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

PyMail
Postfix/Python/DTN
Email Interface

Version 0.6

n4c-wp2-022-pymail.doc

Folly Consulting Ltd
Elwyn Davies
www.folly.org.uk/

ABSTRACT

Starting in May 2008, N4C is a 36 month research project in the Seventh Framework Programme (www.cordis.lu/fp7). In cooperation between users in Swedish Lapland and Kočevje region in Slovenian mountain and partners, the project will design and experiment with an architecture, infrastructure and applications in field trials and build two test beds.

This document describes the PyMail system used to send email in DTN bundles. This version contains information on the progress and evaluation of the system during the period of the 2009 Summer tests in arctic Sweden.

Due date of deliverable: 31/12/2009 Actual submission date: date/month/year

Document history			
Version	**Status**	**Date**	**Author**
0.6	Completed Section 6, spelling check and minor corrections.	16/03/2010	Elwyn Davies
0.5	Completed relay installation notes	08/03/2010	Elwyn Davies
0.4	Completed outstation installation notes	08/03/2010	Elwyn Davies
0.3	Added comments from Avri Doria and editorial fixes	21/12/2009	Elwyn Davies
0.2	Filled in outline.	30/11/2009	Elwyn Davies
0.1	Added outline for evaluation and progress deliverable.	30/11/2009	Elwyn Davies
0.0	Created (primarily to document architecture and Postfix configuration)	6/11/2009	Elwyn Davies

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

CONTENT

1. INTRODUCTION AND MOTIVATION

The PyMail system was initially developed to provide an interface between a Postfix mail server running in a DTN equipped Internet gateway and email clients running in small, portable nomadic clients connected through a DTN network. All the nodes are equipped with DTN bundle agent stacks, initially using the DTN2 reference implementation.

The motivation for constructing this system was to provide a convenient means for testing the proposed developments of the DTN naming and addressing scheme in the dtn: uri scheme in conjunction with the PRoPHET dynamic DTN routing protocol. The intention is that nomadic devices would expose EIDs both for the basic name of the node and also the mail acceptance service for one or more users who have access to the node. The PRoPHET routing protocol would route bundles addressed to the relevant mail service accordingly.

The interface in the Internet gateway is mediated through a suite of Python programs that send and receive email messages from the mail server and client and convert them to and from email bundles that are handled by the bundle agent. Each email is encapsulated in a single bundle. For mail being sent to nomadic clients, a separate bundle is generated for each nomadic destination so that they can be routed independently. Email for other destinations that has to pass through a gateway may have multiple destinations specified in a single bundle. The email destination addresses are placed in a 'Delivered To:' extra header at the beginning of the email in its internal form. The Postfix email server is used in the Internet gateway to provide the email service.

The nomadic nodes are also equipped with a suite of similar Python programs that interface with the bundle agent and a local email client. Incoming mail is provided to the local email client through a minimalist POP3 server and outgoing mail is handled by a minimalist SMTP server. These programs are both implemented in Python. This allows the use of (almost) any mail client that may be used on the nomadic device. The initial implementation has been tried out on various machines, including Asus EEE PCs and Nokia N810 Internet tablets, using Wi-Fi communications for exchange of routing information and email bundles.

2. ARCHITECTURE

The PyMail system contains four independent components that run as processes within an OS environment. Two main combinations are provided for depending on the context in which the system runs. The system as

originally envisaged is designed to transport email in DTN bundles (to RFC 5050 standard) between a gateway machine with access to the conventional SMTP-based RFC 2822 Internet email network and small) nomadic machines intended primarily for a single dedicated user that communicate primarily using DTN. The network architecture is illustrated in Figure 1.

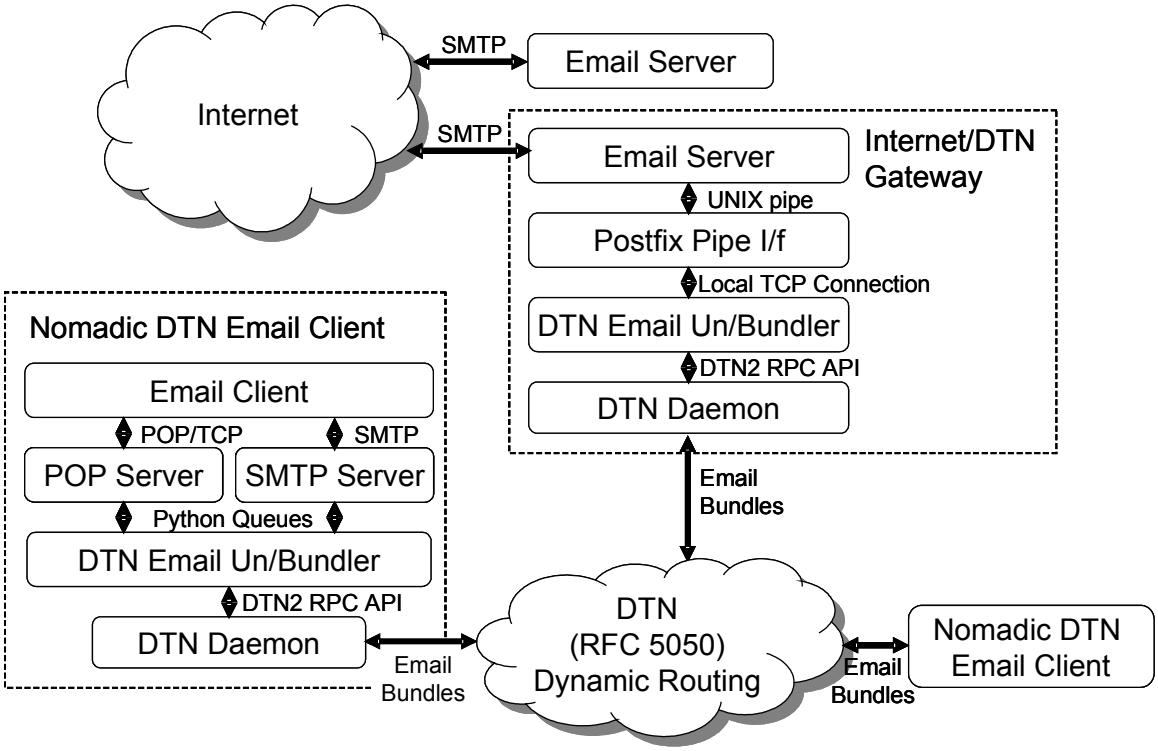


FIGURE 1 BASIC PYMAIL NETWORK ARCHITECTURE

In the basic case, there are two distinct configurations used on the Internet gateway and on the 'outstations'. The Internet gateway is intended to run the Postfix mail server which provides the routing for emails, a DTN2 bundle daemon to transmit and receive bundles, and some interface logic written in Python. The outstation is intended to run one of the various email clients that provide a user mail interface, a DTN2 bundle daemon to transmit and receive bundles, and some interface logic written in Python.

2.1 INTERNET GATEWAY ARCHITECTURE

The components are:

- Postfix mail server
- Pipe transport process
- Bundle encapsulator/decapsulator (Email bundler and unbundler)
- DTN2 bundle daemon

The gateway provides a mechanism that can route emails intended for a specific domain to the bundle daemon where they are sent out into the DTN network, and a parallel mechanism that receives emails as bundles that need to be routed to other domains through the Postfix server.

The outgoing path for sending emails into the DTN network works as follows:

- The Postfix server is configured with the domain(s) that are to be accessed through DTN as relay domains. These are known as 'nomadic (email) domains'.
- The Postfix is configured with a new 'pipe' transport specification that invokes an instance of the pypop_smtpout.py process for each individual email to be sent to the nomadic email domains.
- The transport map configuration is configured to use this new transport for the nomadic email domains with the server set to invoke a separate pipe process for each email.
- The encapsulator/decapsulator runs continuously as a daemon. It is multithreaded. On the outgoing path, a thread accepts emails from the pipe transport process and examines the destination addresses. It saves a copy of the email in a local maildir directory (dtnout) and queues an event for the outgoing DTN thread. There are separate DTN threads to deal with outgoing mails, incoming mails and report bundles indicating that outgoing mails have been received. The outgoing DTN thread accepts events resulting from emails being received from Postfix via the pipe transport and turns them into bundles. The destination EID is derived algorithmically from the email address in the 'Delivered To:' header. and delivery reports are requested. This is currently a limitation as the EID of the nomadic outstation has to match the email address of the user to whom email will be delivered. The copies of emails are tagged with the bundle id of the transmitted bundle and retained until a delivery report is received or delivery is deemed impossible. If a delivery report is not received after twice the specified expiry time for the email bundle (currently set to 3 days – the delivery report will have an equal expiry time) a new event is queued in order to retransmit the bundle. After (say) 3 retransmissions without a delivery report delivery is deemed impossible. [Retransmission is not yet implemented in the code supplied.]
- A 'standard' DTN2 bundle daemon is used to transmit and receive RFC 5050 formatted bundles. Due to the nature of the application API, separate connections are used for sending and receiving bundles. This allows the interface logic to continuously monitor the incoming interface which would not be feasible if the same connection was used for input and output. It also allows both interfaces to sit in I/O waits so that the interface application does not consume unnecessary compute cycles. The registrations used by the interface application are set to DEFER with a very long lifetime (2 years!) so that incoming bundles are retained even if the mail interface is not running (after it has been run once). Typically we would expect to use a dynamic DTN routing protocol such as PRoPHET in order to distribute emails within the nomadic domain. See Section 2.4.2 for an alternative mechanism that uses part static and part dynamic protocols.

The incoming path for receiving emails from the DTN network works as follows:

- The DTN2 bundle daemon receives incoming email bundles (and also relevant delivery reports).

- The encapsulator/decapsulator runs a thread which registers the appropriate demultiplexing address (dtn:<local eid>/email/in) and the email bundles are delivered to this thread. The DTN2 bundle daemon will send a delivery report at this stage. [Note: this is not necessarily quite correct – there is a chance that the bundle would not be correctly sent onwards but there is no way in DTN2 to postpone the delivery report. It would require sending a positive acknowledgement for full verification.] The bundles are decapsulated and stored in a local maildir (dtnin) ready for onwards transmission to Postfix. The receiving thread enqueues an event for the incoming email. A separate thread receives these events and implements an SMTP client which connects to the standard SMTP server socket provided by Postfix on localhost. The received emails are sent using the standard SMTP protocol to Postfix. Once they have been successfully received by Postfix

- Postfix must be configured to provide promiscuous relay for localhost connections. The emails are received by Postfix and dispatched into the Internet or delivered to local mailboxes as appropriate.

