**Networking for Communications Challenged Communities:
Architecture, Test Beds and Innovative Alliances
Contract no: 223994**

N4C
DTN Node Design

Trinity College Dublin/Intel
Stephen Farrell
www.tcd.ie
stephen.farrell@cd.tcd.ie

ABSTRACT (Max 400 word)

This document describes the hardware and software design for Delay Tolerant Networking nodes being developed by Trinity College Dublin and Intel, to be used in summer 2009 and subsequent tests. Note that this document represents a snap-shot of an evolving design, and details of the design may change as a result of testing before the actual trials. The entire design will in any case be revisited as a result of lessons-learned for subsequent trials and in order to better integrate with and support the work of other partners in the N4C project.

Document history		
Status	**Date**	**Authors**
TCD/Intel initial drafts (-00..-03) Earlier versions were n4c-tcd-003-router-design, this version is renamed.	2008-09-11	Stephen Farrell, Alex McMahon, Kerry Hartnett
DRAFT deliverable 5.1	30/04/09	Author: Stephen Farrell Contributors: Alex McMahon, Eoin Meehan, Kerry Hartnett, Shane Brodie, Stefan Weber
FINAL deliverable 5.1	11/05/09	Stephen Farrell

Dissemination level	
	Level
PU = Public	PU
PP = Restricted to other programme participants (including the Commission Services).	
RE = Restricted to a group specified by the consortium (including the Commission Services).	
CO = Confidential, only for members of the consortium (including the Commission Services).	

CONTENTS

1 INTRODUCTION

This document describes the hardware and software design for the DTN [DTN] nodes to be used in N4C testbeds. The goal of N4C [N4C] is the development of a lasting testbed for Delay- and Disruption-Tolerant Networking. The testbed will be validated via a set of pilots over the duration of the project. ***Since this is a design document that pre-dates initial deployment some aspects are subject to change. Those where change is expected are highlighted in bold, italics, like this.***

There are three types of DTN node described in this document:

- A DTN router is a battery/solar powered DTN node assembled from common-off-the-shelf (COTS) products that provides DTN store-and-forwarding and application services as described below. DTN routers are designed to be used in the summer villages as the core of a local network.
- A DTN mule is a COTS netbook, to be deployed in helicopters servicing the summer villages. DTN mules do not currently offer application services, only supplying DTN routing functionality.
- A DTN gateway is a standard personal computer (PC) that that is well-connected and mains-powered and acts, as the name suggests, as a gateway between challenged areas where DTN routers and mules are deployed and the Internet. Some DTN gateway functions are distributed, whilst others must be deployed adjacent to the challenged areas.

Note that these three types of DTN node are not the only kinds that may be part of the N4C project, for example, the Hiker's PDA [PDA] is a combined DTN node/ mule that also offers application services.

Similarly, it is important to note that the detailed designs here are not meant to be “exclusive” – there are other designs that continue to be developed in the N4C project with overlapping functionality – the intent is to test these various designs so as to eventually achieve a combination of DTN and application functionality so that a lasting testbed can be achieved. For example, DTN nodes developed in the earlier SNC project [SNC] have been used as the basis for the summer 2008 testing and will continue to be used in the project since they are an already-tested platform. The expectation is that the more successful aspects of the various node designs will be merged as the project continues so as to produce a final, proven, DTN node design as a major output of the project. Essentially, this represents a spiral development model, which nicely suits the overall project structure of iterated summer and winter trials, at each stage making use of what's available then and improving on that incrementally.

Initial deployment of these nodes is planned for August 2009 in Sweden. At the time of writing (May 2009) TCD and Intel are currently in the process of completing hardware and software integration in the lab in Dublin. The N4C architecture specification [ARCH] describes how these nodes fit into the overall N4C project. Section 2 describes the hardware for these nodes, section 3 the software and section 4 the project plan.

2 HARDWARE

In this section we mainly describe the DTN router hardware since the other systems are complete COTS systems. However, for completeness table 2.1 below describes the hardware platforms for each of the DTN nodes.

Node Type	Hardware platform
DTN router	Intel Atom based single board computer + power + networking
DTN mule	Asus EEE PC 901 netbook (or a fit-PC Slim if an SSD version is available in time http://www.fit-pc.com/new/about-fit-pc.html)
DTN gateway	Standard desktop PC

Table 2.1 – DTN node types

2.1 DTN Router Hardware Requirements

Before describing the selected hardware we first present the set of requirements that we used when analysing the various COTS subsystems that are currently available.

