A Delay and Disruption Tolerant Transport Layer Protocol

Stephen Farrell

A thesis submitted to the University of Dublin, Trinity College
in fulfilment of the requirements for the degree of
Doctor of Philosophy (Computer Science)

September 2008

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work. I agree that Trinity College Library may lend or copy this thesis upon request,

[illegible]

Stephen Farrell
Dated: September 15, 2008

Acknowledgements

First, foremost, and always, thanks to Fiona, Pete, Aengus and Jenny for literally everything that means anything.

My work within Trinity College Dublin has mainly been carried out under the various SeNDT contracts. In addition to the author, the collaborators in the SeNDT project included Vinny Cahill, Dermot Geraghty, Ivor Humphreys, Paul McDonald, Barbara Hughes and James Garland. Enterprise Ireland funded SeNDT under their Commercialisation Fund programme.

The core of this thesis is an extension to some work that has involved collaboration with Manikantan Ramadas from the Indian Space Research Organisation, (formerly with the University of Ohio), and Scott Burleigh from NASA JPL, but also with other participants in wider DTN community.

Many thanks to Professor Vinny Cahill, my thesis advisor, for his patience and willingness to put up with a fair bit of meandering along the way. Thanks also to Dr. Christian Jensen for his support during my initial months here.

Finally, thanks to the college itself – whatever that means – but it's certainly a special place.

Abstract

The area of delay-tolerant networking (DTN) is concerned with highly challenged networks in which application layer "sessions" commonly lack contemporaneous end-to-end connectivity. For example, efforts to create networks involving Mars orbiters must deal with visibility constraints that mean that devices are not contactable for extended durations. Some wireless sensor networks, (where sensors are only occasionally contactable, for power management or location reasons), also impose similar networking challenges.

Currently, the most important technique used in delay tolerant networking is based on the overlay network approach as exemplified by the bundle protocol. The bundle protocol is essentially a store-and-forward protocol that can be layered above a range of lower-layer protocols via a, (so-called), convergence layer. Convergence layers often use standard transport protocols, like UDP or TCP, but sometimes make use of more esoteric protocols, such as a file based transport via USB token. There is also a delay-tolerant point-to-point protocol called the Licklider Transmission Protocol (LTP) that is designed to be used by a convergence layer on very high latency links, as might be encountered in deep-space communications.

As an overlay network that has to be able to run over various lower layers, the bundle protocol cannot be optimally efficient in all such cases. Sometimes, fields within the bundle protocol will replicate information present at lower layers. This is an inherent consequence of being a lower-layer agnostic overlay and is not in itself, a flaw. However, it does mean that there is an opportunity to develop an alternative protocol that is more tightly bound to some specific lower-layer (e.g. UDP or IP), and hence can be more efficient in cases where that lower-layer is in use for all "hops" between source and destination.

This thesis describes such protocols, which we term DTN transports, and specifes a specific DTN transport protocol, (a variant of LTP), called LTP-T, for "LTP transport." There are a number of other benefits to addressing DTN requirements at, essentially, layer 4 of the OSI reference model. With LTP-T for example, we specify a more deterministic placement of the nodes in the store-and-forward network where storage occurs, thus leading to more reactive error recovery in many cases.

Thus, the main thesis of this work is that DTN transport protocols generally, with LTP-T as a proof-of-concept, provide an alternative to the overlay approach, (as exemplified by the bundle protocol), that is sometimes better suited to handling delay and disruption. Our conception of a DTN transport protocol is the substantive contribution of this thesis to the area of delay-tolerant networking.

To evaluate LTP-T we constructed a network that realistically emulates a deep-space network consisting of nodes on Earth, and on and around Mars. Our results show cases where LTP-T outperforms the bundle protocol (running over UDP). Our main performance metric is goodput, which is the number of application-layer bytes successfully transferred per unit of time where communication is possible. Based on this emulation, and other tests, we derive a characterisation of a set of situations where each of the protocols is a better option.

Table of contents

List of Figures

List of Figures (Cont'd)

List of Tables

List of Equations

Chapter 1

Introduction

Delay-tolerant networking (DTN) is concerned with highly-challenged networks in which application layer sessions commonly lack contemporaneous end-to-end connectivity. [FA03] The most important current approach to DTN is centred on an overlay protocol called the bundle protocol (BP). [SC07] In this thesis we develop the concept of a DTN transport protocol and document the design and evaluation of one such protocol, called the "Licklider Transmission Protocol – Transport" (LTP-T). [FA05] We show that a DTN transport has advantages when compared to the overlay approach and ultimately characterise a number of cases where LTP-T and the BP perform differently.

1.1 What is a DTN?

A couple of examples may help the reader understand the kinds of networking challenges that DTNs are aiming to address. One of the main factors motivating the use of networking for deep-space communications is the fact that communications between Earth and a Mars lander are more mass and power efficient when data is sent via an orbiter. [FR04] Efforts to create networks involving Mars orbiters and landers and Earth stations clearly encounter extremely high latency links, but must also deal with visibility constraints that mean that devices are not contactable for extended durations.

For example, there are regular periods when Mars is near the Sun for a number of weeks, and during which very little or no effective communication can take place. [AL98] A more common, though less extended, disruption occurs due to rain at an Earth station, which can disrupt scheduled communications [TO03] leading to re-transmissions being required, and such re-transmissions may have to wait until the next contact opportunity, which could be hours later.

The Sensor Networking with Delay Tolerance (SeNDT) project's planned lake-water quality monitoring network [FA06] involves the deployment of a wireless sensor network in which the sensors are only contacted occasionally primarily for power management reasons. This creates networking challenges that are similar to those described in the deep-space communications scenario above. For example, SeNDT water-quality sensors deployed in a lake use a close-range radio frequency (802.11b) [IE99] and will only occasionally be visited by a node/router – the data-mule - with (eventual) Internet connectivity.[2]

It is worthwhile noting that, since the SeNDT nodes will typically only contact the data-mule (a boat), [LE05] this constitutes a different networking scenario from the usual Mobile Ad-hoc Network (MANET) case, where any pair of nodes may be the source and destination for a packet. In the SeNDT case, the data-mule is planned to be a fishing boat. If the sink node (the destination for sensor data) is back on shore, somewhere on the Internet, and is only connected to the sensor(s) via the boat then the effective latency of the sensor/sink application layer will be similar to that of an Earth/Mars link since the boat must move before the relevant traffic is delivered.

Such a network can also suffer extended periods when no boats come in range of sensor nodes, e.g. in the lake where SeNDT nodes have been prototyped, during winter flooding tethered boats are sometimes under water for weeks. Such periods are similar to the Martian solar conjunction. Disruptions due to radio frequency (RF) shadowing or rain will similarly occur at times and the use of multi-hop sensor node to sensor node communications is also similar to the use of orbiters as data relays, though in the SeNDT case this is done to increase the area connected rather than to save power or mass.

The first example demonstrates that there is at least some need for DTNs and the second shows that DTNs need not be solely concerned with deep-space communications but can also be useful in terrestrial networks. We will see further argument along these lines later.

---

[2] SeNDT is a TCD project in which the author is active http://down.dsg.cs.tcd.ie/sendt/ (Throughout the thesis, URLs are added as footnotes.)

1.2 Major DTN Activities

This work is not taking place in a vacuum - there is an Internet Research Task Force (IRTF) DTN research group (DTNRG[3]) working to develop (experimental track) Internet RFCs for DTN protocols. That group had its genesis in a previous IRTF research group on an Interplanetary Internet or Interplanetary Network (IPN) that itself grew out of some work done in the late 1990s and thereafter largely by NASA on their future network requirements.

In 2003, the US Defence Advanced Research Agency (DARPA) issued a call for proposals[4] for what they called "disruption-tolerant networks" thus creating a potential cause of confusion. However, the DARPA change of terminology crystallised the fact that delay is not the only source of disruption in terrestrial networks. Link-specific delays are in fact, probably the exception, and disruption is really the more general term.

For the present though, the acronym DTN can be expanded either way and both expansions are considered to be essentially equivalent. In fact, for the purposes of this thesis we will generally not distinguish between delay and disruption, since one can justifiably consider disruption to be the more general term, and regard delay as simply an extended disruption.

The current emphasis for many DTN projects is focussed on developing and trialling the basic DTN protocols (e.g. the BP) and their implementations. A good example is the DARPA-funded SPINDLE[6] project that is building BP-based prototypes for use by the US military, with a focus on the development of a modular architecture for the BP stack implementation. There have also been a few initial investigations [AK04,HA06t] (described later) into issues related to the use of transport protocols in a DTN environment, but to date, DTNs have been of marginal interest to the broader transport community, for

[3] See http://www.irtf.org/ and http://www.dtnrg.org/ It should be noted that the author has been, and continues to be, actively involved in the DTNRG (as co-chair) and also (as co-author) in the development of the LTP specifications and (again as co-author) in work on BP security.

[4] http://www.darpa.mil/sto/solicitations/DTN/

[6] http://www.ir.bbn.com/projects/spindle/index.html

example, Fall and McCanne [FA05j] quite properly discuss currently important transport performance issues without any mention of DTNs.

More broadly, the overall level of scientific interest in DTNs has been demonstrated by the holding of a Dagstuhl seminar [BR05] on the topic in 2005, with another planned[7] for 2009. DTNs have also been the topic for a number of SIGCOMM workshops.[8] The June 2008 IEEE Journal on Selected Areas in Communications,[9] was a special issue dedicated to DTNs, and included a paper, co-written by the author, setting out an architectural retrospective view of DTNs. [FA08j] A special issue of the Elsevier journal on Computer Communications on DTNs, (with the author as one of the guest editors), is also currently in preparation.[10]

1.3 Existing DTN Protocols

In order to understand the approach taken in this thesis one must of course first consider existing work on DTNs. As stated earlier, today's mainstream approach to DTN is based on the BP. [BU03, FA03, SC05] The BP is essentially a store-and-forward protocol that can be layered above a range of lower-layer protocols via so-called convergence layers (CLs). [FA03] CLs often use standard transport protocols, like UDP or TCP, but sometimes make use of more esoteric protocols, like file-based transfer via USB token [DE04].

There is also a delay-tolerant point-to-point protocol, called the Licklider Transmission Protocol (LTP), [RA08] that is designed to be used as a convergence layer on very high-latency links, as might be encountered in deep-space communications. In order to deal with such latency, without first requiring negotiation or other handshakes (that waste scarce bandwidth), LTP assumes that the peers share "massive" amounts of state, [BU08] for

---

[7] http://www.dagstuhl.de/en/program/calendar/semhp/?semnr=2009071

[8] E.g. http://www.sigcomm.org/sigcomm2005/cfp-wdtn.html and http://www.dritte.org/nsdr08/

[9] http://www.jsac.ucsd.edu/Calls/delayanddisruptioncfp_extended.pdf

[10] http://www.elsevier.com/authored_subject_sections/P05/CFP/cfp_dtn.pdf

example, knowing when the peer will be listening for transmissions (modulo the light-trip-time between the peers).

As an overlay network that has to be able to run over various lower layers, the BP cannot be optimally efficient in all such cases – sometimes, fields within the bundle protocol will replicate information present at lower layers. To take a trivial example, a cryptographic checksum could be added both in the BP [FA07s] and by a CL running over IPsec [KE05]. This is an inherent part of being a lower-layer-agnostic overlay and is not in itself a flaw. However, it does mean that there is an opportunity to develop another protocol that is more tightly bound to some lower layer(s), and hence may be more efficient in cases where that lower layer is in use for all hops between source and destination.

The BP also introduces the concept of a "custodian" which is an overlay node at which packets (called bundles) are stored, before being forwarded later. The BP furthermore defines bundle reports and various acknowledgements that travel towards the source in the network. The fact that the BP is an overlay introduces a need to decide where (topologically) to locate such functionality and introduces a requirement to make sure those nodes remain available as required, even possibly in the face of an adversary. [AB03] Such difficulties are actually inherent in any overlay where additional functionality is being placed inside the network, instead of at the edges, as would be the case if one could follow the end-to-end principle more faithfully. [BL01, RE98, SA84]

LTP of course, cannot offer end-to-end services, since it is designed purely as a point-to-point protocol. So LTP cannot by itself be an alternative to the BP since its packets lack, for example, a destination field. LTP can however, be layered on top of UDP and used directly by applications.

We therefore have both an overlay protocol and a point-to-point protocol, thus leaving open the space for the development of a protocol filing the gap where we would normally use a transport layer.

1.4 Problem Statement

The main problems considered here are the consequences of choices related to the location of functionality (e.g. re-transmission) within a DTN and how those choices interact with various protocol features (e.g. reliability or security). The challenge faced is to explore this design space to develop protocols that meet a sufficiently broad set of requirements so as to warrant the wider-scale testing of those protocols with the ultimate goal of eventually incorporating DTN features and functionality into the standard suite of Internet protocols. This thesis represents yet another step along that road.