2.2 OUTSTATION ARCHITECTURE

The components are:

- Email client (most standard email clients are usable, e.g., Thunderbird, Evolution)
- Bundle encapsulator/decapsulator with SMTP server and POP3 server
- DTN2 bundle daemon

The PyMail outstation software provides an interface to the DTN2 bundle daemon to allow the sending and receiving of emails in RFC 5050 formatted bundles. The current system allows for a single email user on the outstation and ties the EID of the outstation to the email address of the user. The nomadic users are confined to a single domain within this system. Parallel paths are provided for incoming and outgoing emails based on a lightweight email server implementation integrated with the bundle encapsulator/decapsulator. The email server provides a very basic SMP server implementation for outgoing emails and a basic POP3 server for incoming emails. As with normal email servers, the email client need not be active in order for emails to be received and stored.

The outgoing path for sending emails works as follows:

- The email client connects to the SMTP server in the localhost provided by the encapsulator/decapsulator daemon. Messages are sent using the standard SMTP protocol.

- The encapsulator/decapsulator runs continuously as a daemon. It is multithreaded. On the outgoing path, a thread accepts emails using the SMTP protocol. It implements a very basic set of SMTP capabilities that are

suitable for a single client system. It saves a copy of the email in a local maildir directory (dtnout) and queues an event for the outgoing DTN thread. There are separate DTN threads to deal with outgoing mails, incoming mails and report bundles indicating that outgoing mails have been received. The outgoing DTN thread accepts events resulting from emails being received from the email client via the SMTP server and turns them into bundles. The destination EID is either derived algorithmically from the email address in the 'Delivered To:' header if it is another address in the nomadic email domain or is the fixed EID of the Internet gateway. Delivery reports are requested. Again there is currently a limitation as the EID of the nomadic outstation has to match the email address of the user to whom email will be delivered. The copies of emails are tagged with the bundle id of the transmitted bundle and retained until a delivery report is received or delivery is deemed impossible. If a delivery report is not received after twice the specified expiry time for the email bundle (currently set to 3 days – the delivery report will have an equal expiry time) a new event is queued in order to retransmit the bundle. After (say) 3 retransmissions without a delivery report delivery is deemed impossible. The email is discarded and, if possible, a delivery failure notification email will be sent to the original sender. [Retransmission is not yet implemented in the code supplied.]

- A 'standard' DTN2 bundle daemon is used to transmit and receive RFC 5050 formatted bundles. Due to the nature of the application API, separate connections are used for sending and receiving bundles. This allows the interface logic to continuously monitor the incoming interface which would not be feasible if the same connection was used for input and output. It also allows both interfaces to sit in I/O waits so that the interface application does not consume unnecessary compute cycles. The registrations used by the interface application are set to DEFER with a very long lifetime (2 years!) so that incoming bundles are retained even if the mail interface is not running (after it has been run once). Typically we would expect to use a dynamic DTN routing protocol such as PRoPHET in order to distribute emails within the nomadic domain. See Section 2.4.2 for an alternative mechanism that uses part static and part dynamic protocols.

The incoming path for receiving emails from the DTN network works as follows:

- The DTN2 bundle daemon receives incoming email bundles (and also relevant delivery reports).

- The encapsulator/decapsulator runs a thread which registers the appropriate demultiplexing address (dtn:<local eid>/email/in) and the email bundles are delivered to this thread. The DTN2 bundle daemon will send a delivery report at this stage. [Note: this is not necessarily quite correct – there is a chance that the bundle would not be correctly sent onwards but there is no way in DTN2 to postpone the delivery report. It would require sending a positive acknowledgement for full verification.] The bundles are decapsulated and stored in a local maildir (dtnin) ready for onwards transmission to Postfix. A separate thread implements a simple POP3 server

that accesses the maildir and can provide access to the received mail from any email client that uses POP3 - the mail could also be accessed through a maildir client such as Evolution if desired. The POP3 server is not explicitly notified when new mail arrives – it is assumed that the email client will periodically poll the POP3 server and will recognize when new mail appears without explicit action. Emails that have been retrieved through POP3 are deleted (actually moved to the deleted area of the maildir) at the end of each POP3 connection.

- The email client retrieves the email through POP3 (or the maildir 'protocol') and displays it for the user.

2.3 COMPARISON WITH 'VILLAGE' EMAIL ARCHITECTURE

The nomadic email solution presented in this document uses DTN to deliver bundles containing emails directly to nomadic outstations. An alternative solution, The Village Email architecture, has been demonstrated within N4C. This uses DTN bundles as a means to transport emails between an Internet gateway and an email server machine located in a enclave (e.g., a Sámi village) that has local IP connectivity within the enclave but no conventional connection to the (not quite) global Internet.

Outstations within range of the server machine, typically using standard Wi-Fi connectivity, are able to access email through the server in the enclave. DTN is used as a transport for groups of email. The N4C implementation provides a mirroring mechanism that uses multicast delivery to provide mirrors of the email store on the Internet gateway in one or more email servers in enclaves.

Individual outstations need not necessarily be equipped with a DTN bundle agent as they will use IMAP or POP3 over IP to access the email server in an enclave.

Since the email server machines are either completely fixed in one location or only moved occasionally, the village email architecture can work satisfactorily with static DTN routing. It is not necessary for the data mules that carry DTN bundles to use dynamic routing such as PRoPHET and it is easy to leverage semi-scheduled delivery mechanisms such as the helicopter flights that visit many of the Sámi villages on a relatively regular basis.

On the other hand users need to be within range of the servers and email cannot be delivered out to more isolated positions. A combination of the two architectures may prove to be the optimal solution for the large Sámi areas or in circumstances where people within a village need communications with one of their nomadic members.

2.4 ALTERNATIVE USES

2.4.1 Relaying from an Internet Gateway to an Enclave Gateway

In a situation where rather than sending emails from an Internet gateway to nomadic outstations, emails are to be sent across a DTN network to disconnected enclave (e.g., a village) that has an email server machine that can be accessed by local machines using standard IP.

The PyMail software normally used on the Internet gateway can be slightly adapted so that all emails are sent to the enclave gateway by using a fixed EID for outgoing email bundles. In this case static routing may well be used. This is an alternative implementation of the village email architecture.

The enclave gateway runs the same software as the Internet gateway but with an alternative EID so that all outgoing emails are sent to the Internet gateway.

2.4.2 Combining Enclave Relay and Nomadic Solution

Unfortunately the current implementation of the DTN2 bundle daemon can only provide one routing method in a single instance. If you wish to use static routing to get the bundles to an enclave gateway and then onwards using dynamic routing to nomadic outstations, the enclave gateway will have to run two DTN2 daemons. This is not a problem but one of them (at least) has to be configured to use non-default (TCP) ports for bundle links and control access.

The linkage can either be done using a Postfix server as an intermediary, or there is also a DTN 'NAT' module that can rewrite the DTN addresses while relaying emails between a pair of DTN2 bundle daemons.

The arrangement for the DTN NAT configuration is shown in Figure 2. The NAT components function as Application Layer Gateways that receive all the email bundles that need to be relayed between the nomadic routing domain and the static routing domain. They rewrite the destination DTN EIDs in the email bundles received from one of the bundle daemons appropriately and then resend them though the ‘other’ bundle daemon. The NAT components are implemented in Python using much of the same code as is used for the DTN bundle daemon interface at the Internet gateway and the nomadic email client nodes.

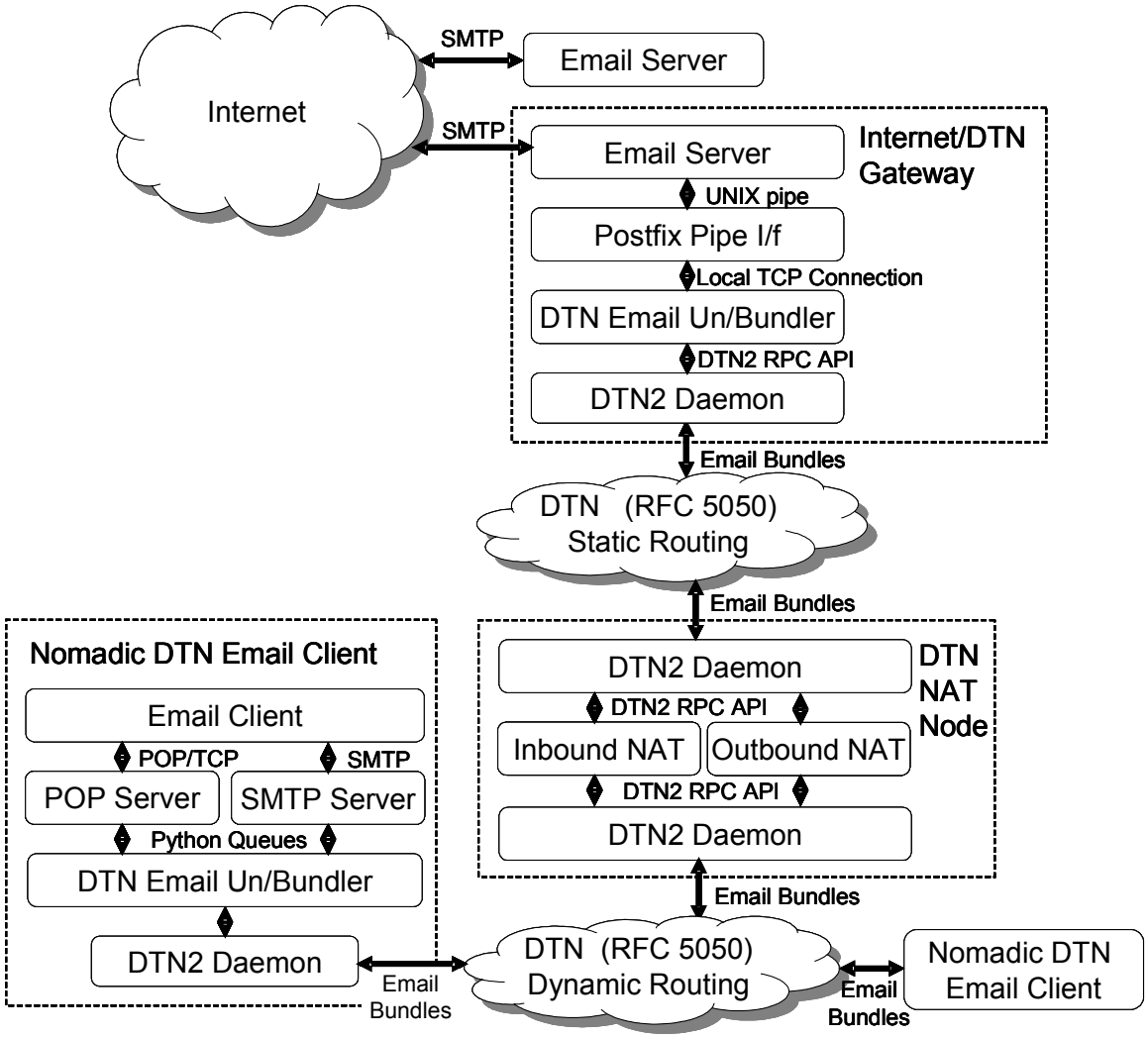


FIGURE 2 COMBINATION OF STATIC AND DYNAMIC ROUTING

3. DEVELOPMENT

This section gives an outline of the processes needed to get the infrastructure running on the equipment used for testing before and during the Summer 2009 trials.