1. MUST be capable of acting as a DTN router (running DTN2's dtnd etc.)
2. MUST be capable of acting as a WiFi access point for the summer village
3. MUST be capable of offering application layer services to people in the summer village (web, email etc.)
4. MUST have sufficient and extensible storage as required by DTN and applications. (We did not model storage requirements but rather focus at this stage on extensibility.)
5. MUST be able to operate without mains power for extended durations (up to months)
6. MUST be able to operate without effective battery re-charging for at least two days, SHOULD be able to survive one week without re-charging, though possibly only with reduced load
7. SHOULD have sufficient peripherals so as to allow the attachment of various other hardware devices that may prove useful in future (e.g. screens, sensors)
8. MUST be robust enough to be “luggable” and to survive in the challenging environment (though we do not require winter capabilities at this stage)
9. SHOULD be over-specified in terms of power and robustness.

In particular that last requirement (and in fact most of the others) is motivated by our earlier experience with the SeNDT [SENDT] DTN sensor network nodes.

2.2 DTN Router Hardware

Thanks to Intel being a partner in N4C, the project is in a position to leverage their expertise and resources and so decided early on to base the DTN router on the Intel Atom processor. [ATOM] That processor offers significant functionality with reduced power consumption and with an increasing range of platforms

available. The main hardware components for the DTN router are summarised in Table 2.2 below. Figure 2.1 is a photograph of these components on the lab bench.

Function	Selected Product
Processor Board	PROTEUS from Eurotech http://www.eurotech-ltd.co.uk/en/products.aspx?pg=PROTEUS&pid=10120
Network	Microtik RB411 Router board http://routerboard.com/popup.php?kods=92 Engenius EMP-8603 wireless card http://www.irishwireless.eu/shop/item.aspx?itemid=268
Solar Panel	Sunshine Solar FastFIX 20w12v http://www.sunshinesolar.co.uk/khxc/gbu0-prodshow/SS20WP.html
Battery	Camden 5085329 7AH 12V gel http://ie.farnell.com/camden-electronics/beg120075/battery-europa-gel-12v-7-5ah/dp/5085329
Enclosure	Eurobox IP66 cabinet http://www.euroboxenclosures.co.uk/mild-steel-cabinets.php

Table 2.2 – Main Hardware Components

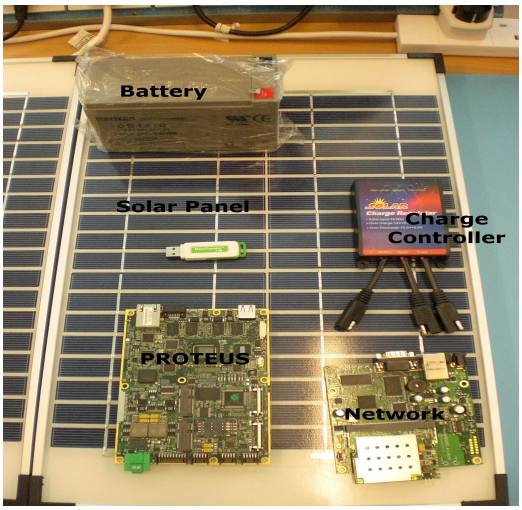


Figure 2.1 – Main Hardware Components
(the USB stick is for scale).

2.2.1 Processor Board

After examining a range of COTS products we selected the PROTEUS board from Eurotech as the basis for the DTN node. The main characteristics of the PROTEUS are shown in Table 2.3 below.

Features	
Processing unit	Intel Atom processor options: - Z530 Processor 1.6GHz (2.3W) - Z510 Processor 1.1GHz (2W)
Chipset	System Controller Hub US15W (2.3W)
Supported OS	Windows XP, XP Embedded, CE 6.0 and Linux
Memory	DDR2 SDRAM - up to 1GB (400/533MHz)
Flash (optional)	Up to 4GB parallel ATA Flash on board
Graphics	- Ultra low power integrated 3D Graphics - 2 x single-channel LVDS 24-bit interface to Board to Board connector - 1 x backlight connector
Audio	HD Audio CODEC and 2W audio amplifier supporting mic in, line in, speakers out and headphones
Touchscreen	4, 5 and 8 wire resistive touchscreen interface
USB Support	- USB 2.0 supporting low/full/high speed modes - 2 x user accessible ports (Pin header, one client configurable) - 2 x ports connected to PCI Express Mini Card sockets - 1 x port to board edge Type A connector
Ethernet	- Gigabit Ethernet port supporting 10/100/1000 BaseT - Output to board edge RJ-45 connector
Serial ATA	- Supports 1 x serial ATA 300 - 1 x SATA connector - 1 x power connector for 2.5" SATA drive
Serial Port	1 x serial port used for general purpose RS232 TX/RX port
PCI Express	x1 PCI Express to PCIe Mini Card socket x1 PCI Express switch to: x1 PCIe Gigabit Ethernet controller x1 PCIe serial ATA controller x1 PCIe Mini Card socket
SDIO/MMC	1 x MicroSD socket
TPM	- Atmel Trusted Platform Module Device, TCG v1.2 compatible - SMBUS interface
Modules (optional)	20-channel GPS receiver SirFStar III chipset module Bluetooth or ZigBee wireless comms provided by onboard modules
Test Support	JTAG Interface (Intel XDP)
Power Requirements	+8.5 to +25V DC (+12V nominal)
Physical/other	COM Express Extended module mechanical footprint Operating temperature: 0°C to +70°C 155mm x 110mm

