Evaluating the feasibility of maintaining Web 2.0 communications during civil emergencies

James Meneghello

School of Engineering and Information Technology,
Murdoch University,
Perth, Western, Australia
Email: j.meneghello@murdoch.edu.au

Kevin Lee*

School of Science and Technology,
Nottingham Trent University,
Nottingham, UK
Email: kevin.lee@ntu.ac.uk
*Corresponding author

Abstract: Maintaining communications across disconnected networks can be a troubling issue, particularly since existing solutions are highly theoretical or have unrealistic hardware requirements. As a result, CANDICE was designed with the aim of allowing extensible accessible facilitation of internet connections in environments not suited to reliable communications – particularly hostile zones. Unfortunately, current web development trends (including a heavier emphasis on user interaction and asynchronous data transfer) ensure that less useful content is accessible without an uninterrupted internet connection. By evaluating the operation of CANDICE in retrieving popular websites, types of content that are well-suited to transmission over disrupted networks can be determined.

Keywords: networking; internet and communications; disconnected networks; Web 2.0; fault tolerance.

Reference to this paper should be made as follows: Meneghello, J. and Lee, K. (2018) 'Evaluating the feasibility of maintaining Web 2.0 communications during civil emergencies', *Int. J. Humanitarian Technology*, Vol. 1, No. 1, pp.66–82.

Biographical notes: James Meneghello is a Senior Data Scientist at Optika Technologies and PhD candidate at Murdoch University, and spends his time exploring new ways to extract social data from the unstructured internet. He completed his BSc in Computer Science and Internet Software Development while co-founding and operating a web development firm in Perth, Western Australia, before continuing on to an MSc in Distributed Systems and a PhD in Social Data Extraction and Integration. He now works in data science, utilising his myriad skills to develop bespoke solutions to highly unique challenges using natural language processing and machine learning.

Kevin Lee is a Senior Lecturer at Nottingham Trent University, UK. He received his BSc, MSc, and PhD from Lancaster University. He was previously a Research Associate at the University of Manchester in the UK, Postgraduate Research Fellow at University of Mannheim in Germany and Senior Lecturer at Murdoch University in Australia. He has published over 60 papers in the areas of distributed systems and adaptive systems.

1 Introduction

While the majority of IP networks incorporate reliability into their architectural design, not all situations can be anticipated and mitigated. Powerful forces of nature, initiation of war and a number of other events can be grouped into what we call 'catastrophe events'. In the case of network infrastructure, even a minor event [such as a ship anchor being dragged along the sea bed (Hamblen, 2008)] can result in interruptions of international links for entire countries. In most cases, some level of redundancy exists; multiple transnational fibre pipes generally exist between a country and its neighbours, allowing traffic to be rerouted in the event of connection interruption. Civil emergencies [particularly those involving governmental (BBC, 2011)], or increasingly, terrorist actions (Solomon, 2015) can result in a complete shutdown of all traffic to an area.

A proposed solution to the loss of internet traffic involved in such a situation is to replace the disconnected link with a system of couriers carrying physical disks, mimicing the function of packets in an IP network. While IP networks have protocol functionality in place to provide a range of options regarding reliability, fault tolerance and speed, existing SneakerNet (Gray et al., 2002) techniques are not designed with these attributes in mind (being primarily used for small volumes of data).

In instances where no direct user interaction is possible, web robots often encounter difficulty when retrieving content from sites due to a lack of server configuration standards. Assumptions must be made about external server configurations that are often incorrect and must be tailored on a site-by-site basis, which tends to be a tremendous feat when dealing with the amount of content potentially available to users. Current web development trends are also making delayed networking increasingly difficult. With heavier emphasis on client-side user interfaces and asynchronous data transfer from the to server such as Asynchronous Javascript And XML [AJAX (Garrett, 2005)], less static content is available without some kind of user interaction on a micro-connection basis.

In an attempt to evaluate the facilitation of communications in such environments, CANDICE was developed – and by evaluating the effectiveness of transferring popular sites across the CANDICE system, the possibility and reliability of managing effective communications can be determined.

The remainder of this paper is as follows. Section 2 provides background information and previous relevant solutions that have been determined to help in some way with communicating across disconnected networks. Section 3 describes the design and implementation of the system developed for the purpose of addressing communications across disconnected networks (CANDICE). Section 4 evaluates the potential in Web 2.0 communications in a number of situations. Section 5 discusses the results of the evaluation. Finally, Section 6 presents some conclusions.

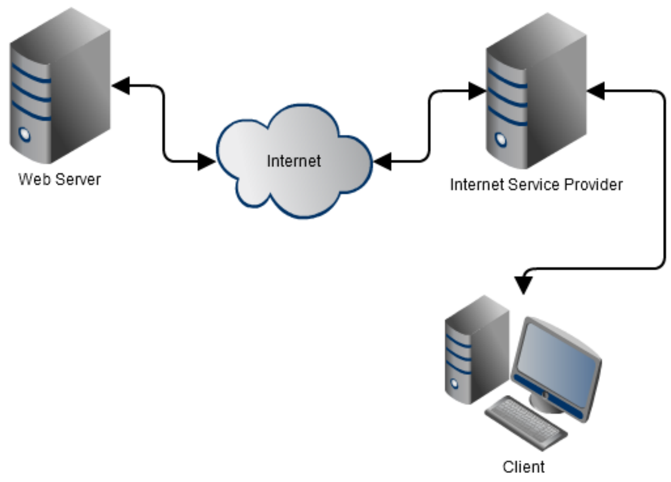
2 Background

Under normal circumstances such as those represented in Figure 1, a user is able to access the internet through a connection to their internet service provider, using a phone line, fibre-optic cable or other medium. The ISP then passes that traffic through an upstream provider that facilitates international communications, using fibre-optic cables designed to handle large quantities of data. The network is considered to be disconnected when the international uplink is either disabled or severed completely, with varying degrees of delay in repair.