The Internet gateway runs on a conventional Intel Atom based server that is Folly Consulting's public server (*rosebud.folly.org.uk*). The operating system on this machine is Gentoo Linux version 2.6.28. It is running Postfix 2.5.5, Python 2.5 and the most recent version of the DTN2 code (as at 12 June 2009). with some additional bug fixes in the PRoPHET code resulting from investigations that Elwyn Davies carried out. Bundle information is stored in a Berkeley DB storage.

The nomadic units were a 'swarm' of Nokia N810 Internet tablets of which there were ten units. The operating environment for these machines is discussed in Section 3.2.

Additionally, in order to leverage the linkage being used by Trinity College Dublin(TCD) for their 'village' email system, an Asus EEE PC was also used to provide 'DTN NAT'. This will be explained later in Section 8.3.5.

3.1 INTERNET GATEWAY ENVIRONMENT

The operating system on this machine is Gentoo Linux version 2.6.28. It is running Postfix 2.5.5, Python 2.5 and the most recent version of the DTN2 code (as at 12 June 2009). with some additional bug fixes in the PRoPHET code resulting from investigations that Elwyn Davies carried out. Bundle information is stored in a Berkeley DB storage.

Development of the custom Python code for this machine and compilation of the DTN2 code, including the Python API, used the 'standard' Linux environment and needed no special adaptations. The configuration of the various components is covered in Section 4.

3.2 N810 ENVIRONMENT

The Nokia N810 runs the Maemo Diablo operating system with the Hildon user interface on an ARM Atmel processor. Because of the limited resources of the N810, tools have to be cross compiled in a 'Scratchbox' environment (see http://maemo-sdk.garage.maemo.org). Folly Consulting set up a Scratchbox environment and generated a DTN2 suite for the N810. There were a couple of prerequisites that were not in the standard N810 builds: The Berkeley DB database system (version 4.7 was built from source) and the TCL interpreter (version 8.5 was built from source). Since they were not required, the XML interface was omitted (the Xerces package was therefore not required) and the Bundle Security Protocol was not included.

Python 2.5 was also needed for the interface programmes. Python 2.5 for N810 is available but is somewhat non-standard. A long list of Python packages is required in order to have a fully functioning Python system on an N810. (see appendix). In particular, in order to build the Python dtnapi module, the distutils package is required.

In order to conserve space on the fast memory of the N810, it is essential to use 'stripped' executables. The DTN2 daemon programme in particular is approximately 25MB with all symbol tables in place but reduces by about 90% when 'stripped' of symbol tables.

The standard email client on the N810 was used to send and receive email on the outstations.

The Hikers PDA application was also installed on the N810s. In line with the Hiker's PDA control interface, Hildon UI menu items were installed to provide control of the PyMail application allowing it to be started and stopped by the user. This was important to help with battery lifetime management, as the use of ad-hoc Wi-Fi is not kind to the battery and can lead to a relatively short battery discharge time.

Details of installation and configuration are described in Section 5.

3.3 OUTSTATIONS ON OTHER MACHINES

A PyMail outstation can be run on essentially any machine that supports the DTN2 bundle daemon, Python runtime and an email client.

At present this essentially excludes machines running Microsoft Windows because the DTN2 code does not work well (or maybe at all) on Windows. Note that since the DTN2 API can potentially work between machines that are IP connected, the Python portion and the email client can run on a different machine to the DTN2 bundle daemon.

For most Linux distributions on machines more capable than the N810 tablets, installation is much simpler because the code can be compiled locally and the Berkeley database and TCL interpreters are usually pre-installed.

3.4 STATIC TO DYNAMIC RELAY MECHANISM

In order to minimize the amount of configuration and routing mechanisms during testing in the Swedish Arctic area in summer 2009, it was decided that it would be convenient to use the same routing and mule mechanism to transfer nomadic email bundles from the Internet gateway machine to the village router at Stahloluokta and then use the nomadic system to take the bundles on to nomadic machines in the field around Stahloluokta. However the route from Dublin to Stahloluokta used static routing and a limitation of the current DTN2 implementation is that it can only handle one routing scheme per bundle daemon. The PyMail nomadic email system is intended to be used with a dynamic routing protocol such as PRoPHET. It was decided that the appropriate solution was to establish a static route from the Internet gateway implemented on the Folly Consulting server (rosebud) in a Maidenhead, UK data centre through an SSH tunnel to the Trinity College Dublin (TCD) server (basil) and then onward following the same route as the village emails (see Section 2.3) to the helicopter base at Ritsem in the Padjelanta national park and via helicopter mule to the TCD DTN village router. A relay machine (implemented on a Asus EEE-PC running the Xandros distribution of Linux) would then run two bundle daemons. One daemon would provide a static route through the village router back to the Internet gateway in the UK. The second daemon would provide a proxy Internet gateway running the PRoPHET routing protocol. All nomadic email would pass through this pair of daemons.

In order to implement this mechanism using DTN2, it was necessary to provide an EID (address) translation mechanism at the DTN bundle level. Bundles going to nomadic users were initially addressed to the relay node. At the relay node a Python application registered with the static routing daemon to receive all the nomadic email bundles (just a single EID). The bundles were then inspected and a new destination EID inserted based on the destination email address. The constructed bundles were then sent out via the dynamic routing daemon.

In the reverse direction, a second Python application registers with the dynamic routing bundle daemon to receive all email bundles with the Internet gateway EID (this does not need to be modified because the routing systems are independent). The source EID is modified to reflect the form used in the static domain and sent on to the static routing bundle daemon for transmission to the real Internet gateway in the UK.

The details of the configuration are described in Section 6.

4. CONFIGURING THE INTERNET GATEWAY

A number of things have to be installed and configured in order to set up the Internet gateway. The following sections give details.

4.1 BUNDLE EID CONSTRUCTION

Because of limitations in the EID handling of DTN2, which effectively only allows a node to have a single 'base' EID (similar to a host name), the EIDs used in the PyMail system are closely tied to the associated email domain. In the system that was deployed during Summer 2009 in the Swedish Arctic, the email domain used was **nomadic.n4c.eu**. Thus the Internet gateway has the EID **dtn://gateway.nomadic.n4c.eu**.

The destination EID for email bundles being sent to nomadic nodes is derived algorithmically from the email address of the (single) user of that nomadic node. Thus an email being sent to **username@nomadic.n4c.eu** will be encapsulated in a bundle with destination EID **dtn://username.nomadic.n4c.eu/email/in** where the **email/in** part is used to demultiplex the bundle to the correct (email) handler at the destination.

The source EID used for email bundles sent out by the gateway is then **dtn://gateway.nomadic.n4c.eu/email/out** and reports (delivery and deletion) will be returned to **dtn://gateway.nomadic.n4c.eu/email/reports**.

Similarly email sent by nomadic outstations will have the destination EID **dtn://gateway.nomadic.n4c.eu/email/in.** The Python Email Bundle Encapsulator/Decapsulator in the gateway registers to receive email bundles sent to **dtn://gateway.nomadic.n4c.eu/email/in** and reports sent to **dtn://gateway.nomadic.n4c.eu/email/reports**.

The source EID used for email bundles sent by the nomadic node handling the email for **username@nomadic.n4c.eu** will be **dtn://username.nomadic.n4c.eu/email/out**.

In the current version of PyMail these address patterns are hard coded into the Python code: the email domain is set in the main driver (*dp_main.py*) and the EID patterns can be found in the DTN interface (*dp_dtn.py*).

4.2 USERS, PERMISSIONS AND DIRECTORY STRUCTURE

To manage the programs, processes and data for the PyMail system, it is convenient to create a user and group called *dtn* to own all the material. The DTN2 daemon process and the Python programs are run with the dtn user as owner so that the process has limited access in case of a security breach.

A home directory (typically */home/dtn*) should be created for the dtn user and the following directory structure created. The permissions for these directories should be set to minimize accessibility for other users.

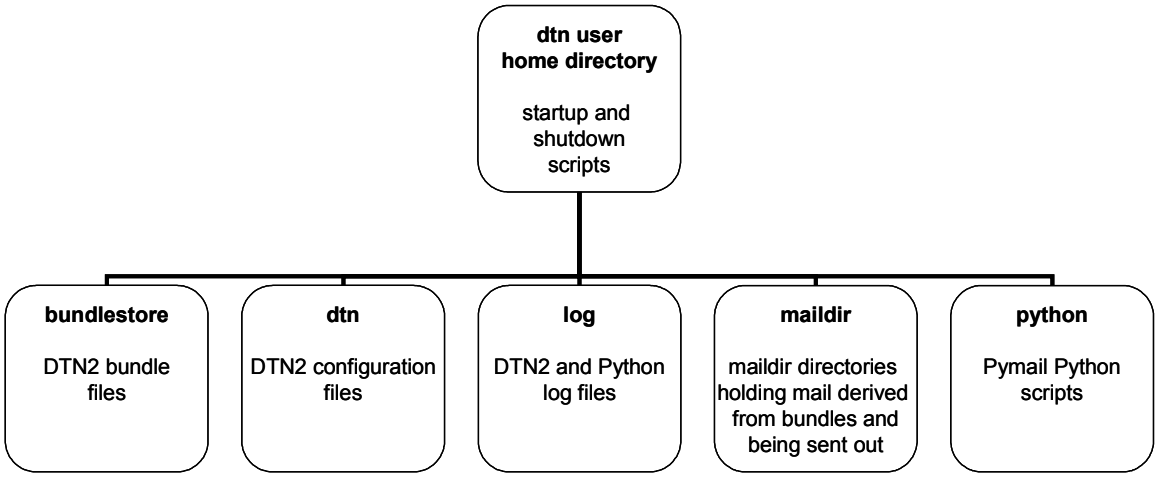


FIGURE 3 INTERNET GATEWAY DTN USER DIRECTORIES

The **dtn** user does not need access to any Postscript configuration or have any association with the owner of the Postscript processes.

Note that the *bundlestore* and the *maildir* need to be in the same filing system because the DTN2 bundle payload files in *bundlestore* and the email files in *maildir* are just 'moved' rather than copied since they are the same data but just conceptually reassigned when the email is encapsulated or decapsulated by the Python program.

4.3 CONFIGURING POSTFIX

To allow Postfix to communicate with the PyMail email encapsulator/decapsulator, a number of things have to be done:

- The domain(s) to which email is to be sent across DTN have to be declared as relay domains so that the Postfix server will forward these emails.
- A new 'pipe' transport (dtnout) has to be declared.
- The pypop_smtpout.py program invoked by the transport has to be placed appropriately and made executable.
- This transport is configured to send one email per instance.
- The dtnout transport is specified for use by all the DTN relay domains.
- Ensure that Postfix will provide anonymous relay for localhost SMTP connections.

