Report on an Arctic Summer DTN Trial

Stephen Farrell, Alex McMahon, Eoin Meehan, Stefan Weber and Kerry Hartnett

***Abstract*—Delay- and Disruption-Tolerant Networking (DTN) is an emerging area of networking research that will benefit from real-world trials and testbeds. We describe a week long DTN trial carried out during the summer of 2009 in the Laponia area of Northern Sweden that involved the provision of basic email and web services to users who were 57km distant from any power or networking infrastructure. The trial validated our design and successfully demonstrated the use of email via helicopter-transported data-mules. With the aim of making it easier for others to replicate this kind of trial, the hardware and software used are described with references to full specifications, as are the results of the inital 2009 trial. We also describe plans for an extended trial in summer 2010 in the same area.**

I. Introduction

Delay- and Disruption-Tolerant networking (DTN) [1] is an area of networking research that has grown over the last decade. [2], [3] The Internet Research Task Force's DTN Research Group[1] has developed an architecture [4] and a number of protocols for use in networks that cannot make use of standard "chatty" Internet protocols. These include the Licklider Transmission Protocol (LTP) [5] (not further considered here), and the Bundle Protocol (BP), specified in RFC 5050, and with a reference implementation called DTN2[2] which was the basis for the trials reported on here. [6]

One of the goals of the DTN research community is to establish how well the BP, and to a lesser extent, DTN2, perform in realistic trials. To that end, the "Networking for communications challenged communities" (N4C) project[3] has been funded to explore how the BP, and more generally, DTN, can be used to meet the needs of real users in real testbeds, with the overall goal of the project being to establish the technology for a lasting DTN testbed that can outlive the project funding. N4C started in 2008 and in this paper we report on the summer 2009 trials carried out in the arctic in conjunction with the first workshop on Extreme Networking[4] at which we presented a brief description [7] of the work reported on in detail here.

The N4C project's arctic trials, including this one, are designed for, and with the involvement of, the local Saami reindeer herding population. The ultimate goal of the project for these potential users is to use technology in order to better support their traditional lifestyle and culture, which involves spending some weeks in the mountains of Laponia in summer, when the reindeer are gathered for calf-marking. [8] For the Saami, there is a significant financial and social cost each time they must take a helicopter trip to town simply in order to make a phone call or send a text or email message. There are no commercial communications providers operating in this area, so the prospect that DTN might provide even very basic messaging or web services such as news and weather has a lot of appeal for these users.

So our trial has essentially two high-level goals: for us, as network researchers, we would like to validate the DTN architecture, protocols and their implementation and to make real-world data and experience available to other researchers; for the Saami, they would like to validate that DTN can provide basic networking services that they can use whilst in the mountains of Laponia.

It is important to note that the implementation described here has mainly been designed for the extended trials in summer 2010, and the 2009 trial was intended to, and did, demonstrate that the basic equipement design, software and networking could work in a short trial, prior to expending the resources required for the longer trial.

Since the goal of this paper is to enable experimenters to either replicate, or improve upon, our work, we provide references and links to detailed materials (e.g. a full "bill of materials" for the hardware used). All of those materials are linked from our web site - see section VII for details.

A. Outline of the paper

In the next section we briefly consider related efforts, then we describe the overall setup of the summer trial and the hardware and software used. We then describe the results of the trial and lessons-learnt. We finally draw some conclusions from our work in the summer 2009 trial. Throughout the paper, we will describe planned differences between the implementation for the 2009 and 2010 trials that result from the work done in 2009.

II. Related Work

In this section, we briefly describe other trials that resemble ours in terms of trialling DTN with real users in the real world, or that otherwise closely resemble our work.

The list of these real-world trials is limited, especially in contrast to studies based on simulations - which is also reflected in the available literature where novel architectures are often selected with preference to reports of real world evaluations. A survey carried out as part of the N4C project [9] considered a much broader range of DTN related developments and evaluations.

The Saami Network Connectivity (SNC) project [10] was a precursor to the N4C project, involving some of the same partners, that established the initial interest in DTN amongst

Kerry Hartnett is with Intel Labs. All other authors are with Trinity College Dublin, Ireland. Contact: stephen.farrell@cs.tcd.ie

[1]http://www.dtnrg.org/ All URLs accessed at 2010-09-23 unless otherwise noted.

[2]http://dtn.sourceforge.org/

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

[4]http://www.extremecom.org/

the user community described here. SNC demonstrated limited small-scale connectivity in the same region of northern Sweden beween 2006 and 2008, based on an earlier version of the BP, using a more "nomadic" approach to the type of equipment used in the experiments, and without aiming to support standard applications like email and web access as was done here. Some of the tests done in the SNC project were repeated as part of the initial work on N4C in 2008. The main difference in approach between SNC and N4C could be summarised by saying that providing any connectivity of any form was a goal for SNC, while N4C aims to provide a more capable network, that can interoperate with Internet applications.

The Motopost DTN trial [11] implemented a DTN style email service based on a modification of the SMTP protocol, rather than using the bundle protocol. The Motopost system involves kiosks and data-mules exchanging routing and other configuration information prior to the SMTP exchanges and has a particular naming/addressing scheme that is designed specificially for their trial, so it is unclear how easily it could be extended to support other applications. However, the data-mule based architecture used is quite similar to that presented here, and the authors report that end-users do consider privacy (or perhaps confidentiality) a real requirement.

In contrast, Hyyrylinen, et al. [12] describe a way to handle mail using the BP that has quite a different approach to ours in that they require all mail client devices that need to operate in a challenged region to be DTN enabled whereas we only require a standard mail user agent, and our village DTN router (described below) handles all the DTN aspects. While their approach offers additional flexibility, for example, the ability to use ad-hoc connections to route mail between two end-user nodes, that comes at the cost of requiring that the DTN stack be installed and configured on the end systems, which is at least a barrier to adoption and sometimes makes it impossible to use interesting classes of device such as WiFi enabled phones that can talk SMTP and IMAP but which do not allow installation of arbitrary network applications.

While the relatively short time available in the 2009 trial meant that we could not achieve many "live" HTTP request/response roundtrips, we are not, of course, the first to tackle the fairly obvious task of accessing web content via a DTN.

As part of their "Drive-Thru Internet" project, Ott and Kutscher [13] presented a classification of various methods by which HTTP could be supported in a DTN context. Using their terms, we are addressing the DTN gateway approach here and do not assume that any web server supports DTN or has provided a list of resource dependencies. As might be expected, their method for encapsulating HTTP in bundles is similar to our approach, though it is not clear whether their system could handle the extended latencies involved in our trials, where a user may emit a HTTP request but only expect to see results a day or two later, after most components are likely to have been rebooted for power management or other reasons.

Balasubramanian et al. [14] developed a DTN-like caching web service called Thedu, deployed on top of Dieselnet, that is similar in many respects to our approach. Thedu does not use the BP for encapsulating HTTP requests and responses, but does address web search, which we have not yet tackled and Thedu is more advanced when it comes to handling pre-fetching of web resources compared to our 2009 web access service, which encapsulates HTTP requests in bundles and has a simple server-based pre-fetching model.

Neither of the above web access schemes appear to specifically address user privacy concerns as described below, which may arguably be more important in the extremely high latency shared context in which we operate.

Another aspect in which our approach may differ from the related work, but which was not fully explored in 2009 relates to user mobility between villages. In our planned 2010 trial, we will have multiple village DTN routers and so will face some issues related to multicast-like operation. The design presented below is intended to handle these issues, for example, we intend to replicate message store content on all relevant DTN routers, so that regardless of which router is near a user, they can send and receive using the same account.

TierStore [15] is a DTN2 based file system that has been used in a number of TIER[5] deployments. TierStore takes a different approach to that taken here in that it provides a distributed file system that uses the BP for file system synchronisation actions. TierStore aims to allow application developers to develop or run their applications on the local filesystem with the DTN aspects taken care of mostly below the filesystem API, so TierStore aims for portability of applications onto DTN.

In contrast, we aim to provide DTN services via protocol interoperability, e.g. encapsulating SMTP messages into bundles, which is more of an application layer gatewaying approach. TierStore has been used to provide both web and mail services by running local web and mail applications on top of the TierStore filesystem (though so far, we have not seen detailed reports of its deployment in the field). We feel our design is more likely to work with existing applications and user agents overall since we would argue that the way in which application servers use protocols are generally better specified than the way in which they use file systems, which (if true) would mean that it is easier to develop a robust system based on application layer gateways rather than programmatic APIs. The clear benefit of the TierStore approach of course is that it works immediately for an application that only makes simple use of the file system and that can deal with possible inconsistencies in the state of the file system.

Given that the above argument is essentially subjective, it would be very interesting to see a comparison of how the two approaches compare in the same or a commensurate real-world trial.

III. Background to the Trial

The area for the 2009 and 2010 trials (shown in Figure 1) is part of the Padjelanta national park (67°22'N,16°48'E) and

[5]http://tier.cs.berkeley.edu/

therefore has almost no road, power or networking infrastructure. As a national park, there are many restrictions on the development of such infrastructure, even temporarily.

As there are no roads, the area is serviced by a small number of helicopter companies, that have semi-regular flights, between for example Ritsem and Staloluotka, for which the flight duration is about 20 minutes. Since our remote site for the 2009 trial was in Staloluotka, the Ritsem-Staloluotka helicopter link was our only data-mule route. In 2010 we may make use of other flights in addition as we plan to have addtional remote sites.

The helicopter companies also charter their aircraft for ad-hoc flights, and even with their regular service may make detours to pick up, or drop off, people and goods, mainly on behalf of the local population. (The rules of the national park constrain the locations where the helicopters are allowed land, except when they are carrying some of the local population.) The helicopters thus provide a roughly scheduled service.

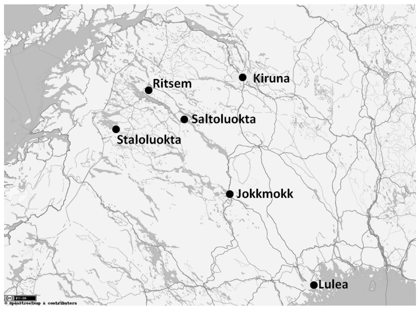


Fig. 1. Map of Test Area (c) OpenStreetMap and Contributors. The Internet gateway with GSM connectivity was in Ritsem, the village DTN router was installed in Staloluotka, approx. 50km distant.

One of the helicopter companies (Fiskflyg[6]) co-operated with our trial, allowing the installation of data-mules on two of the helicopters they operate in the area. When there were some issues with those (described below), the pilots also carried netbook computers handed to them by our personnel in Staloluotka. In Ritsem, the pilots only needed to briefly bring the netbook into their cabin for the few minutes between flights for the necessary exchanges to occur. Local co-operation like this was critical to the success of the trial as will be seen below.

In the winter, this area is essentially deserted. In the summer, between late June and the end of August, there are two different types of user with which we are concerned, the first being the local population of Saami reindeer herders who have summer camps (or villages) that they use in this area for, roughly, six weeks around July of each year. The second are tourists (hikers) who mainly use paths that are maintained by the local tourist agency and who stay in tourist cabins along the paths. (The Saami summer camps and the tourist cabins are generally not co-located, nor are the Saami camps directly on the tourist trails.) The N4C project aims to provide these user communities with some level of network access, even whilst within the national park. There are of course some other classes of potential user, e.g. the tourist cabin attendants and the helicopter pilots, but we do not specifically target these so far, although we do hope that they may themselves make use of the 2010 trial services.

For the Saami population, the aimed-for benefit is to allow them to spend more time in the summer camps - at present, they have to return, via expensive helicopter trips, for even the most trivial communications. (Though reportedly, people sometimes hand their mobile phones to the helicopter pilots with an already-written text message ready for sending once the helicopter reaches an area with mobile coverage.) For these people, any form of communication, however limited, has the potential to save significant time and cost and could therefore really enhance their ability to stay in the mountains and continue their chosen lifestyle, which is their main goal for the summer.

For the tourists, and perhaps the cabin-keepers, (employed by the tourist agency), the benefits of our applications are less clear. However, since the helicopter flights generally are to and from the tourist cabins, these are good locations in which to locate the temporary infrastructure described below. In fact, we also saw quite an amount of interest amongst tourists in the cabins - perhaps this reflects a desire for communications even whilst hiking in the wilderness, or perhaps it merely reflects boredom.

As stated above the 2009 trial was only a week-long test to demonstrate that the equipment and overall design can work, so we took advantage of another type of user: the participants in the ExtremeCom 2009 workshop, who took part in an approximately 50km hike setting out from Ritsem and arriving three days later in Staloluotka. These were IT and even DTN-literate users and were a useful group to test the initial deployment of the hardware and software described here.

However, it is worth noting that the system described below is really designed for the first two types of user, and our goal in 2010 will be to provide email and web services for the Saami population and, to the extent to which they wish to use the system, the tourists.

The N4C project benefits from the direct participation of some of the Saami people. Such local co-operation is essential. We cannot overemphasise the importance of this - without the assistance and goodwill of local people, there is no possibility of a successful trial. Experimenters wishing to carry out trials in locations such as this really must ensure that they first establish, and then maintain, such contacts. During 2009 this co-operation included interacting with the helicopter company and tourist-cabin attendants where otherwise language issues would have posed difficulties. However, local advice and sensitivity were also significant when it came to siting the temporary infrastructure equipment, for example, moving rocks to anchor the stand was prohibited for non-locals (who might accidentally disturb local flora), and locating the equipment near the church in Staloluotka would have shown disrespect.

Aside from the more (and less) obvious logistical and organisational challenges inherent in any field trial, at this stage in the development of DTN, the main challenge in carrying out a DTN field trial in an area such as this is

[6]http://www.fiskflyg.se/

transtioning technology designed for laboratory use to use in the real world. For example, while the DTN stack used has been in laboratory use for a number of years and has seen some limited field use, there were a number of issues (detailed below) with logging that required significant work before, during and after the field trial. As a result of trials such as this the DTN technology should be much better suited for future field trials.

In addition, trials such as this validate the DTN architecture, for example, the ability to really use a data mule is demonstrated here, which involves more than just deploying the technology, but also requires that the technology is suitable for operation by the untrained (in DTN) personnel involved. (In our case helicopter pilots.)

This trial also required an innovative integration of the DTN stack with off-the-shelf hardware products and open-source software to produce a sufficiently robust DTN node, and also development of a router stand suited for use in this trial, and in future, a range of other field trials.

Lastly, this trial involved integration of the DTN stack with standard web and mail services, so that the end-user devices did not have to have any DTN stack installed. While that represents a fairly obvious variant, it has not, to our knowledge, been done before in this way. These application integration aspects of the trial should prove useful to the DTN community once we have merged that source code with the existing DTN open source reference implementation.

IV. Experimental Setup

The system described here has been designed to allow end-users to use familiar applications and interfaces and to hide the DTN. End-users do not have to install any special software and are able to use a standard laptop with wireless access, or indeed any device that can support IEEE 802.11 (b or g, at 2.4GHz) such as various types of mobile phones, or PDAs.

The logic behind these design constraints is that the user populations concerned should only need the most basic conception of DTN, essentially being able to view it as sending their email via helicopter with no DTN specific knowledge being required to use the service and with no requirement to install DTN software on the end-users' devices.

Our overall system for the 2009 trial is shown in Figure 2 and consisted of two computers in Ritsem, one of which had basic Internet connectivity via a 2G GSM modem and the other of which acted as a BP relay; a unit which we call a "Village DTN Router" in Staloluokta and data-mules carried by the helicopters. In addition, users could make use of their own (or the experimenter's) laptop or other wireless device. In addition to laptops, both Nokia 810 PDAs and iPhones were used during the trial by other participants in the ExtremeCom 2009 event, and from the N4C project.

The village DTN router essentially acts as a traditional WiFi hotspot, but instead of connecting directly to the Internet uses the BP to send and receive bundles encapsulating application traffic.

The unit in Ritsem was connected, via a Virtual Private Network (VPN), back to servers in Trinity College Dublin (TCD) that de-capsulated and processed the relevant application traffic as described below.