2.1 The effect of disconnected networks

During a communications shutdown international links are often disabled or severed, as represented in Figure 2 as a disconnected link. As a result, connections are not able to flow freely from client users to international web servers. The majority of larger internet sites are based in a number of locations around the world (but require constant connectivity to other sites), so major communications tools such as Twitter, Facebook and even e-mail services are usually affected.

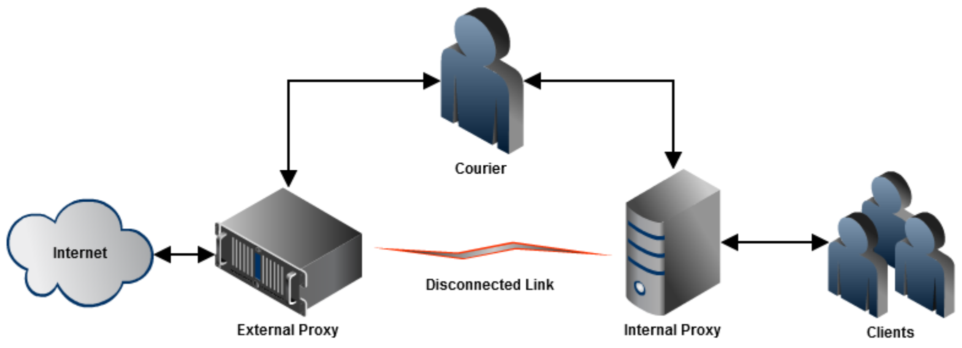
Figure 1 Normal web connection architecture (see online version for colours)



While external links are disconnected, the connections existing from the ISP to client usually remain intact. In order to shut them down, a concerted effort to sever backhaul links would need to be made – generally requiring an extensive resource outlay. Because these individual connections still exist, there remains an opportunity to restore some connectivity through manually recreating a physical link from some point in the internal network to some point with internet connectivity (usually outside civil borders). By using a courier to manually ferry data between internal and external proxies, we can restore connectivity to home users or individual organisations.

Communications are an important service in times of emergency – they allow for improved safety, group organisations and the ability to contact lost relatives. There are several useful applications [such as Ushahidi (Okolloh, 2009), an open-source real-time crisis mapping application] that can be of extraordinary benefit to welfare and/or protest groups by consolidating emergency information (such as meeting locations) and other communications into a single point of access. If atrocities are being committed, having the ability to disseminate evidence amongst the wider global community allows it to intervene and prevent the situation from worsening.

Figure 2 Proposed physical model (see online version for colours)



2.2 *Reconnecting networks*

There is a significant amount of research previously undertaken (Fall et al., 2007; Burleigh et al., 2003; Menoher and Mraz, 2009; Gray et al., 2002; Demmer et al., 2004) into numerous areas related to the topic, as it encompasses a broad spectrum of fields such as fault tolerance, delay-tolerant networking, disconnected networks, SneakerNetting, general IP networking and web content studies. Content matter specifically relevant to this system and situation has been included below.

Delay tolerant networking offers numerous solutions to this problem, including courier-style systems. While deemed too complex for the simple extendable solution that is desired, Delay Tolerant Networking Protocol (DTN2) and its associated RFCs (Fall et al., 2007) provide a solution designed to bring internet connectivity to societies that have unstable communications networks. An earlier iteration of DTN was previously developed to provide distributed delay-tolerant filesystems, named TierStore (Demmer et al., 2004). TierStore has since been used to develop other applications that operate in a delay-tolerant manner, such as a distributed wiki system (Du and Brewer 2008), while DTN2 has been used in applications such as intermittent and interplanetary networking (Burleigh et al., 2003).

Web development trends, systems and standards are key to the problem as they are often tailored towards making delay-tolerant networking more difficult (though unintentionally). The increase in user interactivity is inversely proportional to the ease of non-interactive access, and providing a smoother, richer user interface comes at the expense of flexibility. The global shift to Web 2.0 paradigms (Murugesan, 2007) has made automated content mirroring without user interaction difficult, with individual processes needed on a site-by-site basis. RuralCafe (Chen et al., 2009) is designed to enhance web accessibility in developing countries. Research describing the physical aspect of the system (SneakerNet) (Gray et al., 2002), was completed a number of years ago – but the technique became less relevant with the introduction of wide-scale broadband accessibility in developed countries.

3 CANDICE

In order to provide a stable platform for evaluating the feasibility of reliable CANDICE was developed. By implementing a pluggable architecture able to be extended to handle numerous protocols and web interfaces, CANDICE can be used as a proof-of-concept to analyse popular web content while providing a potentially viable method of handling communications. While the initial functional requirements are minimal (allowing only basic static content retrievable by robot), the system's extensible nature can allow a wide variety of content to be transferred with additional plug-in development.