While there have been some attempts to examine meeting DTN requirements at the transport layer, there has yet to be a fleshed-out demonstration that this is possible and that, as an architecture, the result provides comparable performance to the overlay approach. This is the specific question that this thesis attempts to answer.

The methodology followed is to review the main causes of delay and disruption in order to derive a set of protocol requirements that are specific to the DTN context[11] and to examine how existing protocols, and in particular the overlay approach as instantiated by the BP, meet these requirements.

Based on this review, we define our concept of a DTN transport protocol and analyse in detail how that compares with the overlay approach. We then describe one DTN Transport protocol (LTP-T), developed specifically in order to test the DTN transport concept. We describe a set of tests and results in order to demonstrate: a) that LTP-T is a viable DTN transport protocol, and b) that in some cases LTP-T can significantly outperform the BP. We finally consider how well LTP-T meets our set of DTN specific requirements in order to point the way forward for future work in the area of DTN transport protocols.

[11] We do not specifically discuss generic protocol requirements, e.g. lack of livelock, since those do not help distinguish between the relevant options.

1.5 The Contribution

In this thesis, we propose what we call a "DTN transport" to mean a protocol offering end-to-end services similar to current transport protocols and that meets the requirements imposed by DTN (derived in the next chapter).

We use this term because it reflects our view that DTN protocols will commonly be layered over IP (or UDP/IP), as the thin-waist [CR99] of the network. Our DTN transport concept also follows two additional principles: all nodes adjacent to disruptable nodes implement the DTN transport and secondly, all nodes that implement the DTN transport provide storage. For example, a DTN transport will use the next layer down (UDP or IP[12]) to get as far as possible towards the destination. One result is that only nodes that are adjacent to hops that are subject to delay or disruption need to implement the DTN transport. Equivalently, if all-to-all traffic is to be allowed, then all nodes that are adjacent to hops that are subject to delay or disruption must support the DTN transport.

In this way, we reduce the options in comparison to the overlay approach, thus making the network more deterministic and predictable and therefore increasing the reliability of the overall network.

The concrete DTN transport protocol that we use to test this architecture is LTP-T which is in fact designed as a set of extensions to LTP, using the standard LTP extensibility mechanism, [FA08] for example, one such extension being a "destination" field. An end-to-end LTP-T path then consists of a sequence of LTP hops, but with LTP-T we also mandate that those LTP sessions run over UDP (or IP), and that LTP-T nodes should be placed adjacent to hops that are liable to suffer from delay or disruption.

So, the main thesis of this work is that the DTN transport architecture is an alternative to the overlay architecture (so that LTP-T is a possible alternative to the bundle protocol) that is, under circumstances that we identity later, better suited to the task in hand. Our

[12] From our DTN transport point-of-view, we can regard layering above UDP and IP as essentially equivalent, in fact the main choice between these relates to implementation – whether to write kernel or user-space code.

conception of a DTN transport protocol is the substantive contribution of this thesis to the area of delay tolerant networking.

To test whether LTP-T is a viable DTN protocol, we constructed a network that realistically emulates a deep-space network consisting of nodes on Earth, and on and around Mars, and demonstrated that LTP-T can operate in that environment. We also carried out a range of tests that show cases where LTP-T outperforms the bundle protocol (running over UDP). Our main performance metric for these tests is goodput, which is the number of application-layer bytes successfully transferred per unit of time where communication is possible. Based on this emulation and other tests, we derive a characterisation of a set of situations where each of the protocols is a better option.

To summarise, the main contributions of this thesis are:

- The definition of the DTN transport concept and the elucidation of requirements for a DTN transport protocol.
- The design and implementation a DTN transport protocol, and the publication of source-code for that implementation.
- The evaluation of that design and implementation via a network emulation encompassing a realistic Earth-Mars networking scenario.
- The development of an LTP implementation concept, built around the idea of punctuated time (described in Chapter 4).
- To examine some of the performance implications of the BP and LTP protocols.

1.6 Scope

Not everything connected to DTNs or transport protocols is in scope for this thesis. So, in order to avoid misunderstanding it may be useful to specifically draw attention to a few things that are out of scope.

As stated above, DTNs cannot assume normal end-to-end connectivity, as is required for example by TCP (see section 2.4.2 for more details of how TCP and variants are problematic in DTNs). This means that we need not consider many of the TCP variants that have been proposed for use on the terrestrial Internet.

To be clear, we are not proposing LTP-T as a generic transport protocol, but only as one suited for use in DTNs and so we don't need to compare LTP-T against the full panoply of actual and proposed transport protocols. Were LTP-T ever to become an Internet standard[13], then it would presumably require a very clear applicability statement covering this, as was for example, done for the stream control transmission protocol (SCTP). [CO02]

Routing and security for DTNs are areas where the hard work is only beginning. Whether and to what extent routing schemes for MANETs [XI02] or other ad-hoc networks will be suitable for use in DTNs is an open question. And while there has been some work on DTN security [FA07s, SY07] , that has not so far encompassed the development of any DTN-friendly key management schemes – current key management schemes generally require multiple exchanges which are problematic in DTNs. [FA06s]

One could also envisage applying some of the broad range of techniques used to design autonomous systems to DTNs, for example, methods to predict future connectivity based on historic patterns. [CU02] Such optimisations can improve the performance of the protocols that are considered here, but their application is not part of this study since we are mainly concerned with demonstrating that the DTN transport architecture is a realistic alternative to the overlay approach.

[13] The BP and LTP specifications are Experimental track RFCs, and hence are not Internet standards, though they are Internet RFCs. The LTP-T specification has not, so far, been published in Internet-draft form.

1.7 Document Layout

The remainder of this thesis is laid out as follows:

- Chapter 2 describes the general background including causes of delay and disruption, derives a set of requirements to be met by a DTN transport protocol and discusses relevant existing protocols.
- Chapter 3 describes the on-going work in the DTNRG on the DTN-specific protocols being developed there – the BP and LTP.
- Chapter 4 analyses the reasons for wanting to use a transport solution for DTN.
- Chapter 5 describes the LTP-T protocol and its implementation.
- Chapter 6 documents our evaluation of LTP-T.
- Chapter 7 draws conclusions and describes planned and potential future work.

Chapter 2

Background

In this chapter we present the background to delay and disruption tolerant networking (DTN) and the current state of the art in this field. The presentation here is mainly based on our book[1] on DTNs [FA06] and on a related article in IEEE Internet Computing [FA06i].

We begin by describing the basic problem with using standard transport protocols in a DTN and outline some applications that could make use of DTNs. We then go on to describe some common (and exceptional!) causes of delay and disruption and from this, we derive the DTN-specific requirements that DTN protocols should meet. (We do not list more standard protocol requirements, such as the lack of livelock etc.)

We then describe a range of approaches to DTN-like problems that have previously been attempted and why they each fail to meet some important DTN protocol requirement. In this way, we provide evidence that some new architecture is needed, be it an overlay or our DTN transport concept. In Chapter 3 we look in detail at existing DTN protocols and subsequently, (in Chapter 4) analyse why this leaves room for an alternative architecture to be developed – what we call a DTN transport.

2.1 The Basic Problem

The main justification for the development of any DTN protocol depends on there being interesting networking scenarios in which important Internet protocols cannot be used. We begin with one, critical, example – the fact that the Transmission Control Protocol (TCP) [PO81] sometimes breaks. TCP underlies most of the applications we use every day – email, the web and enterprise single-sign-on to mention a few. So if we have a set of

[1] The book goes into significantly more detail on most of the material from this chapter – we have tried to take the approach of selecting sufficient of that material to motivate and understand the remainder of the thesis.

application scenarios that are interesting, and where TCP simply cannot work due to delay or disruption, then we will have established the basic need for DTN protocols. Of course TCP is not the only transport protocol currently defined, but as we'll see later, none of the current standard transport protocols meet the set of requirements imposed by DTNs.

2.1.1 Martian Networking

Much of the work described in this chapter has its roots in a research project to develop an Interplanetary Internet (IPN) [BU02, AK04i]. The basic idea was to try to make data communications between Earth and (very) remote spacecraft about as easy as between two people on different sides of the Earth. As it happens, before any application data can be sent using TCP there is a handshake required that consumes one round-trip time (RTT). TCP also specifies a number of timeouts, the longest of which, the default user timeout, is specified to be five minutes.

The result is that if no data is sent or received for five minutes the connection is broken. Putting those facts together, once a spacecraft is more than two and a half minutes away in terms of light-trip time (LTT), then every attempt to use a TCP connection will fail, and no application data will ever be transferred. In the case of Mars, at its closest approach the RTT is about eight minutes, with a worst-case RTT of about forty minutes. The result is that standard TCP cannot ever work for Earth/Mars communications. There are, in fact, other reasons why TCP will fail at Earth/Mars distances, but one is sufficient. So, there is, (at least), this deep-space application where we cannot use one of the most important standard network protocols, and where there are DTN networking requirements to meet.

2.1.2 Some Terrestrial Applications

Even if a Martian network cannot use TCP, that, in itself, might not be seen as sufficient reason to be interested in going to the trouble of developing DTN protocols. However, there are also terrestrial networking situations where DTN protocols are required, and in this section, we will outline a couple of these. One reason for including these here is basically so as not to create the impression that DTN is really just IPN.

One application of the Sensor Networking with Delay Tolerance (SeNDT) project was mentioned in the previous chapter. In addition to dealing with lake water quality, the same hardware platform is also used for a noise monitoring application. Monitoring noise in urban areas is typically required in order to ensure that traffic and other noise remains within acceptable limits. Normally such monitoring is carried out by local government authorities and involves a monitoring station that can either continuously sample and store ambient noise or else can record events such as when noise levels cross thresholds. In this case, the end users were Dublin City Council and the Irish National Roads Authority, so the end-user requirements were real, not invented.

Traditionally, such monitoring has required the use of highly expensive microphones connected to simple data-logging equipment; however, some colleagues (also in Trinity College Dublin) are now exploring the use of arrays of less expensive microphones, but with more capable data processing. [MC07] Their hope is that such a monitoring station will produce results which are of an equivalent quality, but at a much lower cost per station. The unit then calculates various noise statistics that can be sent back via various communications methods, e.g. SMS, GSM data service, or WiFi.

The DTN requirement comes with the addition of two further end user requirements, first that they be able to re-locate nodes periodically and second, that they be able to occasionally get access to actual noise samples, in particular those associated with peaks (i.e., loud noises). The first requirement implies a need for flexibility in how data is returned and rules out some options (e.g. using landlines), and the second implies a need to be able to occasionally return large volumes of data, but in a cost-effective manner. Taking the above requirements, and the fact that the data being gathered is not time-critical, this application is very well suited to using a DTN protocol.

There is another context in which similar monitoring is done, but with somewhat of an opposing set of requirements for data transmission. In this case the context is planning of future infrastructure such as roads or new airport runways where the requirement is initially to monitor the quiet, and not the noise. However, we can clearly use mostly identical equipment though we are now much more likely to locate stations in areas without wired power and even perhaps out of coverage of commercial data networks.

There is also a related application worth noting here – a sensor network for detecting cane toads in Australia [SH04] that makes use of a mixture of very small sensors (so-called motes[7]) as well as more capable nodes. The application requirement here is to detect the extent of the spread of the cane toad, which is an introduced species that is considered to be damaging to the local environment.

The toads are detected via an analysis of their calls, with the motes including microphones and the more capable nodes running the analysis software. In this context, the authors estimated that they need a mote in each 20m diameter hexagonal cell and one more capable node for each 2,000 motes. Thus they calculate a requirement for about 1,000 motes per square kilometer.

Clearly in order for this to be feasible the cost of the motes (including batteries, provisioning, and deployment costs) must be very small. The authors also don't address how to gather up the thousands of motes at the end of the experiment – one of the perhaps least considered aspects of many sensor networks.

The main benefit from a DTN protocol (and application) here would be that it simplifies the application layer code by handling disruption in the stack. In this case, we would need whatever protocol is used to be present on various hardware platforms, which is yet another good reason to investigate using a "standard" set of DTN protocols.

The above are only a couple of representative sketches of how DTN protocols can solve real problems. Our DTN book [FA06] covers these and many additional applications, for example, bus-based networks, networking in developing regions, DTNs as a backup or enhancement for some emergency services and many more.

In most of these terrestrial cases, individual network hops (point-to-point connections) do not suffer overly from delay or disruption[8], so one could well develop an application layer solution where the end-to-end delays and disruption are handled "outside" or "above" the network. However, if one did develop useful end-to-end DTN protocols then that could

---

[7] http://www.tinyos.net/
[8] Basically, this is why the deep space application remains the "poster child" DTN.

save repeated development of application layer code for each new application thus saving effort and presumably increasing the overall consistency of the application and of the network.