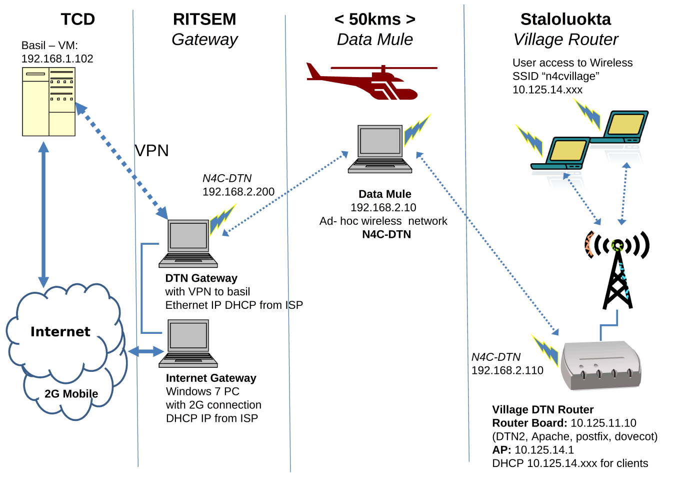


Fig. 2. Network design. From left to right: TCD is connected to the gateway at Ritsem via a VPN running over a GSM 2G link; the gateway connects via ad-hoc WiFi to the data-mule carried by the helicopter; the data-mule connects via ad-hoc WiFi to the village DTN router when the helicopter is present; end-user devices connect via infrastructure-mode WiFi to the village DTN router; DTN protocols are not used on this last link.

A. Email Transaction Walkthrough

Before describing each of the hardware and software components we first present a brief walkthrough of how an inbound email message is handled at the various nodes using the bundle protocol. This should give the reader useful context in understanding the detailed descriptions.

An email is sent to user001@village.n4c.eu from somewhere on the Internet and eventially arrives, as normal, with no DTN protocols used, at the destination mail server, which is hosted in TCD. There the email is received and stored in the user001's message store, which in our case, consists of a set of files, one for each mail message. Periodically, (each morning), a process checks for changes in the overall message store and constructs a compressed tar file containing all the changed files, one of which is our mail message for user001. That tar file will be sent via the DTN using the BP, which is a store-and-forward delay-tolerant protocol.

The tar file is sent as a bundle payload, to the DTN2 daemon (dtnd) running on basil, (our mail server in Dublin for this experiment) which accepts custody of the bundle. If the link between basil and the DTN gateway node in Ritsem is up, then the bundle will be forwarded immediately, and the DTN gateway node will return a new administrative bundle to basil, accepting custody of the bundle containing the tar file. At this point, basil can remove the bundle from its persistent store. If the link from basil to the gateway is down, then the bundle will be stored (on disk) at basil until the link comes up again and the above exchanges have succeeded. The same store-and-forward step and custody handling is repeated at each DTN node in our configuration.

The link between basil and the DTN gateway node runs over a VPN established between these hosts. This is because the Internet gateway node's public IP address is dynamically assigned by the 2G GSM proivder in Ritsem, and the VPN allows basil to use a static address for the DTN gateway node. So, the 2G network provides a public IP address to the Internet gateway node in Ritsem, which in turn provides a private IP address to the DTN gateway node. But the DTN gateway (as necessary) establishes the VPN connection to basil, and uses a known private IP address on that VPN interface. This allows basil's DTN daemon to establish connections to the DTN gateway as required, and vice versa.

When the bundle has reached the DTN gateway, it is stored, awaiting a connection to be setup between the DTN gateway node and a data mule. When that connection is established, then the bundle is forwarded to the data mule and custody is passed to the mule.

Once the data mule moves to Staloluotka (via helicopter), then the same exchange occurs between the data mule and the village DTN router, which is the destination for the bundle in question. At that point, the bundle is delivered to a process that extracts the mail files and places them into the copy of the message store on the village DTN router. Once this has happened, then when user001 next connects to the village DTN router's message store (IMAP) service, then the mail can be read.

So, essentially, our mail message that arrived at the top left of Figure 2 has flowed from left to right in the picture, until (some hours later) is has arrived at the village DTN router at the bottom right of Figure 2, at which point it can be read by user001, using a laptop or PDA as shown at the top-right of the figure.

B. Village DTN Router Hardware

In this section we describe the village DTN router hardware used in 2009, with an emphasis on the reasons for our design choices. Almost identical hardware will be used in 2010.

The primary requirements faced were robustness and use of off-the-shelf components - the hardware chosen had to be able to withstand field use, and given that one of our goals is that other experimenters should be able to construct similar devices, basic engineering skills and a minimal toolkit must be all that is required to build devices.

The full bill of materials and detailed wiring, layout and build instructions for making such a node are available from our web site.

A Village DTN Router is made up of a single board computer, a WiFi access point, batteries for main power and solar panels for charging. Everything except the solar panels is mounted in a single weather proof enclosure. Figure 3 shows a router operating in Staloluotka during the trial - all of the the electronics are in the box at the base of the stand. (For scale, the solar panels are roughly A3 sized, and the base of the stand is 92cm on each side.)

1) Printed Circuit Boards: The single board computer used is the Proteus[7] single board computer from Eurotech. This is an Intel Atom Z530[8] based board which in a single package

[7]http://www.eurotech-ltd.co.uk/en/products.aspx?pg=PROTEUS&pid=10120
[8]http://ark.intel.com/Product.aspx?id=35463

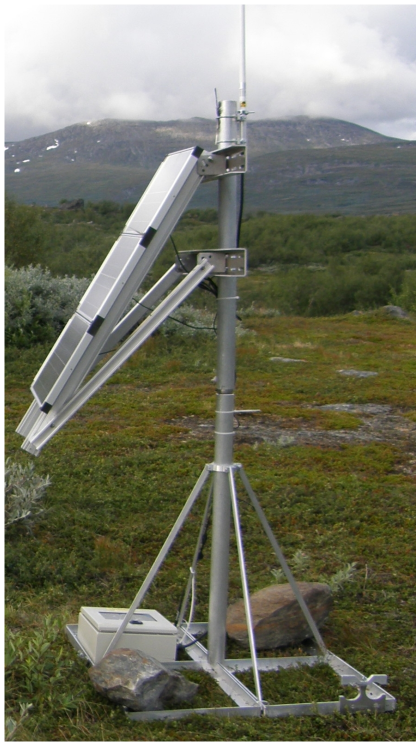


Fig. 3. Deployed Village DTN Router. This is the router shown at Staloluotka during the trial in summer 2009. One can see the stand, 3 x 20W solar panels, infrastructure-mode WiFi antenna and the enclosure (at bottom) containing the electronics. (See Figure 4 for an internal view.)

has 2GB flash, SDIO slot, 2 x PCIe slots, GPS, SIM-card reader, serial ports and USB ports. Overall, the Proteus board used 7.2W, which closely matches the stated usage of 2W for the Atom CPU and 4.3W for its accompanying US15W Chip set.

While the Proteus board may be considered somewhat high-spec for a device that is intended to be deployed in a challenged network environment, (and is also currently quite costly), the flexibility afforded by so many interfaces and capabilities has allowed us to handle what could otherwise have been a number of setbacks during development. (For example, a late-breaking requirement for a second ad-hoc mode WiFi radio.) Basing our development on an arguably "over-specifed" board is therefore yet another aspect of overall system robustness. Our expectation is that in the relatively near future, either the pricing of the Proteus will be reduced, or, once we are more confident of the actual operational requirements we may be able to replace the Proteus with a board that is a more exact fit to those requirements, and is less costly, since there is nothing particularly special about, for example, the fact that the Proteus uses the Intel Atom chip.

As stated, the Proteus is today somewhat expensive and is by far the single most expensive component, costing about twice as much as the second most expensive component (the mounting). However, while we re-evaluated our use of the Proteus in late 2009, we have not so far found that any of the cheaper boards on the market actually save much, if anything, when one takes into account the fact that in all cases so far, some additional components are required to get cheaper alternatives to work. Given that the Proteus was one of the first Atom-based single board computers on the market when we started our design process, and given that we are still (18 months after reeceiving our first Proteus) receiving significant BIOS updates, there is probably a good general argument to be made that selecting a high-end device is better when dealing with newer technology, whereas with more mature technology there are more tailored, tested and cheaper options usually available.

The WiFi Access Point is a Mikrotik RB411[9] single-port router board with an Engenius EMP-8603[10]miniPCI wireless card. For our 2010 trials, we are investigating replacing both of these with a single WiFi PCI card mounted directly on the Proteus. In 2009 however, the use of the Microtik reduced the number of software components that had to be configured and tested, which was deemed worthwhile at the time.

One of the benefits of all of these boards is they each have on-board voltage regulators which is important in systems powered by type of battery used. These batteries do not give a constant voltage, but range from approx 13V when fully charged to approx 10V when depleted. The onboard voltage regulators meant we could connect all these devices directly to the input power, without having to install a separate voltage regulator.

The Proteus also has diode protection - an additional board safety feature that was particularly useful when wiring-up or debugging during trials. We have seen other boards (data-mules) damaged due to accidentally reversed polarity, which can easily happen during configuration or testing in the field.

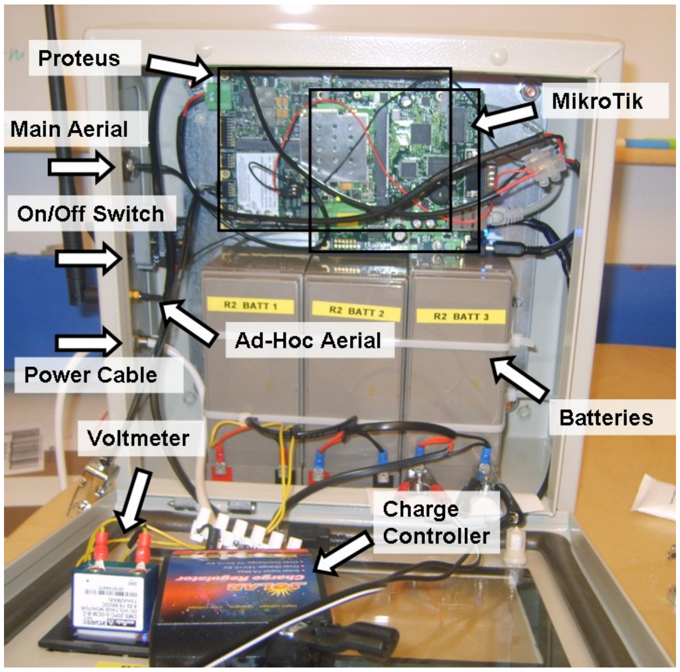


Fig. 4. Village DTN Router interior. For scale, the Proteus is 155x110 mm. The ad-hoc WiFi antenna can be seen on the left. Other components are described in the text.

[9]http://www.routerboard.com/pricelist.php?showProduct=38

[10]http://www.engeniustech.com/datacom/products/details.aspx?id=250

The relatively small form-factor of the above boards was also a factor, although not really in terms of reducing the overall size of the equipment as one might expect, but rather in allowing for flexibility in how components are set out, and secured for transport within the enclosure. The overall enclosure sizing is essentially determined by the batteries. (In fact, for our 2010 trial, we have made some substantial simplifications in component layout within the enclosure and in the wiring harness as a result of lessons learned during 2009.)

Figure 4 shows the internals of a village DTN router.

2) Power: Power for the system is provided by three 12V 7.5AH valve-regulated lead-acid (VRLA) batteries. Using three batteries rather than one allowed us to experiment with mounting and positioning options within the enclosure, but at the expense of having more connecting wire. Lead-acid chemistry batteries are better suited to solar charging, and VRLA batteries do not have to be kept upright as the electrolyte is sealed in. This is important as it means the Village DTN Router can be transported safely and also reduces restrictions on its final mounting position. (Vertical or horizontal.)

For development purposes, we used a Studer SBM-02 power monitor[11] that allowed us to log battery voltage levels, power consumed and charging levels. This device was not required for operation, but was essential during development in order to assess the actual power consumption and hence the useful battery life.

We also included a volt meter[12] with an LED display mounted on the inside of the window to give feedback as to the state of the batteries in field. For 2010, we are considering a more sophisticated digital monitoring system, in conjunction with more sophisticated power management, that is enabled by the latest revision of the Proteus BIOS.

Power to charge the batteries is provided by three 20W 12V "fastFIX Mono" solar panels.[13] One of the reasons for choosing 3 separate panels rather than a single larger panel is that the entire system (in fact two systems, plus spares) had to be shipped from Dublin to Lulea in northern Sweden. The equipment then had to be assembled and taken by car to the helicopter base at Ritsem, then flown by helicopter to Staloluokta. The system therefore had to be man-portable, and "fold down" into a suitable form-factor. Three smaller solar panels are easier to manage than a single larger panel. The three panels were connected in parallel to give a theoretical maximum 60W of power and connected to the charge controller in the main system enclosure.

In 2009, we had an "arm" assembly that allowed us to change the elevation angle of the solar panels to face the sun directly, but we noted that most other solar panels in the area were simply mounted vertically to the ground. At the latitude concerned, vertical mounting seems to be just as effective so we eliminated these arms for our 2010 trial. A side-effect of the 2010 mounting was that we could easily swivel the solar panels to face the sun, which ancedotally seemed effective.

[11]http://www.studer-inno.com/?cat=battery_monitoring

[12]http://ie.farnell.com/1339314

[13]http://www.sunshinesolar.co.uk/khxc/gbu0-prodshow/SS20WP.html

We expect to be able to measure the effectiveness of this from 2010 data.

The final power related item in the enclosure is the solar-charging regulator.[14] VRLA batteries must be prevented from discharging completely as this reduces their lifespan and ability to hold charge. Conversely batteries must also be prevented from overcharging as this will cause generation of excess hydrogen and deplete the electrolyte. A solar-charging regulator is designed to be connected to the solar panels, the battery and the load. It monitors the current flowing in and out, disconnects the load if the battery voltage drops too low, (below 11.1V in this case), and disconnects the solar panels if the battery voltage goes too high (14.5V). For 2010, we plan to use a more capable solar charge controller,[15] that offers a number of advantages, for example using pulse width modultation (PWM) which allows the battery to be more completely charged.

For the 2009 version of the village DTN router, the boards consume 0.9 amps at 12V and the batteries store 21AH, giving an approximate lifetime running at full power of roughly 22 hours. In reasonable sunlight the solar panels produce approximately 36W of power (not, of course the full 60W) and in the Arctic in summer we get about 19.5 hours of daylight, though with fixed mounting of the solar panels good charging conditions are only experienced for a fraction of that time. Simple calculation therefore shows that we need 7 good hours of sunlight in order to fully charge batteries from empty while drawing no load.

From this one can (correctly) conclude that our 2009 village DTN router required quite an amount of good sunlight in order to keep operating. In order to do better, one has to make use of lower power standby states which, unfortunately, were not available with the version of the Proteus BIOS used then. However, since we only required the system to operate for a short period, there was almost always enough power.

Since then, a BIOS update has resolved this issue, so that the 2010 village DTN router will be able to put itself to sleep when power levels are reduced and wake on a real-time clock alarm when power levels are improved. Testing of these power management features is ongoing at the time of writing but will be crucial for the longer trial planned for 2010.

In addition, this BIOS update has activated some power saving features of the Intel Atom processor that were unavailable in 2009. For example, the CPU can automatically change from running at 1.60GHz to 800MHz when idle, and move to a C6 state, saving power on the system. For example, with high activity (C0 state) the CPU draws 750mA, whilst in C6 state power consumption drops to 620mA, a saving of about 20%, which in a power sensitive system extends battery life considerably.[16]

3) Enclosure and Mounting: All of this equipment is inside a metal, IP66 compliant enclosure with a toughened glass window. We preferred a metal enclosure over plastic due to previous experience of the project members, and the option of a toughened glass window meant the status of the Proteus and other LEDs and our voltage meter could be read without having to drill additional holes in the enclosure, potentially compromising its ingress protection.

[14]http://www.sunshinesolar.co.uk/khxc/gbu0-prodshow/REG7.html
[15]http://www.sunshinesolar.co.uk/khxc/gbu0-prodshow/E-REG10A.html
[16]http://www.hardwaresecrets.com/article/611/6

The enclosure chosen was the Eurobox mild-steel model 83502 cabinet[17] which has a removable plate in one wall to allow for holes to be drilled for cables and switches. Initially three holes were planned, one for the power cable from the solar array, one for the antenna cable, and one for an on/off switch. One additional hole was required for a second, ad-hoc mode, WiFi antenna as described below.

When designing a device to operate outdoors, it is of course important to minimise the number of holes and all switches and glands must provide the same or, if at all possible, a higher level of ingress protection than the enclosure itself. Details of the specific components chosen may be found in the bill of materials on our web site.

All apertures were further sealed with waterproof bituminous tape, which should be included in field kits, since ingress protection is otherwise put at risk if for example, an antenna must be temporarily removed.