Table 2.3 – PROTEUS Board Features

From http://www.eurotech-ltd.co.uk/en/products.aspx?pg=PROTEUS&pid=10120

Some points to note about the PROTEUS board:

- While this version of the board is not rated for winter temperatures, the manufacturer has is a low-temperature version in preparation.
- We expect there to be a wider range of Atom based single-board computers available for 2010 tests – at the time we had to select hardware for the summer 2009 tests, there were few manufacturers that could ship a board meeting our requirements in a timely fashion. So we may change to (a cheaper) board for future tests, depending on results and on the availability of alternatives.

The PROTEUS has 2GB of on-board flash memory that will be used to store the Operating System and applications. Additional storage will initially make use of an SD card with at least an additional 4GB of cache storage. (Initial testing of higher density SDHC cards has shown that up to 16GB cards can work.) In the event that the SD storage is insufficient, or unreliable (some SD card file systems have been seen to be problematic with multiple suspend/resume cycles), then the PROTEUS board can support either an SSD or SATA disk, but we hope to avoid that form of storage as it consumes more power than SD.

2.2.2 Networking

Our initial plans were to use an external Power-over-Ethernet (PoE) WiFi access point (AP) to provide networking for the DTN router. The advantage would have been that only the access point would need to be external which would have reduced the costs associated with rugged, external enclosures thanks the the ability of PoE to support reasonable distances between the DTN router and AP. However, testing showed that the external devices available consumed too much power (about 8-9W) and so we changed plan to include an internal AP and wireless card that use less power (1-4W) and are housed in the same enclosure as the PROTEUS with an external antenna. We plan to experiment with both low and high-gain antenna options, where the high-gain antenna should require less power for the same transmission range.

2.2.3 Power

We selected lead-acid gel batteries for reasons of safety, cost and capacity. The selected units each have a 7AH capacity, and the enclosure selected allows for up to three batteries to be included giving a total of 21AH power available. “Full-on” power consumption for the PROTEUS+networking is 2.35A, meaning that the unit will fully discharge the batteries in less than 9 hours of “full-on” operation. Power consumption in various “sleep” modes is currently being tested but is estimated to be around 0.45A which maps to slightly more than two days battery. So the power budget clearly requires both re-charging the batteries and operating according to a sleep/wake duty-cycle. The management of the duty-cycle is currently being investigated.

The solar panels selected can produce 20W at 12V and can be added in series (up to 4 per enclosure). We plan to initially test with two solar panels per unit in order to meet (or exceed) the requirements stated above. In addition to the solar

panels, there is also a need for a solar charger that connects the batteries, solar panels and PROTEUS board, routing current as appropriate.

For Dublin-based external testing, (but not for field deployment), we will be using a Studer high precision battery monitor (model SBM-02, http://www.studer-innotec.com) in order to measure the power consumption and battery charging while units are placed on the roof of a building on the TCD campus. This will allows us to test the management of the unit duty-cycle so as to increase confidence that deployed units will meet our power requirements.

2.2.4 Enclosure

The enclosure selected is an IP66 steel Eurobox enclosure (shown on the right in Figure 2.2 below) which has a window giving us the option to attach a screen, LEDs etc. to the PROTEUS board to offer status information without opening the box or requiring additional holes to be drilled.

Figure 2.2 – Eurobox enclosures

3 SOFTWARE

In this section we describe the software components to be deployed on the DTN nodes for the summer 2009 trials. Later trials will involve additional components and/or changes to the versions and functionality described here. Once the initial, summer 2009, trials are complete, then we will integrate software from other N4C participants as appropriate into the same build environment. This is consistent with the overall project approach of spiral development based on semi-annual field trials.

3.1 Off-the-shelf Software Components

We have prepared a standard software build for DTN nodes using the following off the shelf software components.

3.1.1 Operating System and Basic Networking

Each of the DTN nodes will run the Ubuntu mid-8.04 (Hardy) distribution. This distribution uses a kernel (2.6.24-lpia) that is tailored for running on lower power devices such as the PROTEUS and netbooks.

