An N4C DTN Router Node Design

Stephen Farrell[1], Stefan Weber[1], Alex McMahon[1], Eoin Meehan[1], Kerry Hartnett[2]
Trinity College Dublin[1], Intel[2]
stephen.farrell@cs.tcd.ie

Abstract

We describe the design for a Delay Tolerant Network (DTN) node to be used in the N4C project's summer 2009 trials to be held in conjunction with the ExtremeCom workshop. The design uses COTS products and applications; the DTN2 implementation of the Bundle Protocol (BP) and provides basic network access, e-mail and web access via the DTN.

1 Introduction

We outline the hardware and software design for one type of DTN [1] router to be used in N4C testbeds. This paper is a shortened version of an N4C project deliverable to which the reader is referred for further details. [2] The goal of the N4C project [3] is the development of a lasting testbed for Delay- and Disruption-Tolerant Networking. The testbed will be validated via a set of pilots over the duration of the project. The overall approach of the N4C project is to use a spiral development model with bi-annual trials, with each iteration evolving the design of the DTN components and applications.

The DTN router described here is a battery/solar-powered node assembled from common-off-the-shelf (COTS) products that provides DTN store-and-forwarding and application services as described below. The DTN router is designed to be initially used in the N4C summer village trials that coincide with the ExtremeCom workshop as the core of a local network.

2 Selected Hardware

We use the PROTEUS board [4] from Eurotech as the core of the DTN router. This has 2GB of on-board flash memory for Operating System and applications. Additional storage uses an SD card with 4GB+ of storage. Networking is based on an internal WiFi AP (Microtik) [4] and wireless card (Engenius) [5] that are housed in the same enclosure as the PROTEUS with an external antenna.

We use a lead-acid gel battery [6] for reasons of safety, cost and capacity. These each have a 7AH capacity, and the enclosure allows for up to three batteries to be included giving a total of 21AH power available. "Full-on" power consumption for the PROTEUS+networking is 2.35A, meaning that the unit will fully discharge in less than 9 hours of "full-on" operation. Power consumption in various "sleep" modes is currently being tested but is estimated to be around 0.45A which maps to slightly more than two days. The power budget clearly requires both re-charging the batteries and operating according to a sleep/wake duty-cycle. The management of the duty-cycle is currently being investigated.

The solar panels [7] can produce 20W at 12V and can be added in series (up to 4 per enclosure). In addition to the solar panels, we use a solar charger that connects the batteries, solar panels and PROTEUS board, routing current as appropriate. The enclosure selected is an IP66 mild-steel box from Eurobox. [8]

3 Software

The PROTEUS board runs Ubuntu mid-8.04 (Hardy) with a 2.6.24-lpia kernel. The DTN router acts as a WiFi access point and hot-spot for local users that have their own laptops, issuing IPv4 addresses via DHCP and offering some static web pages with instructions as to how to use the DTN aspects of the network. Apache and Squid are used for web content, for mail we use either Postfix or Exim as a mail transfer agent (MTA) and Dovecot as message store (MS) accessed via IMAP. Mail and HTTP requests are encapsulated using the BP between the DTN router and a gateway machine that is well-connected to the Internet. Once HTTP requests are received at that gateway, then we use wget and puf in order to crawl for responses that are then returned to the village router, again via the DTN.

In the summer 2009 trials in Sweden the main path for DTN transfers between the DTN router and the gateway will be via a data mule (a netbook) on the helicopter that periodically services the village. We use the DTN2 (version 2.6) implementation of the BP.

4 E-Mail Operation

The basic idea for mail is that users create a new mail account with a first-come-first-served email address in our mail domain for use during (and around) the trials. Users are responsible for setting up forwarding of their existing mail to this new account as desired. We provide a "bastion" MTA and MS on the gateway that handles these new mail accounts in the usual fashion.

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

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

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

Each mail submitted from the village is separately encapsulated in a bundle using the BP. A simple script called from the DTN router's MTA uses the existing dtnsend application in order to encapsulate and send the bundle to an endpoint identifier (EID) representing the bastion MTA running on the gateway.

Once a bundle arrives at the bastion MTA endpoint, the existing dtnrecv application accepts the bundle from the dtnd, and the decapsulated mail message will be submitted to the bastion MTA via SMTP.

The MS uses Maildir format, so our task becomes one of file synchronisation of the bastion MS content and the DTN router MS content. This is done by a cron job that picks up file changes on each MS instance and sends those to all relevant other instances. For a village DTN router MS instance, that involves sending to the DTN endpoint identifier (EID) of the bastion MS instance. For the bastion MS, that will in future involve sending to a non-singleton EID that represents all village MS instances, but currently involves a singleton EID. Files that require synchronisation are detected via their modification times. We encapsulate both the file name and the file content (if the content has changed) or just the file name if the content remains the same. Encapsulation will be as a compressed tarball in both cases, with a zero sized file if only the file name has changed. (That occurs when a file is read for example.)

Files are sent via the dtnsend and dtnrecv applications as described earlier.

5 Web Access

The basic principle followed is to make as much as possible available to as many users as possible with as little work for those users as possible. So users with a web browser can simply issue HTTP requests, which are re-directed to the Apace instance on the DTN router and offered the chance to submit their request via the DTN, and asked whether they consider the request private or not.

The privacy issue arises because with very few users and very high transaction latency, being able to see what's in the local cache can easily expose who has been accessing what. We use cookies to bind specific transactions to specific browsers without requiring users to set up web accounts. Access to private cache entries requires the browser to present the relevant cookie. Users that don't accept cookies can still see all public content and can request additional public content, but cannot request private transactions.

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

Each HTTP "transaction" has a transaction ID associated with it, added as a new HTTP header (X-DTN-trans-id) to the request and is included with the response bundle payload. For each request the gateway does some crawling and the set of responses are place in a tarball named using the transaction ID. So we encapsulate each HTTP request in a single bundle, but an entire set of HTTP response payloads is encapsulated into a single response bundle. (In future we may add a BP extension block describing the URLs contained in the payload.) The heuristics to be used for crawling are still under development.

6 Conclusion

We have outlined the design for a DTN router node being developed for the N4C project. Planned future work includes field-testing this design and extending it in various ways, e.g. adding support for less static web content (AJAX) and additional applications (IM via jabber).

7 References

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

[2] Farrell, S. et al, "N4C project deliverable 5.1 "N4C DTN Node Design," May 2009. (work-in-progress)

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

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

[4] http://routerboard.com/popup.php?kods=92

[5] http://www.irishwireless.eu/shop/item.aspx?itemid=268

[6] http://ie.farnell.com/camden-electronics/beg120075/battery-europa-gel-12v-7-5ah/dp/5085329

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

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

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