3.1 Overview

The CANDICE system comprises a number of components, illustrated in Figure 2. Each of these components is required to be functional for the system to operate.

3.1.1 Client

Clients represent any user that wishes to request or submit information to be communicated to the internet. This could be a number of entities; a user working in an activist organisation with a computer, a user sitting in a cafe with an open wireless access point using a smartphone or any other user operating a device that is able to use a standard web browser. A client should not be required to possess any specialised hardware or software, and this category should be able to encompass as large a group of people as possible.

3.1.2 Internal proxy

An internal proxy is a simple computer operating as a server with some specialised scripts installed. It can be placed at any point in a network – connected to a wireless access point in a cafe, as a server in an ISP or even set up in a library. All other computers connect through the internal proxy, which intercepts requests and stores them to be transferred via courier to the external proxy. There can be multiple internal proxies operating within any area.

3.1.3 Courier

Couriers manually visit proxies to transport data across the disconnected links and can operate with anything from a USB thumbdrive hidden under clothing to a car full of hard drives. Smartphones that enter the proximity of proxies can be used to transfer data. Couriers would generally experience hostile conditions during the course of their journey, and unfortunately are not completely reliable – being human, they are prone to coercion and/or death.

3.1.4 External proxy

An external proxy is a computer with an active internet connection with a script that processes requests from couriers. The script then retrieves/submits the requests, saving any feedback or requested data for transport back via courier. There can be multiple external proxies, and any external proxy should be able to handle requests from any internal proxy.

3.2 Requirements for comms. across disconnected networks

Each component of the system has a number of requirements that it should adhere to; individually specified as 'shall' or 'should'. 'Shall' requirements are mandatory, critical parts of the system that should be functional in order for the system to operate correctly.

'Should' requirements are optional, but provide a significant boost to efficiency and speed of transfer.

- shall allow, at a minimum, the user to directly request a static web page (or site) accessible through a direct URL, without the need for dynamic interaction (such as AJAX)
- shall only require an HTTP client to request web pages (such as a web browser)
- shall be accessible with a minimum of hardware requirements, using only commodity hardware
- shall allow at least the request of static web content
- should provide extensibility of request types through a plug-in architecture
- should provide redundancy of transmissions
- should prevent interception of transmissions
- should have the ability to operate in a hostile environment, though not necessarily without loss of life or delay.

3.3 Constructs

- *Disconnected link:* An internet link that has been severed, generally an upstream/backhaul connection that operates at the ISP-level or higher.
- *External proxy:* A computer with persistent internet connectivity with specialised software installed to facilitate communications with a number of internal proxies.
- *Internal proxy:* A server or computer with functional internal networking but interrupted internet connectivity, acting as a transparent proxy for a number of client computers to intercept web requests and record them for manual transport to the external proxy.

- *Transparent proxy:* A server that intercepts web requests made hierarchically below the server and modifies them, without the need to modify the client.
- *Client:* A device (such as a PC, smartphone, etc.) with a web browser and a connection to an internal proxy, either directly (through WIFI or similar) or indirectly (if the internal proxy is operating at an ISP-level).
- *Package*: A hard drive containing a number of requests (if headed towards the external proxy) or responses (if heading back to internal proxies). Encrypted at a disk level, to ensure no interception.
- *Request:* A web request sent by a client to an internal proxy (and then packaged for transmission to an external proxy), requesting a website, web page or submitting some form of data for upload. Converted for storage and transmission by the internal proxy when outgoing (and vice versa for external when incoming).
- *Response:* A reply to a request from a client, containing a web page, site or error codes, i.e., error 404 (could not access website). Is transmitted from external proxy to internal proxy, and then on to the client.
- *Courier:* A physical data carrier that transports one or more packages from internal to external proxies, visiting one or more of each to propagate requests and responses.