DTN router nodes in villages will act as a hotspot, providing basic connectivity for user's laptops via DHCP. DNS services are provided using BIND 9.

3.1.2 DTN Software Components

The following DTN software components are the starting points for N4C:

- DTN2 reference implementation: provides the bundle protocol (BP) and various convergence layers and routing schemes, including an early implementation of the PRoPHET routing protocol. [DTN2]
- LTPlib - TCD's LTP (and LTP-T) implementation – not currently planned for the summer 2009 trials, but included for later. [LTPLIB]

3.1.3 Infrastructure Applications

The following are the main infrastructure applications included in the standard DTN node build. (Note these are part of all builds but will only be used in the DTN router and gateway for summer 2009, and not in the data mules.) All of these packages are widely used on the Internet.

- E-mail – Postfix and EXIM message transfer agents (MTA) and Dovecot message store (MS).
- Web browsing – Apache and Squid for local and off-line web browsing.
- Web crawling – wget and/or puf – running on the DTN gateway
- Domain Name System (DNS) - BIND 9 – for DNS on DTN routers

3.2 E-Mail Operation

The basic idea for mail is that users create a new mail account for use during (and around) the N4C trials. If a user wishes to read their “home” mail, then they are responsible for setting up forwarding of their mail to this new account.

We will provide a “bastion MTA” which will handle both inbound and outbound mail from these mail accounts. That MTA is well-connected to the Internet and can do all the usual mail processing, e.g. DKIM signing, mail scanning for AV and spam, publication of DNS MX, and spf resource records for the mail domain and provision of a message store accessible via LDAP.

When a user is well-connected, they can simply use this as an additional mail account. Once a user enters a challenged region, then we need to use the DTN in order to send and receive mail, e.g. in the summer village. This requires us to handle user set up, SMTP for outbound mail and message store access for inbound mail, each of which is described below.

The details of the mail domain to be served and the policies to be followed will evolve over time and in consultation with other N4C partners, but for the purposes of planning and testing the DTN nodes we describe specific parameters and policies for handling email. The domain names and hosts to be used may change before the system is deployed.

3.2.1 Initial Mail Account Set Up

The mail domain will be: **village.n4c.eu**. The local part of mail addresses will be assigned on a **first come, first served** basis. That is: users choose the local part of their mail address within the village.n4c.eu domain. We use a standard web sign up, but with **manual moderation** since we will only be creating a small number of accounts for initial trials and do not need to e.g. use CAPTCHAs so as to avoid spambots – users fill in the form and once they have selected a unique local part, that requires manual moderator approval before the account is set up.

IMAP accounts will use numbered user identities, e.g. **user001** etc. When a user chooses a unique mail-address local part, that will be bound to a specific IMAP user account. This allows us to pre-provision all of the IMAP accounts that we will use in the DTN routers in advance so that we avoid a requirement to create new IMAP accounts via the DTN – we only need to synchronise mail, which is simpler. IMAP accounts will each have a pre-set **password** that the user will receive via email after the mail account has been successfully set up.

At this point in time, it is not yet clear if we will support password change at the IMAP servers, but we do not plan to support synchronised password updates. That is, even if password change is supported, a user will need to change their password on each separate instance of an IMAP server with which they interact. For example, if the summer 2009 trial were to involve more than one village (which is not planned) and a user were to move from one village to another, having changed their IMAP account password in one village; in the new village, they would be back to their initial password.

The IMAP server's DNS name will be **imap.village.n4c.eu** and this MS will be usable via **IMAP/TLS** from anywhere on the Internet for users with a valid mail account. We will also provide an SMTP server, called **smtp.village.n4c.eu** that will accept mail submission from any **authenticated user**, that is, users have to authenticate to this MTA in order to send mail.

Users are free to use any mail address in the “From:” message header field for mail they originate, but will only receive mail via the mail address that is bound to the IMAP account as above. So a user that has forwarded (some or all of) her “home” mail to the village.n4c.eu account can reply to that mail using her “home” mail address.

All of the initial account set up has to be completed before the user visits a challenged region, but this can be done well before the user leaves home, e.g. from another country for N4C project partners.

3.2.2 Village Account Set Up

When the user first visits a challenged region they can simply bind to the message store instance at the DTN router. The DTN router will “pretend” to be imap.village.n4c.eu so that the connection from the user's MUA will simply work. However, the content of the MS will be dependent on message files having been synchronised via the DTN. The same will be true of the mail submission server, **smtp.village.n4c.eu**.

The net result is that a user that has set up their mail account before visiting the village will simply be able to use their mail user agent (MUA) immediately and will have mail service, albeit that they are some DTN hops away from the Internet.