The following sections show how to do this. The Postfix configuration files are typically in */etc/postfix* but this potentially depends on the Linux distribution in use.

4.3.1 Adding DTN relay Domains

In the Postfix *main.cf* file modify the line specifying *relay_domains* =...thus:

```
relay_domains = nomadic.n4c.eu, nomadic1.dtn.relay
```

where the right hand side is a comma separated list of domain names if required.

4.3.2 Adding the DTNOUT Pipe Transport

In the Postfix *master.cf* file add a new (logical) line:

```
# DTN mail outgoing interface
dtnout     unix  -       n       n       -       -       pipe -v
  flags=D user=dtn argv=/home/dtn/python/pypop_smtpout.py ${recipient}
```

The last line is a continuation line as it starts with white space. The position of the new line within *master.cf* is not significant.

The effect of this is to create a new sub-process connected to the Postfix daemon through a Unix pipe whenever there is an outgoing email to be sent to a nomadic email domain. The invoked process is named via the **argv** parameter; modify the path as appropriate for your installation. The **flags=D** parameter request that Postfix prepends a *Delivered To:* line to the email. The **user=dtn** sets the owner of the sub-process; choose a different user name as appropriate. the **$(recipient)** parameter means that the email of the recipient is on the command line of the sub-process as well. Finally the **-v** modifier for **pipe** is optional; it prompts additional logging when the transport is invoked. Some of these functions only work because a separate process is invoked for each email.

4.3.3 Configuring the DTNOUT Pipe Transport

Add a new line to *main.cf*:

```
# Set the concurrency for delivery to dtnout
dtnout_destination_recipient_limit = 1
```

This ensures that a separate sub-process is spawned for each email destined for the DTN relay domains. If huge numbers of emails were involved, we might wish to consider whether this is a satisfactory solution.

4.3.4 Connecting to DTN RELAY Domains with DTNOUT

In *main.cf* specify a transport map file. The most convenient is probably one that uses regular expressions for the map, thus:

```
# Transport maps for DTN transports
transport_maps = regexp:/etc/postfix/transport_regexp
```

The transport map file then contains lines such as:

```
/^.*\@nomadic.n4c.eu$/          dtnout:
/^.*\@nomadic.dtn.relay$/       dtnout:
```

4.3.5 Providing Anonymous Relay for Localhost SMTP Connections

In *main.cf* there are various ways to specify which machines you trust. Look for the comments relating to **mynetworks**.

If you only need to trust the local host, the easiest way is specify

```
mynetworks_style = host
```

If more than one host needs to be trusted, specify, for example

```
mynetworks = 127.0.0.0/8, 192.178.324.0/24
```

4.4 CONFIGURING THE PYTHON PARTS

The Python code is written for version 2.5 of the interpreter, but should run equally well on version 2.6. It will not run with version 3.x of the Python interpreter at present. This choice was made because version 3 is not available for the N810, and also when the project was started version 3 was very immature.

The Python files are conveniently placed in */home/dtn/python/*. The Python parts use the 'standard' Python logging system. The Postfix pipe 'adapter' and the Email Bundle Encapsulator/Decapsulator uses separate logging files. In the current version the log configuration file name (*/home/dtn/dtn/dtn_pymail_log.conf*) is hard coded into the source, and needs to be changed if the directory structure is altered. The names of the log files used by the two programs are in the configuration file and may need to be altered if the directory structure is altered. The logs are set to 'rotate' at midnight each day.

For the Postfix pipe 'adapter', the code is in:

- pypop_smtpout.py

This is a standalone Python program. The file should be made executable so that it can be run by the Postfix transport subsystem. The log is opened at */home/dtn/log/dtn_postfix_pipe.log*.

The Email Bundle Encapsulator/Decapsulator Postfix interface uses the following code pieces:

- dp_main.py
- pypop_maildir.py
- dp_pfmailout.py
- dp_pfmailin.py
- dp_dtn.py
- dp_msg_send_evt.py
- SocketServer2_6.py (Note: this is a copy of the standard SocketServer Python library code from version 2.6. The SocketServer code in version 2.5 is badly broken.)

It may be necessary to change some hard coded values in dp_main.py if the directory layout changes or the structure of the domain names used is changed.

The path names in the last line represent

- The root of the maildir mailbox directory tree

- The location of the log file
- The name of the Python logging configuration file

Earlier on in the file (line 102), the name of the nomadic mail domain is defined.

It may be convenient to make the dp_main.py file executable, but this is not essential as the start up script (*/home/dtn/start_mail_link.sh*) doesn't assume it is. The log is opened at */home/dtn/log/dtn_pymail.log*.

4.5 CONFIGURING DTN2

Ensure that the Python DTN2 API is created and installed. This is not the main **`make`** command for DTN2. It is necessary to change to the *applib* directory and issue the commands **`make pythonapi`** and **`make pythonapi_install`**. The latter command may require root permissions depending on where it is writing.

Assuming that the system is using PRoPHET dynamic routing, the following configuration needs to set in the */home/dtn/dtn/dtn.conf* file for the DTN2 bundle agent daemon (**dtnd**):

```
# Storage Directory
set dbdir /home/dtn/bundlestore

# Routing type
route set type prophet

# Route local eid
route local_eid "dtn://gateway.nomadic.n4c.eu

# PRoPHET parameters
# Age out node parameters every 12 hours - 12 * 60 * 60 seconds
prophet set age_period 43200

# Decay probabilities by units of a day -
#   kappa is set in milliseconds - 24 * 60 * 60 *1000
prophet set kappa 86400000

# Route local eid
route local_eid "dtn://gateway.nomadic.n4c.eu

# TCP interface (default address and port)
interface add tcp0 tcp

# Configure local neighbour discovery
discovery add ip_disc ip port=4301
discovery announce tcp0 ip_disc tcp cl_port=4557 interval=10
```

4.6 STARTUP AND SHUTDOWN SCRIPTS

For starting the system there is a main start-up shell script (*/home/dtn/start_nmadic_mail.sh*) and two subsidiary shell scripts that it calls

(*/home/dtn/start_dtnd.sh* and */home/dtn/start_mail_link.sh*) that respectively start the DTN2 bundle daemon and the Python Email Bundle Encapsulator/Decapsulator.

The process id (number) of the Bundle Encapsulator/Decapsulator (Python) process is recorded in */home/dtn/dp_main.pid* for use when the process has to be stopped.

For stopping the system there is a shell script */home/dtn/stop_nomadic_mail.sh* which kills the Python process whose id is recorded in */home/dtn/dp_main.pid* and uses **dtnd-control stop** to shutdown the DTN2 bundle daemon.

5. CONFIGURING AN N810 OUTSTATION

Installing PyMail on a Nokia N810 tablet is more complex than on a more capable machine because a number of packages are not installed by default, and there are not standard Debian Maemo packages available for them. These packages have to be complied from source in the Maemo 4 Scratchbox environment. Note that as of the end of 2009, the N810 is obsolete and new machines can no longer be bought. The N900 is available as a substitute, with more capabilities and running Maemo 5. This document will be updated to cover the N900 in due course.

5.1 SETTING UP SCRATCHBOX AND BUILDING REQUIRED COMPONENTS

This document is not going to go through all the details of getting the Scratchbox development environment running on your (Linux) cross-compilation platform. The details are rather Linux distribution dependent - it is straightforward and pretty automatic (I believe) on a Debian based distribution. Gentoo (which I use) is somewhat more complicated.

Before starting it is necessary to ensure that the Linux kernel is compiled with the binfmt_misc module enabled so that Armel executables can be run, and if you wish to experiment with the graphical user interface you must have the X-server Xephyr capability installed. There is a Debian package for this. If compiling using Gentoo, you need to set the USE="kdrive" flag on the configuration.

Documentation for installing Scratchbox and the necessary tools can be found at Maemo 4.1.2 SDK Releases. The Scratchbox installer script does the required work. Note that this is still Scratchbox version 1.0.11 rather than Scratchbox 2. These is a different project which is user supported which uses Scratchbox 2. I have no experience with this. There is a huge manual for the release at Maemo Diablo 4.1 Release Documentation which also includes some notes on setting up the development environment.

Once you have Scratchbox installed and checked out, use apt-get to install the Python 2.5 and Python 2.5 development packages inside Scratchbox (packages: python, python2.5 and python2.5-dev).

Download the source code for the various packages that have to be installed:

- TCL Interpreter (Version 8.5.7+) from Tcl/Tk Developer Exchange Repository.
- Berkeley Database (release 4.7.25 - DTN2 hasn't been checked out with 4.8 I believe) from Berkeley DB Release Archive. Obtaining the source code is now simpler - you used to have to register.
- DTN2 and Oasys (Sourceforge or N4C Code Repository)

Extract the source inside the Scratchbox environment. Compile and install the packages within Scratchbox with the Armel tool chain. It is unlikely that you will be using the external router or convergence layer capabilities in DTN2 so configure Oasys without the Xerces XML package. I have not attempted to use the Bundle Security Protocol in the PyMail environment as yet, so disable the BSP in DTN2.

For convenience the installation directories for these packages should be set to */usr/local/*. This will make it easier to control installations in the N810. The Berkeley database library will install into */usr/local/BerkeleyDB.4.7* and Tcl into */usr/local/lib*. In order to link Oasys and DTN2, these two directories have to be added to */etc/ld.so.conf* and then **ldconfig** must be run in order to update the linker cache.

Ensure that the Python DTN2 API is created and installed. This is not the main **make** command for DTN2. It is necessary to change to the *applib* directory and issue the commands **make pythonapi** and **make pythonapi_install**. The latter command may require root permissions depending on where it is writing.

It is necessary to 'strip' the *dtnd* DTN2 bundle daemon executable as it is approximately 25 Mbytes including all the debug symbol tables. This would take a large chunk of the available program storage in the N810. After executing **strip dtnd** the executable will be closer to 2.5Mbytes for a saving of about 90%!

It is also possible to reduce the amount of material to be transferred onto the N810 - in particular eliminating the manual pages. A list of files actually transferred is contained in the appendix.

5.2 BUNDLE EID CONSTRUCTION

The EIDs used for bundles are described with the Internet gateway configuration in Section 4.1. In the current version of PyMail these address patterns are hard coded into the Python code: the email domain is set in the main driver (*dp_outstation.py*) and the EID patterns can be found in the DTN interface (*dp_dtn.py*).

5.3 USERS, PERMISSIONS AND DIRECTORY STRUCTURE

The N810 only expects to have one user (other than root). Normal user files are installed in */home/user*. Large files that are *not* executables, such as log files can be stored on */media/mmc2*, which stores onto the internal flash memory card. It appears that it is not possible to execute programs from the

flash memory. The system components are installed as **root** files on */home/user* and */media/mmc2* are stored as **user**.

