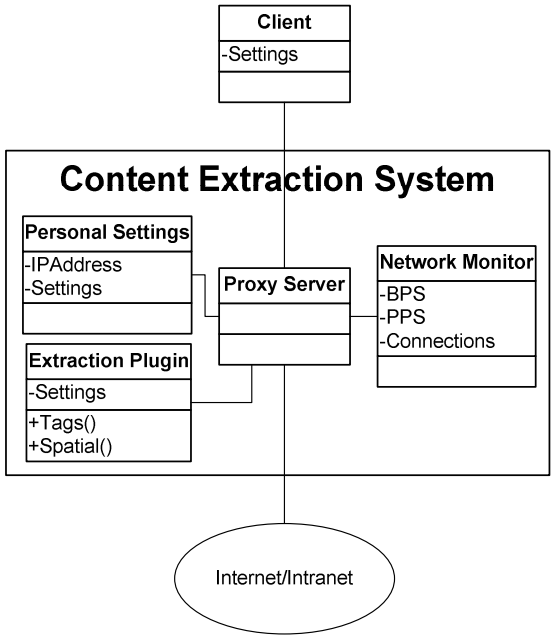


Fig. 2. Adaptive content extraction system architecture (overview).

A. Proxy Server

The adaptive extraction system has been implemented as a proxy server. In this way, all requests from clients received through a wireless access point can handled accordingly. The extraction functionality was implemented as an extension to Crunch 2.0, an extensible, thread-based proxy server which was developed by [8] using Java. Reflection provides the opportunity to develop plug-ins for the Crunch system. Each plug-in takes a DOM tree representation of an HTML document as input, which can then be altered as required before being returned to the proxy. Crunch provides functionality to perform certain content extraction methods, as described previously, implemented as plug-ins which can be enabled or disabled at any stage.

B. Content Extraction

This section of the system handles the actual manipulation of content by examination of the underlying mark-up. As

mentioned earlier, two extraction techniques have been developed; spatial representation and tag extraction. The algorithms are implemented as a plug-in to the Crunch system.

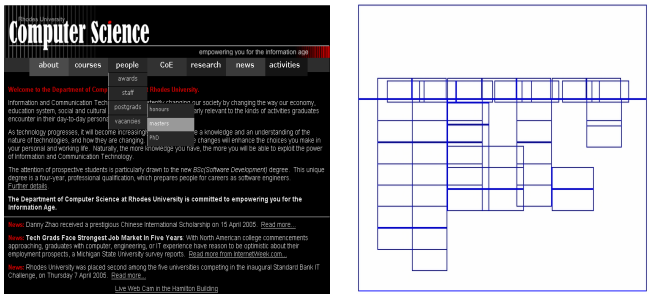
Spatial representation is based on the approach described by [5]. Content pages are represented as a thumbnail image divided into differently coloured content blocks. Each block links to the corresponding content group contained in the DOM tree. The image representation is created by rendering relevant nodes in the tree using the ICEBrowser SDK for Java [10]. Once the nodes have been rendered, it is possible to obtain the coordinates of the bound box in which each node is contained. These coordinates are then used to create a standard html image map of the content page.

The spatial representation algorithm was tested using a controlled data set, consisting of a variety of HMTL web pages. Each page contained one or more tables, with every table containing an unlimited number of text boxes, images, video or sound. The documents contained no script or nested tables. Fig 3 illustrates the effects of translating a test page using the spatial representation algorithm. Fig 3a shows the original page as rendered by Internet Explorer, Fig 3b displays the image map generated from the HTML, and Fig 3c shows the result of selecting content block *i* displayed in Fig 3b.

a b c

Fig. 3. Results of the spatial representation algorithm. (a) The original HTML page. (b) The page represented by the spatial algorithm, containing content blocks *i*,*ii* and *iii*. (c) The result of selecting content block *i* from (b).