One of our goals in the N4C project is to ensure that other experimenters can make use of our village DTN router in future trials. One of the less obvious aspects of that is that solar panels and radio antennae need to be securely mounted, and suitable mountings are frequently not available, in particular easily transportable mountings.

We therefore designed a foldable aluminium stand, sturdy enough (with some ballast) to hold the three solar panels and the mounting brackets, and the system box, but light enough to be carried by one person, and fit in the passenger compartment of a helicopter. There are also weight limits that apply when transporting equipment on the helicopters we used in 2009, hence the use of Aluminium. CAD drawings for this design are available from our web site.

C. Data-Mules

Data-mules are nodes that physically move and carry bundles so as to connect otherwise disconnected parts of the DTN. In our trials, data-mules do not run any application layer code, but only act as BP nodes.

Two types of data-mule were deployed for the 2009 trials. One was a single board computer (SBC) mounted in an enclosure in the helicopters, the other, initially intended as a backup in case the helicopter mounted unit failed, (which occurred), was an Asus EEE PC 901 that was handed to the pilot. Before getting to the failure and how we handled that, we first describe what we planned that ultimately failed, since such hard-learned lessons are worth preserving.

The SBC helicopter-mounted data-mules ("Wireless Router Application Platforms," referred to as "WRAPs") were selected by other partners in the N4C project, and were previously used in the SNC project. These are essentially end-of-life off-the-shelf products, so we only provide a brief overview of the hardware here. The board used was the PC engines

[17]http://www.euroboxenclosures.co.uk/mild-steel-cabinets.php

WRAP 2.E[18] running the OpenWRT operating system and with a Tonze PC686X[19] mini PCI wireless card. These units were enclosed in a "mini-box" outdoor enclosure also from PC engines[20].

As part of a separate deployment that overlapped with our trial, the WRAPs had been deployed in the helicopters to support other DTN tests that made use of ad-hoc WiFi networking. Since the WRAPs were already deployed we attempted to interface with them via the second radio on the village DTN routers. This was a late surprise in that we only learned of it a few weeks before the start of the trial, (we had previously assumed that all nodes would be using infrastructure mode WiFi). As described above, once we found out that ad-hoc WiFi was required we added the second radio to the Proteus board and proceeded with our integration and testing. At this point we also developed a backup (described below) in case the late surprise caused problems.

However, once in the field during testing in Lulea and Ritsem, we experienced many problems with ad-hoc WiFi interoperability and so had to deploy our backup data-mule solution. We never reached a satisfactory conclusion as to exactly why this was the case, but while two ad-hoc radios based on the same chipset seemed to be able to associate consistently and reliably, we found that combinations of different chipsets gave highly variable results, in a lot of cases with no WiFi association forming at all.

So during the actual trial, none of the data reported on here was transported via the WRAPs but all used our backup data mule solution.

Our backup data-mule solution was an Asus EEE PC 901 netbook, running the Ubuntu 8.04 operating system, using ad-hoc mode WiFi, with the DTN2 implementation of the BP (as had the WRAPs).

We had been able to test the ad-hoc WiFi for these during development, and knew that they would work. In principle, we could have had the EEE PC data mules use infrastructure mode WiFi, however we did not, because we only finally decided to use these data mules after we were in the field and it was too risky to change the configurations of all other nodes at that point.

The modus-operandi for these mules was to keep them powered up whilst at the helicopter base in Ritsem, and in WiFi contact with the DTN gateway node described below. A pilot making a trip to Staloluotka simply carried the mule (still powered up) in the helicopter and handed it to our personnel on arrival in Staloluotka. Our personnel in turn brought the data-mule into range of the village DTN router (and checked it was operating correctly) so that the bundles were transferred, and then handed the data-mule back to the pilot. On arrival back in Ritsem, the pilot simply plugged the data-mule back into its power supply which was in range of the DTN gateway device described below. All of the traffic reported on below was carried by the EEE PC data-mule.

Having said all of the above, the helicopter mounted data mule remains a better solution (if it works), and so we plan to attempt this again for our 2010 trial, since we won't always be present to swap the EEE PC with the pilot in 2010. For the 2009 trial, the WRAPs were powered directly from the helicopter 12V power supply and were mounted below the front passenger seat. For our 2010 trial, we plan to re-use these devices mounted in the same locations, but this time running infrastructure mode WiFi and with a battery backup since testing in 2009 showed that there were a number of potential communications opportunities lost due to the pilot turning off the helicopter ignition switch immediately on landing, before the WRAP had succeeded in establishing contact with another node.

[18]http://www.pcengines.ch/wrap2e1.htm
[19]http://www.cyberbajt.pl/produkt/1444/
[20]http://www.pcengines.ch/case2c.htm

D. Internet/DTN Gateway

The Internet/DTN gateway was composed of two devices - a tablet PC with a 2G/3G USB dongle to provide basic Internet connectivity and DHCP address service, and another Asus EEE PC 901 which acted as a DTN gateway. The tablet PC was also used for other experiments using the WRAPs and so was essentially a piece of "legacy" equipment.

Since the tablet PC ran Microsoft Windows, and the DTN2 stack has not been ported to that operating system we needed the EEE PC to receive bundles from the data-mule and to forward them via the Internet to a server (basil) back in Dublin which was the source or destination endpoint for DTN services.

Interestingly, the 2G connection from the tablet PC often experienced outages, and so the EEE PC relay was in fact quite useful since those outages often co-incided with the arrival or departure of helicopters. (We hypothesised that people frequently used their GSM phones around the time they took a helicopter ride, which may have affected the 2G bandwidth in the otherwise sparsely populated cell.)

We secured the connection back to basil using a virtual private network (VPN), which was installed on the DTN gateway EEE PC using OpenVPN[21] which connected to the our server, basil, in Dublin. Basil then routed email and executed web requests. The VPN was necessary as we were using static routing for DTN and such routes need fixed IP addresses or DNS entries. The VPN eliminated any issues that might have arisen since the tablet PC could get different IP addresses when it received its own Internet-facing IP address from the 2G/3G network - essentially it meant we controlled all the IP addresses of all of the DTN nodes.

E. Radio Links

Our initial design was for all clients (including data-mules) to operate in WiFi infrastructure mode. The main antenna for the village DTN router was mounted on the top of the aluminium stand over the solar panels. We chose a weather-proof, 12dBi high-gain, vertical polarisation, omni directional antenna[22] and an n-type connector to a large diameter co-axial cable. Large diameter co-axial cables are less flexible but result in far less signal attenuation (of the order of 0.5dB per meter).

[21]http://www.openvpn.net/
[22]http://www.itelite.net/products-desc.php?id=274

However, the WRAP data-mules installed on the helicopters (by another partner in the N4C project) had been configured to operate in ad-hoc WiFi mode in order to support other experiments by other partners in the N4C project. This is an example of the kind of "late-surprise" that is very likely to occur in multi-partner field trials. Luckily, the additional capabilities of the Proteus board allowed us to to install an additional PCIe wireless card on the Proteus board, which we configured to operate in ad-hoc WiFi mode, thus allowing us to interoperate with the deployed WRAPs (in principle, if not in practice). The second radio also of course required a second antenna, which was an EnGenius 7dbi vertical polarisation omni-directional antenna.

In the field, these ad-hoc WiFi connections proved very troublesome, with many associations not being established. The main pattern was that two identical devices would work reliably but that mixed devices would not. For this reason we plan to move to an all-infrastructure mode or 2010 testing.

The main lesson we learned here is to be very cautious of ad-hoc WiFi with heterogeneous devices. In any future trials where ad-hoc WiFi is to be used, we will not assume inter-operability, (we would make that asumption for infrastructure mode), but will only use combinations of devices that we have recently tested, with a strong preference for just one device type.

F. Software

Village DTN routers, the Internet and DTN gateways and the EEE PC data-mules all ran Ubuntu Linux LPIA version 2.6.24-16-lpia.

The village DTN router provided a wireless hotspot service to the end-users. The wireless access was provided by the Mikrotik/Engenius board running RouterOS Level 4. The Proteus provided DHCP services, a DNS server, NTP server, Apache Web-server (with mod_auth_forms, PHP5) and Squid proxy server, MySQL server, Dovecot IMAP server and a Postfix SMTP server.

Delay tolerant networking support was provided via the open-source DTN2[23] stack, version 2.6.0 (changeset 3450). All convergence layers were TCP, with node-discovery[24] enabled. Other that this discovery aspect, all DTN routes were static, with the various node's IP addresses configured on each node.

Data-mules provided no end-user services and therefore simply ran the DTN2 stack. The DTN gateway also provided no end-user services and therefore ran the DTN2 stack and OpenVPN to connect to the support system basil at TCD.

1) Naming: The overall approach followed with naming was to configure DNS names for nodes with IPv4 addressing and to derive DTN Endpoint Identifiers (EIDs) from those names. This was a purely pragmatic approach since our aim was to investigate how well existing DTN protocols actually worked in the field, rather than to test new experimental DTN naming features.

As we have seen, our DTN nodes were defined as gateways, routers or data-mules plus the special case of basil, which was well-connected to the Internet at all times.

[23]http://dtn.sourceforge.net/

[24]http://dtn.sourceforge.net/DTN2/doc/manual/configuration.html#discovery

Routers and gateways were multi-homed. Each had an ad-hoc interface and a managed mode interface and separate IPv4 private network address spaces were assigned for each. The 10.125/16 address space was assigned to managed mode interfaces and 192.168.2.0/24 address space was assigned to ad-hoc mode interfaces. A range from the 10.125.14/24 address space was assigned for the DHCP servers on each router, to provide IP addresses to end-users.

Routers and basil each had a DNS server configured with A, PTR, CNAME, TXT and MX records for the set of all relevant DTN nodes.

The DNS domain used for the all field nodes in our tests was "village.n4c.eu". In this trial there were only a small number of nodes, each manually configured.

DTN2 requires one to configure interfaces for routing, so most interfaces were named using the DNS domain and a hostname indicating the type of node (*dtnrouter*, *dtnmule* or *dtngateway*), and a unique string (the last two octets of the interface's IPv4 address.) Figure 5 gives an example.

$$\boxed{dtngateway\text{-}11\text{-}10.village.n4c.eu}$$

Fig. 5. Example Node Name - the Village Router in Staloluotka

The Delay-Tolerant Network architecture [4] requires each node to have at least one EID that uniquely identifies that node. In our case, we used a Convergence Layer Adapter (CLA) identified by prepending 'dtn://' and appending '.dtn' to the *interface identifier*. See Figure 6 for an example. This form of naming is a convention used by many DTN2 deployments.

$$\boxed{dtn://dtngateway\text{-}2\text{-}200.village.n4c.eu.dtn}$$

Fig. 6. Endpoint Identifier of a CLA (CLAID)

A *registration* is a named inter-process connection (IPC) that an application uses with the DTN2 daemon in order to send or receive bundles. This is how the DTN2 daemon (*dtnd*) knows to deliver, rather than forward, bundles and is also how the DTN2 daemon listens for bundles from applications.

The DTN web and email application's EID registrations were a concatenation of the EID of a CLA, a delimiter '/' and a unique service *descriptor*. The service descriptors used were: HTMLgateway, HTMLrouter, mailtogw, and, mailsyncin. Figure 7 provides and example of a registration EID.

$$\boxed{dtn://dtnrouter\text{-}11\text{-}10.village.n4c.eu.dtn/HTMLgateway}$$

Fig. 7. EID for HTMLgateway service

The *dtnN4Crecv* middleware provided the web gateway service with service descriptor HTMLgateway. This could be configured to register as a web gateway on basil, (See Figure 7), or on a router. The same *dtnN4Crecv* middleware also provided two email gateway services, with descriptors mailsyncin and mailtogw. While we do not claim any significant benefit for this setup, (though it did allow for some code re-use), it is necessary need to know this in order to re-use our code or to fully understand the supplementary data.

The main Internet-connected support DTN node was basil, located in Dublin, and its CLAID was *dtn://basil.dsg.cs.tcd.ie.dtn*.

2) Routing: We designed our routing scheme, based on static routes, well in advance of the actual field trial. As a result, at design time, we faced a number of uncertainties, in terms of potential node placement and even in terms of how well our nodes would perform. For these reasons, the routing scheme included routes that, as it turned out, were not required. However, as designed, this scheme allowed us to replace one village DTN router with another without re-configuring in case a node failed.

In the event, one router ("11-10," in the supplementary data) was deployed in Staloluotka and then transported to Saltoluokta (where the workshop component of the ExtremeCom event was held). The second, backup, router ("11-20") was not required, other than for testing, which mainly happened in Ritsem. However, in this section, we describe the full routing setup, including the backup router.

Bundles from the field test, with email or web request payloads were received by basil, which then routed email and executed web requests. Response bundles were routed back to the revelant field nodes.

Bundles transmitted between router nodes and gateway nodes always traversed a mule node (See figure 8).

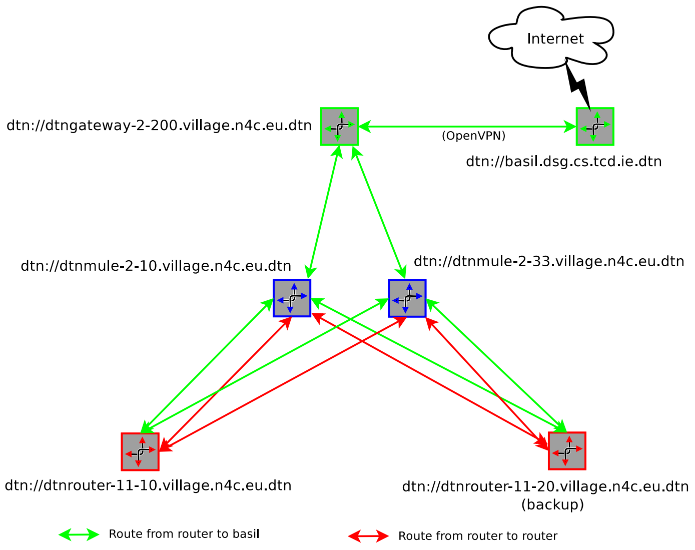
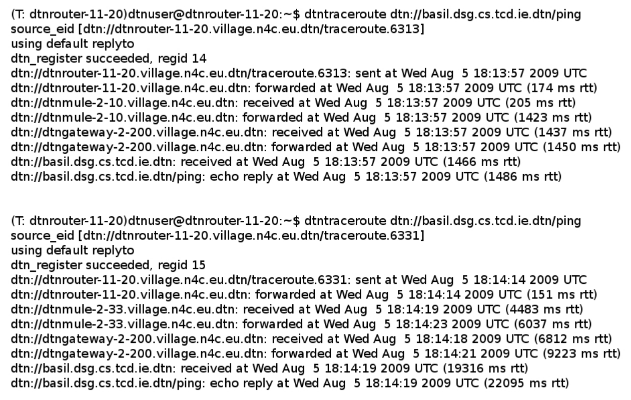

Internet
dtn://dtngateway-2-200.village.n4c.eu.dtn
(OpenVPN)
dtn://basil.dsg.cs.tcd.ie.dtn
dtn://dtnmule-2-10.village.n4c.eu.dtn
dtn://dtnmule-2-33.village.n4c.eu.dtn
dtn://dtnrouter-11-10.village.n4c.eu.dtn
dtn://dtnrouter-11-20.village.n4c.eu.dtn
(backup)
Route from router to basil
Route from router to router

Fig. 8. Static routes. Redundant routes were configured to allow for use of a backup router and mule in case of failure in the field. See Figure 2 for the physcial topology. Bundles from basil were always routed to both the main and backup routers.

Two routers, one Internet and DTN gateway and three mules were deployed. The DTN gateway in Ritsem relayed bundles to the basil in Dublin and vice versa.

The three data mules consisted of two WRAP units, each mounted in a helicopter serving the Ritsem/Staloluotka link, and one Asus EEE PC that was handed to helicopter pilots as described previously.