2.2 Causes of Delay and Disruption

We now concentrate on some common sources of delay or disruption, rather than on handling delays and disruption.

By delay we mean the end-to-end latency of data transmission. Some of those delays are inherent in the transmission medium, or the geometry of the system, but others are due to packets being temporarily stored on intermediate nodes.

By disruption, we mean factors that cause connections to break down, or not be established, normally due to transient or quickly changing aspects of the system and/or its environment.

Table 2.1 summarises the causes of delay and disruption described in this section that give rise to our set of DTN requirements.

Cause	**Section**
• Finite speed of light and the scale of the solar-system (light-trip time) • Planetary (and other) movements do not conform to simple, regular patterns • Positioning inaccuracy (N-body problem) • Visibility restriction in line of sight communications • Over-subscribed communications systems	2.2.1
• Power conservation (on/off/sleep modes) • Radio range restrictions to conserve power • Long-duration outages (e.g. snow on solar panels)	2.2.2
• Node movement out of coverage area (incl. Infrastructure nodes) • Long application-driven duty-cycles • Network interface selection decisions	2.2.3
• Lack of application activity • Beacon-based node wake-up	2.2.4
• Malware/Denial-of-Service (DoS) • Stealth requirements	2.2.5

Table 2.1 – Summary of causes of delay and disruption.

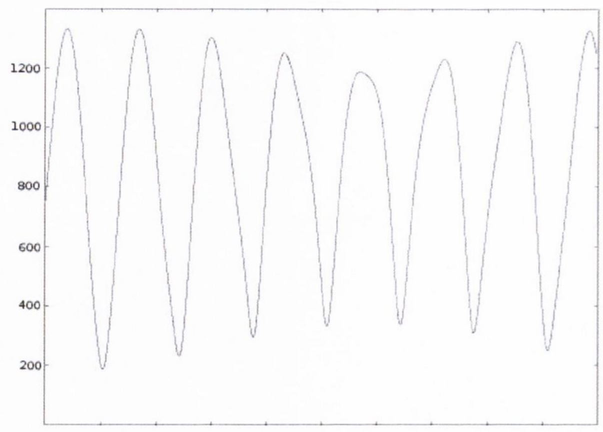


Figure 2.1 - Light-trip times between Earth and Mars from January 1, 2000. Each "wave" represents about 2 Earth years. The vertical axis is the light-trip time in seconds, the horizontal axis covers 15 years. (Diagram generated by the author.)

2.2.1 Light-Trip Times and Ephemeris

Since it presents some of the best examples of unavoidable delay, we look in some detail at delays in deep-space networking. The main constraints in such networks are the finite speed of light and the fact that the solar system is a very, very big place which is very, very sparsely populated with constantly moving bodies. We will however generalize the causes of delay and disruption described here when formulating requirements statements.

Finite speed of light

Because the solar system is so big, the finite speed of light is a major consideration when communicating at solar-system distances. Even though light travels at approximately 300 thousand kilometers per second, the distance from the Sun to the Earth means that light from the Sun takes around 8 minutes to reach us here. We are literally looking at the Sun as it was 8 minutes ago. However, to reach the outer planets, things are much worse. On average, light from the Sun will take about five and one half hours to reach the outermost planet (or one of the largest Trans-Neptunian Objects), Pluto.

Figure 2.1 shows how the light-trip time between Earth and Mars varies over a 15-year time frame. As can be seen the two planets have a close approach roughly every 2 years when light-trip times are down to around 4 minutes, but at maximum separation light-trip times can be up to 20 minutes. This therefore provides our first DTN protocol requirement:

Req. DTN protocols should be able to operate (relatively efficiently) even when link or path latencies are of the order of minutes, hours or days[9].

Another noteworthy point is that the graph in Figure 2.1 is not symmetric, which means that handling such variations is inherently complex, and tends to involve significant amounts of scheduling data, (e.g., multiple megabytes), a fact that has consequences for how we implement support for such situations.

Positioning inaccuracy

Light-trip time is not the only obstacle to communicating at these distances. As it happens, we cannot predict the exact positions of the planets with sufficient precision for all time. The general N-body problem [GR90] in physics ensures that there will always be some uncertainty in the positions.

The N-body problem states that once there are more than three bodies involved, even if we know all about them (how big, where they are, etc.) we cannot calculate their exact locations at any time in the future with arbitrary precision. In fact the uncertainties will fairly quickly become such that we may end up pointing a radio antenna in the wrong direction were we to ignore this problem. Ultimately this problem is due to the fact that we do not (or perhaps cannot) exactly solve the mathematical equations involved precisely. Practically, what we do is to use approximations that we periodically have to adjust to reflect the reality of the solar system.

In addition, there are also other more local variations that cause unpredictability; for example, an orbiting satellite's path will be slightly perturbed by passing over the equator of the Earth which bulges somewhat compared to the poles. Such perturbations can mean that satellite positions may become uncertain within a range of kilometers over time.

[9] DTN requirements are called out like this near the text motivating them. They are also collected (and numbered) in section 2.2.7 below. All but one of the requirements are phrased as requirements that DTN protocols "should" meet – since we are not dealing with a (set of) specific application context(s), we cannot generally resolve requirements as MUST vs. SHOULD vs MAY etc.

Visibility

As well as positional uncertainty, there are also visibility considerations. As the planets rotate, a position on the surface of one planet becomes visible on another. It is worth noting that some of the planets rotate rather slowly (a Venus day is 224 Earth days[11]) and that sometimes we might be more interested in the visibility of a location on the surface of a moon orbiting another planet. This happened recently with the Huygens lander that was part of the Cassini mission to Saturn [JA97]. Huygens landed on Titan, which is one of Saturn's moons, and for a brief while that landing site was one of the most interesting places in the solar system, at least in terms of deep space communications.

More often, orbiting spacecraft have orbits which are shorter than the rotation time of the body they orbit, (e.g., Mars Express has an approximately 6-hour orbit[12] and the International Space Station has an approximately 90-minute orbit[13]), so the visibility of such spacecraft change frequently, since they will typically spend about one-third of each orbit behind the body. This means, of course, that we cannot use line-of-sight communications for those eclipsed periods and deep space communications are all line-of-sight.

> *Req.* DTN protocols should not require simple, regular, strictly periodic nor cyclic patterns of visibility, but should be able to benefit from such patterns where they exist.

Visibility will also be constrained by other factors, for example, the local horizon, so typically one would require an object to reach a certain altitude before considering it visible for data communications purposes. This is both because signals that have to travel further through the Earth's atmosphere will be of lower quality compared to signals that are heading straight up, but also because it is simply not possible to point most large antennae right down to the horizon because they are too heavy to be easily supported in

[11] Most planetary data quoted here is available at: http://www.nineplanets.org/data.html
[12] http://www.rssd.esa.int/index.php?project=MARSEXPRESS&page=orbit
[13] http://www.space.gc.ca/asc/pdf/educator-observing_edu.pdf

such positions, and in any case would be far more likely to suffer from terrestrial interference even if one did build them to operate in such an orientation.

Of course, in long-haul communications one must also take the light-trip time into account, so that one is interested in visibility modulo the light-trip time. In principle, one could begin transmitting when the recipient is not yet visible from the transmitter. Often however enough leeway is allowed before transmissions start, at least for closer deep-space contexts, so that the recipient will actually be visible before transmission starts.

> *Req.* DTN protocols should support changes in the scheduling and/or contactability of nodes.

Handling ephemeris-related factors

In order to handle all of the above factors, space missions use so-called ephemeris[14] files and software. These files store calculated positions of the bodies (planets, moons, and spacecraft) of interest at the various times of interest and the software basically interpolates between these data points to determine the estimated position at any given time in the applicable range. Such ephemeris files must be updated occasionally, for example, as more information becomes available from observation, or as the results of spacecraft maneuvers are taken into account.

Ephemeris files will typically be large multi-megabyte files, which could be considered somewhat similar to an Internet router having to store large routing tables. The main source of software for handling ephemeris files is the NASA Jet Propulsion Laboratory (JPL) ephemeris library,[15] which provides functions to estimate the positions of the various solar system bodies at various times. There is also a JPL ephemeris file format that is the de facto standard used in many interplanetary missions.

---

[14] An ephemeris lists the spatial coordinates of celestial bodies and spacecraft as a function of time.
[15] http://www.ephemeris.com/ or ftp://ssd.jpl.nasa.gov/pub/eph/export/

To get a feel for the complexities involved in such communications, consider sending a command from Earth to a lander on Mars. First, scientists and mission planners meet, sometimes years in advance, to decide what commands to send and when. The set of commands will then go through various stages of approval and format translations. Finally, the commands will be encoded in an often mission-specific manner for transmission to the spacecraft.

The commands may then be sent via the NASA deep-space network (DSN) or European Space Agency (ESA) Tracking Station Network (ESTRACK), both of which consist of a terrestrial data network with large antennae (up to 70 meters diameter) spread around the globe. The large antenna is required in order to supply the power to transmit over such long distances. Global coverage is required in order to make it possible to "see" the entire sky at any time.

Next one needs to point the antenna in the correct direction, at the correct time. Accurately pointing a 70-meter antenna is a nontrivial problem and may require a few minutes to slowly slew the antenna to the correct position. Once in position, the transmitter can begin to send the signal.

However, there is of course no point in sending the signal if the receiver is powered off, which is exactly how a spacecraft receiver will be for most of the time. And if the receiver is in space, it may have to be warmed up before operating, so even before pointing the receiver's antenna (if it is not an omni-directional antenna, that is), there is work to be done. The need for data to "wait" for the next period when both antennae are powered and properly oriented provides yet another source of disruption.

And of course things don't always go according to plan. With the Galileo probe to Jupiter, its main (high-gain) antenna that was intended to transmit data at the rate of about 130 Kbps, failed to deploy correctly, so all data had to be returned via the much smaller omni-directional antenna. Most spacecraft include such an antenna usually as a backup for controlling the spacecraft in case something goes wrong with the main antenna.

Since this antenna has to be contactable regardless of the spacecraft's orientation, it must transmit in all directions, (i.e., be omni-directional), which clearly means that the transmission in the direction of Earth carries much less power, and hence less data. The resulting data rate was 10 bits per second! However, by simultaneously using multiple antennae, in the United States, Spain, and Australia, the data rate when all three were pointing at Galileo could be increased to 160 bits per second. While this may still seem tiny it was enough to save the mission, which went on to be highly successful.[16]

Req. DTN protocols should be able to support a wide range of data rates.

As can be seen, there can be a lot of complexity involved in deep space communications, certainly compared to sending a single e-mail over the terrestrial Internet. We could describe the situation by saying that the plumbing for deep space communications involves lots of complexity that is absent in the terrestrial Internet, but of course given that the Internet is approaching or has passed a billion hosts; scale factors may act to make both situations roughly equally complex!

2.2.2 Conserving Power

Ultimately many systems are constrained by the need to conserve power. Basically, total battery depletion is equivalent to destruction for many of these systems, whether one is dealing with a lander on Mars, or a sensor floating in a lake. Even where a node is recoverable (which it is not in the former case), the cost to recover it may be significant and, of course, even if the node can eventually recharge its battery, the inability to provide service while the node's battery is dead could be significant.

In the future, one may also have to deal with systems that, though they are nearby, are practically irrecoverable if no power is available, say, a sensor node that is simply too small to find by searching. In those cases, what was deployed as a way to monitor the environment could end up itself being a pollutant. So long as the node has sufficient power it can at least send out a beacon signal, possibly with some location information.

[16] http://www2.jpl.nasa.gov/galileo/faqhga.html

Similarly, for devices that are carried about on a person, such as mobile phones, we have power conservation requirements (say, to last for a number of days between recharges), which effectively override most other requirements – in the mobile phone case, power conservation is one of the highest priorities, for example, according to Kim: "The outstanding problem with the current mobile handset is the battery drain on the device." [KI06]

> *Req.* DTN protocols should be usable in systems where power conservation is the over-riding system-level requirement.

So, we can see that conserving power is a requirement for many systems. The next question is how does this cause delay and disruption? Well, first, one of the most power-hungry devices on many wireless devices is the radio. It has been estimated that each bit transmitted or received costs the same amount of power as executing some hundreds to thousands of central processing unit (CPU) instructions. [BL05a]

In a typical embedded system a significant amount of full-on power consumption will be due to radio communications, in many cases these days using WLAN-based protocols (i.e., IEEE 802.11b/g). In fact, the actual power consumption can depend on the details of the traffic pattern; for example, if a streaming application has predictable packet inter-arrival times, then the wireless interface can be more easily tuned to conserve power. [CH02]

Even radio setups with much smaller ranges (radio range and power consumption will increase together) like Bluetooth [KA01] will consume significant power. In fact, when we take into account the overall network power consumption, the longer-range radio solution will normally be the more cost effective. We can fully cover a square kilometer with 100 devices, each with an ~70m effective range (say 802.11b devices), but we need 2,500 devices if each has only an ~30m effective range (like Bluetooth).