Although the algorithm was successful in this controlled environment, these results could not be replicated for all possible HTML document structures. Embedded script has the potential to cause artifacts in the final image map. Fig 4a shows an example where JavaScript was used to create a drop-down menu on a page (http://www.cs.ru.ac.za/). The entire script is rendered during page analysis, creating erroneous content blocks in the final image map (Fig 4b). Differences in document structure may cause inconsistencies in the way content is represented. For example, a nested table, consisting of numerous images, would be incorrectly represented as a single, large content block instead of exposing each image.

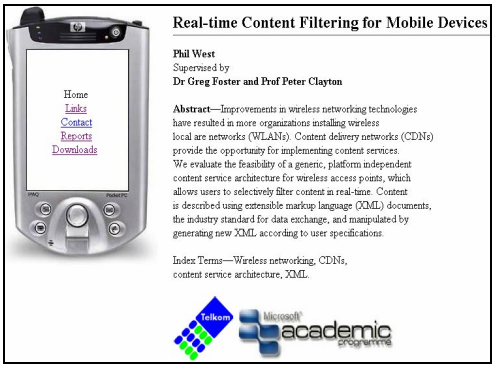


a b

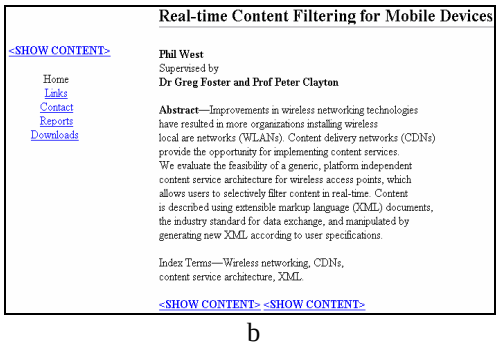
Fig. 4. Screenshots illustrating the artifacts created by applying the spatial representation algorithm to pages with script. (a) The test page (htpp://www.cs.ru.ac.za/). (b) The results of applying the spatial representation algorithm.

In tag extraction the DOM tree is parsed to identify pre-defined tags, which are then replaced with a link to the content described. Tags that require extraction are explicitly specified in the tag extraction settings, for example specifying "<IMG>" will replace all the images in requested documents.

Experiments conducted using the tag extraction algorithm showed the approach to be successful for all possible correct HTML and XML structures. However, it is not possible to extract within embedded content such as Macromedia Flash, but only to either extract all or none of this content. A second limitation is introduced by the use of XML, where it is possible to name tags using unique, non-standard identifiers. This is not an issue with HTML documents since tag names are standardized. Fig 5a shows an example web document. Fig 5b illustrates the same HTML document with the tag extraction algorithm successfully applied, in this case with all images replaced by links.



a



b

Fig. 5. Results of the tag extraction algorithm. (a) A web page with no extraction applied. (b) The same page after tag extraction was applied to images.

C. Personalization

Functionality is provided for users to select their own extraction settings, supporting users who wish to control the type of content they receive. This approach allows users to set the parameters for the tag extraction algorithm. A Browser Helper Object (BHO) was developed to extend the clients web browsing capabilities. The BHO provides drop down menus from which users control the extraction criteria, which are then appended to web requests. Settings are appended on the first request, and any subsequent request made after the criteria have been altered. The proxy server maintains a SQL database, containing session information about each user's extraction preferences.

D. Adaptive Extraction

Previous work on content extraction has had an influence on network performance metrics [11]. This is due to the fact that during the extraction process data are discarded or manipulated, usually resulting in a reduction in size of the content page being processed. However, this influence has been a side effect and not measured or controlled. The extent to which content extraction influences performance metrics, such as bandwidth usage, is evaluated. The extraction framework is extended to allow for adaptation to network conditions as a method for increasing the available bandwidth in legacy networks.