Two examples of the paths bundles follow when transmitted from a router to a gateway via a data-mule are shown in Figure 9. These traces show the situation during testing in Ritsem, before the router was deployed at Staloluotka, hence the low latencies.


```
(T: dtnrouter-11-20)dtnuser@dtnrouter-11-20:~$ dtntraceroute dtn://basil.dsg.cs.tcd.ie.dtn/ping
source_eid [dtn://dtnrouter-11-20.village.n4c.eu.dtn/traceroute.6313]
using default replyto
dtn_register succeeded, regid 14
dtn://dtnrouter-11-20.village.n4c.eu.dtn/traceroute.6313: sent at Wed Aug  5 18:13:57 2009 UTC
dtn://dtnrouter-11-20.village.n4c.eu.dtn: forwarded at Wed Aug  5 18:13:57 2009 UTC (174 ms rtt)
dtn://dtnmule-2-10.village.n4c.eu.dtn: received at Wed Aug  5 18:13:57 2009 UTC (205 ms rtt)
dtn://dtnmule-2-10.village.n4c.eu.dtn: forwarded at Wed Aug  5 18:13:57 2009 UTC (1423 ms rtt)
dtn://dtngateway-2-200.village.n4c.eu.dtn: received at Wed Aug  5 18:13:57 2009 UTC (1437 ms rtt)
dtn://dtngateway-2-200.village.n4c.eu.dtn: forwarded at Wed Aug  5 18:13:57 2009 UTC (1450 ms rtt)
dtn://basil.dsg.cs.tcd.ie.dtn: received at Wed Aug  5 18:13:57 2009 UTC (1466 ms rtt)
dtn://basil.dsg.cs.tcd.ie.dtn/ping: echo reply at Wed Aug  5 18:13:57 2009 UTC (1486 ms rtt)


(T: dtnrouter-11-20)dtnuser@dtnrouter-11-20:~$ dtntraceroute dtn://basil.dsg.cs.tcd.ie.dtn/ping
source_eid [dtn://dtnrouter-11-20.village.n4c.eu.dtn/traceroute.6331]
using default replyto
dtn_register succeeded, regid 15
dtn://dtnrouter-11-20.village.n4c.eu.dtn/traceroute.6331: sent at Wed Aug  5 18:14:14 2009 UTC
dtn://dtnrouter-11-20.village.n4c.eu.dtn: forwarded at Wed Aug  5 18:14:14 2009 UTC (151 ms rtt)
dtn://dtnmule-2-33.village.n4c.eu.dtn: received at Wed Aug  5 18:14:19 2009 UTC (4483 ms rtt)
dtn://dtnmule-2-33.village.n4c.eu.dtn: forwarded at Wed Aug  5 18:14:23 2009 UTC (6037 ms rtt)
dtn://dtngateway-2-200.village.n4c.eu.dtn: received at Wed Aug  5 18:14:18 2009 UTC (6812 ms rtt)
dtn://dtngateway-2-200.village.n4c.eu.dtn: forwarded at Wed Aug  5 18:14:21 2009 UTC (9223 ms rtt)
dtn://basil.dsg.cs.tcd.ie.dtn: received at Wed Aug  5 18:14:19 2009 UTC (19316 ms rtt)
dtn://basil.dsg.cs.tcd.ie.dtn/ping: echo reply at Wed Aug  5 18:14:19 2009 UTC (22095 ms rtt)
```

Fig. 9. Example routes as shown via the DTN2 dtntraceroute command. Each traceroute bundle is shown arriving and departing at each node, in the first case from router to the mule, then gateway and finally basil. The second case is the same except the route is via a different mule ("2-33").

Each node was configured with static routes to the EIDs of all the available applications and the CLAs of every other node in the topology (Please see Figure 8).

Static routes were configured on:

- Each router for every EID not contained on that router via the EID of each CLA of each mule in the topology.
- The gateway in Ritsem for every EID registered on a router via the EID of each CLA of each mule in the topology and directly to the EID of the CLAs of basil for EIDs that did not contain village.n4c.eu.
- Basil, for every EID that contained village.n4c.eu via the EID of each CLA of the Ritsem gateway.
- Each mule to every EID of each CLA of each router and gateway in the topology.

All of the static routes were cofigured as TCP "ONDEMAND" links. When such links are dropped (e.g. after a data-mule travels out of the coverage area of a village DTN router), then the DTN2 daemon will periodically attempt to re-establish the TCP connection. In our configuration, this was attempted after a random delay of between 1 and 10 seconds.

In addition to static routes, we also used the link discovery features of the DTN2 stack. The main reason to turn on link discovery was due to uncertainty in planning - until very late in the day, we could not be sure which type of mule (WRAP or EEE PC) would be used, and we were uncertain as to which precise mule instances would be used right up until after testing in Ritsem had begun. Since the link discovery mechanism allows for bundle forwarding even in the absence of specific routing information, and we essentially only provided a single real route (via helicopter) this was a useful backup.

As it turned out, link discovery was also useful in order to support other N4C experimeters who were at the ExtremeCom event and who were experimenting with a different "nomadic" email solution.

Link discovery works by having each DTN2 daemon emit a UDP broadcast message periodically (every 10s) containing contact information for that node. A node receiving such a message (i.e., a data-mule coming into range of a village DTN router) would establish a TCP connection to the relevant address and port. That TCP connection was then treated as a potential route for any bundle.

However this mix of "ONDEMAND" and discovered routes can lead to cases where a bundle will be transmitted both over the just-reestablished pre-configured static link and over the discovered link.[25] Nonetheless, we felt this was worthwhile since our overriding concern was to get bundles back from the field and optiminsing usage of link capacity is a secondary issue.

Each node accepted custody for each bundle it received and requested custody for each bundle it transmitted. That meant that each node was capable of re-transmitting any bundles as required.

3) Time: The BP requires that nodes have rougly synchronised clocks in order to mark bundles with a creation time, and so that those bundles can be dropped after a configured expiry period.[26]

Since transit times were of the order of one day, bundles had to have an adequate lifetime in order not to be expired whilst in transit.

NTP was therefore configured on all nodes. The village DTN router had an NTP server and provided time to end-user hosts in the WiFi hotspot if requested. Routers in turn requested time from data-mules which requested time from gateways. The DTN gateway at Ritsem took time from an NTP server on basil which was well-connected to the Internet. As we'll see later, NTP did in fact correct the clocks on the nodes during the trial.

4) N4C Middleware: Before explaining the web and mail applications, it is first useful to understand how we handled encapsulation of mail and web traffic in the BP. The applications *dtnN4Cmiddleware* and *dtnN4Crecv* handled this encapsulation.

dtnN4Cmiddleware was designed to receive a user-typed Uniform Resource Locator (URL) from an HTTP request on a local TCP socket, to generate a bundle with that URL plus a transaction identifier (see below) as payload with a destination EID of the application (on basil) that would process the request. The bundle was transmitted with a source EID of the router that made the request (See Figure 10).

The *dtnN4Crecv* application was used for both N4C web and email applications on both the routers and basil.

On basil, *dtnN4Crecv* was used to receive these bundles, parse the "Transaction ID" and 'URL request' from the bundle payload, or application data unit (ADU). In this context,

[25]One could argue that DTN2 should not do this where the destination CLA, host and port are the same, but that is the current behaviour.

[26]At the time of writing, the DTNRG are investigating ways of relaxing this "rough real-time" requirment.

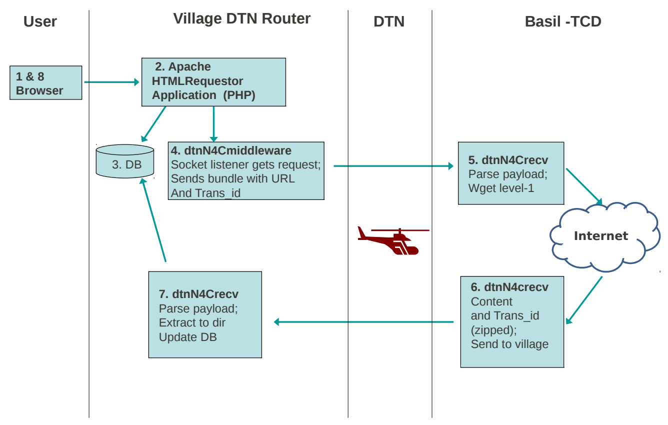

User
Village DTN Router
DTN
Basil -TCD
1 & 8 Browser
2. Apache HTMLRequestor Application (PHP)
3. DB
4. dtnN4Cmiddleware Socket listener gets request; Sends bundle with URL And Trans_id
5. dtnN4Crecv Parse payload; Wget level-1
Internet
6. dtnN4crecv Content and Trans_id (zipped); Send to village
7. dtnN4Crecv Parse payload; Extract to dir Update DB

Fig. 10. URL request application. (1) The user sends a request for a URL to the village DTN router from her browser. (2) On the village DTN router, Apache receives the request which is handled by the HTMLRequestor PHP application. This writes state information to the (3) database and passes the request data via a local socket to the the (4) dtnN4Cmiddleware application which encapsulate the request. When the bundle arrives at TCD, (5) the dtnN4Crecv application decapsulates the bundle content and fetches the web content which is then encapsulated (6) by another instance of the same middleware and sent back to the router, (7) where dtnN4Crecv decapsulates the web content and updates the database; when the user re-connects, (8) the web content can be displayed. All encapsulated requests and responses include a unique transaction ID.

dtnN4Crecv received the ADU on its *HTMLgateway* registration. On reception of the HTML request *dtnN4Crecv* obtained a predetermined set of the requested data using a web crawler, generated a new bundle in which that data was the payload and transmitted that bundle to the EID that was the source EID of the bundle it had received. The payload of the response bundle was the compressed file containing the results of web crawling, with a filename based on the 'Transaction ID' from request bundle.

On reception of a response bundle, *dtnN4Crecv* uncompressed the payload and created a directory with a name that was the 'transaction ID' indicated by the payload. Access to the newly created directory was controlled by the web application described below.

In the context of email application *dtnN4Crecv* was used to receive an email, parse the status report and output the original bundle creation timestamp and sequence number 'creation.seq'. When used in the context of email applications, *dtnN4Crecv* used the two email registrations.

G. Web Request Manager

The HTTP request service allowed users in a village to submit requests for URLs and to track those requests. The overall intent was to provide some level of offline browsing, so that the parts of web sites that were suited for such browsing could be viewed even from remote areas.

In order to submit requests, users only needed to point a web browser (on their phone, PDA or other computer) at the site in question and they would be re-directed into the transaction handling workflow. Transaction tracking was required since the typical latency involved in web requests was of the order of a day.

Web transactions could be flagged as either private or public. A public transaction was one where the original requestor did not mind that everyone else in the field trial could access the results from the cache. A private transaction was one where only the original requestor could access the results. (Where “only” here reflects a relatively trusting environment, certainly less threatening than the public Internet.)

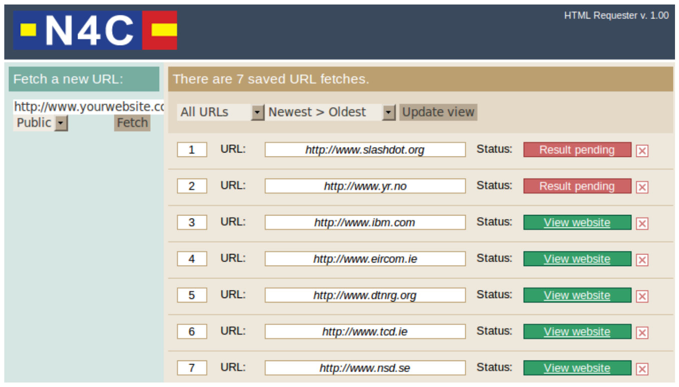
The point of private transactions is that a cache-hit for a public transaction reveals that someone else has also asked for the relevant URL. In a context like ours, with very few people involved, it can be trivial to know who it was that requested that URL, and that could have privacy implications. For example, if the URL related to some medical condition.

When designing this web service, we did not want to force users to create any new web accounts to make use of private transactions. This was mainly for usability reasons and so that we would not be in a position where we had to manage anything more sensitive (such as a password possibly re-used from a more sensitive application context) than the private transaction itself. Given that the village DTN router boxes were not in a secure location and that users have a tendency to re-use passwords, we felt that this was a better option.

Private transactions were therefore protected via simple possession of a cookie. If the web client (browser) permited cookies to be set then a transaction could be private. If not, only public transactions could be supported.

The web portal was implemented using an Apache module, mod_auth_form[27], PHP and a MySQL database (please see Figure 10).

Any HTTP request to TCP port 80 or 443 was intercepted and redirected to this portal (please see Figure 11) which used PHP to query whether the browser would accept cookies and then redirected the browser to either a restricted page/directory or to a public directory, depending on the browser configuration.



There are 7 saved URL fetches.

1	URL:	http://www.slashdot.org	Status:	Result pending
2	URL:	http://www.yr.no	Status:	Result pending
3	URL:	http://www.ibm.com	Status:	View website
4	URL:	http://www.eircom.ie	Status:	View website
5	URL:	http://www.dtnrg.org	Status:	View website
6	URL:	http://www.tcd.ie	Status:	View website
7	URL:	http://www.nsd.se	Status:	View website

Fig. 11. Portal with web requests. The URL entry dialog is on the top left, the top of the right pane shows the number of requests, the bottom of the pane has an entry for each request - green if a response has been received, or red (e.g. items 1 and 2 in the figure), if no response has arrived so far. Clicking on a green box re-directs the user to the the received web content.

If the browser prohibited cookies, a default cookie value was used by the server side indicating that the transaction was anonymous. If the browser permitted cookies, a random 32-character cookie called ’uid’ (“User ID”) was set if necessary, and added to a MySQL database table. This cookie was long lived and scoped for the village.n4c.eu domain.

The mod_auth_module was however designed assuming that users would enter usernames and passwords and then be assigned short-lived session cookies. In our case, we needed to map the long-lived uid cookie to these short-lived session cookies, which were used to manage the user’s session on the portal.

On the portal page (whether public or private), the user then had to enter (or possibly re-enter) the desired URL, at which point the bundle based HTTP request processs flow started. Once the HTTP responses for that transaction had been received those were associated with the uid cookie value and the session cookies, and a page refresh would then indicate that the transaction had completed and offer a link to the transaction results.

When the user clicked that link the URL fetched was below the village.n4c.eu domain and the local part of the URL specified the site from which the content had been fetched.

1) Web Application Components: We now descibe how the web application handled requests using the bundle protocol. See figure 10 for an overview of the data flow described here.

1) **User submits URL via browser.**
2) **WEB based portal to accept web requests, part of the application was written in PHP.**
    a) URL entered.
    b) Request designated ’Private’ (own use) or ’Public’ (open to everyone)
    c) Tracked user cookie as ’User ID’ and ’Session ID’
    d) Assigned a ’Transaction ID’ to URL request
3) **Database (MySQL) this part of the application tracked the URL request**
    a) DB Members,
    b) ’Transaction ID’, URL, session cookie (SID1), encrypted cookie ID
4) **Middleware on router**
    a) PHP to middleware SOCKET to DTN API used send only URL and ’Transaction ID’ in first 32 bytes of bundle payload. The Bundle moves through the DTN on the mule.
    b) The data mule accepts bundle custody
    c) Forward bundle to next hop
    d) Give up bundle custody
    e) Gateway at the permanent connection point in Ritsim
    f) Accept bundle custody
    g) Establish OpenVPN tunnel
    h) Forward bundle to next hop at OpenVPN termination (BP over OpenVPN)
    i) Give up bundle custody
5) **Bastion Server Basil in TCD**
    a) Middleware accept custody of bundle
    b) It parses payload
    c) Wget - (perform heuristics) partially mirror site to directory named after ’Transaction ID’
6) **Middleware on Basil in TCD**
    a) Compress as transaction_ID.tgz
    b) Create bundle with transaction_ID.tgz as payload and Destination EID of village.
    c) Forward bundle to next hop at OpenVPN termination (BP over OpenVPN)

[27]http://comp.uark.edu/ ajarthu/mod_auth_form

  d) **Gateway on the return journey**
  e) Accept bundle custody
  f) Forward bundle to next hop
  g) Give up bundle custody
  h) **Mule**
  i) Accept bundle custody
  j) Forward bundle to next hop
  k) Give up bundle custody
7) **Village Middleware (village)**
  a) Accept custody of bundle
  b) Parse payload
  c) Extract to dir in session cookie/id and user cookie protected area.
  d) If content not marked private use Wget mirror site directory through the guest squid cache 'Transaction ID'.
  e) Update portal and database
8) **User logs in to check request status**

H. Email Service

The email service was configured to allow a user connected to a village DTN router to send and receive email to and from the Internet. Mail account provisioning was not provided, so mail users had to make use of pre-provisioned accounts, for example *user001@village.n4c.eu*, to which they could forward mail from the Internet, or from which they could send mail to the Internet. The logic here was to allow users to use standard email user agents (MUAs) in the hotspot-coverage of the village DTN router, without having to require those users to be at all aware of the DTN.