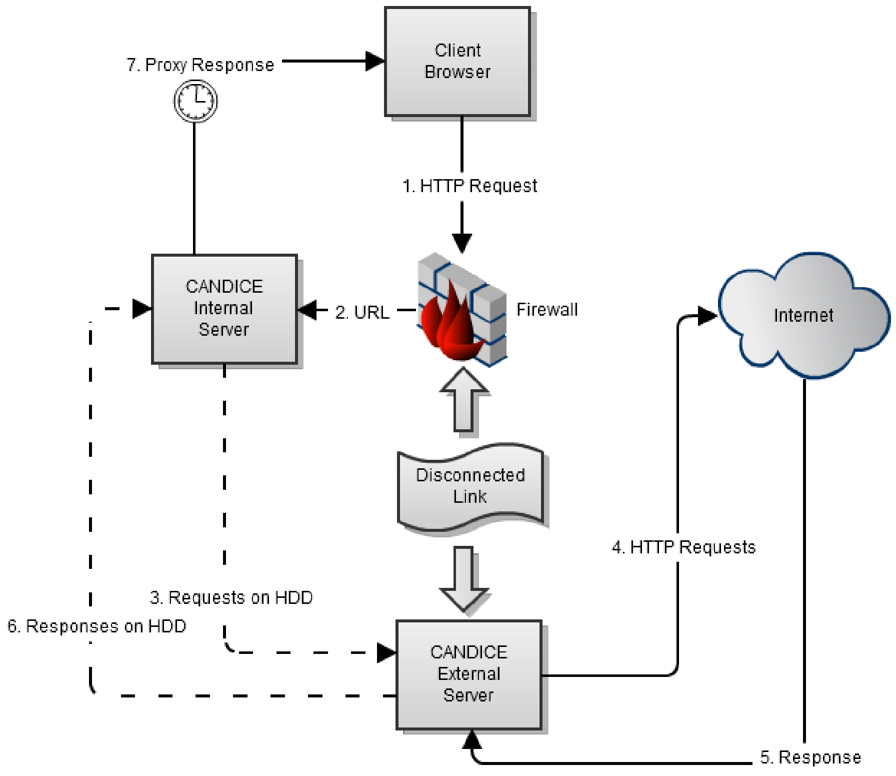
3.4 System description

The logical model is designed to provide a simple, effective and scalable solution for bridging disconnected networks, as presented in Figure 3. The basic data flow between components and entities is represented in order to show a basic path from client request to response.

The system begins with a *client* making a request to a website which is currently not available due to disconnected links. The user types a URL into their web browser (as an example, http://www.google.com) which is sent via wireless or their wired network to a computer, operating as a transparent gateway running the CANDICE internal proxy software.

The *internal proxy* takes web requests and checks for a currently stored copy of each site. If found, the cached site is returned along with a toolbar allowing the user to request an update (useful for news sites). Otherwise, a page is returned querying whether the user wishes the site to be cached in the future. If the user requests the site, a database flag is set and the page transitions to a waiting page that periodically queries the system for the cached site while the browser is still open.

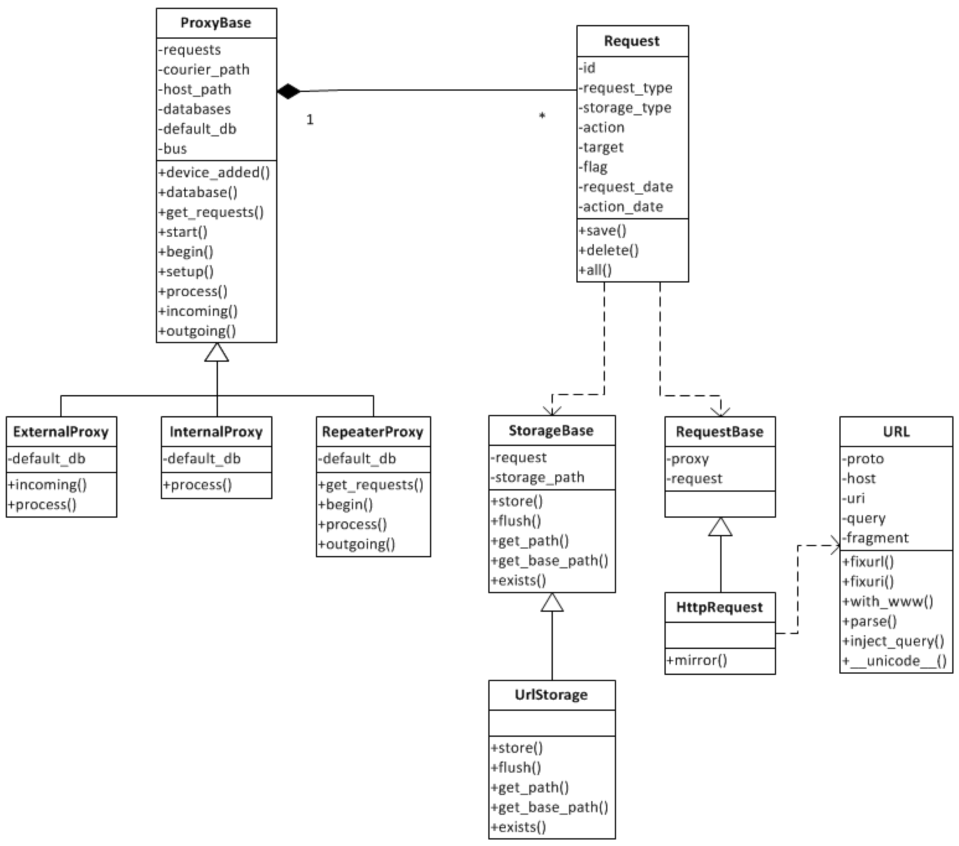
Figure 3 Proposed logical model (see online version for colours)



Once a site has been requested, a courier inserts an encrypted thumbdrive or hotswappable hard drive into the internal proxy. The proxy utility script automatically detects the presence of the device (and the existence of Courier.db on the drive) and copies any outstanding URL requests from the internal proxy to the courier's device. The courier then removes the device from the server and packs it for delivery.

The *courier* transports the device from the internal to the external proxy. This could be a short trip to the nearest available internet connection, or could be a dangerous trip across a border through hostile environments. If the courier is captured, lost or killed, the disk's encryption should prevent eavesdropping or manipulation – and if the courier does not arrive at the destination within a predetermined amount of time, the request statuses are reset.