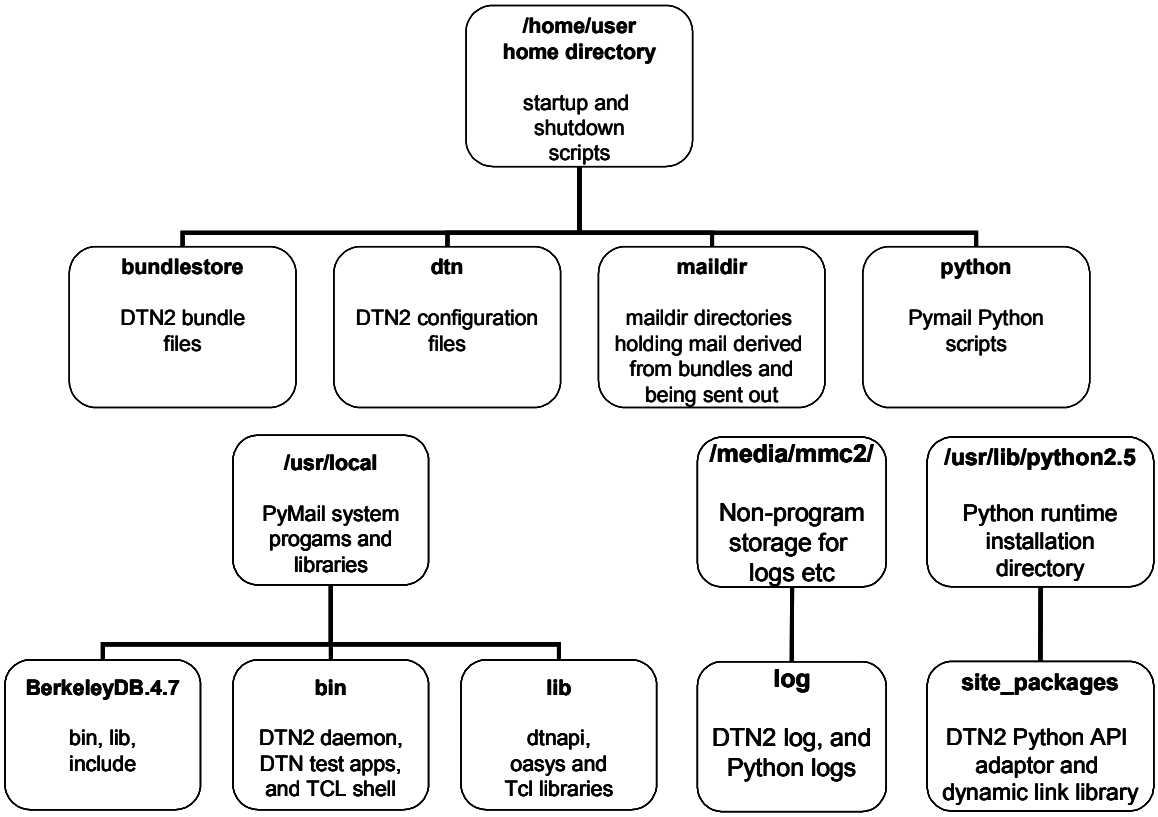


FIGURE 4 N810 DIRECTORY STRUCTURES

Note that the *bundlestore* and the *maildir* need to be in the same filing system because the DTN2 bundle payload files in *bundlestore* and the email files in *maildir* are just 'moved' rather than copied since they are the same data but just conceptually reassigned when the email is encapsulated or decapsulated by the Python program.

5.4 INSTALLING COMPONENTS

In order to add programs to the N810 and set up the filing system as needed it is convenient to have remote access to the N810. Start by installing the openSSH system on the N810 as documented here. It is best to install both the openssh-server and openssh-client. As part of the installation process a password is set for the superuser (**root**).

With openssh installed, the N810 can be accessed from a another machine. Typing on a full size keyboard is significantly easier!

The next stage is to install the Python version 2.5 runtime system. This comes from the Maemo Extras repository. The process is described on this page - the OS2008 (Diablo) version is needed for the N810 - but it is overly restrictive. It is not necessary to be in RD mode. Having superuser access using ssh is sufficient. Using `apt-get install python2.5-runtime` in a root shell is quite adequate. Note that this process involves downloading a lot of packages from the repository. If you are loading more then one N810 it is

much faster to download onto one machine and then copy the package files between the apt cache directories at */var/cache/apt/archives* on the two machines. Then after enabling the Extras repository and issuing **apt-get update**, the install will use the pre-cached packages rather than accessing the Internet.

[I have also considered changing the file system type on */media/mmc2* to **ext3**. This requires the **e2fsprogs** and **e2fslibs** packages. Installing the Norut **Geoblog** application also potentially requires the **subversion** program package. Other useful packages are **python2.5-setuptools** also needed for **Geoblog**, but may also be useful in future for **PyMail** when there is a Python installer (planned). I may also choose to use **sqlite** as the database for **PyMail**.]

The remaining components (Tcl, Berkeley Database and DTN2 package) cannot be installed from a repository but have to be copied from the Scratchbox where they were previously compiled. The files are installed into */usr/local*. The simplest technique is to create a **tar** archive of the files on the Scratchbox development machine starting at */usr/local* , copy this to the N810 (using **scp**) and expand the archive back into */usr/local* on the target N810. The only item that this does not cover is the DTN2 Python API. The files from */usr/lib/python2.5/site-packages* have to be copied separately. The files are *dtnapi.py* and *_dtnapi.so*.

Finally the linker dynamic library cache has to be updated as for Scratchbox. The Berkeley database library will install into */usr/local/BerkeleyDB.4.7* and Tcl into */usr/local/lib*. In order to link Oasys and DTN2, these two directories have to be added to */etc/ld.so.conf* and then **ldconfig** must be run in order to update the linker cache.

5.5 DTN DAEMON

Assuming that the system is using PRoPHET dynamic routing, the following configuration needs to set in the */home/user/dtn/dtn_outstation.conf* file for the DTN2 bundle agent daemon (**dtnd**):

```
# Storage Directory
set dbdir /home/user/dtn/bundlestore

# Routing type
route set type prophet

# Route local eid
# This item should use the email user name part to identify the
# node.  At present this is a constraint in DTN2 as it can only
# use a single EID.
route local_eid "dtn://<email user name>.nomadic.n4c.eu

# PRoPHET parameters
# Age out node parameters every 12 hours - 12 * 60 * 60 seconds
prophet set age_period 43200
```

```
# Decay probabilities by units of a day -
#   kappa is set in milliseconds - 24 * 60 * 60 *1000
prophet set kappa 86400000
# TCP interface (default address and port)
interface add tcp0 tcp

# Configure local neighbour discovery
discovery add ip_disc ip port=4301
discovery announce tcp0 ip_disc tcp cl_port=4557 interval=10
```

5.6 PYTHON PARTS

The Python code is written for version 2.5 of the interpreter, but should run equally well on version 2.6. It will not run with version 3.x of the Python interpreter at present. This choice was made because version 3 is not available for the N810, and also when the project was started version 3 was very immature.

The Python files are conveniently placed in */home/user/dtn/python/*. The Python parts use the 'standard' Python logging system. In the current version the log configuration file name (*/home/user/python/dtn_pymail_log.conf*) is hard coded into the source, and needs to be changed if the directory structure is altered. The names of the log files used by the two programs are in the configuration file and may need to be altered if the directory structure is altered. The logs are set to 'rotate' at midnight each day.

The Email Bundle Encapsulator/Decapsulator Postfix interface uses the following code pieces:

- dp_outstation.py
- pypop_maildir.py
- dp_pop3.py
- dp_smtpserv.py
- dp_dtn.py
- dp_msg_send_evt.py
- SocketServer2_6.py (Note: this is a copy of the standard SocketServer Python library code from version 2.6. The SocketServer code in version 2.5 is badly broken.)

It may be necessary to change some hard coded values in *dp_outstation.py* if the directory layout changes or the structure of the domain names used is changed. It is also possible to place the Python code other than **dp_outstation.py** in the Python site-packages.

The path names in the last line represent

- The root of the maildir mailbox directory tree (normally */home/user/dtn/maildit*)
- The location of the log file (normally */media/mmc2/dtn/log)*
- The name of the Python logging configuration file (normally */home/user/python/dtn_pymail_log.conf*)

Earlier on in the file (line 96), the name of the nomadic mail domain is defined.

It may be convenient to make the dp_outstation.py file executable, but this is not essential as the start up script (*/home/user/dtn/start_pymail.sh*) doesn't assume it is. The log is opened at */media/mmc2/dtn/log/dtn_pymail.log*.

5.7 CONTROL SCRIPTS AND MENU ITEMS

The PyMail system on the N810 is started using the shell script */home/user/dtn/start_pymail.sh*. This script sets up paths and environment variables, then starts the DTN2 bundle daemon, pauses for 10 seconds to allow the bundle daemon to start up and then starts the Python Email Bundle Encapsulator/Decapsulator (**dp_outstation.py**).

The PyMail system is stopped using the shell script */home/user/dtn/stop_pymail.sh*. This script finds the Python process and kills it and then stops the DTN2 bundle daemon using **dtnd-control stop**.

Hildon desktop menu items can be provided to invoke the start-up and shutdown scripts by installing descriptor files in the directory */usr/share/applications/hildon*. The menu item descriptors are in **start_pymail.desktop** and **stop_pymail.desktop**. An icon for these items is located at */usr/share/pixmaps/n4c.png*.

6. CONFIGURING A DTN RELAY SYSTEM

This section describes the configuration of the relay node that is used to link a statically routed DTN connection to a dynamically routed section, so that, for example, email bundles can be distributed to truly nomadic nodes, but using a distribution point that has a more regularly scheduled connections. The scenario was explained in more detail in Section 2.4.2.

The relay node runs two separate DTN2 bundle daemons, one configured for static routing, and the other configured for dynamic routing. Both the API and the control interface for a 'daemonised' or backgrounded DTN2 bundle daemon operate over TCP/IP links for which the TCP port can be specified as part of the configuration. Thus the pieces of Python code that relay between the two bundle daemons can attach to both bundle daemons in a controllable fashion.

6.1 EID MAPPING IN THE RELAY NODE

The EID of the static routing bundle daemon in the relay node is set to **dtn://relay.nomadic.n4c.eu**. In order to make use of the relay node, the Internet gateway node is slightly modified so that the destination EID for outgoing email bundles destined for **username@nomadic.n4c.eu** is set to **dtn://relay.nomadic.n4c.eu/username/email/in**.

Consequently all such bundles are routed to the relay node. The outgoing bundle relay Python code then registers with the static routing bundle daemon to receive all such bundles. The current implementation requires the relay to register all the EIDs for the email users that have nomadic addresses.