For the 2009 trial, we did not attempt to provide mail account provisioning, since the day-long roundtrips, and the limited duration field trial, would not have resulted in significant traffic. Instead, we pre-advertised the mail facility to ExtremeCom participants and encouraged them to setup forwarding. Note however, that the pre-provisioned mail accounts put no limits on the "From:" mail header field, so that mail from the remote parts of the field trial could appear to the recipient(s) as being from the user's standard mail account. In other words, our email service did not insist that the "From:" field in the mail message was the same as the user's pre-provisioned mail account address, which is an anti-spam check many mail services enforce at mail submission time.

For the 2010 trial, we do plan to provide mail account provisioning so that users who chance upon the trial can setup new accounts, even from the remote locations. The 2009 trial was designed with this in mind, (for example, our message store replication scheme envisages multiple-village support), even though such facilities were not exercised during the trial.

Postfix was configured as the mail Message Transfer Agent (MTA) on each village DTN router which then took any email destined for outside the "village" (mail-domain !=village.n4c.eu) and transmitted those in a bundle (one mail message per bundle) to the bastion mail transfer agent (MTA) on basil). Upon reaching basil, email was extracted from the bundle and forwarded via Postfix to the Internet via the N4C mail email server at n4c.eu. This link from basil was via yet another VPN to the main n4c.eu domain's mail server. This last aspect is important to consider since the n4c.eu domain's mail servers are set up so its mail is generally delivered by other Internet mail servers. A naive set up could result in emitted mail being treated as spam, if the mail server for example did not publish Sender Policy Framework (SPF) or MX records in the Domain Name System (DNS).

The village DTN router kept a record of which bundle ID carried which message ID mail header field. Each mail-bundle sent required an end-to-end bundle status report ("delivery report") from the bastion MTA. Such acknowledgements are part of the BP, though the *dtnN4Crecv* had to extract the original bundle-ID from the payload of the bundle status report. When this acknowledgement was received, the email was removed from the village DTN router's outbound mail queue. After 5 days, if a status report was not received for that bundle ID, a "potential non-delivery notification" was sent to the originator.

Any outbound "local" email (mail-domain = village.n4c.eu) was simply delivered by Postfix to the local Dovecot IMAP message store (MS) and no bundles were directly created for these mails, though since such messages were delivered to the MS, they were synchronized via the BP as described below.

Incoming email from the Internet was received by a Postfix MTA on basil and delivered to the MS, another Dovecot IMAP server, on basil. The MS on basil and the village DTN router(s) were clones, with basil as the master and the MS instance(s) on village DTN routers as replicas.

A daily synchronising job packaged up all new and modified email files in the MS on basil into a tar file and transmitted this file in a bundle to the village DTN router. This packaging process made use of the fact that the MS file store is based on the Maildir file format, allowing the synchronisation process to easily handle cases when a mail is read, deleted etc. When this bundle was received a corresponding sync process on the village DTN router inserted these new emails into the local replica of the MS, where they were then available for client MUAs.

In order to ensure any emails sent locally on the village DTN router are available on basil, a similar sync process happened in the opposite direction.

The synchronisation process worked in delivering emails newly created on village DTN routers, but it did not track the status of emails. This is planned to be fixed for 2010.

Emails created in an MS that uses Maildir format include the name of the node creating the message, so it was easy to identify only those messages created on "this" MS instance for synchronising with "that" MS instance. Inserting such message files into the relevant user directories caused Dovecot to automatically update its internal state so no manipulation was required.

Figures 12 and 13 show the mail setup in the village and on the bastion mail host (basil).

I. Extensibility

The design documented here is, as stated, intended for use in our larger 2010 trial. That trial will involve four village DTN routers, parallel data mules (two helicopter mounted WRAPs and two EEE PCs) and two gateway sites (Ritsem as here, and one other) and will run for approximately six weeks. While some minor changes and enhancements are planned for

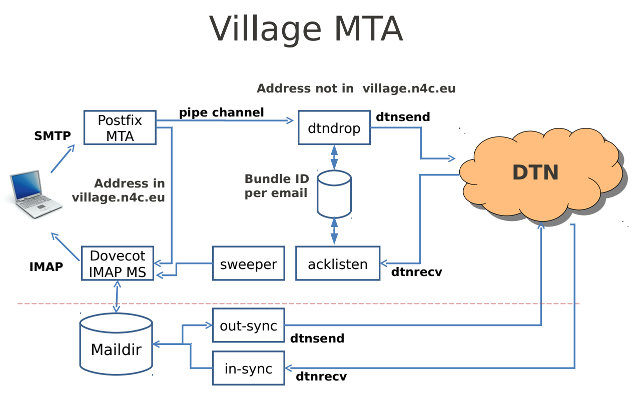


Fig. 12. Email configuration at Village. Standard mail servers (postfix and dovecot) provide service to mail clients. New non-local messages (with addresses not in the village.n4c.eu domain) are sent (by postfix) via the DTN to basil requesting bundle delivery reports; local messages are delivered to the message store (dovecot). The message store is periodically read (by the out-sync process) and changes are sent via DTN to the primary message store on basil. Similarly, periodic message store changes are received from basil (by the in-sync process) and message store content is synchronised. The "sweeper" process handles the case when no bundle delivery report is received back for a message sent - it sends the user a "possible non-delivery report" email message.

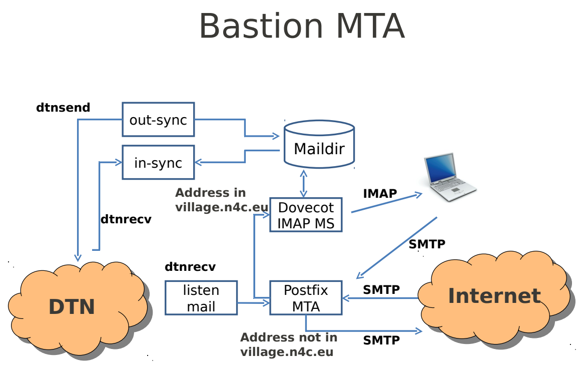


Fig. 13. Email configuration at basil. This is the inverse of the setup in Figure 12, where postfix and dovecot interface to standard Internet mail.

the 2010 trial, all of the core components described here are designed and tested for use in that larger scale trial.

In fact, there are no hard limits inherent in this design that would constrain its use to such trials involving ten's of nodes. The main feature missing were one to wish to carry out say a trial with hundreds or thousands of nodes would be the lack of a way to (securely) distribute configuration information to DTN nodes, via the DTN. In other words, we do not yet have a scalable solution for network management. DTN network management is a subject that is only now starting to be studied within the DTN research community and so is not yet ready for the kind of real-world trial described here.

The system described here could also be used in other kinds of DTN trial, for example, we have considered using our village DTN router as a "master" node in a DTN sensor network during the design process. Essentially, any DTN involving "luggable" solar-powered DTN nodes could make use of the hardware and software described here and we do plan to re-use these sytems for other future trials. Since the core of the village DTN router is a standard Linux environment, any DTN-enabled application that can handle disruption (at the application layer) and properly handles power cycles should be able to be used with our device. As part of the 2009 trial, we also acted as a plain DTN router for other ("nomadic," see below) applications running on handheld devices, so the village DTN router has also been demonstrated to work for other applications not developed by us, nor running on our equipment.

Lastly, there appear to be a range of other use-cases that mirror the requirements of the Saami herders, for example, in developing regions, in (some) emergency situations and in sensor networking as already mentioned. We believe that our implementation of the design documented here could be used in many such cases. Of course, were we or someone else to develop another implementation in future, some details of the selected hardware and software would be changed, but, we believe, the general design would remain the same, and many of the lessons learned documented here would remain applicable.

V. Results

In this section we present the results from our summer 2009 trial in Laponia. The main questions that we would like to answer are:

1) How many users used the system?
2) How many messages or transactions were attempted?
3) What were the message delivery ratios and latencies?
4) How many bundles were transferred?
5) How were contacts between DTN nodes distributed?
6) How did the system perform as a whole?

First however, we describe the data that was collected, and how that was processed. In each sub-section we present a summary of our relevant conclusions and/or lessons-learned.

A. *Test Timing*

The results presented here reflect data gathered between our arrival in Ritsem, (approx. 2009-08-07T17:00:00 UTC) and the time when we turned off the systems in Saltoluotka (approx. 2009-08-14T00:30:00 UTC) which encompasses three different setups, detailed in Table I.

In the first phase, equipment was assembled and tested in Ritsem, which involved setting up the full test topology and end-to-end testing, but with low latencies. In the second phase, the village DTN router was moved to Staloluotka, set up and re-tested by experimenters. Two days later the ExtremeCom hikers arrived in Staloluotka and made use of the system. In the third phase, the same village DTN router was moved to Saltoluokta, where the workshop (presentations) portion of the ExtremeCom event was held, and the system was restarted,

TABLE I
TIMING OF 2009 TRIAL (UTC)

	Location	Start	End	Duration
1	Ritsem	2009-08-07T17:00	2009-08-09T07:00	38h
2	Staloluotka	2009-08-09T16:00	2009-08-12T09:00	65h
3	Saltoluotka	2009-08-12T14:00	2009-08-14T00:30	34.5h
	Overall	2009-08-07T17:00	2009-08-14T00:30	151.5h

TABLE II
NODES AND ROLES IN 2009 LOGS

Short name	Role	Comment
dtnrouter-11-10	router	The main router used
dtnrouter-11-20	router	A backup/test router
dtnmule-2-10	data-mule	The Asus EEE PC mule
dtngateway-2-200	gateway	The gateway at Ritsem
dtngateway-1-102	basil	The well-connected endpoint

mainly in order to “finish” in-progress transactions. (Since the workshop location in Saltoluotka provided Internet access, participants’ interest in DTN was understandably diminished.)

Summary: Field trials like this will involve uncertainty in timing, for example, activities may be delayed by days for logistical reasons, so the trial period will only be precisely identified in retrospect. Experimenters should plan for this and need to ensure that reliable correlation of logs from multiple nodes is possible (e.g. by having sufficiently accurate clocks).

There will almost certainly be an extended setup period, as was the case here, and especially in a first outing, one should expect that to involve debugging and plan accordingly.

With a high-latency DTN, there will also be some outstanding transactions at the end of the trial period. Experimenters should be aware of this when generating results.

B. Data Collection and Preparation

All system, application and DTN logs generated during the test were preserved, as was all configuration information (e.g. the “/etc” directories) for all nodes. Various of these logs include sensitive or personally identifying information and so cannot be made public, however, we have prepared a number of derived log files, based on these files, and those derived logs are available on our web site - see Section VII for details.

Shorthand names for the nodes involved in data collection were as shown in Table II. The backup router was only powered on for a short test during the trial but routes were in place to use it, should that have been required, so it shows up in some logs since bundles were sent to both routers. (This increased the resiliency of our overall system but also generated more bundle traffic which was a secondary goal of the trial.)

The syslog files from each of the nodes were used to derive the number of DHCP clients attaching and the overall timing of the trial.

All DTN2 daemons used recorded logs in *info* mode. While these logs are less verbose than those produced using *debug* mode they do still contain a lot of information that is unnecessary for results-analysis, so scripts were developed to filter and translate these logs into a more useful format. These scripts used some simple “C” code, but mostly grep and awk to parse and analyse DTN2 logs into a more easily processed comma-separated value (CSV) format. Additional grep and awk based scripts produced input suitable for gnuplot, which was used to generate most of the graphs below. (Once again, all those scripts are available - see section VII for details.)

For our summer 2010 trial we plan to add new *info* logging calls to DTN2 to make this step easier and to also include a global bundle identifier to enable tracing the same bundle across many nodes - for the 2009 trial DTN2 logs on intermediate nodes did not contain the bundle identifiers (basically the creation time) which caused quite an amount of work to properly correlate the log entries on the various nodes. Luckily, timing and payload sizes proved sufficient to allow us to correlate most log entries, but this was only possible due to the limited scale of the 2009 trial.

Bundle duplication is an issue that became apparent during analysis of the logs. Duplicates were received on all nodes but subsequently recognised and deleted by the receiving daemon. Most of these duplicates arose from the routing setup which was designed to prioritise delivery at the expense of bandwidth. In effect, a single bundle was likely to be sent twice if both the opportunistic and on-demand contacts between the nodes in question were active. Node reboots following sudden power off events may also have caused resending of bundles residing in that node’s data store where state stored in RAM was lost.

Summary: Logging that has been developed for laboratory testing and debugging during code development will probably not contain the right data to allow correlation across nodes for extended periods of time. Experimenters should examine logs before starting and consider whether the information required for later analysis is present, and how to extract that. Using a file that is some months old will ensure that the experimenter doesn’t already know what to expect when she opens the file. Had we done this prior to our trial, we would have saved some person-weeks of effort in log analysis.

Experimenters should be aware that logs will contain personally identifying information, so anonymisation of MAC and email addresses, URLs fetched, etc. will be required.

The development of a de-facto standard for logging events in a DTN would be beneficial, since it would allow experimenters to develop tools that could take such trace files as input.

C. Power Cycles

While in Ritsem and Saltoluotka the router was operated from mains power. In Staloluotka, the router successfully operated from solar power, though it also experienced ovenight power shortages from which it recovered once sufficient sunlight had been captured.

Figure 14 shows the on/off power cycles for the nodes that were not powered for the full duration of the trial. (“b/router” represents our backup router.) Where the line is high, the device was powered on, where low it had been shutdown, or (when the line is slightly lower) been suddenly powered off.

The data for this figure was derived from the syslog files on the various nodes. This information is also overlaid on some later figures and helps clarify, for example, the pattern of contacts seen.

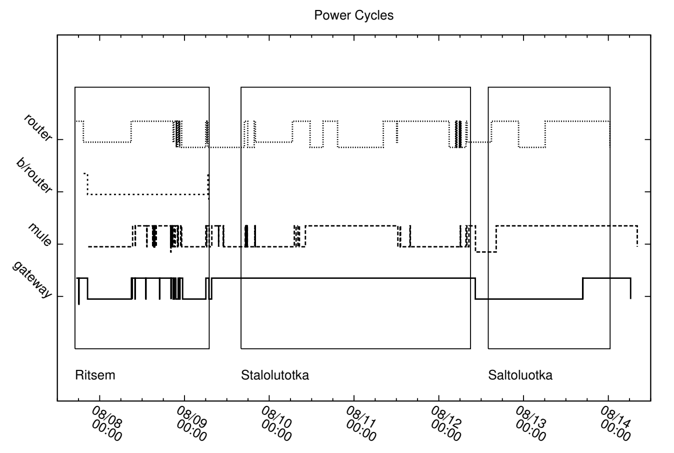


Fig. 14. Power cycles. When the line is high, the device was on. When low, it had shutdown. (And when slightly lower, it had experienced a hard-power-off, e.g. in the case of the mule in the gap between the Staloluotka and Saltoluotka phases of the trial). A sequence of power on/off events for the router can be seen towards the end of the Stalolutoka phase - this was due to insufficient battery power overnight. (And led us to plan improved power management for our 2010 trial.) See Table III for additional detail. This, and subsequent, figures also highlight the different trial-phases, and the horizontal axis represents time.

TABLE III
POWER CYCLE STATISTICS (SECONDS)

Node	Phase	Total On	Cycles	Average On	Percent On
backup	Ritsem	4960	2	2480	3.63
router	Stalo	0	0	0	0
	Salto	0	0	0	0
	Overall	4960	2	2480	0.85
gateway	Ritsem	62372	12	5197.67	45.59
	Stalo	234000	0	234000	100
	Salto	48713	2	24356.5	39.22
	Overall	379848	15	25323.2	64.73
mule	Ritsem	50274	19	2646	36.75
	Stalo	108177	16	6761.06	46.23
	Salto	143760	0	143760	115.75
	Overall	313995	39	8051.15	53.51
router	Ritsem	58471	10	5847.1	42.74
	Stalo	105361	14	7525.79	45.03
	Salto	93655	2	46827.5	75.41
	Overall	257487	26	9903.35	43.88

In total there were 82 power cycles seen and Table III gives some basic statistics for each node, with the total and average given in seconds. Note that the total number of cycles per node include power cycles that happened during, or spanned, the gaps between the three phases of the trial. Where a power cycle extended beyond a period (e.g. in the case of the mule at Saltoluokta), its percentage-on value can be greater than 100% in Table III.