The adaptive algorithm monitors network conditions and automatically applies extraction techniques at set threshold values. Extraction methods are applied in a specified order of precedence, the default being video and sound, then script and finally images. An essential part of the system is the ability to monitor network conditions. In particular, the focus is on the amount of traffic on the network (in bytes). WinPcap [12], an open source packet capture and network analysis library for Windows, was utilized to achieve this. Since the WinPcap library was developed for use with C++, it was necessary to create a Java wrapper class in order to expose the methods required by the proxy server. In this way the proxy has access to network statistics; the average bytes transmitted per second, average packets transmitted per second and the number of connections made in a specified time period.

Sample HTML documents were created to evaluate the effects of the adaptive algorithm on network statistics. Testing was carried out in a controlled environment using Microsoft Application Centre Test (ACT) and Proxy Sniffer, two tools for testing the load capabilities of web servers [13] [14]. A script was created that evaluated the performance metrics of the test web pages by continuously increasing the number of simultaneous connections. All requests were directed through the content extraction proxy server. The amount of available bandwidth was limited in order to simulate a legacy network.

Fig 6 illustrates the general trend discovered using the adaptive approach, showing bandwidth usage as a function of the number of simultaneous connections. With no extraction (— — —), the bandwidth usage increases following a linear trend. The amount of available bandwidth is shown by the bandwidth cap (– - – -); 30 Kbps. The adaptive algorithm (———) extracts content at various levels in order to prevent bandwidth usage from exceeding the cap, first extracting video and sound, then script, and finally images. In this way, the adaptive algorithm frees network resources, which allows a greater number of users to access content through the server. In this case, bandwidth usage with no extraction (— — —) exceeds the cap at 15 users (Fig 6a). Video and sound objects are extracted at this point, reducing the usage to below the cap. This process is repeated for script (Fig 6b) and images (Fig 6c) when the cap is reached again. In this way it is possible to extend the number of users that can access content from 15 to 50 (Fig 6d).

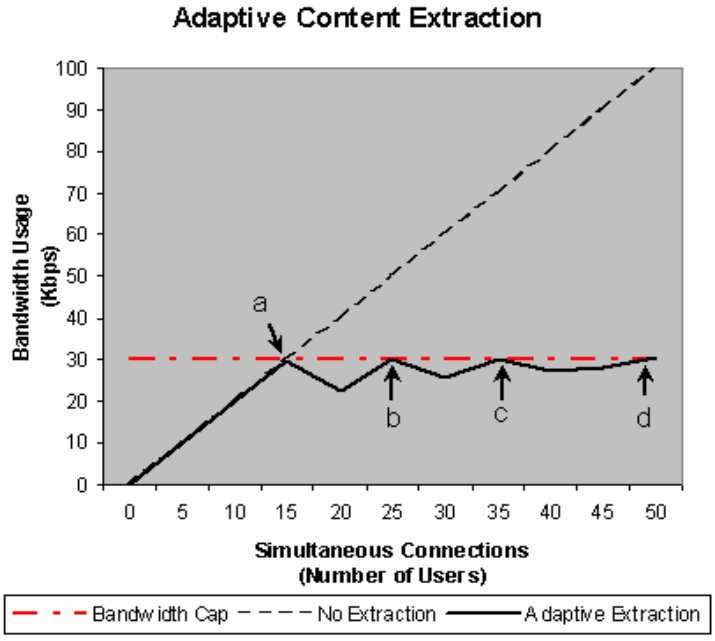
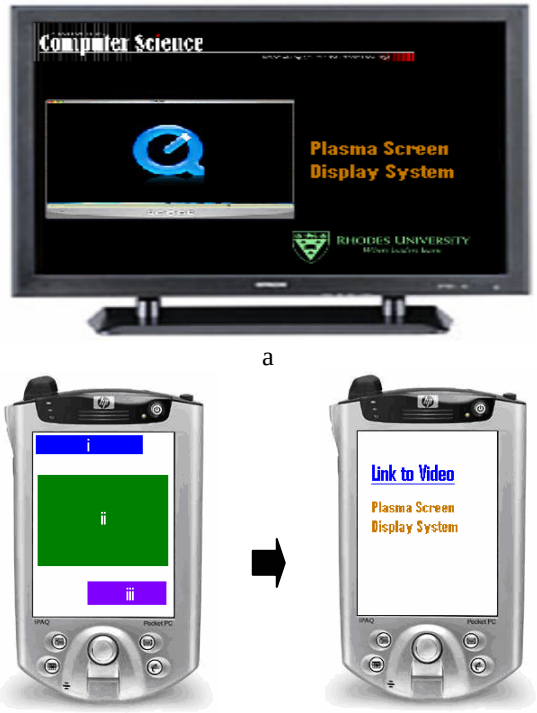