Clearly then, a very simple and effective way to save power is to turn off the radio. However, this causes the obvious problem that the device is offline while the radio is off, and hence constructing the network to handle this voluntary disruption can be difficult.

Powering off the radio also causes delays. If the nodes in a multi-hop path are sometimes on, and sometimes off, then data will sometimes have to "pause" in a node until the node is ready to transmit the data to the next hop on the path.

Regardless of how the transmitter knows that the receiver's radio is on (and it must know that), there will be a delay involved. And those delays may be significant since in some cases the most natural way to handle powering the radio on and off will be related to some application-specific duty cycle, for example, related to day/night cycles in environmental monitoring sensor networks, or roughly related to the frequency with which a node is "visited."

Even were battery technology to undergo dramatic improvements (e.g., based on fuel cells), the fact that for many applications one would like to deploy nodes that can last for significant periods (months, years) means that batteries alone will not be enough to power most systems over their entire lifecycle.

So, many systems will also include some kind of power harvesting (e.g., solar panel, wind generator, piezoelectric, etc.) that can also lead to disruption; for example, a solar powered node that is liable to be shadowed for a significant period (say due to snowfall) will eventually suffer communications disruption.

Most forms of power harvesting appear to be vulnerable to some such long-duration outage. The first reappearance of such a node may well also require some special handling – it may otherwise appear to be a newly introduced node or may no longer have an up-to-date network configuration.

> *Req.* DTN protocols should be able to operate in situations where applications or the environment determine duty-cycles.

2.2.3 Intermittent Availability

Sometimes a node will be physically moving. And sometimes that movement will take the node to "dead" coverage areas between access points.[18] Depending on the duration of the disconnection and the amount of coordination behind the scenes by the access point administrators, TCP sessions can sometimes survive such a handover. However, in general, if the access points are not coordinated or are too far apart, then TCP sessions will fail due to the disruption. Note that this may or may not become visible to an end-user; for example, a mail client may totally hide the disruption, whereas a Web browser will probably fail for outstanding requests, but may begin to work correctly once the new connection is established.

> *Req.* DTN protocols must co-exist with, and be able to make use of, the existing Internet.

Aside from the usual cases of access points being intermittently visible (in radio terms, that is), a very fast-moving node might be moving quicker than the infrastructure can handle the migration of the various aspects of the system state that are required to provide service. Current infrastructure systems are designed and provisioned so that the so-called "binding updates" can in fact be handled as quickly as required. However, if we consider a case where the access points themselves are only intermittently connected to the Internet, then we can see that this can cause disruption and delay.

> *Req.* DTN protocols should operate in the face of significant node mobility, even for infrastructure nodes.

One may ask why an access point might only have intermittent connectivity. Well, it could be that the access point will await cheaper (off-peak) Internet access, or perhaps the access point is on a ship, which can only connect back to shore for a certain distance. Or, perhaps

[18] In this section and throughout, we use WLAN terminology, but we could equally describe the issues here in terms of mobile IP concepts like home agent/foreign agent; or we could use Global System for Mobile communications (GSM) or 3rd Generation Partnership Project (3GPP) terminology - the point being that the delay and disruption issues are the same regardless of the type of mobile communication in use.

the access point is awaiting a satellite pass. The disruption in such intermittent access point cases is fairly obvious in that the node will not have Internet connectivity when the current access point does not. The delay caused will be directly related to the delay before the access point next has connectivity.

The infrastructure may also be unreliable if a node is moving quickly past sparsely deployed access points, say along an autobahn, if access points are deployed at service stations. Ott and Kutscher [OT04] have demonstrated that an IEEE 802.11g node in a car traveling at about 120 km/hr will have about 11 seconds connectivity with an access point at the side of the motorway, which will often not be sufficient to service application layer requests (say for information about the destination). In this case, their "drive-thru Internet" architecture[19] envisages starting the request with the car picking up results as it passes the next access point at the next service station, perhaps 30 km further down the autobahn.

> *Req.* DTN protocols should not assume the same path is always used for application layer requests and responses.

And if intermittent availability is an issue for some infrastructure networks (those with access points), we can see that this will clearly be much worse for networks that have no infrastructure at all. In those typically ad hoc networks, hop-by-hop connections will be established at sometimes unpredictable times, so that the probability of successfully establishing an end-to-end connection (even assuming the endpoints are both available at once) might be very small.

> *Req.* DTN protocols must operate even in the face of the total absence of an end-to-end connection.

There are also some ad hoc cases where delays may be introduced due to the nature of the context or application, rather than the network. For example, if two commuters play one another at chess each morning while both are on the train, then it might be a fair setup to

[19] http://www.drive-thru-internet.org

stop the chess clocks whenever the players are no longer in proximity, (on the basis that the out-of-range player cannot make a move). Such a system would lead to delays of the order of a day being introduced.

Recognizing the facts presented in this section is perhaps one of the main differences between DTNs and other networking approaches including mobile ad hoc networks, peer-to-peer networks, and other similar networking approaches. Those approaches basically attempt to engineer end-to-end connectivity, in the face of ad hoc, disrupted connections, using such routing protocols as Ad hoc On-demand Distance Vector routing, (AODV), [PE03] Dynamic Source Routing, (DSR), [JO96] Optimized Link State Routing (OLSR) protocol, [CL03] Network Mobility, (NEMO), [DE05] or similar schemes. As we'll see, with DTNs one does not generally attempt to build such end-to-end connections, which is a significant difference.

In some networking scenarios it may be hard to decide which interface or contact to use to forward data, so that the data moves "closer" to the destination. For example, in the Earth/Mars scenario, it will be more efficient for a lander to skip an orbiter contact if the orbiter from the 2nd lander/orbiter contact will have an earlier (or more reliable) contact to the deep space network. In such cases, the optimal route to select may involve data first moving "further away" from the destination.

Req. DTN protocols should generally attempt to move data "towards" the destination, even though some optimal routes may involve temporarily moving data "away" from the destination.

2.2.4 Quiescent Environments

One fact about many (though not all) embedded systems is that they tend to spend lots of time doing absolutely nothing. For example, this will be the case with most systems that are not continually monitoring their environment. In such cases, having the device connected to a network solely in order to be available is very wasteful, especially in terms of power consumption. This provides yet another motive to be able to support disrupted and delayed networking.

So, for many systems we'd like a way to have the system turn itself off completely and only have it wake up when something is happening. How we arrange to wake up the system will depend upon what the "something" is that happens. Possibilities might include the availability of power as the sun rises, receiving a beacon signal (usually using a radio receiver dedicated to this purpose) from another system that comes into range, an independently powered sensor reading crossing some threshold value, or movement of the system itself could trigger the wake-up. However, perhaps the most common scheme, and one which should almost certainly be combined with any of the above, is to wake up in response to a scheduled alarm call from an on-board clock, frequently powered via a separate button battery.

Of course, the system also has to handle the opposite problem, deciding when to go to sleep, but we can see that each of the wake-up possibilities suggests a way to decide when to sleep. For example, a system might decide to shut down when the sun sets and its solar panel ceases to charge the battery.

Many processors will also have a range of power-down settings, analogous to the standard personal computer power settings: running; standby, (where system state is stored in random access memory (RAM)); hibernate, (where system state is stored to disk); and "off" (which isn't actually totally off, for many devices, when they're still plugged in). There are a range of standards in this area, with the Advanced Configuration and Power Interface (ACPI) standard [CO03] being perhaps the most commonly supported.[20] Many embedded systems developers will however invent their own scheme for handling equivalents to standby and hibernation.

> *Req.* DTN protocols should be able to operate when the host hibernates or reboots in the "middle" of a "session."

Regardless of how a system is put to sleep and awakened, the fact that different systems in a network are doing this will clearly create delay and disruption in more or less the same

[20] http://acpi.info/

way as if the systems were powering on and off their radios. However, even with individual nodes making individual sleep/wake decisions, there may well be correlations between them. For example, as the sun rises, presumably the more easterly and less shaded systems will power up first, resulting in a wave of power-ons. Similarly, if a sensor threshold were used to control powering on and off, then variations in those levels might cause other patterns in how the systems in the network power on and off, which could negatively impact traffic patterns if only a few sensors at a time are likely to have crossed the threshold value.

One particular case where unexpected interactions could occur would be where each system has a radio for communications and a separate beacon radio that it can use to wake neighbors. In such a case, one can easily imagine a faulty node; say one that is awake too often, acting so as to accidentally deplete the batteries of neighbors, by continually awakening them. So, in practice, most systems that manage power by turning off will also have some built-in safe-guards, for example, ensuring that they awaken at least once per day at a fixed time, and perhaps that once woken they stay awake for a certain period and once put to sleep, that they don't immediately reawaken.

2.2.5 Security Considerations

We have seen that handling the fact that not much happens can be a source of delay and disruption. But some of the things that do happen (either accidentally or deliberately) can create quite bad effects on our putative network. Anyone who is familiar with the problem of Internet worms (the original of the species dates back to 1988 [SP88]) will know that networks are always vulnerable to some security problems.

In terms of DTNs, one of our main concerns is going to be ensuring that network nodes are available when they ought to be available. This means considering so-called denial of service (DoS) attacks, where an attacker consumes some network resource in order to prevent (or disrupt) normal network traffic.

For example, if a DTN node can be woken by a beacon as described above, then an attacker who generates such a beacon will awaken the node, causing it to consume power, perhaps to the extent that the node never has sufficient power to take part in handling normal network traffic.

Req. DTN protocols should be designed so as to be highly robust in the face of DoS attacks.

Another simple, though often easily detected, DoS attack might involve physically shielding the node so that its radio transmissions are blocked. Yet another might involve an attacking node masquerading as a genuine node and generating so much traffic that real traffic never gets forwarded. More traditional networks also suffer from all of these problems, but they are worse in the DTN context due to the fact that since one is generally dealing with a less structured networking environment, it is harder to maintain the shared state that allows nodes to detect these DoS attempts. For example, centralizing firewall logs (as is done for example in some honeynets [CH05]) might be very difficult in many DTNs.

We are also much more likely to be dealing with nodes for which a successful DoS is a catastrophic event. With many nodes, if an attacker can succeed in a battery depletion attack, then they have effectively destroyed the node, and perhaps also partitioned the network so that many other nodes are also badly affected.

In addition to DoS, DTN nodes also have more or less the same set of security requirements that apply to any normal network host. They need support for confidentiality, data integrity, authentication, and the type of security countermeasures that Internet protocol security (IPsec) [KE05] or transport-layer security (TLS) [DI06] address.

Req. DTN protocols should provide (or leverage) confidentiality and data integrity services.

We will see how this can be (partly) achieved later on, but we will also see that there are still open research issues in this area. Basically, it's really hard to do the type of security negotiation involved in protocols like IPsec or TLS when the communicating nodes are using a DTN. The reason is more or less the same as why TCP won't work for DTNs. Such protocols require too many round-trips before data can flow. However, unlike the case of TCP where comparable DTN protocols have been developed, there are, as yet, no DTN analogs for the low-level key management features present in IPsec and TLS.

R.1	DTN protocols must operate even in the face of the total absence of an end-to-end connection.
R.2	DTN protocols should be able to operate (relatively efficiently) even when link or path latencies are of the order of minutes, hours or days.
R.3	DTN protocols should support changes in the scheduling and/or contactability of nodes.
R.4	DTN protocols should be able to operate when the host hibernates or reboots in the "middle" of a "session."
R.5	DTN protocols should be usable in systems where power conservation is the over-riding system-level requirement.
R.6	DTN protocols should generally attempt to move data "towards" the destination, even though some optimal routes may involve temporarily moving data "away" from the destination.
R.7	DTN protocols should not require simple, regular, strictly periodic nor cyclic patterns of visibility, but should be able to benefit from such patterns where they exist.
R.8	DTN protocols should be able to operate in situations where applications or the environment determine duty-cycles.
R.9	DTN protocols should not assume the same path is always used for application layer requests and responses.
R.10	DTN protocols should be able to support a very wide range of data rates.
R.11	DTN protocols must co-exist with, and be able to make use of, the existing Internet.
R.12	DTN protocols should operate in the face of significant node mobility, even for infrastructure nodes.
R.13	DTN protocols should be designed so as to be highly robust in the face of DoS attacks.
R.14	DTN protocols should provide (or leverage) confidentiality and data integrity services.