Of the 82 cycles, 22 ended with a sudden poweroff event, one each for the backuprouter and gateway, two for the data-mule and 18 for the router. The router poweroff events were due to the lack of power management previously noted and is being addressed for our planned 2010 trial. While such events are undesirable (since they can result in corrupted filesystems), one should plan for their occurrence, since they will happen in any real world trial. Roughly 50% of these router shutdowns happened overnight on August 11/12th due to battery depletion, whereas most of the rest were experimenters simply turning off the power.

TABLE IV
"BOTH ON" DURATIONS (SECONDS)

Node1	Node2	Ritsem	Stalo	Salto	Overall
mule	backuprouter	979	0	0	979
mule	gateway	46228	103129	27803	214715
mule	router	46916	40810	88831	176557

TABLE V
MULE/CORRESPONDENT"BOTH ON" DURATIONS (SECONDS)

Correspondent	Location	Time
router	Staloloutka	14334
gateway	Ritsem	78321

In addition to the basic power cycles of each node, we are also interested in the durations for which pairs of nodes that form DTN contacts are both turned on. The reason of course is that unless both are on, no communication is possible. Table IV gives these figures for the relevant nodes. These figures are later used to calculate how efficiently we are using this available time.

Finally, since the mule actually moved between Staloluotka and Ritsem during the "Staloloutka" phase of the trial, we need to separate out the overlapping durations further, so that the times when the mule and the router or gateway were both turned on and both in the same location can be used. Based on the earliest and latest contacts seen, and discounting the gaps representing helicopter rides, Table V shows the times during which the mule and its corresponding nodes were both powered on and in proximity.

1) Summary: Analysis of the power cycles seen in our trial indicated a need for more active power management, which we have addressed for our 2010 trial.

Experimenters should ensure they can detect power on/off events from logs. Sudden losses of power can be problematic to detect, and precisely time, but system logs generally contain predictable (though not always identical) strings at the subsequent power-on, such that the time of the previous log entry gives the best indicator of the time of the event.

D. Basic Activity Levels

The overall population of potential users during the trial was approximately thirty-five, including nine N4C participants, of whom three were involved in setting up and running these tests.

Thirty-six unique MAC addressess were served with IP addresses from the village DTN router via DHCP during the trial. (Some participants had more than one WiFi-enabled device.) Figure 15 shows how these were distributed over time and over each of the phases of the trial.

The histogram in Figure 15 shows the count of unique MACs seen in each 4 hour period. This figure was generated from the syslog on the village DTN router, searching for lines matching "DHCPACK on" (when the router received a DHCPACK from the client) and recording the time and replacing the actual MAC address with an index value.

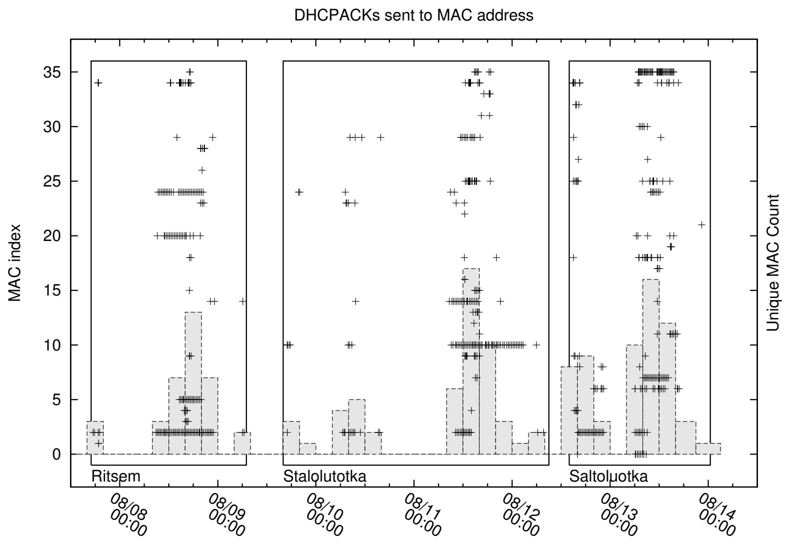


Fig. 15. DHCPACKs from router. Thirty-six unique MAC addresses received DHCP acknowledgements during the trial. The checkmarks in the figure show the times of these events for each MAC, with the vertical scale being the unique MACs. The histogram shows the number of unique MAC addresses seen in each four-hour period. For the histogram, the same vertical axis also shows the number seen, since the absolute numbers involved allow this.)

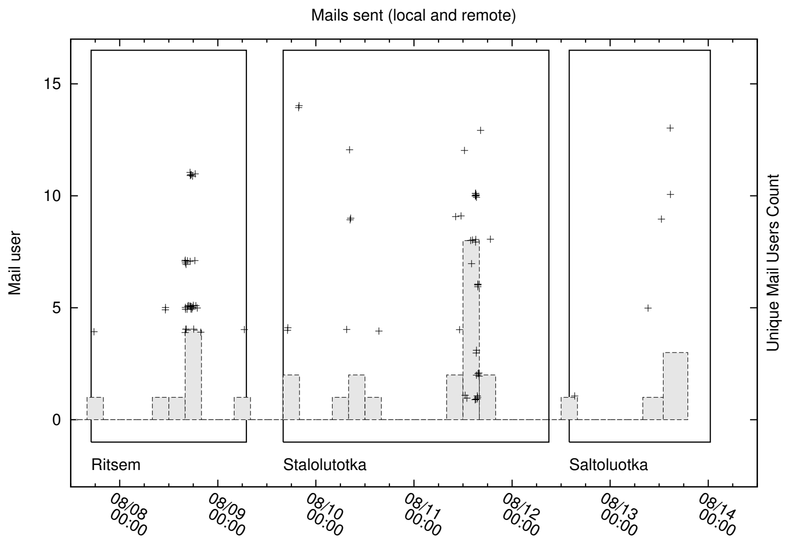


Fig. 16. Mails sent via router. Fourteen users used the pre-provisioned mail accounts. The checkmarks in the figure show the times at which mail messages for each user (vertical axis) were submitted at the village DTN router. The histogram shows the number of distinct users who sent mail in each 4-hour period. In this and subsequent figures, some of the checkmarks are slightly scattered for clarity.

Aside from the number of unique MAC addresses this figure also gives a rough idea of the patterns of basic connectivity in each phase of the trial.

Fourteen of the pre-provisioned mail accounts were used to send mail during the test and in total 87 mail messages were sent (some were purely local), of which 44 were sent by personnel working on the N4C project, two were sent by a tourist that was curious about the system and the remaining 41 were sent by other participants in the ExtremeCom event. Figure 16 shows a view of this outound mail traffic (horizontal lines correspond to each user, crosses have been slightly scattered vertically).

Some 286 unique mail messages were received by 12 ExtremeCom participants as shown in Figure 17. Since mail arrived sporadically, crosses in this figure are again scattered slightly to avoid too many overlaps. Figure 17 also has a slightly different timeline, since some ExtremeCom partipants had already setup mail forwarding before the equipment arrived at Ritsem. For the period shown, the equipment was being integrated in Lulea Technology University who are the co-ordinators of the N4C project.

Figure 18 shows how many messages each user sent and received. From this we can conclude that users that had setup forwarding did receive substantial numbers of real messages

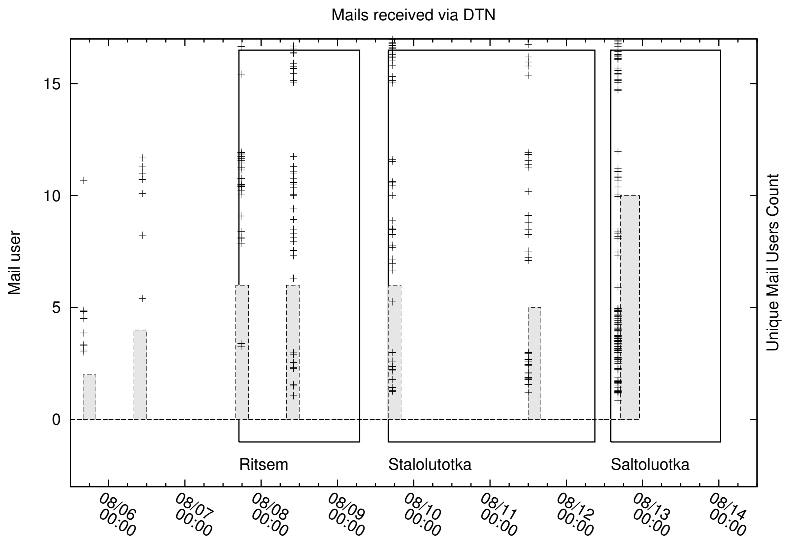


Fig. 17. Mails received at router. Twelve users received mail messages during the trial. The checkmarks in the figure show the times at which mail messages arrived for each user (vertical axis, scattered) at the village DTN router. The histogram shows the number of users who received messages in each 4-hour period. Note that some users had set up mail forwarding prior to the trial starting, so this figure has a longer time-span than others.

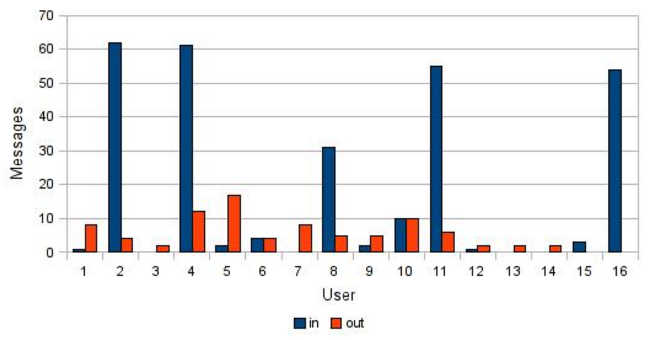


Fig. 18. Mails sent and received per user. The figure shows the number of individual mail messages sent and received by each of the 16 users that either sent or received a message.

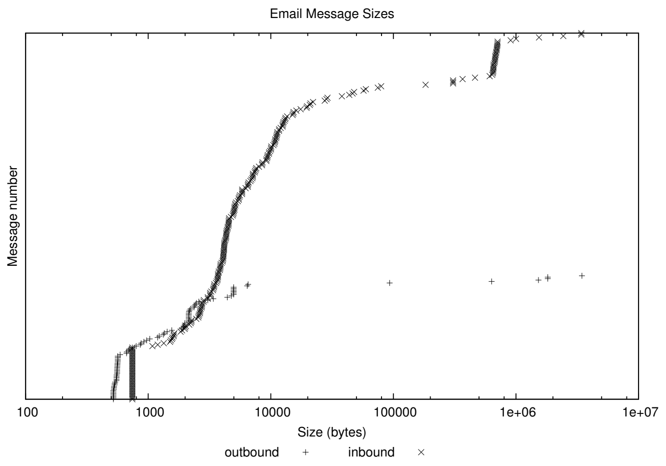


Fig. 19. Email message sizes. The figure shows the sizes of mail messages sent and received during the trial. Both sets were simply sorted by size for the figure and numbered. The distribution of outbound message sizes is as expected, whereas the inbound messages appear larger than for other mail corpuses. See Table VI and the text for details.

during the trial. While the distribution of activity is interesting, it is hard to know whether sound conclusions can be drawn from such a small sample, especially since the ExtremeCom users are not typical of those we expect to encounter in 2010.

Mail message sizes in the trial may be somewhat artificial given that experimenters were testing the system and so re-transmitted a number of messages at different times (only 62 of the 87 messages have different sizes). As usual, there are also outliers, which we believe represent messages containing images. Figure 19 shows the distribution of outbound and inbound message sizes. Table VI shows message size statistics and those reported in a study of email traffic by Shah and Noble [16] and those from the Enron dataset. [17] As can be seen our outbound distribution is not that dissimilar even though our message corpus is orders of magnitude smaller than the others shown. However, our inbound numbers are quite different. At present, we have no explaination for this discrepency.

Browsers from 20 different IP addresses connected to the web service as shown in Figure 20, which was derived from the Apache access log on the router. Twenty-one separate requests were encapsulated in bundles, with 20 responses being returned. However, only 9 of these transactions were started whilst the equipment was in Staloluotka and of those,

TABLE VI
OUTBOUND EMAIL MESSAGE SIZE STATISTICS

Statistic	Outbound	Inbound	Shah & Noble	Enron
Average	106.4 KB	126.3 KB	N/A	N/A
Stdev	473.8 KB	392.3 KB	N/A	N/A
25th Percentile	559 B	2718 B	758 B	919 B
Median	1194 B	4810 B	1901 B	1548 B
75th Percentile	2382 B	12.23 KB	5092 B	2719 B

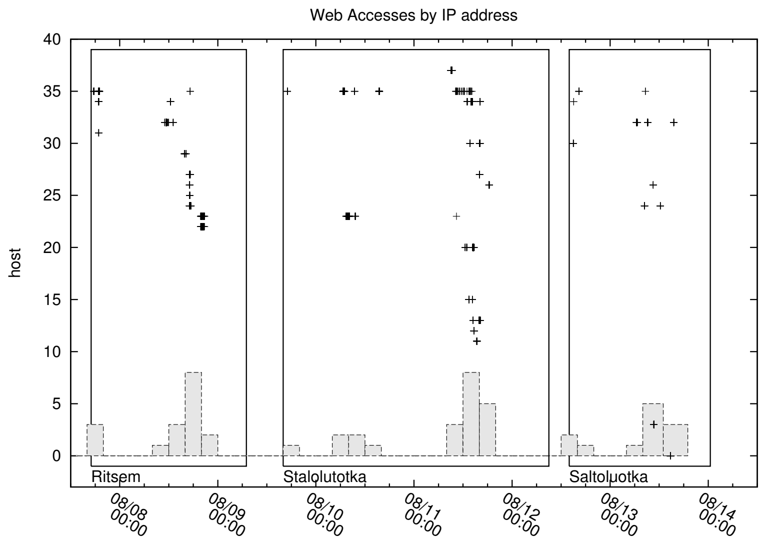


Fig. 20. Web accesses on router. The figure shows the time of each HTTP request (from the Apache log) for each requesting IP address (vertical axis). Twenty of the 36 IP addresses issued via DHCP made HTTP requests, however, the figure includes users checking if responses had been received so only reflects web browser activity levels and not DTN activity. As before, the figure include a histogram of the number of requests seen in each 4 hour period. The hosts here are the same as in Figure 15, though the ordering here is different.

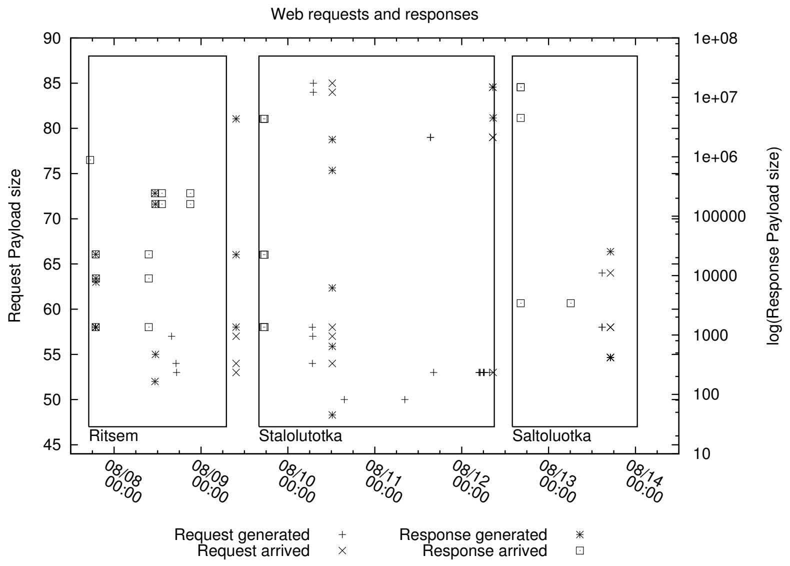


Fig. 21. DTN encapsulated web. The figure shows HTTP requests and responses that were sent via the DTN. Request-size versus generation time at the router and arrival time at basil are shown (left-hand linear size scale). The much larger response sizes are plotted against generation time at basil and arrival time at the router (logarithmic scale on right). The time between generation and arrival can be seen in the figure, since both generation and arrival times are at the same height in the figure.

only 3 were initiated after the ExtremeCom hikers had arrived, so there is no real data as to the useability of our web service from the 2009 trial.

However, the web service components did operate correctly, as can be seen from Figure 21 which plots the web transaction bundle payload size against submission-time and arrival time for requests and responses. In the figure, one can see that requests made roughly correlate with responses and one gets a feel for the latencies involved.

Summary: While all of the numbers are modest, each of the capabilities of the system were exercised and behaved as expected, processing some real traffic on behalf of ExtremeCom workshop participants, as well as traffic generated by N4C project staff.