Figure 4 CANDICE UML class diagram



Once the courier has reached the *external proxy*, the courier inserts the drive. The external proxy script detects the presence of a courier drive and extracts requested sites from the Courier.db file. After downloading the requested sites and/or data, the script appropriately tags the content and copies it onto the courier drive, updating the database as it goes. Once complete (or the drive is full), the courier removes the drive and begins the journey back to the internal proxy.

On the return trip, the courier inserts the drive into the *internal proxy*. The script copies cached content onto the local server, updating the database to reflect the newly cached content. The client's browser, detects new cached content and displays it for the user's perusal.

3.5 Design

The system implementation comprises several important procedures. These algorithms have been implemented in Python (van Rossum et al., 2001) and made to operate on a range of devices. A simplified UML class structure diagram is included as Figure 4. A simple procedure represents the internal proxy utility script, which aims to be a simple redirecting agent that passes requests made through the proxy to the CANDICE web system. While the action could be implemented using Squid (Wessels et al., 1998) and a redirector such as Squirm (Foote, 2005), a lightweight script is more appropriate for this situation. The packet redirection necessary to create a transparent proxy is implemented by simple firewall rules, though a number of HTTP headers included in request need to be rebuilt due to the interception.

Algorithm 1 illustrates the flow of the internal proxy script, which handles the transferral of user requests from internal proxy to courier. The script assumes any courier drive that comes in with data is destined for this proxy, and automatically refills the courier with requests for an external journey.

Algorithm 1 Internal proxy courier script

```
procedure DeviceHandler
    while Device ← DeviceList do
        if Exists(Device / Courier.db) then
            Requests ← SQL.Get(Courier, whereflag = retrieved)
            while Request ← Requests do
                Dir ← Request.ID
                Copy (Dir, CacheDir)
                SQL.Update (Candice, flag = cachedwhereRequest)
                Delete (Dir)
            end while
            RequestedEntries ← SQLCopy(Candice.db, Courier:db, whereflag = requested)
            SQL.Update(Candice, flag = transitwhereRequestedEntries)
        end if
    end while
end procedure
```

The web system detailed in Algorithm 2 comprises a majority of the implementation, handling all client interaction and content hosting. Implemented in Python and Django (Foundation, 2011), the web system is hosted to the user by a number of light-weight Twisted (Labs, 2011) HTTP servers – one for the request system and one to host cached content to the user. All client requests come through the system, and the client pages waiting for cache periodically query the site for new content.

Algorithm 2 Internal proxy web system (simplified)

```
procedure RequestHandler(Request)
    Entry → SQL.Get (whereURL = Request.URL)
    if Entry then
        CreateToolbar ('toolbar' + Entry.Flag)
    else
        CreateToolbar ('toolbarrequest')
    end if
end procedure
procedure TakeRequest(Request)                      ▷ Flag URL to be retrieved
    if Entry ← SQL.Get (whereRequest) then
```

```
        SQL.Update($flag$ = $requestedwhereRequest$)
      **else**
        SQL.Insert($Request$)
      **end if**
      $Toolbar.State \leftarrow Waiting$ Print(Toolbar)
  **end procedure**
```

4 Evaluation

Evaluating the success of the system is an interesting proposal; while couriers may be lost during the operating of the system, the system itself may perform admirably. It therefore becomes important to note the distinction between the system and the human components within it, but a complete failure of all human components comprising the system also reflects badly on the system as a whole. Lack of accessibility, unobtainable hardware requirements or extremely hostile environments can all contribute to system failure – though the former two have been mitigated as part of the project requirements. An evaluation of the system involves the following key points:

- satisfactory compliance with all ‘shall’ requirements listed in Section 3.2
- completion of basic test runs demonstrating the traffic of data between disconnected networks
- a practical example using a hybrid network of virtual machines and hardware, simulating a typical environment.

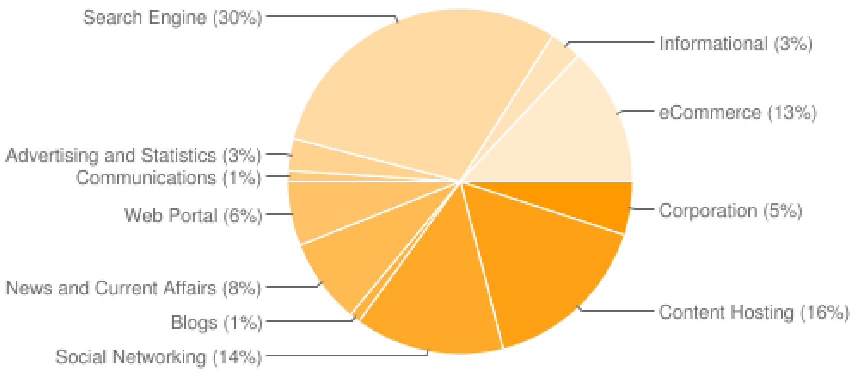
Satisfying these three points should demonstrate acceptance of the system’s fundamental viability as a solution to problems such as that described in Section 2. In addition, there are a number of quantitative and qualitative analysis methods that may be used to determine viability and any problem areas.