Table 2.2 – DTN Requirements.

We should also note at this stage that some DTN nodes, perhaps mainly those in military tactical networks, will also be required to be stealthy or "hide," to the extent that this might cause additional delay in establishing a link, say, if such a node cannot react to a simple beacon, since to do so would expose its existence. While such stealth is mainly a matter for lower layers (i.e. layers 1 and 2), a DTN protocol that required constant "keep-alive" style messages might not be able to be stealthy enough for use in such contexts.

When discussing security, we should also note that compared to a standard personal computer or server, DTN nodes will tend to be placed in much harsher environments. This is obvious for a Mars lander, but is also true for many other DTN applications where the nodes will be vulnerable to, for example, interference from animals (including humans) and the elements, i.e., all of the standard problems that arise when equipment is exposed for longer durations. The failures resulting from such occurrences of course act to increase the likelihood of delay or disruption, further justifying our DTN approach.

2.2.6 DTN Requirements Summary

Table 2.2 above lists all of the DTN requirements together for ease of referencing. We have re-ordered the requirements to place the more important ones first and grouping

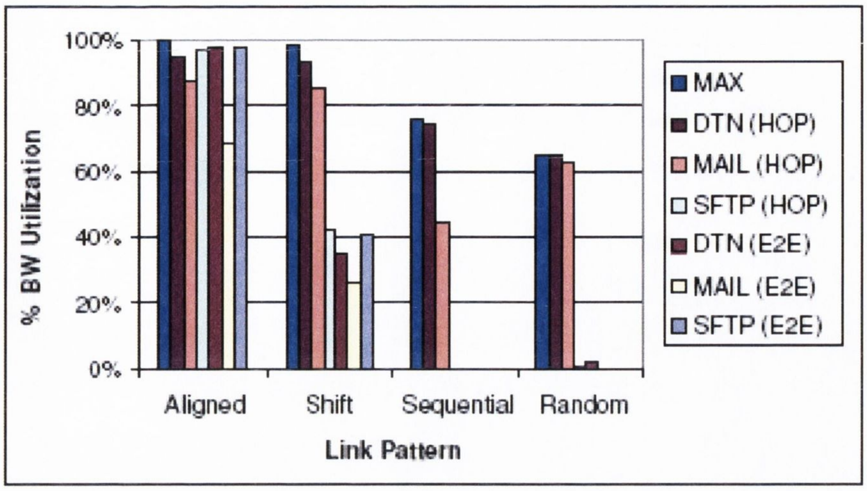


Figure 2.2 - Results from a DTN experiment. (From [DE04])

others. As stated earlier, we are not including generic protocol requirements, (e.g., no livelock), but only those that are specific to, or should be emphasised for, DTN protocols.

In subsequent sections of this chapter, where we can see a requirement that is not met by a solution or protocol then we will include a "**(-R.x)**" in the text. For brevity's sake, we will not however, go into full detail about why each requirement is, or is not imposed or met in each case.

2.3 A Simple Demonstration of Delay (In)Tolerance

In order to demonstrate how some protocols operate in an environment experiencing delay and disruption, Demmer et al. carried out an experiment [DE04] using a sequence of "relays," each of which implemented a number of standard Internet protocols plus a DTN protocol (described later) called the bundle protocol (BP).

The experimental setup had data passing through each of the relays, using a range of protocols and under a range of connectivity conditions. The protocols used were the Simple Mail Transfer Protocol (SMTP) [KL01], a less chatty variant of the File Transfer

Protocol (FTP) [PO85] (here called SimpleFTP or SFTP[21] [DE04]) and the BP. Each of the protocols was run in both an end-to-end ("E2E") mode where IP forwarding was used between protocol daemons running at the source and destination and in a hop-by-hop ("HOP") mode where protocol daemons were running at each node.

At first all of the relays were operated "full-on" and the performance of the various protocols was measured in terms of throughput. Then relays were selectively powered off following various power cycling patterns, either all on or off together ("Aligned"), in a shifting pattern ("Shift") so that there was overlap between each node powering up and the next, then a sequential pattern ("Sequential") in which only one node is powered up at a time, and finally, a random ("Random") pattern in which nodes are powered up and down independently. As shown in Figure 2.2, these patterns of connectivity/availability have a highly detrimental effect on standard Internet protocols. For example, in the sequential, non-overlapping case, no SFTP-based traffic gets through at all. **(-R.1,-R.3)**

As might be expected the DTN protocol survives all this much better than the others, at least in terms of most effectively utilizing the available bandwidth. The reader is referred to [DE04] for full details and discussion.

The conclusion to be drawn is fairly obvious and is that for these types of connectivity setups, standard Internet protocols may have unacceptable performance. **(-R.2)** The more interesting question to ask at this point is whether such communications setups are likely to occur in practice. As explained earlier, one can lose communications for various reasons to do with power management, radio range or interference, and node movements.

In fact, a number of scenarios have similar patterns of communication, particularly related to so-called Vehicular Ad-hoc Networks (VANETS). Wong et al. [WO03] described differences between the random waypoint based model and a more realistic model of inter-vehicular networking that takes a street-pattern into account – one example they give is the fact that traffic lights create clusters of nodes (cars) and that the clusters then disperse over

[21] Not to be confused with secure FTP which is the more common expansion of the acronym SFTP.

a period. Communication between such clusters could, for example, replicate the “shift” pattern described above.

Naumov et al. [NA06] take this examination of VANETs a step further by basing their simulation work on a city-scale “micro” model of traffic movements. Although their chosen network simulator (ns-2 [FA97]) cannot scale to run complete city-scale simulations they nonetheless also conclude that more realistic VANET mobility models (compared to the random waypoint model) imply worse behaviour for standard protocols. **(-R.12)**

VANETs are essentially based on a significant body of work [ZH03, HO02] devoted to what are called mobile ad hoc networks (MANETs). MANETs are networks where each node is also a router and where some kind of cooperative routing protocol is used, with the ultimate goal that by sharing the available bandwidth and connections, any node in the network can communicate with any other.

One might question whether this is the best goal to set for ad-hoc networking, given that it apparently ignores the fact that a one billion node infrastructure network (the Internet) exists and is nearby for almost all MANET/VANET applications. Put another way, if we set the goal for MANET/VANET protocols to be the ability for end-to-end communication solely based on the use of MANET/VANET protocols, then we lose the ability to communicate via some paths that are very likely to be available in many scenarios. A more interesting goal would be end-to-end communications where some “hops” in the path use MANET/VANET protocols, and where others, (most likely those in the “middle” of the path), use standard Internet routing. This wrong-goal problem is often fatal for such efforts even where enormous effort is expended and even where the wrong-goal is, in fact, very close to some real, though subtly different, requirement.[22] **(-R.12)**

[22] DNS security, the WAP forum’s stack and OSF/DCE arguably offer additional examples of the same problem.

One of the best known MANET routing protocols is called Ad hoc On-demand Distance Vector routing (AODV) [PE03] and has as its main aim the on-demand discovery of, and maintenance of state for, routes from a source node to a destination. In many applications, the appropriate destination node for the ad hoc network is a gateway to the terrestrial Internet, and AODV should be usable to create a route between a sensor node and the gateway node.

Another well known MANET routing protocol is the Open Link State Routing Protocol (OLSR). [CL03] In contrast to AODV, OLSR doesn't attempt to build routes on-demand, but rather regularly has nodes exchange topology information, so that details of the relevant next hop are (hopefully) available when needed. OLSR also attempts to constraint the amount of overhead, routing protocol traffic, to within a few hops of each relevant node.

So why aren't AODV or OLSR suited for use in networks exhibiting the communications patterns discussed above? Well, first, they fail if one of the nodes on a route is powered off. This is not a bug. The protocols are simply not designed to cater for devices that are frequently turned off being part of the routes that are built. The obvious effect of this is that when nodes are frequently un-contactable, the routes derived by AODV or OLSR are not valid for nearly as long as would be expected. **(-R.3)**

However, more seriously, once a node (call it "lazy") is powered off, it cannot be used to create new routes (and will indeed cause other nodes who remember our "lazy" node from its previous power cycle to send packets that are never acknowledged, thus wasting resources). In many scenarios where the ad-hoc network communicates with the Internet via a gateway, once all of the nodes that are in radio range of the gateway are sometimes powered off then one faces a situation where no other node can find a working route to the gateway and the entire network essentially breaks down. **(-R.4)**

One can ameliorate this to some extent if we synchronize (even loosely) the on/off cycles of the various nodes. However, in such a situation it only requires one node's clock to be slightly off to disrupt an entire set of routes. We could therefore migrate between the different communications patterns as clocks diverge.[23] **(-R.7)**

Synchronization problems can also happen as the set of nodes changes over time. Say a set of nodes are somehow scheduled to communicate on the hour, for 5 minutes, and to then power off. At some later time, a new batch of nodes arrives on the scene, but with these new nodes communicating at half-past the hour, and again once per hour for 5 minutes. If some nodes act as bridges between these two sets, communicating every 30 minutes, then data can flow across the network, but without there ever being an end-to-end connection. Packets from the first set of nodes will however take at least 25 minutes to get to members of the second set in this setup. Neither AODV nor OLSR would successfully route packets in this scenario **(-R.2)**

In the cases discussed so far, we have only really considered cases where, for example, a TCP connection is to be established between the source and destination. However, even with store-and-forward protocols such as SMTP, the random pattern might perform very badly if mail is forwarded to whatever host is currently contactable. Messages would, in that case, be just as likely to travel "backwards" as towards the destination - this could happen if there is only likely to be a single connected node at any given moment, and if routes are set up so that the currently contactable node is always the next hop Mail Transfer Agent (MTA). **(-R.6)**

In summary, this simple demonstration setup is sufficient to show that standard protocols and even experimental ad hoc protocols can be problematic in some easily constructed scenarios that map to realistic application use cases. As can be seen from Figure 2.2, the demonstration also shows that it is at least plausible that DTN protocols can offer better solutions for these cases.

---

[23] For this reason, one could probably make a good case that routing protocols like AODV should be extensible so as to support in-band clock synchronization.

2.4 Using Existing Protocols for DTN

In this section we review how relevant existing application, transport and lower-layer (sub-IP) protocols might meet DTN requirements. Existing protocols designed specifically for DTNs will be considered in the next chapter. The general approach of a DTN overlay and its consequences are analysed in Chapter 4.

2.4.1 Application Layer Protocols

As mentioned above, one can imagine attempting to meet DTN requirements via the development of application layer protocols. One example that has been suggested in the past has been to "just use SMTP[24]" since SMTP, as a store-and-forward end-to-end protocol has clearly been a success and, in practice, offers sufficient reliability in most cases. However, as also previously noted, since SMTP is layered on TCP, it cannot bridge a hop that is significantly delayed, or sufficiently frequently disrupted. **(-R.2)**

Another approach is to invent a new application layer protocol that sits above whatever transports are available and to handle DTN requirements in this application layer protocol. This has in fact been done in projects that pre-date the DTNRG work on the BP and LTP, for example, Zebranet [ZH04z] and Seaweb [RI05].

Assuming for the moment that our set of DTN requirements are self-consistent, then, since it is therefore possible to build some protocol meeting the requirements, one could of course, choose to run that protocol at the application layer. At that point however, the only reason that one has not developed an overlay network protocol would have to be that there is something application-specific about the protocol design. And that, in turn, means that it is highly unlikely that such a protocol could be used to carry traffic for some other application. In summary then, we can postpone further discussion of application layer DTN to when we consider the overlay approach in Chapter 4.

[24] This has been repeatedly suggested at DTNRG meetings, though seemingly without making it into written form.

Haggle [SU07] demonstrates a different approach that sits "across" layers from the application, down to the data link layer. Haggle basically adds an additional naming layer above current protocols (e.g. HTTP, SMTP) that it uses to resolve meta-data about named data objects. In this way, Haggle attempts to be transport and lower layer agnostic since it only cares about the names of the data objects and not how they are acquired. However, Haggle itself doesn't seem to include support for high-latency **(-R.2)** or highly-disrupted links **(-R.1)**, though presumably Haggle could be extended to support use of such links. Nor does Haggle really provide application independence, since any application that wishes to use Haggle, has to re-engineer its data model to fit the Haggle naming scheme which is not currently used by any widely deployed application.

2.4.2 Transport Protocols

As we have seen there are at least niches where TCP won't work. However, one may well wonder whether or not TCP or some other transport protocol, perhaps specially configured or modified, might avoid these problems. In this section we review existing transport protocols, to see whether or not they suffer similar problems when it comes to meeting DTN requirements. This review of transport layer protocols is largely based on Iren et al's survey [IR99] of transport protocols.