In addition to system and DTN logs, experimenters should ensure they also preserve application logs, e.g. from web and mail servers. Numbers from those logs both help to understand lower layer DTN logging and also act as a check that statistics generated from DTN logs are accurate. This can sometimes be non-trivial, if bundles are sent more than once, or are fragmented, or if bundle endpoint identifiers do not map to application layer identities in an ovbious manner.

Application logs also give a more accurate understanding of user-level interactions, since DTN logs are liable to contain many protocol artefacts (e.g. "ping" bundles or the equivalent).

E. Contacts

A "contact" is a DTN concept referring to the duration during which two DTN nodes can communicate when a connection-oriented convergence layer is in use. Since DTN nodes can only actually send bundles during a contact, it is interesting to see what contacts occurred during the trial.

From our log files, we extracted the details of contacts coming up and going down over the duration of the trial. Figure 22 gives an overview of these. All of the DTN2 links used the TCP convergence layer. The figure also shows the "On" and "Off" power cycles of the relevant nodes from Figure 14, to illustrate the boundaries within which the DTN daemon can and cannot establish contacts. Each contact is represented as a rectangle in the figure, with the width representing the duration of the contact and with various vertical offsets (not really visible in the printed figure, but visible with gnuplot based on the supplementary materials) that represent the type of contact ("ONDEMAND" or "OPPORTUNISTIC") and which node initiated the contact.

Figure 22 was automatically produced from the logs, with one interesting exception, that would probably not have been noticed without the visualisation. The data-mule's NTP daemon moved the time on that node backwards by 1346.44 seconds (roughly 22.5 minutes) at (real time) 2009-08-08 23:22:57 UTC resulting in our logs showing a number of contacts with the router around that time ending before they had started. To make matters more interesting the mule was shutdown before its local clock had caught up with 2009-08-08 23:22:57 UTC and as a result the visualisation scripts interpreted those contacts as staying up until the data-mule had next booted up which was at 2009-08-09 06:02:12 UTC. Those contacts would therefore have been interpreted as lasting for over 6 hours at a time when in fact both mule and router were powered off and could thus skew later generation of statistics. In order to avoid this, we manually "moved" the mule shutdown time forward by 22.5 minutes, so that the actual contact durations were used in later calculations.[28]

While Figure 22 is too coarse grained to be really useful at this resolution, one can see the overall pattern of basil $\leftrightarrow$ gateway $\leftrightarrow$ mule $\leftrightarrow$ router contacts, and the lack of end-to-end connectivity whilst the router was in Staloluotka, which is of course the main motivator for the use of DTN protocols in this situation.

Interestingly, there are more than 2500 contacts represented in Figure 22, which indicates some ineffiency in how DTN2 manages TCP convergencce layer connectivity.

Our configuration also called for both "ONDEMAND" (opened as necessary) and "OPPORTUNISTIC" (probed for whenever non-available) contacts between each pair of relevant nodes. This was mainly a conservative design approach, justified because there was an extended period during which the main designer of the DTN2 setup would not be able to control the node configurations. The "double" approach to using the TCP CL meant that even if others broke the static routing configuration of the ONDEMAND links, the OPPORTUNISTIC links had a good chance of allowing bundles to flow.

Figure 22 also shows a few contacts from the village DTN router to other DTN nodes that were used by N4C partners during our trial, in all phases of the trial. Those are the contact lines extending upwards from the "router" line. Second from the bottom of the figure one can also see contacts to "Folly" which were the other ends of these same DTN paths. At the bottom of the figure one can also see contacts between "basil" and "BBN" which represent contacts between our main Internet-connected gateway and other nodes on the DTN-Bone[29].

We now consider how our contacts map to the possible time during which pairs of relevant nodes could have potentially been in contact. For this purpose we will ignore the "outlier" nodes, that is, the DTN aware users and the connections between "basil" and "Folly" and "BBN." Tables VII to X show the duration of actual contacts versus the time available for each of the phases of our trial. The possible times are derived from the power figures given in Section V-C. For example, the

[28]The file 'gnuplots/contacts/cont.num.moved' contains the manually modified lines with identifying comments. See Section VII for details on how to access this file.

[29]http://www.dtnrg.org/wiki/DtnBone

TABLE VII
OVERALL CONTACT UTILISATION (SECONDS)

Node 1	Node 2	Possible	Actual	Percentage
basil	gateway	379848	313204	82.45
gateway	mule	189907	155353	82.80
mule	router	150081	105186	70.08
mule	backuprouter	4960	2850	57.45

TABLE VIII
RITSEM CONTACT UTILISATION (SECONDS)

Node 1	Node 2	Possible	Actual	Percentage
basil	gateway	62372	44627	71.54
gateway	mule	46228	22036	47.66
mule	router	46916	30110	64.17
mule	backuprouter	4960	2850	57.45

TABLE IX
STALOLUOTKA CONTACT UTILISATION (SECONDS)

Node 1	Node 2	Possible	Actual	Percentage
basil	gateway	234000	186196	79.57
gateway	mule	78321	72998	93.20
mule	router	14334	12152	84.77
mule	backuprouter	0	0	0.00

TABLE X
SALTOLUOKTA CONTACT UTILISATION (SECONDS)

Node 1	Node 2	Possible	Actual	Percentage
basil	gateway	48713	27353	56.15
gateway	mule	27803	27773	99.89
mule	router	88831	63699	71.70
mule	backuprouter	0	0	0.00

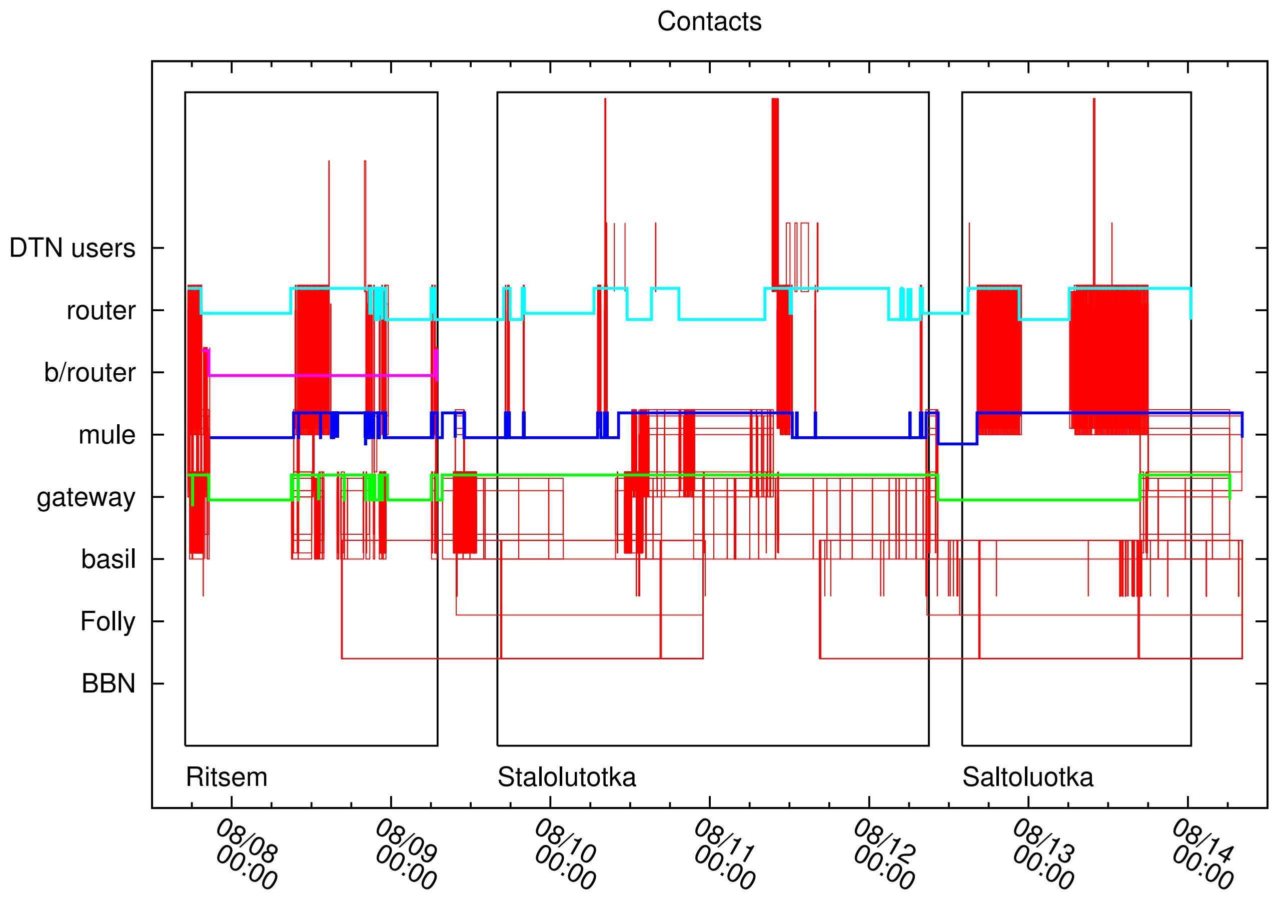


Fig. 22. Contacts during the trial. There were more than 2500 contacts seen during the trial. The figure plots each contact as a rectangle with the horizontal line matching the start and end of the contact and the vertical line matching the two peers involved. Contact peers are named on the vertical axis. At the top, there were three different "DTN user" devices that contacted the village router. At the bottom, there were contacts between basil and Folly, for the "nomadic" mail experiment, and there were regular DTNbone contacts with BBN. Most contacts are too short to be individually visible at this scale, however the supplementary materials include a gnuplot version of this figure that can be explored in detail. The on/off power states of the various nodes are overlaid on the contact rectangles. One can see from the figure that contacts fall within periods when the relevant nodes are powered on and in proximity so this figure gives a good overview of the potential for, and relative complexity of, end-to-end communication in this particular DTN.

gateway $\leftrightarrow$ mule possible time shown in Table VII consists of the times both were powered up, and in proximity.

Summary: The main thing to note here is that the utilisation of potential contact durations was quite efficient, over 93% for mule-gateway contacts and over 84% for mule-router contacts whilst the router was in Staloluotka.

Comparison of the contact information with the power cycles of the various nodes was also a very useful check - with so much data, the more such independent checks one has the better, and the visualisation of contacts shown in Figure 22 was essential in detecting some inconsistancies that would otherwise have affected our results.

However, the number of contacts (more than 2500) was excessive - the TCP convergence layer, as configured, was making and breaking TCP sessions far too often. This both complicated analysis and must represent an inefficiency in terms of reducing overall goodput, and warrants futher examination.

The gateway to basil link (via GSM/EDGE) proved to be much more highly disrupted than we had envisaged. The main reason for this (we suspect) was simple network capacity in the GSM cell at Ritsem - the connection would often break around the time a helicopter landed and our guess is that the embarking and disembarking passengers made phone calls that got higher priority than our GSM data traffic. As this was our "backhaul" to the Internet from the field trial, we were initially quite concerned about this. However, the use of the BP on this link essentially saved the day - had we put the application layer services (e.g. mail relay and web crawling) in Ritsem, rather than in Dublin, then the trial would probably have failed. Using the BP, we simply experienced additional delay.

F. Bundle Flows

As we have seen, a modest amount of application layer traffic was successfully transferred during the trial. Each application layer protocol data unit (in our case, an email message or synchronisation tarball, or a web request or response) is sent as the payload of a bundle. That bundle is forwarded between

TABLE XI
APPLICATION LAYER BUNDLES CREATED

Purpose	From/To	Bundles	Total size
MS Sync	basil/router	14	123064320 B
MS Sync	router/basil	3	286720 B
SMTP encap	router/basil	72	33018032 B
Nomadic	router/basil	48	182749 B
Nomadic	basil/router	5	6680 B
HTML request	router/basil	21	1287 B
HTML response	basil/router	20	41581525 B

nodes, causing some additional administrative bundles to be sent. In this section, we describe how the application traffic maps to the bundle traffic seen. (For a full description of how the bundle protocols works, see RFC 5050. [6])

One difficulty faced here is that DTN2 "info" level logging does not log a unique identifier for bundles.[30] The result is that we cannot easily or deterministically trace all traffic through the DTN. However, our trial was sufficiently small in terms of possible paths and simple traffic numbers that we can make some useful measurements nonetheless.

There were 72 outbound (from the "village") email encapsulating bundles, each of which was contained one email message. This totalled approximately 33MB of bundle payload traffic. (Note that the mail sender process sent duplicate bundles when a single message had multiple recipients, so these bundles represent 62 different mail messages.)

Inbound (from the Internet) mails were received at basil and message store synchronisation bundles were transmitted once per day (at 10am UTC) to the router node and, in parallel, to the backuprouter node. There were 14 such bundles sent from basil, 7 to each of the routers, each of which contained a compressed archive (a "tarball") synchronising the message stores. This totalled approximatedly 123MB of bundle payload traffic.

Three such mail synchronisation bundles were sent from the router node to basil representing changes seen on the router's message store clone, e.g. reading/deleting messages. While these were also scheduled to be sent each day at noon UTC, bundles were only actually sent on the 8th, 11th and 13th. (When there were no changes in the router's message store, no bundles were sent.) These bundles carried approximately 286KB of application layer traffic.

In addition to our own web traffic there was a separate "nomadic" mail experiment carried out that resulted in 48 more small bundles (totalling approx 182KB) being sent from the router to basil, and 5 bundles (totalling 6KB) being sent from basil to the village router.

Web requests contributed 21 bundles (totalling 1287 payload bytes) sent from the village, and 20 response bundlles sent from "basil" (totalling 41.5MB).

Table XI summarises the application layer bundles created.

In addition to our web and mail applications there were two other sources for bundle traffic.

The first additional source of bundles consisted of diagnostic bundles, that is, "pings" and "traceoutes" sent during the trial. There were 125 unique source EIDs used in such bundles.

[30]A defect we plan to fix prior to the 2010 trial.

TABLE XII
BUNDLE TRAFFIC PER NODE

Node	rxbytes	rxbundles	txbytes	txbundles
basil	92504081	1580	228584826	993
gateway	303248030	1858	330954329	1233
mule	311971711	2748	238988113	1802
backuprouter	58404017	744	31460672	451
router	168329157	2730	146656587	1054

TABLE XIII
BUNDLE TRAFFIC PER LINK (MB)

From/To	basil	gateway	mule	b/router	router
basil	-	189.68	17.07	0.01	2.85
gateway	37.85	-	256.54	0.01	12.32
mule	1.73	90.00	-	14.23	104.46
b/router	0.20	16.90	7.05	-	5.78
router	1.42	30.32	98.51	0.01	-

Each run of the dtnping utility generates a numbered EID as its source (so that responses can be correlated) and then periodically sends a bundle to the nominated destination. The default period is 1 second, and the default bundle lifetime is 30 seconds, so in a scenario such as ours many such bundles will expire before reaching their destination.

When a ping bundle reaches its destination it generates one response bundle. In contrast, a traceroute bundle generates multiple responses (one from each node on the path), and 120 unique destination EIDs were used for such bundles. Note that the unique EIDs referred to here do not absolutely rule out many bundles having been sent to or from those endpoints, but this is considered unlikely given the circumstances of the trial, so we treat each as if only one transmission had occurred.

Finally, each of the nodes on the path from the router to basil was configured to accept custody. This means that for each unique bundle transmission from one node to another there will (nominally) be a corresponding bundle returned that contains a custody acceptance administrative record. So, there should be at least twice as many bundles on each link as there are unique application layer PDUs.

Table XII shows the raw number of payload bytes sent and received by each node during the trial and Tables XIII and XIV show the traffic (and number of bundles) sent between each pair of nodes.

Summary: The bundle forwarding and routing used worked as planned and produced the expected behaviour. Examination of the bundle flows showed up some application layer bugs (e.g. the mail service sending unnecessary duplicate bundles).