4.1 Experiment setup

The experiments in this section were executed inside a VirtualBox instance running on an Intel Core i5 2500K with 16 gb of DDR3-1600 RAM. The virtual machine itself was allocated the usage of 1 processor core, 1 gb of RAM, 5 gb of virtual storage for the operating system and 100 gb (initially 50 gb) of virtual storage for the mirroring directory. The virtual drives were initialised on a Western Digital Caviar Black 2tb hard drive, while VirtualBox itself was installed on an OCZ Vertex 2 solid state drive. The internet connection supplied was a 12 mbit ADSL2+ link with one recorded interruption during the run – tests had to be paused and restarted to expand the size of the disk cache being used (to 100 gb). The list of the 100 most visited sites was provided by Alexa (2011/10/2).

For the purposes of these experiments, wget was executed with a number of flags enabled, most notably –m (mirror), –l 3 (maximum depth search 3), –H disabled (so that only content for the specified host is mirrored) and a restricted list of file-types to mirror – HTML, JavaScript, Cascading Style Sheets and images. File types are restricted due to bandwidth and time constraints, as several sites in the Alexa Top 100 are video streaming sites containing multiple terabytes of content each. Figure 5 shows that by partitioning content into coherent categories, interesting comparisons can be made in tests to analyse where problems with content mirroring lie.

Figure 5 Top 100 sites in categories (see online version for colours)



4.2 *Functionality*

Due to recent increases in usage of asynchronous web technology, robot-based mirroring can be difficult on some content – particularly in the case of social networking, or heavily user-interactive sites. Because site usefulness can vary (depending on the site and which part is successfully mirrored), a number of functionality categories must be defined:

- useful: mirror provides useful, relevant, timely information or a service
- functional: mirror provides useful information, but some functionality is compromised
- partial: mirror provides incidental useful information, but most/primary functionality is compromised
- reconfig: mirror would need reconfiguration past experiment standards to allow any level of functionality
- non-functional: mirror retrieves a non-working service
- sabotaged: mirroring is actively blocked.

By examining logs and retrieved content, we are able to qualitatively determine a rating for each site listed in the Alexa Top 100. For purposes of practicality, sites that provide useful information but present it in an inconvenient fashion (such as loss of site design due to URL rewriting) are still noted as such, as the information is intact – just awkward to read. Sites that suffer from improper URL rewriting during the mirroring phase are either listed as reconfig (if reconfiguring the mirroring program would result in a fix) or non-functional (if site functionality is still compromised).

As seen in Table 1, the most popular sites on the internet tend to be heavily user-interactive. Search engines, social networking sites and blogs tend to be among the most popular and least functional, as they often require two-way communication between client and server. By contrast, news and product advertising sites mirror well – exceptions being news sites requiring paid logins. Informational sites (such as Wikipedia) would be quite useful sites to be locally cached, but are relatively unpopular and mirror poorly without specific configurations being applied to the mirror tools.

Table 1 Site categories vs. functionality

Useful		*Functional*	*Partial*	*Reconfig*	*Non-functional*	*Sabotaged*
Search engine	0	0	19	5	6	0
Social networking	0	0	4	4	6	0
Blogs	0	0	0	0	1	0
Content hosting	0	0	4	4	7	1
Informational	0	0	0	3	0	0
E-commerce	1	1	2	5	4	0
Corporation	1	1	1	0	1	1
News and current affairs	2	0	1	3	2	0
Communications	0	0	0	1	0	0
Web portal	0	1	1	3	0	1
Advertising and statistics	3	0	0	0	0	0

4.3 Retrieval responsiveness

Part of the difficulty in mirroring web content is the time required to retrieve content. Particularly in the case of time-sensitive information and intelligence, content retrieved by a courier can potentially expire prior to being delivered back to the client. In order to analyse the effectiveness of the system for handling time-sensitive data, the Alexa Top 100 sites were mirrored and the time taken to mirror each site was recorded.

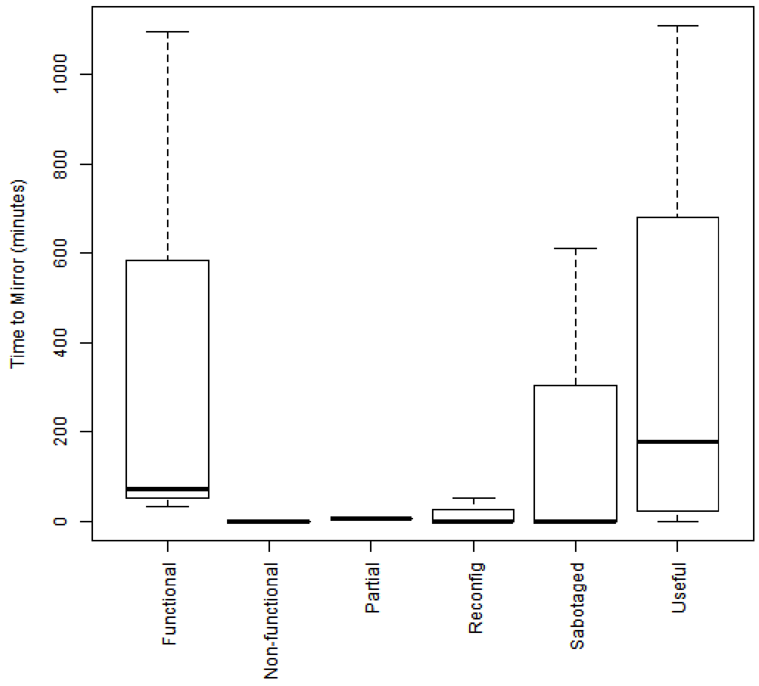
The experiment results were interesting, as shown in Figure 6 – generally the more functional the site mirror was, the longer the mirror took. Sites that were deliberately protected against mirroring wasted a significant amount of time.