As mentioned already TCP's handshake-before-data is hugely problematic for very high latency cases. **(-R.2)** However, even at lower latencies this handshake wastes significant bandwidth that could be used for carrying data. For example with a 1.25 second delay (approximately the Earth-Moon light-trip time), and with a 256Kbps data link (which would be very small in that context) each wasted roundtrip costs 40KB, roughly the size of a small image. The basic message is that any transport protocol that is "chatty" is not suitable for use in DTNs. **(-R.1)** By "chatty," we mean a protocol that requires multiple round-trips before something useful is achieved. This covers not only TCP, but SCTP [ST00] and RTP [SC03] which are basically the standard reliable transport protocols in widespread use today, (both SCTP and RTP have the same problem of consuming a round-trip before data flows).

In addition to the initial round-trip, TCP and other transports react to congestion by being sensitive to dropped or delayed packets. While this works well in today's wired Internet (since links are mostly reliable) it will not work in many DTNs and is already problematic in some wireless contexts since as stated by Montenegro et al: "TCP does not perform well in the presence of significant levels of non-congestion loss." [MO00]

Harras and Almeroth [HA06t] consider TCP-style transport reliability issues in what they call "Delay-Tolerant Mobile Networks" (DTMNs) which are essentially DTNs with many mobile nodes. They consider ways to provide reliability, including hop-by-hop and end-to-end and propose a new approach that they call the network-bridged approach.

The network-bridged approach assumes that nodes can send small amounts of signaling traffic end-to-end, so that when a node receives a packet, it can send a "success" signal back to the originator via this separate channel. **(-R.1)** The motivation for this approach is a scenario where nodes can use a (probably expensive and low bandwidth) cellular network for signaling but must use DTN protocols for bulk data transfer. Based on a modified random waypoint mobility model simulation, they conclude that the network-bridged approach can offer reliability more efficiently than hop-by-hop or end-to-end approaches, but of course, the availability of the end-to-end signaling channel does breach our most important DTN requirement. While the network-bridged approach is interesting and could be a basis for future DTN signaling work, it does not provide a generic mechanism that DTN nodes can use to achieve reliability.

In addition to standard transports there have also been various research proposals over the years for alternative transports or modifications to current transports that could potentially be a match with DTN requirements - in particular DTNs are a bit like very high-speed networks.

Very high-speed networks are like DTNs because the bandwidth-delay product is increased; perhaps sufficiently that signaling traffic becomes problematic due to the wasted bandwidth required. Basically, if the bandwidth-delay product is sufficiently high, and if application traffic is blocked (temporarily) by signaling traffic, then we effectively re-create the problem with TCP in DTNs – the additional roundtrips required mean that the overall suite of protocols are too chatty for use. **(-R.2)**

Kumazoe et al. [KA06] describe how high-speed network variants of TCP perform in a fairly realistic environment, in particular in the presence of numerous "small" standard

TCP sessions and a relatively small number of constant bit rate UDP sessions (like phone calls). Their conclusion is that the 19 (19!) TCP variants they examined for use in high bandwidth-delay environments are badly affected by a requirement to co-exist with other, short-lived TCP flows (e.g. web accesses). **(-R.11)**

According to Kaneko et al. [KA06f] these TCP variants can be classified into three categories: *loss-based* where the congestion window is modified by deliberately causing packet loss[25]; *delay-based* which make use of (estimated) RTT as a network congestion estimator and lastly *loss-based protocols using RTT metrics* which can adaptively switch congestion controls according to the congestion level or estimated RTT.

However, none of these approaches can work, in general, in a DTN – the problem being that estimating the RTT is quite a different proposition since, in a DTN, the RTT may bear no relationship to the level of congestion being suffered. In addition, DTNs, like wireless networks generally, may experience much higher rates of packet drop that are not due to congestion. So both loss-based and RTT estimation based schemes are problematic in DTNs. **(-R.1)**

Overall it appears to be the case that none of the TCP extensions currently proposed for use on the Internet meet our DTN requirements well, since they all either require a round-trip before data is transferred and/or base their congestion handling on changes in RTT. **(-R.2)** However, there have also been some proposals to modify TCP in ways that might be more suited to DTNs and we now examine those.

TP-Planet is a DTN-like TCP variant

TP-Planet [AK04] represents an interesting, though ultimately unsuccessful, attempt to use a standard IP network analysis approach in order to develop a way to handle high-latency links in a transport protocol running directly over IP. As we'll see, the result turns out to be usable only in limited cases of high-latency networks where an end-to-end connection can

[25] These variants generally ramp up transmission very quickly until packet losses occur.

be established and maintained. Put another way, TP-Planet doesn't work where there is no end-to-end connection. **(-R.1)** TP-Planet might perhaps work in some Earth-Moon scenarios, but probably not if the connections are via a relay satellite that isn't always visible to both sides.

The basic approach to tackling problems with TCP has usually been to run simulations of protocols that differ subtly from TCP but which exhibit quite different behavior in simulated stressed environments. TP-Planet follows this approach, in that it first examines the aspects of TCP that are problematic in the face of delay, and then proposes new ways to achieve acceptable performance in those high-latency environments. Unlike the bundle protocol and LTP[26], however, TP-Planet tries to fix these problems by developing a variant of TCP, rather than by inventing an entirely new protocol.

The first problematic aspect of TCP that is considered is slow start. This is the algorithm that TCP uses to slowly increase bandwidth utilization so that at the beginning of a session TCP will spend a considerable amount of (wasted) time waiting for packets to be acknowledged. **(-R.2)** As the session goes on, so long as there are no negative acknowledgments, TCP continues to widen the window of packets that remain to be acknowledged. This widening of the window allows TCP to use more of the available bandwidth. Clearly an algorithm like slow-start should not be run over an interplanetary link, since the inefficiency involved in starting with a small window size would be enormous. [AK02] So, TP-Planet starts by using some environmental knowledge in order to determine the best transmission parameters, but then allows the rate to be adjusted in response to events as the session proceeds. This approach of adjusting the transmission rate based on the end-to-end performance mirrors how TCP works.

TP-Planet monitors how the session is proceeding by inserting some so-called "NIX" packets into the stream, some at a low (IP) priority and some at a higher priority. These packets are like ICMP "echo" packets, in that they are echoed back to the sender by the

[26] Both are described in the next chapter.

recipient. By monitoring the arrival rates of these packets, TP-Planet estimates how the session is progressing, the logic being that if some node on the route is experiencing congestion, then presumably this will cause the delay or dropping of some of the low-priority NIX packets, and so the ratio of transmitted to returned NIX packets at each priority gives an estimate of network performance. Importantly though, this estimate is arrived at in an end-to-end fashion, without having to have knowledge of how intermediate nodes operate. While this approach is different from TCP due to the new NIX packets, the overall idea of establishing an end-to-end performance metric based only on the end-to-end performance of transmissions again mirrors TCP.

TP-Planet, like TCP, uses an additive increase, multiplicative decrease (AIMD) model for tuning the session based on the behavior of the network. This means that when the protocol detects that the network has more capacity, then the number of data packets transmitted can be increased by the addition of a small number of additional packets per second. This is the additive increase step, and in the case of TP-Planet is triggered by noting that the low-priority NIX packets are arriving as quickly as the high-priority ones, thus demonstrating that the network has unused capacity. The multiplicative decrease step occurs when the ratio of NIX packets shows that low-priority packets are being delayed or dropped, thus signaling congestion somewhere in the network.

When this occurs the protocol halves the rate at which it is sending data packets, a multiplicative decrease. This AIMD model is a very well-known approach in networking and its use brings with it a large body of knowledge of, for example, how much memory is required to be provisioned in nodes in order to handle congestion.

TP-Planet also caters for cases where a session is interrupted, say for example due to a short-lived occultation. The idea presented is to reestablish the session without having to suffer the full multiplicative decrease. Basically, by monitoring the NIX packet arrival rates, this condition can be detected fairly quickly. This represents a difference between TP-Planet and TCP, in that TCP basically assumes that all dropped packets were dropped because of congestion, whereas TP-Planet attempts to distinguish between these two cases, so that we don't dial-down the transmission rate unnecessarily. However, the claimed utility of this feature is not clear since for example Mars orbiter occultations last for tens of minutes and not seconds (e.g. while an orbiter is behind a planet).

So, TP-Planet seems to offer an attractive way to leverage lots of existing networking expertise, techniques, and results in order to deal with high-latency environments. However, there is a major problem with TP-Planet that, in fact, appears to make this protocol unusable for many deep-space deployments.

The problem is that TP-Planet assumes that there is a working end-to-end connection for the entire duration of the transport session. That is, in order for TP-Planet to work at all, packets need to have arrived at their destination and must be acknowledged back to the sender, within a timeout, or else the protocol assumes that congestion is occurring and drops the transmission rate, eventually, presumably, to zero or some very low rate. If one of the nodes on the route were an orbiter, and were that orbiter only able to forward packets to a lander during a period when the orbiter is eclipsed (the lander is on the "far side" of the planet), then TP-Planet would not work any better than TCP, which is to say, not at all! **(-R.1)**

This problem restricts TP-Planet to uses where all of the nodes on the route are effectively simultaneously visible to their peers; in other words, it requires an end-to-end connection. Now, while this configuration does occur, it would seem to be very restrictive for a generic DTN protocol.

So, unfortunately TP-Planet appears not to be usable, but it does perhaps show that it is worthwhile considering how standard networking approaches (and related theory) could be applied in DTN cases.

SUMOWIN and Explicit Transport Error Notification

In a similar vein, the survivable mobile wireless networking (SUMOWIN) project was a U.S. DARPA project that looked at a number of issues related to disruption-tolerant networking. One of the outputs of that project was an explicit transport error notification (ETEN) [KR02] scheme aimed at assisting in such networks. As with TP-Planet, the approach here is the standard networking one of using simulations to examine alternatives to standard TCP. In the case of this project, their stated aim was to determine whether or not transport-layer performance could be improved via the use of explicit error notifications.[28]

The basic idea behind the approach is to note (again) that standard TCP assumes the existence of congestion whenever packets are not successfully acknowledged. However, the packet may have been corrupted or lost in-transit rather than dropped from a congested router. If the packet was corrupted or lost, then there may in fact be no reason to apply the multiplicative-decrease antidote that is part of the normal TCP AIMD scheme. Thus, the logic here is that if one can explicitly notify the source and/or destination about errors, as opposed to congestion, then it may be possible to improve the performance of TCP in networks that are likely to see more disruption than congestion.

ETEN defines a few different ways to handle errors. The ETEN message can be returned to the source or sent on to the destination or in some cases (say if the error affects the IP source or destination fields), then perhaps only cumulative error information will be contained in the ETEN messages. However, the ETEN approach, as a TCP variant, still suffers from some fundamental problems as far as a generic DTN protocol is concerned, mainly that we are still wasting time with TCP's chatty session establishment and slow-start. And as with TP-Planet, this approach simply fails if no end-to-end connection ever exists between the source and destination. **(-R.1, -R.2)**

It is worth noting in passing that ETEN is complementary to the standards-track Explicit Congestion Notification (ECN) scheme, [RA01] which might allow a node in the right place, that notes the absence of the ECN flag, to implicitly determine that packet loss is not due to congestion. As an explicit notification ETEN would, were it deployed, presumably offer more certainty and hence produce better overall network performance.

---

[28] http://www.sterbenz.org/jpgs/sumowin/

SCPS-TP – A Space Oriented TCP Variant

The Consultative Committee for Space Data Systems, (CCSDS[29]) is an international standards development organization dedicated to developing data handling protocols specifically for space missions. CCSDS has therefore defined a number of protocols for use in space missions and amongst those is a TCP variant called the Space Communications Protocol Standards – Transport Protocol (SCPS-TP). [CC06, DU97]

SCPS-TP is frequently used between so-called Performance Enhancing Proxies (PEPs) that are used to hide the higher latency involved in TCP paths that include a geostationary satellite – in that case SCPS-TP is said to act as the "inter-PEP" protocol [SC05c] with typical latencies measured in the hundreds of milliseconds (36,000km up and down is about 250ms LTT, routing delays can add as much again). [GL98]

SCPS-TP makes use of a number of features of other TCP variants and defines some new TCP options in order to increase the distance (in terms of LTT), over which TCP remains usable. Among these are two of interest – use of TCP for transactions (T/TCP) and selective negative acknowledgements (SNACK – apparently first defined in SCPS-TP[30]).

T/TCP, defined in RFC 1644, [BR94] specifies a variant of TCP intended mainly for use to support transactional applications where the normal TCP 3-way handshake is problematic and/or where the usual TCP session closing scheme consumes too much time.