[This is not totally desirable. It may be possible to change the format to **dtn://relay.nomadic.n4c.eu/email/in/username**. It would then be possible to use a wildcard in the registration, which I believe works but haven't tried, allowing all possible usernames to be relayed. Of courses, it might be considered better to not make the relaying totally free - anonymous mailers are not in vogue!]

The relay code rewrites the destination EID to the form used in the standard gateway, i.e., **dtn://username.nomadic.n4c.eu/email/in**. The source and report EIDs are left unchanged (i.e., **dtn://gateway.nomadic.n4c.eu/email/out** and **dtn://gateway.nomadic.n4c.ru/email/reports**.

In the reverse direction an equivalent transformation is done on the source address, so that for example **dtn://username.nomadic.n4c.eu/email/out** is mapped to **dtn://relay.nomadic.n4c.eu/username/email/out**. The report address in these bundles should also be altered in an equivalent way.

The EID of the dynamic routing bundle daemon in the relay node is also set to **dtn://gateway.nomadic.n4c.eu**. This overloading of the name could potentially lead to problems, except that the use of different routing schemes ensures that there is no ambiguity about where to send bundles.

6.1.1 Minor Change to Gateway Code to Facilitate Relay

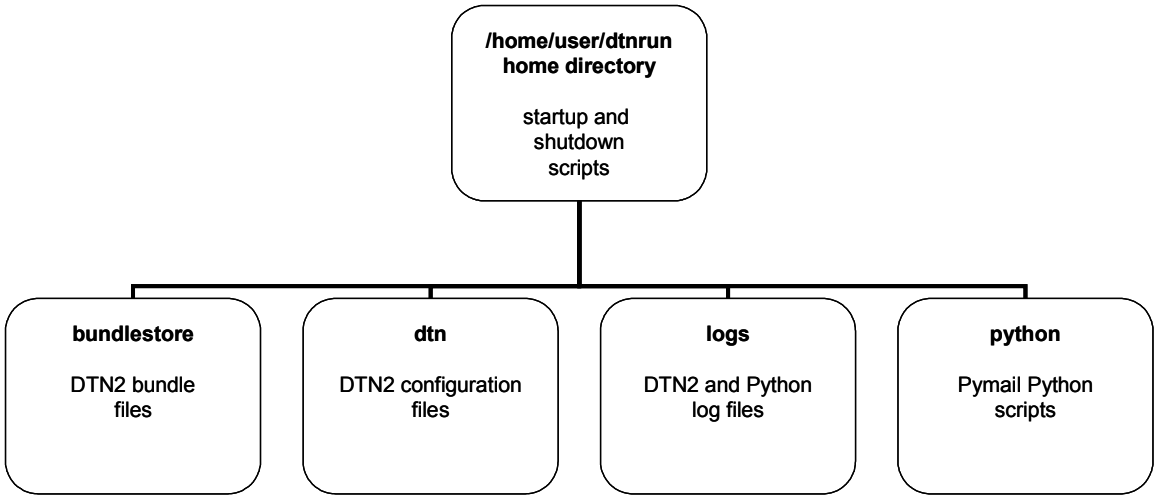
A minor change has been made to the Python code for the Internet gateway to provide the altered destination addresses needed by the relay. The files affected are *dp_dtn.py* and *dp_main.py*.

In *dp_dtn.*py the **__init__** function of the **dtn_send** class has an additional boolean parameter, **relay_mode**, which is set to select the format described in Section 6.1 and cleared to use the original format described in Section 4.1. Two lines of code added at around line 220 of the file choose the correct EID pattern based on this flag.

In *dp_main.py* the flag value is hard coded in the class constructor that creates the **dtn_send**ing thread.

6.2 USERS, PERMISSIONS AND DIRECTORY STRUCTURE

A running on a machine such as an Asus EEE-PC which has only one user, then a suitable directory (typically */home/user/dtnrun*) should be created for the relay files and the following directory structure created. On more sophisticated machines it may be appropriate to create a separate user with its own home directory and set up the directory permissions to minimise access by other users.



6.3 CONFIGURATION OF STATIC ROUTING BUNDLE DAEMON

Ensure that the Python DTN2 API is created and installed. This is not the main **`make`** command for DTN2. It is necessary to change to the *applib* directory and issue the commands **`make pythonapi`** and **`make pythonapi_install`**. The latter command may require root permissions depending on where it is writing.

The configuration file (*/home/user/dtnrun/dtn/dtn_relay_static.*conf) for the DTN2 bundle agent daemon (**dtnd**) running static routing to the village router should contain:

```
# Storage Directory
Set dbdir /home/user/dtnrun/bundlestore

# Set the address for the socket based console protocol
# This selects the 'localhost' and the default port
# Compare with the dynamic routing daemon configuration.
console set addr 127.0.0.1
console set port 5050

# Set the port for the API server
# Use the default port
# Compare with the dynamic routing daemon configuration.
api set local port 5010

# Routing type
Route set type static

# Route local eid
Route local_eid "dtn://relay.nomadic.n4c.eu

# TCP interface (default address and port)
Interface add tcp0 tcp

# Configure a link to the village router
Link add linkToVillage 192.168.2.110 ONDEMAND tcp

# Provide route to gateway node (in UK)
```

```
Route add dtn://gateway.nomadic.n4c.eu/* linkToVillage
```

6.4 CONFIGURATION OF DYNAMIC ROUTING BUNDLE DAEMON

The configuration file (*/home/user/dtnrun/dtn/dtn_relay_prophet.conf*) for the DTN2 bundle agent daemon (**dtnd**) running static routing to the village router should contain:

```
# Storage Directory
Set dbdir /home/user/dtnrun/bundlestore

# Set the address for the socket based console protocol
# This selects the 'localhost' and a non-default port
# Compare with the dynamic routing daemon configuration.
console set addr 127.0.0.1
console set port 5051

# Set the port for the API server
# Use a non-default port
# Compare with the dynamic routing daemon configuration.
api set local port 5011

# Routing type
route set type prophet

# PRoPHET parameters
# Age out node parameters every 12 hours - 12 * 60 * 60 seconds
prophet set age_period 43200

# Decay probabilities by units of a day -
#   kappa is set in milliseconds - 24 * 60 * 60 *1000
prophet set kappa 86400000

# Route local eid
route local_eid "dtn://gateway.nomadic.n4c.eu

# TCP interface (default address and port)
interface add tcp0 tcp

# Configure local neighbour discovery
discovery add ip_disc ip port=4301
discovery announce tcp0 ip_disc tcp cl_port=4557 interval=10
```

6.5 PYTHON CONFIGURATION

The Python code that carries out the bundle relaying consists of two self-contained Python programs that are executed as independent processes sitting 'between' the two bundle daemons. In each case the relay program receives bundles on one daemon and sends them out again on the other with the addresses modified as described in Section 6.1; the roles of receiver and sender are reversed between the two relay programs.

The bundle daemons are configured to use different ports for their RPC API interfaces. The client side of the DTN2 API code determines which port to use by means of the value of an environment variable (**DTNAPI_PORT**) at the time the connection to the bundle daemon is opened using **dtn_open** API call. The Python programs alter the value of the environment variable prior to opening the relevant connection.

- */home/user/dtnrun/python/dp_relay_in.py* - receives from the static domain, sends to the dynamic
- */home/user/dtnrun/python/dp_relay_in.py* - receives from the dynamic domain, sends to the static

Both programs use the same logging configuration defined in */home/user/dtnrun/python/dtn_pymail_log.conf*. The log is written to a file in */home/user/dtnrun/logs*. The file used can be altered by changing the name hardcoded into the call of the main function in each program.

The main body of each relay program is a continuously executing loop that waits for incoming bundles while sitting in a **select** system call so that the program is idle normally. When a bundle is received it is read, the header information modified and then retransmitted through the 'other' bundle daemon. The programs can be stopped with a SIGINT (Ctrl-C from the keyboard typically) which bumps the wait out of select.

6.6 SCRIPTS TO START AND STOP THE RELAY

There are scripts to start the two bundle daemons in */home/user/dtnrun/dtn/start_dtn_village_side.sh* and */home/user/dtnrun/dtn/start_dtn_N810_side.sh*. There is an overall start-up script in */home/user/dtnrun/start_relay.sh*. This starts the two bundle daemons and then after a 10 second pause to allow the bundle daemons to initialize, starts the relay programs.

6.7 TEMPORARY EXPEDIENT DUE TO PROPHET PROBLEMS

Due to problems with the DTN2 PRoPHET implementation, there is an alternative set of configurations that allow the relay node to use static routing to communicate with the outstations (Nokia N810s). This uses alternative DTN2 configuration files for the relay (*/home/user/dtnrun/dtn_relay_adhoc_static.conf*)and for the outstation (*/home/user/dtn/dtn/dtn_outstation_static.conf*).

Note that several of these configuration files contain hardcoded numerical IP addresses since DNS lookups are not available to either the outstations or the relay node. It might be appropriate to encode these addresses in the */etc/hosts* file of the node so that textual lookups are used even where DNS is not available.

7. EXPERIENCE AND PROGRESS

The development of PyMail is still work-in-progress. The reason for creating this application suite is to provide a convenient environment in which service oriented DTN naming and dynamic routing can be tested, in addition to

providing a useful application for use on portable machines in a communications challenged environment.

Work up to the end of N4C summer testing in 2009 has revealed a number of challenges which we will seek to address during the remainder of the N4C project.

This section gives an outline of the work that has been carried out up to the middle of August 2009.

The aim of this work was to provide a nomadic email system that could be trialled by the hikers taking part in the Extremecom 2009 event in the Swedish Arctic during early August.

7.1 LABORATORY WORK

During the period of May to July 2009 the PyMail application was designed and the coding of the Python interface programs was carried out by Folly Consulting Ltd.

The choice of Python for the interface programs was driven by the desire to have a system which could be simply portable to a wide variety of environments as well as a demonstration of the Python interfaces in DTN2 which have not seen much use in practical applications.

The application suite was then integrated with DTN2 and Postfix, initially using laboratory machines. An Intel Atom based desktop machine running Gentoo Linux provided the main development environment including a Postfix mail server. The outstation was initially implemented on an Asus EEE PC 1000 running Xandros Linux, before moving to Nokia N810.

To compile DTN2 for the N810 (ARM Atmel processor) the Scratchbox virtual development environment was used.

It was also necessary to install Python 2.5 on the N810s. This requires the downloading of a large suite of packages to be installed on the N810s. Given that eventually ten N810s were in use, a short cutting mechanism was used to bulk download the relevant Debian packages on to the N810, avoiding the need to download all the packages individually over the Internet.

It was intended to also load the Norut Hiker's PDA application onto the N810s but due to logistical problems with holidays and installation instructions, we were not able to install this application until very late in the day whilst out in the field.