4.4 Compressibility

Websites are made up of a number of content components: HTML source, Cascading StyleSheets, Javascript, images and a number of other media types. In order to more effectively compress these caches for transport by courier (whose carrying capacity is limited), the resultant repositories of content retrieved were archived using gzip with standard-level compression (a trade-off between speed and size) to determine how well the content can be archived.

While movies and other less compressible binary types have been excluded from the mirroring tests, compressibility amongst all sites manages to be quite effective, as seen in Figure 7. Those sites with very low compression ratios tend to have extremely small original sizes, and therefore the archive dictionaries tend to take up more space than the content itself. Functionality of the sites tends to influence compression, but only in certain categories and is generally indicative of content.

Figure 6 Categorised time to mirror top 100 sites



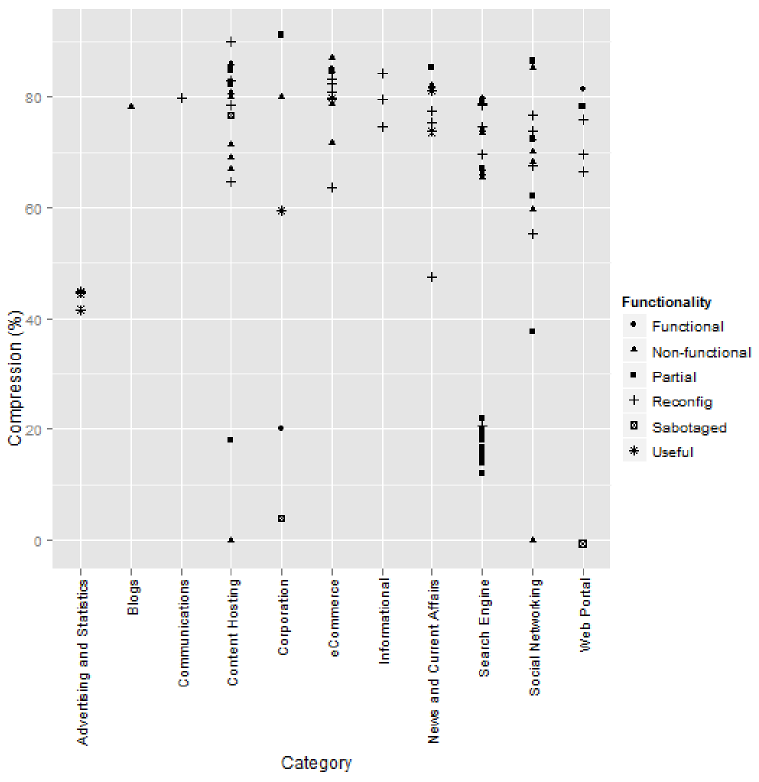
4.5 Analysis

There are numerous links that can be drawn between data collected to derive further meaning. A good example is in compressibility; sites that are non-functional are generally of very small size (<32 KB) and will rarely compress well. Figure 7 shows that a number of sites that do not compress well are also non-functional or only partially operational.

Inversely, even sites with large archive sizes (such as LinkedIn, with a mirror size of approximately 17 GB and a compression ratio of 86.5%) manage high compression ratios due to the nature of its content. These figures would look substantially different for sites with some types of content included – particularly Adult Video Sharing sites which tend to mirror effectively but the content does not compress well.

Many sites mirrored in the experiment require reconfiguration to enable a range of useful content to be retrieved. Some merely require that the crawler be allowed to span hosts (that is, to retrieve from other domains). This can be important with extremely popular websites due to their use of Content Delivery Networks. For example, while http://www.facebook.com serves static html and scripts, images from Facebook are served from http://fbcdn.com, which is a load-balanced and regional content delivery network. While this allows for faster retrievals of data for normal users, this effectively prevents a mirror script from mirroring photos unless reconfigured. During the mirroring process, it was discovered that most sites suffer from link rewriting problems. When a site is mirrored locally, references to other pages, images and content are rewritten to point to the local site – but this has not happened in some instances, leading to missing page styles and/or unavailable content. While most content remains useful without styling, further investigation is needed into the cause.

Figure 7 Compression by category and functionality



5 Discussion

Information derived from these results is indicative of a fundamental problem with Web 2.0 communications – it relies entirely on interaction with specific users. The primary function of most social networks, search engines and communications tools is compromised due to the nature of transmission. Without site-specific API handlers being developed to manage both sides of the disconnected network, there is little viability in initiating and maintaining communications developed routinely as part of Web 2.0. Search engines are generally useless without the ability to POST back to the engine, while the CANDICE framework would easily allow such development, the delay involved in user interaction makes a search-specific solution such as RuralCafe (Chen et al., 2009) more suitable.

By contrast, some sites work extremely well over a disconnected link. News, corporations, products and some e-commerce sites tend to provide useful information about the subject at hand. Some e-commerce sites also provide standard phone-based ordering, allowing the user to browse the mirrored content for the desired product and order by normal phone (if connections exist) or e-mail (if a CANDICE plug-in allows). News websites, while highly time-sensitive, also tend to provide a reasonable amount of useful content.