T/TCP avoids the 3-way handshake via the use of some shared state, so that the initial SYN packets from the sender (and subsequent packets) can contain a connection count (CC) TCP option that essentially identifies the connection and protects against erroneous processing of repeated packets. If the shared state is not present, (e.g. following a reboot), then T/TCP falls back to the normal 3-way handshake, which could clearly be problematic in a DTN if it occurred frequently. By itself T/TCP still requires contemporaneous end-to-

[29] http://www.ccsds.org/

[30] Personal communication from Robert Durst, who said he had "adapted it from work that Richard Fox had done in RFC 1106 (his NAK)".

end connectivity **(-R.1)** and hence is problematic for use in DTNs, but at least it no longer requires the 3-way handshake for every connection.

The SNACK scheme used in SCPS-TP allows TCP senders to specify a list of "holes" (gaps in the set of packets that were received). Once the peer receives this information it re-transmits all of the relevant packets, required to fill the holes. The list is however carried as a TCP option, and so is somewhat limited in terms of the number of bytes available – essentially the sender is limited to describe a small number of holes in each segment.

There is also a security implication here, that perhaps would not have been generally considered at the time SCPS-TP was developed – SNACK provides a potentially damaging denial-of-service attack, with significant amplification **(-R.13)** – if a bad actor forged a packet containing SNACK options that specify large holes, then the receiver will needlessly re-transmit many bytes, thus potentially contributing to a denial of service at the peer, or some intermediate router.

So SNACK appears to increase the requirement for some lower layer security, which in the case of SCPS-TP would presumably be IPSec. [KE05] IPSec in a DTN context however, is problematic since its key exchange scheme, the Internet Key Exchange (IKE), [KA05] requires multiple roundtrips. **(-R.2)**

In the main use-case for SCPS-TP, (as an inter-PEP protocol) the satellite hops are at GEO distances or less and so are not nearly as delayed as, for example, a deep-space link. As we've seen, SCPS-TP still requires contemporaneous end-to-end connectivity and also increases the requirement for IPSec, and so, while SCPS-TP can address some DTN contexts, it does not provide a general transport layer for DTNs that would cope well with higher delays. **(-R.2)**

Datagram Congestion Control Protocol

The Datagram Congestion Control Protocol (DCCP) [KO06] is a transport protocol that "provides bidirectional unicast connections of congestion-controlled unreliable datagrams"[31] that, at the time of writing, is being standardized in the IETF working group of the same name[32].

DCCP attempts to provide a reliable connectionless transport that is efficient for large flows but which is also TCP-friendly, in various senses, e.g. not unduly causing bad TCP performance when TCP and DCCP flows share common links. For an application, using DCCP would be very like using UDP, but with the main difference that the UDP-consumer has to implement its own congestion controls, whereas the DCCP-consumer has a choice of a couple of built-in congestion control schemes, called TCP-like congestion control and TCP friendly congestion control (TFRC).

DCCP however is (not unexpectedly) also targeted mainly at flows between current mainstream Internet hosts, for example calling for an initial default timeout of 200ms, and having the same maximum segment lifetime as TCP, namely 2 minutes. Crucially, DCCP also begins with a three-way connection initiation in the same way as TCP and is therefore similarly problematic when used in a DTN. **(-R.1,-R.2)**

2.4.3 Congestion Control Schemes

While describing many of the above protocols we made the point that they reacted to packet loss with an assumption that the cause for that loss is congestion. While that assumption is clearly not safe in a DTN, we do still also have to consider congestion handling.

Although the whole area of DTN congestion control is in its infancy, there are a couple of (more-or-less) protocol-independent congestion control schemes that are worth a mention here.

[31] http://www.read.cs.ucla.edu/dccp/
[32] http://www.ietf.org/html.charters/dccp-charter.html

TCP Friendly Congestion Handling

As mentioned above DCCP includes a congestion avoidance scheme called TCP Friendly Rate Control (TFRC) [HA03]. In this section we briefly consider how this, or similar rate control schemes, might match our DTN requirements.

The basic idea with TFRC is to try to limit the extent to which potentially "greedy" protocols (like DCCP, or a DTN protocol) might impact other flows in the Internet. A protocol is greedy if it attempts to monopolize the bandwidth on a link, and most naïve approaches to DTN protocols will, in fact, be greedy, since to some extent greedy is the other side of the coin to chatty. (A chatty protocol gets to test the channel a number of times and can adjust its behaviour so as to be fair to other flows; a non-chatty protocol cannot do that, and is therefore more likely to try to grab an unfair amount of bandwidth.)

TFRC is a receiver-based scheme, which means that the receiver gives feedback to the sender as to recent packet arrivals, and the sender uses this to calculate a sending rate so as to remain fair to other flows that are on the set of hops used by the TFRC flow. **(-R.2)** Essentially, this calculation is based on the TCP throughput equation [PA98] and has the goal of allowing the TFRC flow consume about as much bandwidth as would a TCP flow over the same path.

TFRC however, doesn't explicitly take into account the potentially high latencies as might be found in a DTN and there is evidence that TFRC underestimates its fair allocation of bandwidth with higher delays (e.g. 500ms) [RH05] **(-R.2)** There is also a recent "small packet" variant of TFRC [FL07] where the use of a TFRC variant tailored for the use of 1500 byte packets is defined. Whether this may perform better for higher latency links is not yet clear, for example Sathiaseelan and Fairhurst [SA07] conclude that "the current algorithm…suffers for paths with appreciable delay" when considering voice traffic over satellite links.

DTN-Specific Models for Congestion Control

Burleigh and Jennings [BU06] provided a model for DTN congestion control based on an analogy with financial transactions where nodes in a store-and-forward DTN consider their temporary storage of packets as an investment – their model proposes a set of investing rules that determine when the packet should be stored, dropped or negatively acknowledged. Since the model has only been tested in very simple scenarios its suitability as a generic approach to DTN congestion control is still an open question. The model is also currently quite specific to congestion control for the bundle protocol, (e.g. it assumes packets have an expiry time field) though most aspects of it might generalize to cover other DTN protocols.

Seligman et al, [SE06] propose another DTN specific scheme where storage congestion is ameliorated by moving some stored bytes to another "nearby" router that can then further forward the bytes, possibly at a later stage. While this scheme doesn't address cases where overall "local" storage is congested, it does appear to nicely solve some problems where the sender doesn't know which of a set of potential storage points to use, and so allows a form of load-balancing that might be quite useful.

In summary however, the proper handling of congestion for DTN protocols is not yet a well-understood area and is actually perhaps less pressing than is congestion control in non-DTN cases, due to the fact that many current DTN scenarios are very application specific and don't involve much, if any, sharing of bandwidth. Presumably, as DTN matures, congestion control will however, become an increasingly important aspect to consider.

2.4.4 Lower layer protocols

In addition to the DTN-specific protocols we will examine later, we can identify a number of other DTN-like protocols that have been defined over the years. In this section we very briefly consider three of these that operate at or below the network layer.

The first was actually intended as a joke, but has in fact been fairly influential, in terms of both how it has been perceived, but more so in terms of how it provides a demonstration that other physical media might produce unusual results (e.g. very high loss rates) when used to carry IP traffic.

This is of course the IP over avian carriers, or Carrier Pigeon IP (CPIP) [WA90] protocol. In a classic example of the computer literate being over-literal, this was in fact implemented in Norway in 2001, producing ping traces as shown in Figure 2.3. (Since no

```
Script started on Sat Apr 28 11:24:09 2001
vegard@gyversalen:~$ /sbin/ifconfig tun0
tun0 Link encap:Point-to-Point Protocol
  inet addr:10.0.3.2 P-t-P:10.0.3.1 Mask:255.255.255.255
  UP POINTOPOINT RUNNING NOARP MULTICAST MTU:150 Metric:1
  RX packets:1 errors:0 dropped:0 overruns:0 frame:0
  TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0
  RX bytes:88 (88.0 b) TX bytes:168 (168.0 b)
vegard@gyversalen:~$ ping -i 900 10.0.3.1
PING 10.0.3.1 (10.0.3.1): 56 data bytes
64 bytes from 10.0.3.1: icmp_seq=0 ttl=255 time=6165731.1 ms
64 bytes from 10.0.3.1: icmp_seq=4 ttl=255 time=3211900.8 ms
64 bytes from 10.0.3.1: icmp_seq=2 ttl=255 time=5124922.8 ms
64 bytes from 10.0.3.1: icmp_seq=1 ttl=255 time=6388671.9 ms
--- 10.0.3.1 ping statistics ---
9 packets transmitted, 4 packets received, 55% packet loss
round-trip min/avg/max = 3211900.8/5222806.6/6388671.9 ms
vegard@gyversalen:~$ exit
Script done on Sat Apr 28 14:14:28 2001
```

Figure 2.3 - CPIP ping trace.

treatise on DTNs would be complete without referencing CPIP, this thesis can now be considered to have discharged its duty in that respect!) However, the CPIP reminds us that strange physical layers can produce interesting results – in the case of CPIP, any packet arriving is interesting.

In the Postmanet project [WA05], Digital Versatile Disks (DVDs) were used as the sole physical layer medium between rural schools in India and a central Internet connected station. Practically, the Postmanet scheme is tailored to use for Web browsing, but differs from other offline browsing approaches in that the client here is never connected to the Internet, but only sends and receives bytes via DVD.

DVDs are automatically written at the client end, then sent, via the postal system, to a central repository that is connected to the Internet. There, the DVDs are automatically read, queries are submitted, and new DVDs containing response content are written. All central DVD handling uses an automated DVD robot processor, so that responses can be returned to the clients cost effectively.

What's interesting about Postmanet is the argument that the capacity of physical storage media will continue to increase more quickly, and more cheaply, than more commonly used communications media. While this argument doesn't quite seem compelling, it is at least very suggestive that other applications could make use of such a network, and not only perhaps for communications in such rural areas.

One could envisage some applications (e.g., delayed-video-on-demand) that could even make commercial use of a network such as this. In fact this is somewhat reminiscent of various commercial DVD (movie) rental services. The relative symmetry of the channel might also be interesting, for example, allowing backup and recovery services to be offered. All in all, the ability of a DTN protocol to be layered on top of this physical medium is very interesting. With such a layering, one could allow applications to make use of the transmission medium without the application itself having to be aware that this is occurring.

But the main point of Postmanet is the following: if a new 4.7GB of data arrives each day in the post, then that is the equivalent to a continuous data rate of ~456Kbps being used on a 24-hour basis[33]. With either HD-DVD or Blue-ray capacities one could achieve data rates equivalent to, or higher than, basic DSL service. Of course, it may not be possible to actually provide a useful 4.7GB of data each day, but then again perhaps not all DSL connections are usefully busy all the time either. And to make use of the bandwidth one needs applications that can handle the latency (e.g. wwwofffle[34]) and perhaps new application programming interfaces (APIs) as well.

We have one further lower-layer DTN-like protocol to consider: Fidonet.[35] [BU93] Fidonet was a more or less ad hoc[36] network that developed through the 1980s, before it was easily possible to connect to an Internet service provider (ISP). Though Fidonet is really only of historic interest, some aspects of it are reminiscent of some of our DTN requirements. The network was based around the idea of computer-to-computer phone calls. When a message

[33] 4.7*8*1024*1024*1024/(60*60*24*1024)= 456
[34] http://www.gedanken.demon.co.uk/wwwoffle/
[35] http://www.fidonet.org/
[36] "Ad hoc" is used here in the sense of being self-organized without commercial carrier involvement, not in the current sense of ad hoc networking.

was to be sent from one node to another, then at some point the first computer would directly dial the second and transfer the message. One of the reasons for Fidonet was simply the costs associated with long-distance calling. At the time, in the United States, local calls were generally not billed per minute, so any two local computers could dial one another anytime at essentially zero residual cost.

However, long-distance calling was billed per minute and was much cheaper at night, so it made sense to route messages so that those destined for remote locations were first centralized in the local calling area, and were then forwarded late at night. In fact, in Europe and elsewhere even local calls were billed per minute, so dialing up late at night made even more sense.

This naturally gave rise to a hierarchy, roughly related to the phone network's calling areas, but also to a naming and routing scheme and a set of associated protocols. Essentially, the requirement to centralize messages for the long-distance connection and also to make use of cheaper calling times meant that Fidonet ended up representing a quite sophisticated DTN!

The Fidonet hierarchy was structured into so-called zones where each zone was essentially a continent, then a local network (equivalent to a telephony area code), and finally, a node number and an optional "point," so, for example, "1:105/6.42" represented a machine in the US (zone = "1"), located in Portland, Oregon ("105"), where the host was host number 6, and the "point" was "42." A list of all the hosts and their contact points (i.e., the number to dial) was maintained in each locality with modifications widely distributed to other locations at frequent intervals (e.g., weekly). The "point" concept was for a machine that could connect to the network, but which wasn't generally addressable, as it didn't publish contact information. Each such machine had to hang off of a registered node with published contact information and that node was held responsible for all (mis)behavior of its associated "points."