3.2.3 Outbound Mail from the Village

Outbound mail from the villag will be sent from the MUA to the submission server at the local instance of smtp.village.n4c.eu which will be a Postfix or Exim instance. (Testing of both is onging.) We may perform some limited outbound mail filtering at this point, most likely simply to limit outbound mail to some reasonable size, but possibly also some anti-spam checks if such prove necessary.

Each mail submitted will be separately encapsulated in a bundle using the BP via a transport channel exposed by the Postfix or Exim MTA. A simple script can call the existing dtn-send application in order to encapsulate and send the bundle to the dtnd. There will be a single endpoint identifier (EID) representing the bastion MTA to which all such bundles will be routed. That endpoint will be located on the DTN gateway.

Once a bundle arrives at the bastion MTA endpoint, the existing dtn-recv application accepts the bundle from the dtnd, and the decapsulated mail message will be submitted to the bastion MTA, via SMTP using standard OS tools. Note that this does not require that the bastion MTA be hosted on the DTN gateway, but only that there is a reliable connection via SMTP between the DTN gateway and the bastion MTA.

Mail reliability is provided by requesting bundle receipts, so that for each mail sent to the bastion MTA a receipt bundle is returned. The dtn-recv application has been modified to provide the required bundle identifiers so that the sending system knows that the mail has arrived at the bastion MTA and need not be re-transmitted.

Once the message arrives at the bastion MTA, normal mail submission checks will be carried out before the mail is forwarded. Policies related to this will be developed as the trials proceed. For example, the bastion MTA is likely to only accept mail via an authenticated SMTP connection where the peer is either a registered mail account or the DTN gateway. Were this the only policy, then it could leave open some potential for spam to be sourced in the challenged region, but we don't expect deliberate spam sending to occur. However, if someone in the challenged region is using a zombie host for some botnet then we could see some spam being sent, in which case we may have to put some filtering of outbound mail in place.

3.2.4 Message Store Access from the Village

Message stores in this mail system will use the Maildir format, in which each message is stored in a separate file in a known place in the IMAP server file system. The Maildir format is the more suitable of the two standard formats for message store content synchronisation, which is the task at hand. (The other standard format – mbox format – would require more bytes to be sent over the

DTN or more software development due to the fact that mbox format stores an entire folder of messages in a single file, which is thus harder to synchronise.)

So our task here becomes one of file system synchronisation via the DTN. There is one aspect of the Maildir format that requires us to do slightly more than naïve file synchronisation – with the Maildir format, the file name reflects the status of the message in IMAP, so the file name changes when the mail is first read, or when the mail is replied-to etc. This means that we must ensure that we synchronise based on a canonical form of the Maildir file names rather than simply using the actual names in the file system. However, each file does have a unique part to its name, usually based on the time (in seconds since 1970-01-01 format) and a local unique number.

We will run a cron job that picks up file changes on each message store instance and send those to all relevant other instances. For a village DTN router instance, that involves sending to the EID of the bastion MTA's MS instance. For the bastion MS, that will in future involve sending to a non-singleton EID that represents all village MS instances. In the summer 2009 trials, since there will only be one village DTN router involved, that EID can represent a singleton endpoint.

Files that require synchronisation will be detected via their modification times (using the find command). We will have to encapsulate both the file name and the file content (if the content has changed) or just the file name if the content remains the same. Encapsulation will be as a compressed tarball in both cases, with a zero sized file if only the file name has changed. (That occurs when a message is read for example.)

Files will be sent via the dtn-send and dtn-recv applications as described for SMTP traffic in the previous section.

3.3 Web Access

The basic principle followed by the design of web access for DTN nodes is to make as much as possible available to as many users as possible with as little work for those users as possible.

The DTN router will offer a hotspot web site so that on first contact a web browser will be redirected to a local page with instructions on how to use the system generally, how to view local content and how to submit and manage requests for non-local content.

Generally request handling will make optional use of cookies in order to provide a better web user experience when dealing with content that has to traverse the DTN, and so as to respect users' privacy to the extent possible. See the section on cookies below for details.

In terms of the general web architecture, each DTN router will host an instance of the Apache web server with a Squid web cache. Web caches will be updated by encapsulating requests and responses via the DTN (see the section on encapsulation below), with the DTN gateway sending actual HTTP requests to the Internet and crawling for additional content (see the section on crawling below).

Some preconfigured content will be pushed from the DTN gateway to each of the DTN routers via a non-singleton EID so that users can always see a reasonably "fresh" version of the selected content.

Finally, in future trials we will augment the web infrastructure with better support for search and web application frameworks (e.g. AJAX). For now, we are aiming to support essentially static web content for the summer 2009 trial.