Fig. 6. The effects of utilizing adaptive content extraction as a method of bandwidth management. (a) The bandwidth cap is reached, and video and images are extracted. (b) Script is extracted. (c) Images are extracted. (d) The bandwidth cap is exceeded.

III. Implementation

Two working versions of the adaptive extraction framework have been successfully implemented. An XML-based content delivery system was developed for use as an information or advertisement service. Content displays, described by XML documents created using a centrally controlled system, are rendered on a plasma screen (Fig 7a). The extraction framework was adapted to expose this content to users through a wireless access point, providing a simplified method of accessing the data (Fig 7b). Content blocks may consist of multiple content types, as shown by block *ii* (Fig 7b) which corresponds to the video and text objects (Fig 7a). When block *ii* is selected tag extraction

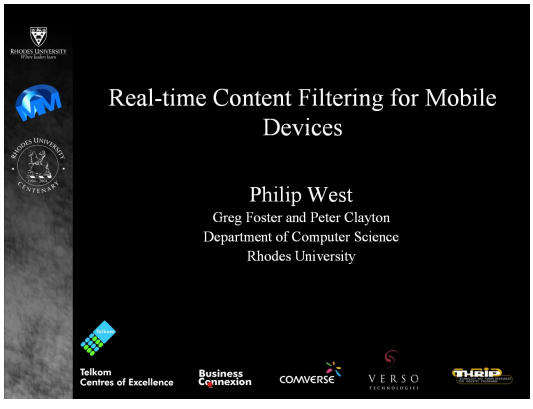
removes rich content, replacing the video with a link (Fig 7c).



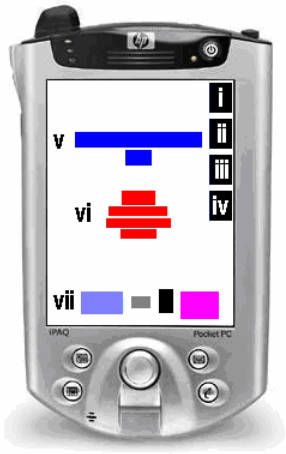
a b c

Fig. 7. The content extraction framework adapted for use with an XML-based display system. (a) A plasma screen displaying a video, 2 images and a block of text. (b) The spatial representation of this display. (c) The result of tag extraction, after selecting block *ii* (b).

The second implementation, currently under development, is designed to expose lecture presentation content to students with mobile devices. Slide show documents are converted to HTML (Fig 8a), which is then made available via a hotspot. Spatial representation is utilized to reduce the size of each slide (Fig 8b). Blocks *i*, *ii*, *iii* and *iv* are appended to each slide representation to allow students to navigate. Selecting block *i* will display the entire slide, block *ii* will display only the background image, block *iii* will navigate to the next slide, and block *iv* will navigate to the previous slide. A rich-client interface is under development and will allow students to selectively store and annotate relevant content.



a



b

Fig. 8. The results of the spatial representation algorithm as adapted for use with slide show presentations. (a) An example PowerPoint slide. (b) The slide spatially represented.