The modest number of appliation layer interactions resulted in the expected number of bundles being sent and received. That application layer (payload) traffic totalled 189MB, sent

TABLE XIV
BUNDLE TRAFFIC PER LINK (BUNDLES)

From/To	basil	gateway	mule	backuprouter	router
basil	-	629	136	5	128
gateway	409	-	619	4	114
mule	34	378	-	55	1205
backuprouter	49	105	145	-	123
router	41	87	766	6	-

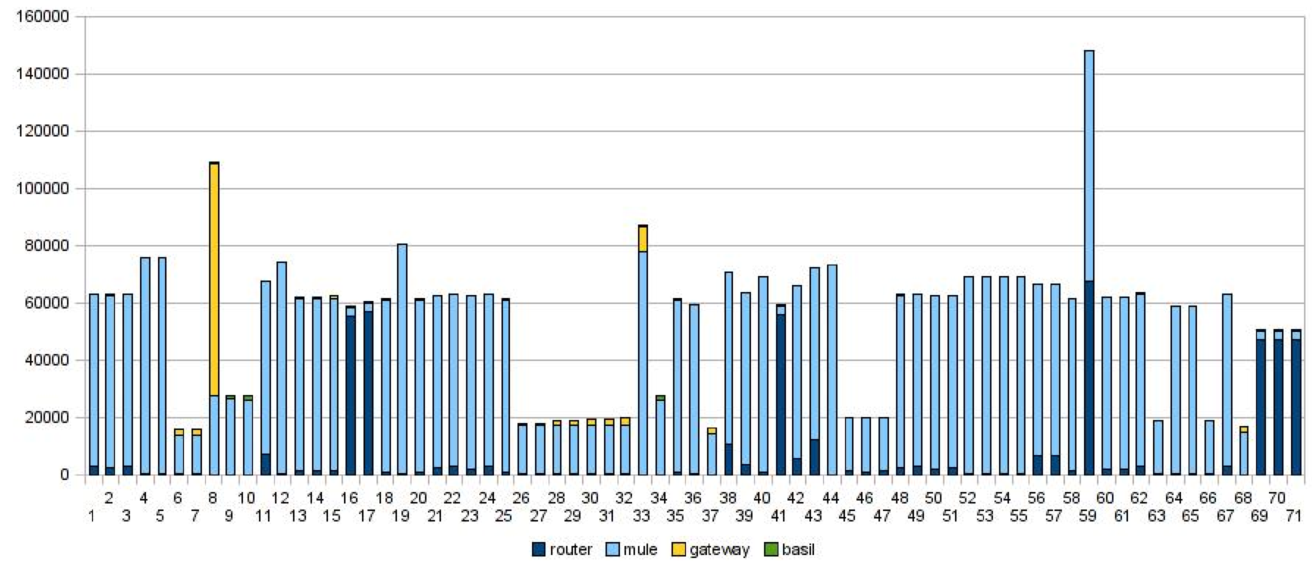


Fig. 23. Bundle queuing latencies. For the 71 bundles that were delivered end-to-end during the trial, (ordered by creation time), the figure shows the queueing time at each of the relevant nodes. In most case, queueing in the mule predominates, as expected for this setup (where the mule was powered on in Ritsem between helicopter trips.) The average latency overall was 15.12 hours, but with a large standard deviation (6.87 hours).

TABLE XV
StaloLuotka Delivery

Class	Number	Total	Min	Max	Average
All	79	104409127	45	14845799	1321634
Delivered	71	83109381	53	14845799	1170554
Non-Delivered	8	21299746	45	14632960	2662468

in 183 bundles. However, there were over 1000 bundles sent between basil and the gateway, when administrative records, bundle duplication and diagnostic traffic were included, so there is possible scope for improvement here, e.g. if custody records could be amortized over a number of bundles. In our case however, since the actual traffic didn't approach the link capacity (see below), this inefficiency was not a problem.

Experimenters should carefully consider the utility of diagnostic traffic when in the field. In the lab, there is essentially no cost to such traffic, however, in the field, especially where latencies are high, or connections are highly disrupted, such traffic can waste significant time (e.g., as experimenters await ping responses which are in many cases not likely to arrive if the ping request has expired). Such traffic also complicates logs slightly, and we found that any such complication can be a barrier to understanding what the network is doing, or has done.

G. Delivery Ratio and Latency

To estimate the delivery ratio we consider web and mail (i.e. no pings, traceroutes or nomadic) bundles sent while experimenters were in Staloloutka and that consquently needed to make use of the data-mule and experienced high latency. There were 79 such bundles representing a total of 104MB, with a minimum size of 45 bytes and a maximum size of 14MB.

Of the 79 bundles, 8 bundles were never delivered, of which two were message store synchronisation bundles destined for the backuprouter, which was powered off. These constituted approximately 18MB of the 22MB of "lost" application layer traffic. Of the others, one was a web request from the router to basil and 5 were web responses from basil to the router. All of these bundles expired at the data mule, and details of their expiry are shown in Table XVI. Looking at the times in Table XVI one sees that the web responses were all expired at about the same time, following a long period during which the mule was in Ritsem (see Figure 22 for the mule-movement timing). This was basically a configuration error - we did not allow sufficient time for the bundle lifetime for web bundles.

There were therefore 71 of 79 bundles successfully delivered, giving a bundle delivery ratio of 90%. Table XV summaries these bundles delivery statistics.

In terms of application layer bytes delivered, and ignoring the bundles destined for the backuprouter, we could claim a delivery ratio of 97%, including those bundles brings the delivery ratio down to 80%.

Bundle latencies for the 71 successfully delivered bundles were, as might be expected, dominated by queuing time. Figure 23 shows the queueing time for each of these bundles. As can be seen queuing on the mule dominates in most cases, with queuing on the router being the next most noticeable facet. The average latency for these 71 bundles was 15.12 hours (with a standard deviation of 6.87).

In terms of direction, 28.77MB (66 bundles) of the 71 delivered bundles were sent from the router to basil, and 50.49MB (5 bundles) from basil to the router.

Note however, that there is a possibility for error here, since our logs do not contain unique bundle identifiers we have to use the (non-unique) combination of size, source and destination as a proxy for a bundle identifier. As stated already, a number of same-sized bundles were sent during the trial so

TABLE XVI
EXPIRED BUNDLE DETAILS

From	To	App.	Sent	Size	Expired Time	Time to Expiry
basil	b/router	ms	2009-08-10 10:00:02	4116480	2009-08-13 10:00:02	259200
basil	b/router	ms	2009-08-11 10:00:02	14632960	2009-08-18 14:57:43	622661
basil	router	hrep	2009-08-10 12:16:38	1952338	2009-08-11 08:29:58	72800
basil	router	hrep	2009-08-10 12:15:58	45	2009-08-11 08:29:18	72800
router	basil	hreq	2009-08-10 12:15:57	591033	2009-08-11 08:29:17	72799
basil	router	hrep	2009-08-10 12:15:58	6199	2009-08-11 08:29:18	72800
basil	router	hrep	2009-08-10 12:15:58	641	2009-08-11 08:29:18	72800
router	basil	hreq	2009-08-10 15:33:45	50	2009-08-11 15:51:10	87444

there are some duplicates which makes it impossible to be sure about the queueing time for a particular bundle. The redundant routing strategy used also accentuates this issue - while that strategy was sound from the point of view of delivery ratio, it can result in additional copies of a bundle being forwarded thus further complicating data analysis. (Of course, data analysis is quite properly a lower-priority than delivery.)

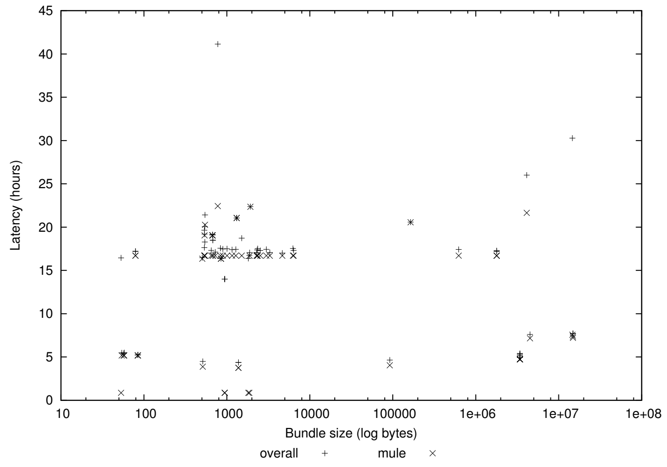


Fig. 24. Bundle size vs. latency. This figure plots bundle size against overall latency for the 71 bundles delivered end-to-end, and also bundle size versus queuing time at the data mule. One can see most overall and mule-queueing time points co-incide showing again that queueing in the mule dominated overall latency. One can also see that there is no obvious relationship between bundle size and latency.

Figure 24 compares bundle size vs. latency for our 71 bundles for both the overall latency and the date mule latency. There are two things to note: first, there is no overall dependency of latency on size, and secondly that the datapoints for the mule are almost identical with those for the overall latency.

Lastly, it is interesting to consider how much of the available bandwidth was actualy consumed, to help with further deployments. Unfortunately, however, we do not have information as to the actual bandwidth figures for the links that were used in 2009 (a defect we plan to rectify in 2010) but we can still make some estimates.

If the VPN connection from the gateway at Ritsem back to basil carried 64000 bits per second at the application layer, and the WiFi connections between the data-mule and the router and gateway carried 1 Mbps, then the traffic sent during the Staloluotka phase (50MB up and 28MB back) would have used at most less than 3% of the network capacity. Put another way, with the offered load of application traffic, the VPN link was only required to carry 1811 bits per second and the WiFi links only needed to carry 29950 bits per second, both of which are quite modest. (These figures are based on the connectivity durations given earlier.)

Summary: Essentially all application layer bundles sent were accounted for, and their handling was fully consistent with the trial design. However, due to some misconfiguration of bundle expiry we only achieved a bundle delivery ratio of 97%, even though the DTN in question was very simple and only made use of static routing.

Bundle latencies averaged 15.1 hours and were dominated by queuing time in the data mule. This was because the EEE PC mule spent significant time static in either Ritsem or Staloloutka. Had the helicopter-mounted WRAPs carried our traffic, the queueing latency would mostly have been in the DTN gateway at Ritsem or the village DTN router. Again this was exactly as expected.

The traffic levels from our 2009 trial were well below the capacity of the network, even considering all the disruptions. This suggests that our design should be able to scale sufficiently for the 2010 trial and quite some distance beyond.

VI. CONCLUSIONS

In conclusion, we have presented the design of our "village" DTN router and presented results from our trial in 2009 and outlined our plans for an extended trial in 2010.

Overall, we consider that the device and the software infrastruture worked well, to the extent that the 2009 trial exercised the components, and has allowed us to identify a number of improvements to the hardware platform, applications and the DTN stack that should help both us and other experimenters to carry our similar DTN field trials.

More broadly, the conclusions we reach that should be of interest to other experimenters are as follows:

The DTN arcitecture, the bundle protocol and the DTN2 implementation basically worked as advertised, and were rather efficient in terms of using the available contacts. However, there was so little traffic, that this conclusion is rather weak for now. We expect to strengthen this from the results of our 2010 trial. There are also areas of potential improvement highlighted earlier that should be addressed, mainly relating to the DTN2 implementation. The only bundle protocol change suggested from this trial is to consider amortizing custody acknowledgements for multiple bundles into one bundle containing a number of administrative records. However, this trial only made use of static routing, and even so, we had a number of issues with forwarding (duplication) and expiry (miscofiguration).

We believe that successful field-trials require more than one run. In our case, since the area is so seasonally constrained, that means doing trials over multiple years. In other areas, successive trials could be done over a shorter period. The reasoning here is that since real field trials involve so much logistics and experimenters only really learn about the real problems in the field, it is unlikely that results from a first outing will be that significant and additional trials will be required to get best value from the effort expended. In our case, the 2009 trial reported on here is essentially a precursor to the 2010 trial, however, it is a necessary precursor, and we feel that other experimenters would be wise to plan similarly. Having said that, the 2009 trial did produce real results that are of interest in themselves and other experimenters can use our logs in order to help them further develop other aspects of DTN.

Trials like this one can be repeated elsewhere, using this or similar technology, and we hope that other experimenters will take the opportunity to attempt more such trials and that funding agencies can see this particular trial as evidence that such efforts are a worthwhile way to develop DTN and to make progress in meeting the real requirements of real users.

VII. Resources

Supplementary information, including source data and generation scripts for all the figures and tables shown here may be found on our web site.[31] The main log file from which almost all other data is extracted is called "N4C_Summertrial_dataset.csv". Subdirectories contain scripts to extract and process the data, for example the "contacts" directory contains the scripts and other code required to generate Figure 22 which extracts the contact information from the main log file.

Initial (informal) information about our 2010 trial is now available on another web site[32]. Once the 2010 trial data has been safely recovered and analysed we plan a companion paper to this one reflecting the results of that larger trial, which involved 5 village DTN routers, 4 mules, two gateways and ran for 6 weeks, all based on the designs documented here.

Acknowledgement

This paper reflects the work of many people involved in the N4C project, the DTN community and the DTNRG. The authors are especially indebted to N4C partners, in particular Fritz-Ake Kuoljok (Fritte) and Samo Grasic from LTU and Elwyn Davies from Folly Consulting who gave invaluable assistance for the 2009 trial. Darragh O'Keefe carried out significant work on data analysis whilst a student with TCD. We thank the anonymous reviewers for their very helpful comments.

This work was carried out as part of the FP7 N4C project funded by the EU under contract No. 223994.

[31] http://dtn.dsg.cs.tcd.ie/n4c-summer09/

[32] http://down.dsg.cs.tcd.ie/s10inf/

Author Contributions

EM and KH were primarily responsible for developing the village router hardware and for the 2009 trial deployment. AMcM was primarily responsible for building and packaging the software for all nodes and developed the web service. EM developed the email service. SF and SW gave overall guidance and participated in design. SF participated in the ExtremeCom hike. All authors contributed text for this paper and contributed to data analysis; SF did the overall editing.

References

[1] S. Farrell and V. Cahill, *Delay and Disruption Tolerant Networking*. Norwood, MA: Artech House, 2006.

[2] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," in *Proc. ACM SIGCOMM '03*. New York, NY, USA: ACM Press, 2003, pp. 27–34.

[3] K. Fall and S. Farrell, "DTN: an Architectural Retrospective," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 5, pp. 828–836, June 2008.

[4] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay–Tolerant Networking Architecture," Internet RFC 4838, April 2007.

[5] M. Ramadas, S. Burleigh, and S. Farrell, "The Licklider Transmission Protocol - Specification," IETF, RFC 5326, 2008.

[6] K. Scott and S. Burleigh, "Bundle Protocol Specification," Internet RFC 5050, Nov 2007.

[7] S. Farrell, S. Weber, A. McMahon, E. Meehan, and K. Hartnett, "An n4c router design," ExtremeCom 2009, 2009, http://www.extremecom.org/.

[8] H. Beach, *A year in Lapland : guest of the reindeer herders / Hugh Beach*. Smithsonian Institution Press, Washington :, 1993.

[9] E. Davies, "DTN - The State of the Art," http://www.n4c.eu/Download/n4c-wp2-012-state-of-the-art-101.pdf, 2009.

[10] A. Lindgren, A. Doria, J. Lindblom, and M. Ek, "Networking in the land of northern lights: two years of experiences from dtn system deployments," in *WiNS-DR '08: Proceedings of the 2008 ACM workshop on Wireless networks and systems for developing regions*. New York, NY, USA: ACM, 2008, pp. 1–8.

[11] S. Naidu, S. Chintada, M. Sen, and S. Raghavan, "Challenges in deploying a delay tolerant network," in *CHANTS '08: Proceedings of the third ACM workshop on Challenged networks*. New York, NY, USA: ACM, 2008, pp. 65–72.

[12] T. Hyyryläinen, T. Kärkkäinen, C. Luo, V. Jaspertas, J. Karvo, and J. Ott, "Opportunistic email distribution and access in challenged heterogeneous environments," in *CHANTS '07: Proceedings of the second ACM workshop on Challenged networks*. New York, NY, USA: ACM, 2007, pp. 97–100.

[13] J. Ott and D. Kutscher, "Bundling the web, http over dtn," in *Proc. WNEPT 2006*, 2006.

[14] A. Balasubramanian, Y. Zhou, W. B. Croft, B. N. Levine, and A. Venkataramani, "Web search from a bus," in *CHANTS '07: Proceedings of the second ACM workshop on Challenged networks*. New York, NY, USA: ACM, 2007, pp. 59–66.

[15] M. Demmer, B. Du, and E. Brewer, "Tierstore: a distributed storage system for challenged networks," in *FAST '08: 6th USENIX Conference on File and Storage Technologies*, 2008.

[16] S. Shah and B. D. Noble, "A study of e-mail patterns," *Softw. Pract. Exper.*, vol. 37, no. 14, pp. 1515–1538, 2007.

[17] B. Klimt and Y. Yang, "Introducing the enron corpus," in *CEAS*, 2004.