3.3.1 Cookies and Privacy

We first explain a privacy concern with the combination of web infrastructure and DTN. When a user makes a web request on the Internet they may have some expectation of privacy, at the very least they don't expect other users in the same local network to be able to easily determine which content they are browsing. While users' attitudes to privacy are very hard to model or predict, we expect that preserving at least this level of privacy is worthwhile.

In future we will examine extending privacy support so that users may have some expectation of privacy even in the face of DTN administrators, via the use of the BP security mechanisms defined by the DTNRG. (The implementation of those mechanisms in DTN2 is not yet sufficiently well tested to warrant a dependency upon them for the initial trials, but we expect that to improve as the project progresses.)

The specific privacy issue that arises in a DTN is where there are few users and where satisfying requests takes a noticeable amount of time, as will be the case in almost all N4C trials. The essential problem is that if I want to see a URL and get immediate access to the that content, then I can infer that one of the (few) other local users has previously accessed that same content. With a sufficiently small number of users, for some sensitive URLs, that can constitute an unacceptable privacy breach.

Earlier designs of our web infrastructure involved a requirement for users to set up web accounts in order to be able to issue HTTP requests so that only the requesting user could see the response. However, iterating the design it became clear that we can achieve almost the same affect, but with much less burden on the users via the use of cookies.

We basically separate all HTTP transactions into "public" and "private" transactions and handle each slightly differently. In this context a HTTP transaction covers all the flows from the browser sending a request, then some DTN encapsulation of the HTTP request and response, until the web content is finally viewed by the requesting user.

For "public" requests, we allow all related responses to be cached by any relevant DTN node and make that content available to any other user. If a user has a desire for a "private" HTTP transaction, then we associate a cookie for the domain of the request in the user's browser and set up access control so that only a browser with that cookie can see the eventual content (or a place-holder while the transaction is in-process). In this way, users that are interested in privacy can simply accept cookies (that last beyond a session) and need not set up any new account. We refer to these cookies as "**transaction" cookies**.

There are also some other uses of cookies in our design, mainly so as to better handle the difference between a user's first contact with the DTN router's web server and the case where a user has previously visited that server. In the former case, we wish to re-direct the user to a front page that offers information and instruction, in the latter case (e.g. where a user clicks on a bookmark) we wish, if possible, to offer the user the best version of that content possible. We call this the "**master**" **cookie** since it reflects whether or not the user has ever visited that DTN router's web server. (In fact, current implementation plans will result in the same value for both the master cookie and all transaction cookies and the DTN router will handle the mapping to transaction Ids.)

Note that in no case will any cookie set in the user's browser be sent over the DTN. This is to prevent cookie-stealing type attacks that would otherwise be trivial to mount for any DTN node on the path.

Users that do not accept cookies (or who delete cookies) will initially be limited to purely public browsing, but will still be able to see as much content as possible. In future we will provide a way for users who wish to do so, to set up accounts, so that they can turn off cookies (or recover from deletions) and still access their own "private" transactions.

There is an open issue as to how (or whether) to handle "previously private" content. For example, some HTTP transactions may only be sensitive for a relatively short duration and it may be safe to allow other users to see the content if it has been in the cache for a number of DTN round-trip durations (since they can infer less about its visibility in that case). Such more complex privacy handling will not be available for the initial trials.

3.3.2 Starting the Transaction

We describe one use-case here, where the user is requesting a page that has never been accessed, (and so is not in the cache), will be considered private, (and so has controlled access), and where the user has set up their browser to use the DTN router as its proxy and also accepts all required cookies. We can assume without loss of generality, that the user's master cookie for the DTN router's domain is set in the browser and that that cookie value is sent with all requests to the Apache and Squid instances on that router.

When the user's browser issues an HTTP request, for content that is not currently cached, then the proxy (Squid) will re-direct the request back to a PHP script running from the web server (Apache) on the DTN router. That script (provisionally called **dtn-http-trans-handler**, which is used throughout this use-case) offers the user a form asking whether the user wants to proceed with the transaction, and whether or not the user considers the transaction private. In this case, the user answers "Yes" to both questions, and that HTTP request (a GET request with a query-string reflecting the user's choices and the requested URL) is returned to the script.

At this point the script initiates the DTN transaction (see the section on encapsulation below), populates the cache for the requested URL with a place-

holder, sets the access control requirement for that page, and re-directs the user's browser to that place-holder in the cache.

The place-holder page will contain a link back to the dtn-http-trans-handler script to allow the user to check on the status of the transaction in case the user returns to the page before the actual content has arrived back via the DTN. In this way each user will see the status of their own transaction even if more than one user requests the same page in a private fashion at the same time (where "same" here is modulo the transaction time).