In addition to testing the new Python programs, the laboratory testing also included work on the PRoPHET implementation in DTN2. It became clear that the PRoPHET implementation still contained a number of bugs, and a significant amount of work was required to identify some of these problems and reach a point where emails could be exchanged using the PRoPHET routing mechanism. The debugging work on this delayed some of the testing and streamlining of installation. The version of PRoPHET in DTN2 cannot be seen as satisfactory for a number of reasons: it does not conform to the

experimental standard currently being processed through the IRTF as regards the transmission of routing metadata in bundles rather than a separate connection and the code as implemented does not meet the conventions set out for DTN2.

The working assumption during this period was that it would be possible for bundles to be routed from the Folly Consulting server in the UK to the field trial area using the PRoPHET routing protocol with the bundles delivered into the heart of the field trial area via the Fiskflyg helicopter relay from Ritsem to Stahloluokta Tourist Camp. However it became apparent that the DTN2 software could only work with one routing protocol in a given bundle agent - it is not currently possible for a single bundle agent to handle different routing protocols on separate links. This proved to be a problem later and has not yet been resolved as of the time of writing.

To avoid the need for extensive configuration work on site, the N810s were each allocated a predefined email account and user name. Because of limitations in the EID naming scheme in DTN2 at present, this effectively determined the local EID of the N810 nodes. It also allowed a simple algorithmic mapping from email address to DTN EID.

7.2 DTN RESEARCH GROUP DISCONNECTATHON

Just prior to the N4C summer trials the IRTF DTN Research Group held a meeting in Stockholm, Sweden in association with the IETF 75 meeting (July 26 -31). During the IETF meeting a number of groups working on DTN implementations gathered for the ‘Disconnectathon’. The object of this gathering was to carry out interoperability trials for the various implementations of the RFC 5050 Bundle Protocol and other DTN components.

During the Disconnectathon the N810 and EEE PC implementations were exercised and some additional configuration work was done in conjunction with TCD to set up static DTN routes that would allow bundles to be routed from Folly Consulting’s ‘rosebud’ server which was providing the email gateway to the Internet through TCD’s server ‘basil’ to the machines which would be deployed during the field trials. As part of this exercise - and in conjunction with the Disconnectathon, TCD and Folly Consulting machines were interconnected in a nine hop ‘route’ demonstrating the interoperability of a number of different DTN implementations.

7.3 PRE-DEPLOYMENT WORK IN LULEA AND RITSEM

Subsequent to the Disconnectathon, TCD, Folly Consulting and LTU staff involved in the summer trials spent a week in LTU and later a couple of days at the tourist lodge in Ritsem prior to flying out to the field trial area at Stahloluokta.

During this period further work was done on the installation of the PyMail program on the N810s and the Norut Hiker’s PDA software was added to the N810s. The opportunity was taken to create menu options on the N810 to control the start-up and shutdown of the PyMail system.

A number of practical issues with both the TCD and PyMail systems became critical during this period. Both systems were initially relying on Wi-Fi ad hoc mode for certain communication links. A good deal of time was spent during this period attempting to improve the reliability of the ad hoc links. However it became clear that the ad hoc mode drivers on the various systems were individually unreliable and could not be relied upon to provide the network merge functionality needed in order to ensure that units physically coming together would be able to start a communication when they were in radio range. This resulted in a rapid reworking of the TCD village email system to use only Wi-Fi infrastructure mode. The N810s and Asus EEE PC proved to be somewhat better but it was still decided that for relaying to the Internet infrastructure mode would need to be used.

Furthermore, there were ongoing difficulties with the PRoPHET implementation in DTN2 especially in the N810s. In particular, there were difficulties with ensuring that bundles were correctly rescheduled for transmission after the bundle agent had been shutdown and restarted.

These limitations in the capabilities of DTN2 and the available hardware at Ritsem and in the helicopters meant that it proved impossible to run a PRoPHET based routing system from the PyMail Internet gateway directly to the nomadic email clients. All these difficulties resulted in the need to make some belated changes to the way in which nomadic email bundles would be delivered to the field trial area and then distributed further.

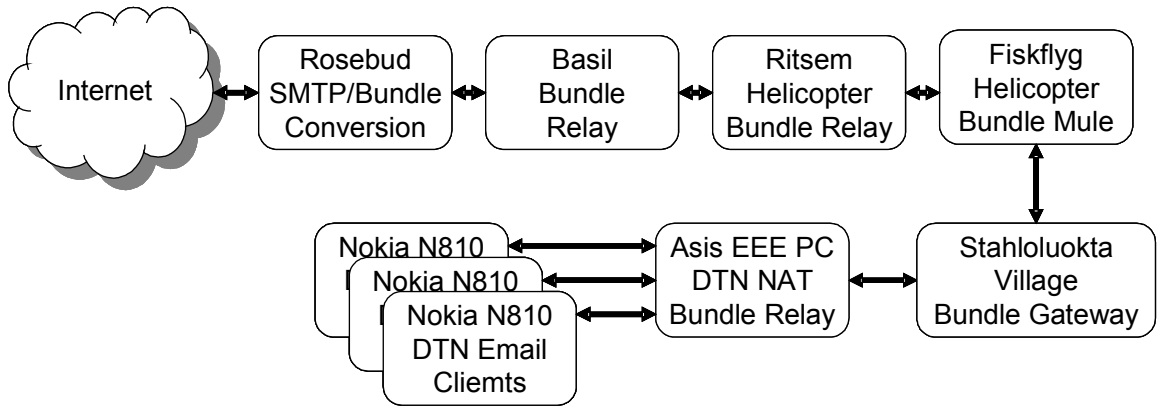


FIGURE 5 BUNDLE TRANSFER PATH DURING SUMMER TESTS

The scheme as finally deployed, shown in Figure 5, involved statically DTN routing bundles to/from the Internet gateway on Rosebud (in the UK) via Internet links via TCD's Basil gateway (in Dublin, Ireland) to the field gateway at the Fiskflyg offices at Ritsem heliport. Here the bundles were transferred over Wi-Fi to a 'data mule' machine (Asus EEE PC) that was carried in one of the helicopters on its regular (semi-scheduled) trips to Stahloluokta. On arrival, the data mule connected over Wi-Fi to the solar powered 'village router/gateway' developed by TCD and bundles were transferred again. The nomadic email bundles were then picked up from the village router by another EEE PC which acted as the base station or relay for all nomadic email within the field test area. Nomadic email clients were able to exchange bundles both with other nomadic clients and with the base station.

These changes also necessitated some changes in the way in which bundles were addressed. The original plan was to address bundles directly to the nomadic email clients when they were generated at the Internet gateway. However the amount of configuration of static routes in the links from rosebud to Stahloluokta with this scheme would have been prohibitive. Thus an address translator was implemented in the Stahloluokta nomadic email base station relay. All bundles from the gateway were now addressed to the relay. They were locally delivered to a Python program that rewrote the destination DTN address based on the email delivery address and passed them to a second bundle agent that provided the delivery to nomadic email clients.

The intention was to provide a number of the hiking participants in Extremecom who passed through Ritsem on Saturday 8 August on their way to the start of the Extremecom walking phase with N810 units. This would allow them to experiment with during the hike. Although this was put into practice, the amount of time available with the hikers before they set off was inadequate to provide training on the new applications. The weather during the hiking was also very wet and problems with limited battery lifetime meant that no significant email traffic was generated during the hiking.

7.4 WORK IN THE FIELD

After the Extremecom hikers set out form Ritsem, the Folly Consulting, TCD and LTU developers flew to Stahloluokta to set up the TCD village router and await the arrival of the hikers. During the three days before the hikers arrived, the helicopter data mule service was exercised and some nomadic emails were sent from a local station in Stahloluokta.

Ensuring that the various pieces of equipment had adequate power proved to be challenging even with reasonably good weather and a small generator.

Some further work on configuring the N810s was also carried out.

As mentioned, the hikers were unable to make significant use of the email application whilst hiking and it will be necessary to consider the best usage of the system for the future.

The whole party travelled back to Ritsem and then onto the conference venue at Saltoluokta where some additional emails were dispatched. The village router system was dismantled before leaving.

8. EVALUATION AND WAY FORWARDS

The development of PyMail and the trial deployment during the Summer 2009 field trails has provided a very useful view of the user and technical problems that stem from trying to provide seamless application integration across the boundaries of the existing Internet and DTN deployments in communications challenged environments.

In this section we summarize the successes and the issues that arose in the development and deployment of the PyMail system during summer 2009, and

then propose a way forwards that will make it a more useful part of the test beds during the remainder of the N4C project,

8.1 SUCCESSES

8.1.1 Use of Python DTN2 API

The DTN2 Python API proved to be convenient and worked as advertised. The only difficulty encountered was during cross-compilation where some adaptation of the Python distribution makefile was necessary due to some macros being set inappropriately (this is not a problem with the DTN2 distribution).

To sidestep the problems caused by the effectively half-duplex nature of the DTN2 API, separate connections to the DTN bundle agent daemon were used for sending and receiving bundles. This allows the receiving connection to be permanently ready for reception and minimises the processor cycles used by the receiver by having the receiver sitting in an IO wait state in a 'select' system call. The send and receive aspects are handled by separate threads with their own connections and consequently there is no need to break out of the receive 'select' call in order to send a bundle - the functions are essentially independent.

8.1.2 Implementation of DTN2 on Nokia N810 and Asus EEE PC 1000

The intention with both of these devices was to implement PyMail on the default operating system. For the N810 this is Maemo and for the Linux based EEE PC this is Xandros. (Some of the other systems being developed in N4C have moved to Ubuntu on the EEE PC because of problems with Xandros.)

This was reasonably straightforward. Compiling the DTN2 code did not require any special modifications to the code.

Due to the very limited memory on the N810, it was necessary to use 'stripped' executables - by default, DTN2 is linked with debugging symbol tables which makes the programs very large. Removing the symbol tables using the 'strip' command reduces the size of the DTN2 bundle agent daemon (*dtnd*) by about 90% (from approximately 25MB to around 2.5MB).

For the N810 it was also necessary to install

- the Berkeley Database system as the N810 does not support a *dbm* style database by default,
- the Python 2.5 system to allow the use of the Python scripts, and
- the TCL interpreter to support the DTN2 control connections.

Both the bundle security extensions and the XML-based extension system were omitted for the N810 build as they were not necessary for this deployment. The intention was to use the built-in memory card in the N810 to provide storage for bundles and emails but this was not completed in time for the trials, so all files were stored on the internal memory of the device. This was not a problem for these trials as data volumes were very low, but for the future this would be essential. The speed of writing is a potential problem for this memory. One solution is to place the files in main memory initially and

copy the files in the background as necessary - further work will be done on this problem in due course. Scripts to start and stop PyMail were connected to the N810 menu system (as has been done with the Hiker's PDA system).

No particular issues were encountered on the EEE PC - the standard local development environment provided all the necessary tools.

Since the trials the rationale for maintaining Xandros as the EEE PC operating system has been negated by a change of policy at Asus. Asus are no longer supporting a Linux distribution as a pre-installed option, and the Xandros support has been steadily withdrawn. This and other matters are discussed in Section 8.3.7.