IV. Conclusion And Future Research

The filtering techniques implemented in this project have positive implications for small screen devices, pay-for-data, and legacy networks. Extraction algorithms can potentially reduce the size of rendered HTML and XML documents. Exposing content to users provides the functionality to selectively choose which data is downloaded, reducing the cost of using pay-for-data networks such as GPRS. This extraction system can be further enhanced using the set of filters introduced by [8]. The adaptive extraction algorithm can be used to manage the bandwidth of legacy networks, allowing more users access.

Future investigation will focus on increasing the performance and functionality of the algorithms. Presently, tag extraction replaces content within a document by inserting a link pointing to this extracted content. When a link is selected the content is loaded in a new page, removing it from the original context in the document. The aim is to implement an approach where the content will be loaded in the correct place in the appropriate document, thus maintaining the intended context. A usability study of the system is planned. At this stage results indicate that algorithms function correctly, however, feedback from users is required to further evaluate the positive and negative outcomes of the adaptive content extraction approach. Further experimentation is required to evaluate fully the effects of the adaptive algorithm on network performance metrics.

References

[1] Ridgeway, M., ".NET Wireless Programming", SYBEX Inc., 1151 Marina Village Parkway, Alameda, CA 94501, United States of America, 2002.

[2] Alam, H., and Rahman, F., "Web Document Manipulation for Small Screen Devices: A Review". Second International Workshop on Web Document Analysis (WDA2003), Edinburgh, August 3, 2003.

[3] Beck, A. and Hofmann, M., "Enabling the Internet to Deliver Content-Oriented Services", Proceedings of the Sixth International Workshop on Web Caching and Content Distribution, Boston University, Boston Massachusetts, USA, June 20-22, 2001.
[4] Buyukkokten, O., Garcia-Molina H. and Paepcke, A. “Text Summarization of Web Pages on Handheld Devices”. Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL), Pittsburgh, PA, U.S.A., 2-7 June, 2001.
[5] Chen, Y., Ma, W.Y., and Zhang, H.J. “Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices”. Proceedings of WWW’03 Budapest, Hungary, May 20-24, 2003.
[6] Han, R., Perret, V., and Naghshineh, M., “WebSplitter: A Unified XML Framework for Multi-Device Collaborative Web Browsing”. Proceedings of the *ACM* Conference on Computer Supported Cooperative Work (CSCW), 2000.
[7] Kunze, M. and Rosner, D., "An XML-based Approach for the Presentation and Exploitation of Extracted Information". Proceedings of the First International Workshop on Web Document Analysis, Seattle, WA, September 8, 2001.
[8] Gupta, S., Kaiser, G., Neistadt, D. and Grimm P., "Automating Content Extraction of HTML Documents". World Wide Web Journal, January 2004.
[9] Gupta, S., Kaiser, G., Neistadt, D. and Grimm P.(2003) "DOM-based Content Extraction of HTML Documents". Proceedings of WWW’03 Budapest, Hungary, May 20-24, 2003.
[10] ICEBrowser SDK. Available: http://www.icesoft.com/
[11] Lyijynen, M., Koskinen, T., Lehtonen, S., and Pesola, J., “Content Adaptation on LANE Active Network Platform”. Proceedings of the 7th International Conference Telecommunications (ConTEL), 721- 724 vol.2, 11-13 June, 2003.
[12] Degioanni, L., Baldi, M., Risso, F., and Varenni, G., “Profiling and Optimization of Software-Based Network-Analysis Applications”.
[13] Hasan, J., and Tu, K., “Performance Tuning and Optimizing ASP.NET Applications”. Apress, 2560 9th Street, Suite 219, Berkeley, CA 94710, United States Of America, 2003.
[14] Proxy Sniffer: Web Load Test & Web Page Analysis Tool. Available: http://www.proxy-sniffer.com/.

P. West, a Telkom bursary holder, obtained a BSc Honours degree in Computer Science from Rhodes University, Grahamstown in 2003 and is currently reading for his MSc in Computer Science at the same institution.