There may be some corner cases here that need further work, e.g. how to disambiguate transactions if two users ask for the same URL with different query strings at the same time.

3.3.3 Local Caching

In addition to place-holder pages, the Apache server will make available an index pages of cached content that that user can browse. Users that have made "private" requests can also see the list of the private requests that they have issued together with status information for each. Access control here depends on the master cookie having been set. This is implemented in the **dtn-http-status** script.

There is an open issue here related to non-browser clients, e.g. software update processes that make HTTP requests. Graceful handling of such requests may be problematic and is a matter for investigation. The behaviour of AJAX clients and other HTTP POST requests with post-data also needs to be checked.

3.3.4 Request Encapsulation

When the browser is eventually re-directed back to the proxy to access the place-holder content, the proxy adds a new HTTP header (**X-DTN-trans-id**) and forwards the request to what it (Squid) thinks is the next upstream proxy, but which is actually a new application (provisionally) called **dtn-http-req-encap** that encapsulates the HTTP request as a bundle that is sent to the DTN gateway's inbound HTTP request EID. This program responds to the proxy which then updates the transaction status.

At the DTN gateway the **dtn-http-req-decap** application listens at that EID and receives the request and passes it on to the HTTP satisfaction engine.

3.3.5 Satisfaction Engine and Crawling

The HTTP satisfaction engine (SE) runs on the DTN gateway and, for requests with no crawling required simply issues the HTTP request and passes the response received (in the case of success) to the response encapsulation engine described below.

In most cases, the SE will quickly receive a HTTP response with a 200 OK code and will then crawl a subset of that content in order to better meet the user's expectations of the overall HTTP transaction. We are currently evaluating using wget and/or puf for this crawling stage. Initially, very simple crawling strategies

will be used, but these are expected to evolve during testing and trials during the project.

The SE will maintain the transaction ID which will be placed into the HTTP response using the same HTTP header value.

There is an open issue as to whether to always associate the transaction ID with every crawled response or sometimes only with the "direct" response. Doing the former seems right, but may result in too much overhead if "private" requests involve a lot of "vanilla" crawled responses. For example, it may be sufficient to not associate simple "GIFs" for radio buttons with the request in some cases since those tend to be commonly used across an entire site.

The handling of HTTP errors and re-directs during crawling and the depth and breadth of crawling will initially be based on heuristics set at the DTN gateway. In later versions we will investigate parametrising this process so that the end-user (in the village) can exercise some control over the process. The crawling engines under consideration (wget and puf) both support many variations in each of the above respects.

In the first instance the DTN gateway will not act as a long-term cache but will simply issue requests and construct sets of responses into a directory tree for sending back to the requester. We will investigate the use of longer term caching at the DTN gateway (or elsewhere on the Internet) in future versions.

3.3.6 Response Encapsulation

After the crawling process is complete the DTN gateway will have a set of files to encapsulate into a bundle to send back to the challenged region(s). The **dtn-http-resp-encap** encapsulation program will create a single tarball (.tgz file) and make this the payload of a bundle. That bundle will be sent to the non-singleton EID of the village routers. (Only one initially.)

In later versions, we will investigate the use of a meta-data block describing the payload, for example, a meta-data block that lists the URLs of the content from the payload could be useful in routing the bundle.

3.3.7 Cache Update

Once the bundle arrives back at the village it will be received at the relevant EID and the **dtn-http-resp-decap** application will explode the tarball and associate the content with the relevant transaction and the relevant cache files and access permissions will be updated so that when the user next checks, she can (modulo possession of the appropriate cookies), see the content that completes her transaction.

3.3.8 Search and Feeds

In the first instance we will not be supporting any specific search in this web infrastructure. However, we will in future offer a search option that is essentially an additional level of indirection when compared to the above. The difference with search is that the SE must first get some results from a search engine (e.g. Google) and then select some search results for retrieving and crawling.

In similar fashion, we will investigate supporting the current DTNnews application from the DTNbone. The processing of news feeds being in many respects similar to the handling required for search requests and results.