A core advantage of the internet is its ability to adapt and evolve. Many types of content are continually being created and collated in ways never before seen. In order to establish connections to such a fluid structure of services, connection methods need to be handled almost on a per-case basis. Connecting to Facebook over a disconnected network is almost entirely dissimilar to connecting to Twitter, for example. While these sites provide accessible Application Programming Interfaces that can be used to interact with the site using scripts, there is no formal standard for robot-initiated connections to web systems. By providing an extensible architecture to handle disconnected network connections, CANDICE provides the ability for programmers to develop plug-ins to the system to manage many different types of traffic – including standard e-mail, FTP connections and a large number of others.

However, the system is not a silver bullet. Without modifications to framework code and user interaction, connections are considered shared and cached, excluding user-specific sessions from taking place (such as individual Facebook log-ins). While the system can request HTTPS content, it cannot manage HTTPS across a disconnected link – the request would have to be serialised to disk and recreated externally.

6 Conclusions

This paper presents an extendable framework that aims to ease communications between disconnected networks in an accessible fashion. The framework is used to evaluate the feasibility in providing effective connectivity to popular sites, and problems in dealing with forced user interactivity over disconnected links are discussed. Evaluation showed that the majority of popular content accessed on the internet requires significant real-time user interaction and is unsuitable for delay tolerant networking. However, specific types of content such as news and blogs can be mirrored with relative ease and transferred for distribution over disconnected networks. The study has shown that it is feasible to maintain internet communications in unstable civil situations. It is hoped that by exposing the difficulty in accessing some styles of websites over disconnected networks, it might encourage the re-design of those sites.

References

BBC (2011) *Libya Removes Itself from the Net* [online] http://www.bbc.co.uk/news/technology-12653078 (accessed January 2016).

Burleigh, S., Hooke, A., Torgerson, L., Fall, K., Cerf, V., Durst, B., Scott, K. and Weiss, H. (2003) 'Delay-tolerant networking: an approach to interplanetary internet', *IEEE Communications Magazine*, Vol. 41, No. 6, pp.128–136.

Chen, J., Subramanian, L. and Li, J. (2009) 'RuralCafe: web search in the rural developing world', *Proceedings of the 18th International Conference on World Wide Web (WWW '09)*, ACM, New York, NY, USA, pp.411–420.

Demmer, M., Du, B. and Brewer, E. (2004) 'TierStore: a distributed storage system for developing regions', *6th USENIX Conference on File and Storage Technologies*, 26–29 February 2008, San Jose, California, pp.35–48.

Du, B. and Brewer, E.A. (2008) ‘Dtwiki: a disconnection and intermittency tolerant wiki’, *Proceeding of the 17th international conference on World Wide Web’, WWW ‘08*, pp.945–952, ACM, Beijing, China.

Fall, K., Scott, K.L., Burleigh, S.C., Torgerson, L., Hooke, A.J., Weiss, H.S., Durst, R.C. and Cerf, V. (2007) *Delay-Tolerant Networking Architecture* [online] http://tools.ietf.org/html/rfc4838 (accessed January 2016).

Foote, C. (2005) *Squirm* [online] http://squirm.foote.com.au (accessed January 2016).

Foundation, D.S. (2011) *Django Web Framework* [online] https://www.djangoproject.com/ (accessed January 2016).

Garrett, J.J. (2005) *AJAX: A New Approach to Web Applications* [online] http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications (accessed January 2016).

Gray, J., Chong, W., Barclay, T., Szalay, A. and Vandenberg, J. (2002) *Terascale Sneakernet: Using Inexpensive Disks for Backup, Archiving, and Data Exchange*, May 2002, Technical Report MS-TR-02-54, Microsoft Research, Redmond, WA 98052.

Hamblen, M. (2008) *Cut Cables Force Worldwide Internet Traffic Rerouting*, January, PCWorld online] http://www.pcworld.com/article/142068/article.html (accessed January 2016).

Labs, T.M. (2011) *Twisted* [online] http://twistedmatrix.com/trac/wiki (accessed January 2016).

Menoher, J. and Mraz, R. (2009) *Secure Cross Border Information Sharing Using One-Way Data Transfer Systems*, White Paper, Owl Computing Technologies, Inc, April 2009, CT 06877, USA.

Murugesan, S. (2007) ‘Understanding Web 2.0’, *IT Professional*, Vol. 9, No. 4, pp.34–41.

Okolloh, O. (2009) ‘Ushahidi, or ‘testimony’: Web 2.0 tools for crowdsourcing crisis information’, *Participatory Learning and Action*, Vol. 59, No. 1, pp.65–70.

Solomon, E. (2015) ‘ISIS to cut private internet access in parts of Syria’, *Financial Times*, 20 July.

van Rossum, G. et al. (2001) *Python Language Website*, World Wide Web [online] http://www.python.org (accessed January 2016).

Wessels, D., Nordstrom, H., Jeffries, A., Rousskov, A., Chemolli, F., Collins, R. and Serassio, G. (1998) *Squid Internet Object Cache* [online] http://www.squid-cache.org (accessed January 2016).