8.1.3 Delivery of Emails Directly to and from a Nomadic User

A small number of emails were successfully delivered in both directions from the lab and field locations.

8.2 ISSUES

8.2.1 User Acceptance and Usability

The limited amount of testing that we were able to carry during Extremecom 2009 revealed that it is unlikely that recreational hikers will voluntarily activate a nomadic device during normal hiking. In particular requiring some action from a person carrying a nomadic device to exchange data with a another passing hiker is not likely to meet with user acceptance. The weather during a good part of the hike was inclement (heavy rain) and hikers would be unwilling to open their packs or even necessarily pause when passing others in such circumstances.

Hence any exchange of data needs to be triggered without user intervention. This has implications for the design of the system and interacts with the battery lifetime and Wi-Fi modes as described in the next two issues when in use by Hikers and other casual users.

Such casual users need to be incentivized to make active use of the device while actually hiking - it is necessary to actively try out information services on a wider scale to see if these can give sufficient incentive. On the other hand email usage at tourist villages would certainly seem to be popular.

The system was not trialled by employees for work related purpose this year. It seems that a work related system would have a greater chance of acceptance.

8.2.2 Ad-Hoc Wi-Fi Mode

The intention was to use the Wi-Fi Ad-hoc mode to connect nomadic devices to each other and the Internet relays and gateways. Although the Nokia devices and the EEE PC using Xandros performed reasonably well with ad-hoc mode, the field trials encountered a large number of issues with ad-hoc mode. In particular the connections to data mules had to be reconfigured to use infrastructure mode in most cases because of difficulties with getting arbitrary

pairs of machines of different types and with different operating systems to connect reliably.

The problem here appears to be that ad-hoc mode is not very well tested by manufacturers and software developers, so that the standard behaviour expected does not always occur. In many cases the network merge function that is supposed to occur when nodes come into range of an existing communication cluster appears not to work correctly, leading to a partitioned network. Overall the difficulties with ad-hoc mode are a major stumbling block for nomadic operation. This is a serious issue with regard to basic design parameters of the N4C framework.

8.2.3 Battery Lifetime and Application Management

The battery lifetimes of many of the devices that are potentially useful as nomadic clients are limited. Typically only a few hours of active use is possible. The problem is exacerbated by using ad-hoc Wi-Fi mode which requires periodic active beacon transmission which is a major power drain both from the point of view of radio transmission and the need to wake up the device fully.

Power drain could be lessened by requiring the user to start the application that uses ad-hoc Wi-Fi bit as discussed in Section 8.2.1, casual users would not like this sort of scheme.

Very careful attention to application design is needed to ensure that applications do not keep the node processor active unnecessarily, preventing the device from entering as deep a sleep mode as possible.

8.2.4 N810 GPS System

The Nokia N810 contains a built-in GPS system which can provide location information to aid in logging of events. Unfortunately it is very slow and power hungry. However, since nomadic email clients are mostly expected to be carried by people moving around outside buildings and vehicles, GPS location is a viable means of locating clients.

8.2.5 DTN2 PRoPHET Implementation

The PRoPHET implementation in DTN2 has a number of issues, both of architectural design and buggy implementation. A number of these bugs were identified and fixed during lab testing but it has become clear that there is a more generic issue with DTN2 which does not appear to maintain sufficient state about bundles that have or have not been transmitted when the bundle agent is shutdown and restarted. Both PRoPHET routing and static routing seem to have encountered problems with bundles either being transmitted multiple times or not transmitted at all because of this problem. This will require considerable work to ensure that shutdown/restart works as expected.

The PRoPHET implementation also does not match either the PRoPHET specification or the other main implementation of PRoPHET (resulting from the SNC project) as regards how routing metadata is exchanged between meeting

nodes. The DTN2 implementation transfers this information in bundles whereas the specification uses an out-of-band link. This leads to issues when nodes are connected for a significant length of time and should re-exchange metainfomation.

Difficulties with the PRoPHET scheme meant that most of the field deployment had to fall back to static routing which limited the possibilities of multi-hop forwarding. In practice other issues meant that this did not affect the email delivery significantly, but it obviated the idea of using the nomadic clients as data mules.

8.2.6 DTN2 Addressing

Currently, DTN2 can only route bundles to a node based on a single EID base component (similar to a host name) - the bundles can be demultiplexed based on additional components once they arrive at the node. This is inconvenient for 'service based' delivery as is needed for email addresses. It would be highly desirable to be able to route to an EID that represents an email address that does not depend on the node EID of the (nomadic) node where the user email client is currently located. Essentially a node should be able to advertise multiple EIDs, some of which refer to services as discussed in recent IRTF drafts relating to the dtn: URI scheme.

8.2.7 Heterogeneous Routing in DTN2

A DTN2 bundle agent cannot use more than one routing mechanism in a single instance. Thus a single bundle agent cannot provide a means of interlinking areas of a DTN network using static routing and PRoPHET routing. This meant that the eventual field deployment of PyMail had to use a DTN NAT scheme in which bundles were passed between two bundle agent instances in a relay node, one providing static routing through the 'village router' back to the Internet gateway and the other providing PRoPHET routing to the nomadic clients.

Some thought need to be given to how heterogeneous routing can be added to DTN2.

8.3 WAY FORWARDS

Further development of the PyMail application will require some further development both of the infrastructure DTN2 code and the custom code for PyMail. Some thought also needs to be given to usability and the user model.

8.3.1 User Model

As noted in Section 8.2.1, casual users were not inclined to use the email system while hiking, partly because of the effort needed to work with the prototype system in difficult conditions and partially because there were not clear benefits to using the system with the facilities available.

We need to work with the Norut Hiker's PDA application and information providers to see if incoming emails (e.g., regarding weather and local facilities) could be a useful 'killer application' that would move casual users to make use of the nomadic clients.

We also need to put the tool in the hands of local professionals for employment related purposes.

8.3.2 Wi-Fi and Battery Lifetime

Apart from ensuring that the PyMail application is as friendly as possible as regards allowing the nomadic clients to enter sleep mode for as much time as possible, the existing hardware does not appear to offer many options for improvement. Increasing time between ad-hoc mode beacons might reduce power consumption at the risk of missing communication opportunities.

One way forward may be to implement the system on cellular telephones that are already optimized for long battery lifetimes. However the use of ad-hoc mode Wi-Fi for communications may still reduce the effective battery lifetime even on cellular phones. One advantage appears to be that Nokia cellular phones (at least) have good ad-hoc mode implementations.

For the future it is possible that either Intel's 'My Wi-Fi' proposal or the upcoming 'Wi-Fi Direct' proposal in the 802.11 standards committee may provide a lower power consumption solution, but details are sparse and it is unlikely that commercial hardware will be available during the remaining part of the N4C project.

Finding a solution to the general problem of low power mobile neighbour encounter discovery in a wireless environment is a key research issue that would greatly enhance the utility of nomadic clients.

8.3.3 DTN Improvements – Addressing

Proposals exist to enhance the dtn: URI scheme[1]. Experimentation with these proposals will require significant changes to the DTN2 code. It is anticipated that this work will be carried out, but it is not absolutely essential to experimentation with PyMail.

8.3.4 DTN Improvements – PRoPHET Implementation

In order to make further progress with dynamic routing an improved implementation of PRoPHET is required. The DTN2 implementation requires major rework to make it fit for purpose and to implement the standard specification that is now approaching publication as an experimental standard RFC.

Folly Consulting is investigating ways to create a bundle agent that will provide the required functionality and have a smaller 'footprint' than the DTN2 implementation. See Section 8.3.7 for further details.

8.3.5 DTN Improvements – DTN NAT Workaround

Modifying DTN2 to allow it to provide different routing protocols on specified links requires a great deal of effort and may require considerable restructuring of the bundle agent. It is unlikely that this work can be carried out during the N4C project. A workaround that provides the required

[1] There are currently two DTN Research Group drafts addressing this issue: draft-irtf-dtnrg-dtn-uri-scheme-00 and draft-irtf-dtnrg-dtn-uri-scheme-00

functionality was implemented during the 2009 testing period. Separate bundle agents are deployed on a relay node with one agent providing (say) static routing for some links and another agent providing (say) PRoPHET dynamic routing for other agents. In order to make this work, an addressing scheme similar to the Network Address Translation (NAT) scheme familiar in IP networks has to be adopted. An interface program ('Application Layer Gateway' in NAT parlance) routes bundles between the two bundle agents on the relay node, and modifies the EID destination addresses to the appropriate value for the target network based on information from the email addresses in the bundle payload.

This scheme also allows different bundle agent implementations to be used for the various routing protocols which may be important if the DTN2 PRoPHET implementation is not upgraded during the N4C project (see Sections 8.3.4 and 8.3.7.)

8.3.6 Duplicate Suppression and Retransmission

Since a DTN network is not able to guarantee that an email will be delivered (e.g., if a node does not come in range of a relay before the bundle expires), it may be necessary to retransmit a email if no delivery notification is received before the bundle is likely to have expired. It is also possible that an email will be received multiple times because of DTN routing injecting multiple copies into the network. It was intended that the PyMail interface programs support retransmission and duplicate suppression. This will be implemented for the next session of testing.

8.3.7 Hardware and Operating System Developments

The hardware and available operating systems on equipment suitable for use in nomadic email clients is evolving rapidly at the moment. The Nokia N810 used in the 2009 trials is now obsolete and has been replaced by the N900 which incorporates cellular telephone technology in addition to the Wi-Fi technology in the N810 but retains the Maemo Linux operating system for the non-telephony applications. Asus EEE PCs no longer provide the Xandros operating system as standard; they are now sold with Microsoft Windows as default..

In order to cope with these developments it seems essential to have an implementation of DTN that will support the RFC 5050 bundle protocol and a small set of routing protocols (at least epidemic, static and PRoPHET routing) on an implementation platform that is deployable on both Linux and Windows platforms, and is compatible with DTN2 at the bundle protocol layer. Folly Consulting is investigating how this can be achieved by improving the SNC PRoPHET code using Qt as the implementation platform.

8.3.8 Testing in a Less Power Challenged Area

Testing the PyMail application in the Sámi areas proved a salutary experience. The basic environmental challenges of power availability and weather tend to overwhelm the protocol and implementation issues. It seems important to do considerable additional live testing in an area where power is less of an issue

to ensure that the application is stable and dependable before retrying in an environment with the full set of challenges. This would give the best chance to determine the usability of the application for the future.

9. SUMMARY FOR NOMADIC EMAIL APPLICATION

Development of the PyMail Nomadic Email system has demonstrated that the nomadic email system has potential, but considerable work is needed on the DTN infrastructure to make it a realistic, deployable system. The work during the Summer 2009 testing campaign has highlighted a number of areas where work is needed and a plan for development and further testing, especially in less power challenged areas, is being studied.