3.3.9 Pre-Cooked Content

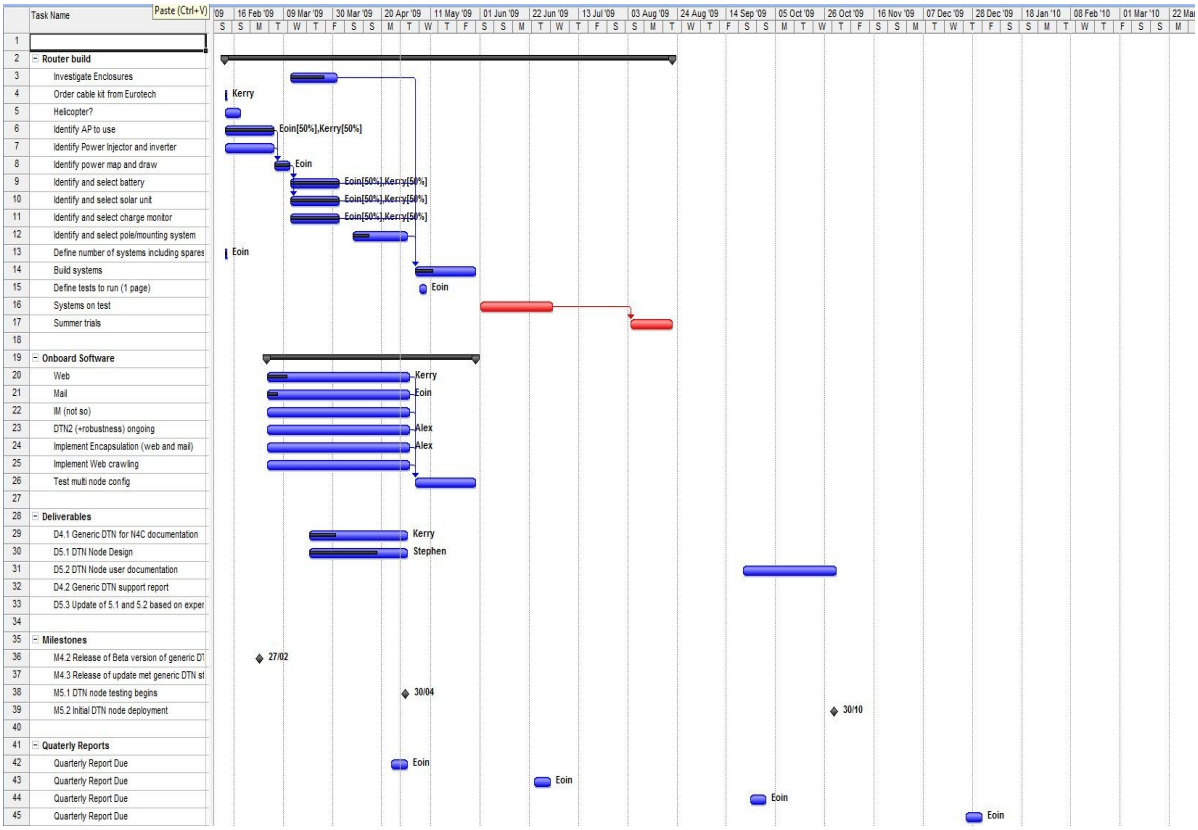
It is envisaged that some content will be sufficiently popular so that it is worthwhile periodically pushing that content out to the village(s). That will simply be retrieved via a cron job running on the DTN gateway that feeds configured URLs into the SE. Pre-cooked content will use a range of fixed transaction IDs that are specially treated at de-capsulation time.

4 PROJECT PLAN

The design documented here is currently (May 2009) in the process of being implemented in TCD (with TCD and Intel staff) for a first trial during the summer 2009 trial in Sweden. Our overall approach is to develop and test the DTN router and other hardware and software in the lab in TCD during April and early May; to further test the system outdoors in TCD during May and June, with July for debugging and additional software testing and finally bringing the equipment to Sweden for the summer trials in August. Figure 4.1 below shows the current Gantt chart for this part of the project.

Once the summer 2009 trails are completed, then we will evaluate the results seen and plan for the winter 2010 trials in Sweden. If resources and time permit, we may also bring the equipment to Slovenia in the autumn of 2009 so as to get additional input for summer 2010 and later trails.

Figure 4.1 – Project Plan at May 2009.



5 REFERENCES

[N4C] http://www.n4c.eu/

[DTN] Farrell, S., & Cahill, V., “Delay- and Disruption-Tolerant Networking,” ISBN 1-59693-063-2, Artech House, 2006.

[ARCH] N4C D2.1 “System Architecture,” n4c-wp2-004-sys-arch-04, September 2008.

[SENDT] Paul McDonald et al., "Sensor Networking with Delay Tolerance (SeNDT)," First International Workshop on Distributed Sensor Systems (DSS'07) in conjunction with IEEE ICCCN 2007, Turtle Bay Resort, Honolulu, Hawaii, USA, August 16, 2007.
http://down.dsg.cs.tcd.ie/sendt

[SNC] http://www.snc.sapmi.net/

[PDA] N4C WP3 Scenario foe Hiker's PDA – Use Cases.

[ATOM] http://www.intel.com/technology/atom/index.htm

[DTN2] http://www.dtnrg.org/

[LTPLIB] http://down.dsg.cs.tcd.ie/ltplib/