Over the 1980s and 1990s, Fidonet grew to the point where there were tens of thousands of nodes registered worldwide. However, it was eclipsed by the widespread availability of the Internet, which shows that no matter how technically appealing a DTN protocol may be, once there is a simpler solution available, that's what will be used. That lesson may, in the future, be relearned were, say, an urban IEEE 802.11b/g-based data mule network to become outmoded by a WiMAX[37] network that, whenever it does arrive, promises continuous metropolitan area coverage.

2.5 Summary

In this chapter we examined the background to DTNs, in particular the causes of delay and disruption that affect these networks and we derived a set of requirements that we would like DTN protocols to meet. We saw that standard protocols, most importantly TCP, but also many proposed variants, are problematic to the point of being unusable in some DTNs. In the end, we have seen that there is a niche where current protocols do not suffice, so our next step is to examine the main current proposals for DTN-specific protocols.

[37] http://grouper.ieee.org/groups/802/16/

Chapter 3

DTN-specific Protocols

In this chapter we present the main open research group developing DTN protocols (the DTN Research Group) and outline the two main DTN protocols that are currently being developed by that group.

The first of these is called the bundle protocol (BP), which in essence works in very much the same way that email works today, but aimed at being integrated into other applications rather than being an application for end users. The second is called the Licklider Transmission Protocol (LTP) and is a point-to-point protocol tailored for use with very high-latency links.

The BP is an example of what is generally called an "overlay network protocol" [AB03] and is depicted in Figure 3.1. As an overlay network protocol it can be run above the current suite of Internet protocols as well as over the more esoteric protocols used by spacecraft or those proposed by researchers for dealing with complex sensor networks and other challenged environments.

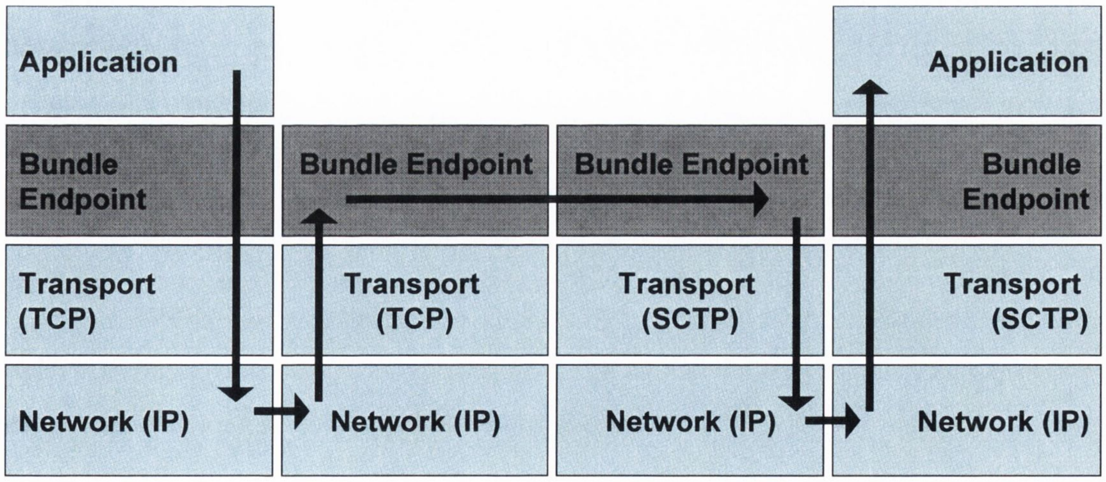


Figure 3.1 - The overlay network protocol approach showing the bundle layer in dark gray (adapted from [CE07]).

	Deep space network	Sensor network
"Edge" nodes	Spacecraft	Sensors
"Middle" nodes	Earth stations	Data mules
"Sink" nodes	Internet host	Internet host
Edge/Middle Connectivity	Via schedule and ephemeris	Via schedule and radio range
Middle/End connectivity	Via application layer gateway	Scheduled/ when mule "at home"
Main network constraint	Spacecraft power	Sensor node power
Application traffic profile	Telemetry and science payloads	Telemetry and sensor data
Scale	Few, highly valuable nodes	Many relatively inexpensive nodes
Computational complexity	Highly constrained processors but an ability to expend significant computational resources on coding/processing.	Relatively constrained processors and limited storage
Overheads	Encoding must be highly terse since bandwidth is at a premium	Good bandwidth for individual links but multi-hop protocols could result in bandwidth scarcity

Table 3.1 - Deep-space/Sensor network analogy.

Many of the problems that arise in DTNs are quite well addressed by the BP. However there is also sometimes a need for delay tolerance at a lower layer in the network; mainly to handle cases where there is either very high latency or intermittent connectivity between one host and the next. The classic example is the connection between the orbiter and the Earth station mentioned in Section 2.1.

There are also terrestrial applications that require similar behavior, for example, if no contact between two machines will ever be sufficiently long to complete an application layer exchange. In such cases we need some way to handle forwarding of data one part at a time, where there may be a delay of hours between each partial transmission. Essentially in that case we may need a delay and disruption tolerant point-to-point protocol, like LTP.

In fact, some of those terrestrial sensor networking cases, (particularly where they use "data mules" [SH03]), are actually tightly analogous to deep-space networks. For our present purposes it is perhaps sufficient to note that data-mules that physically traverse a field of sensor nodes are quite like Earth stations in a deep space network as can be seen from Table 3.1 which compares such networks. As with all analogies, this ultimately breaks down, perhaps primarily in terms of the numbers of sensors expected versus the number of spacecraft, however, it is worth noting that current sensor networks also tend to involve limited numbers of nodes and that the total number of spacecraft supported by the DSN today is perhaps not that dissimilar to an experimental-scale sensor network.

Before we deal with the detail of the BP and LTP, it will help to understand the various groups that are involved with this work.

3.1 The Genesis of DTN Research

At base, the "vision" behind the DTN effort is to try to extend the Internet architecture to cater for applications where delays or disruption are significant factors. That vision, in large part, was initially fostered and promoted by a relatively small group of people including Vint Cerf and some engineers at the NASA Jet Propulsion Laboratory (JPL) and elsewhere within NASA who started working on the so-called Interplanetary Internet (IPN) back in 1998. [BU02ipn]

The IPN group eventually morphed into an Internet society (ISOC[1]) special interest group, the IPNSIG[2], which had (and still has to an extent) a public web site and mailing list where discussions on relevant topics were held. The group working on the IPNSIG began development of an architecture for an IPN and made some progress towards developing protocols conforming to that architecture. In fact, much of that work survives into the current versions of the DTN architecture and protocols. Some work specifically on the IPN is also still ongoing within NASA.

However, the IPNSIG faced the problem that it is very hard to experiment with an inter-planetary network when no such network exists. It would also be very expensive to try to create such a network. At the same time, some people (including the author) were investigating how IPN concepts might apply to terrestrial applications, [FA03e] in particular sensor networks, which, as we've just seen, have a lot in common with a putative IPN. Since experimenting with a sensor network is a lot easier to do, it became clear that the IPNSIG was no longer the best venue to do work on this topic.

---

[1] http://www.isoc.org/

[2] http://www.ipnsig.org/

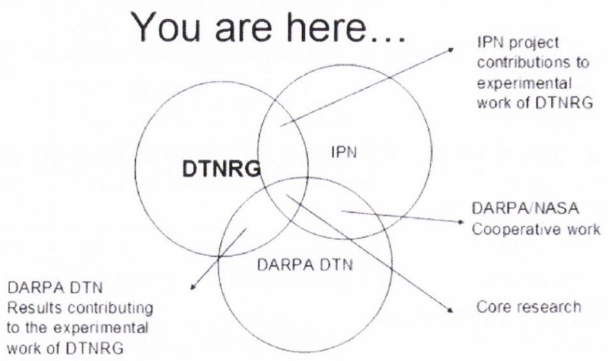

You are here...
DTNRG
IPN
DARPA DTN
IPN project contributions to experimental work of DTNRG
DARPA/NASA Cooperative work
Core research
DARPA DTN Results contributing to the experimental work of DTNRG

Figure 3.2 - A diagram relating the main organizations working on DTN topic. (Image credit: Vint Cerf & DTNRG members.).

For this reason, an Internet Research Task Force (IRTF[3]) research group was formed to look at the more general area of Delay-Tolerant Networking (DTN) – that group is called the Delay Tolerant Networking Research Group (DTNRG[4]) and is currently the main open venue for work on the DTN architecture and protocols. The main protocols (the BP and LTP, described below) under consideration by the DTNRG are being developed with the aim of producing experimental RFCs[5] documenting the protocols.

Just to confuse matters somewhat, the United States Department of Defense, under their defense advanced research projects agency (DARPA[6]) issued a call for proposals in early 2004 [BA04] for what they called "disruption-tolerant networking" (also DTN!), which is yet another generalization of the same concept. The difference is that, until the DARPA call, the main focus of the DTN work was on high-delay cases like the IPN or sparse sensor networking (where sensor readings are not needed in real time). However, there are other types of disruption that can occur, e.g., radio shadowing, frequently passing in and

[3] http://www.irtf.org/

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

[5] An RFC is a Request For Comment, the Internet's archival document series. Experimental RFCs specify protocols that are of interest to the Internet community, but that are not yet ready for widespread deployment.

[6] http://www.darpa.gov/

out of range of a base station etc., and that are not properly reflected in the phrase "delay tolerance". Whether the D in DTN will come to mean '*disruption*' or will continue to mean '*delay*' is not yet clear, but in any case, the same architecture and protocols can hopefully serve in both contexts.

In any event there are therefore a number of different, but overlapping and collaborating, groups of people working on this topic. Figure 3.2 is a graphic that was used at a recent DTNRG meeting to explain this to the audience. The DTNRG work is, at the time of writing, coming to fruition. The DTN architecture [CE07] has been published as Internet RFC 4838, and the BP specification [SC07] as Internet RFC 5050. LTP is described in three Internet RFCs: RFC 5325 covers the motivation for the protocol [BU08], RFC 5326 specifies the protocol itself [RA08] and lastly, RFC 5327 on protocol extensions, [FA08] mainly addressing security issues. Work on other topics, such as BP security and DTN routing, is ongoing and will hopefully lead to further RFCs in the months and years ahead.

3.2 The Bundle Protocol

The bundle protocol packages a unit of application data along with any required control information into a "bundle", which is very similar to an email message. The bundle is then forwarded along a route consisting of a number of intermediate machines that may each store the bundle for significant periods. So, the bundle protocol is an overlay network store-and-forward protocol.

For example, if the source machine is a lander on Mars, it may create a bundle but not be able to forward it to a Mars orbiter for a few hours until the orbiter is next overhead. When the orbiter receives the bundle it, in turn, may have to store the bundle until its next scheduled contact with an Earth station. When the bundle is received by the Earth station it can be quite quickly forwarded on to its destination, perhaps the desktop machine of a scientist studying some Martian rock formation. The overall delay could be hours, or even longer if sufficiently intense rain disrupted the orbiter to Earth station contact, in which case it may take days for the data to eventually arrive at the scientist's desktop computer.

The overlay network approach as represented by the bundle protocol represents the mainstream DTN approach in terms of the number of people who are working on its specification and development. The bundle protocol is described in two main documents:

The DTN architecture document [CE07] introduces the general overlay architecture, puts this in context in terms of applicability and introduces key architectural terminology. The Bundle Protocol Specification [SC07] specifies the formatting of bundles and the processing rules associated with sending and receiving bundles. There are also a few subsidiary documents related to security, an overview [FA07s] and a document defining security extensions for the bundle protocol [SY07].

Aside from DTN, overlay networks have been proposed for a number of reasons, ranging from increased resilience [AN01], to security and privacy [DI04, CH81] and accelerated content delivery [SA02]. To the best of our knowledge none of these has considered very high-latency communications, with the exception of some anonymizing networks [MO03] which in fact create high latency as part of their approach to disguising traffic patterns. Nevertheless, it is clear that an overlay approach can address DTN requirements – be that via development of a specific new overlay (like the IPN as originally envisaged), or else via modifications to some current overlay network. However, given that most overlays to date tend to be application- or function-specific (other than the X-bone [TO01]), a DTN-specific overlay network based around the BP is a reasonable option to investigate now.

3.2.1 Basic Bundle Protocol Concepts

A DTN node is considered to be any entity that runs an instance of the BP and so can in principle send and/or receive bundles. There may be exceptional nodes that can only ever transmit, (e.g. a very simple sensor). Nodes are identified by endpoint identifiers (EIDs), which are the BP's equivalent of addresses. Syntactically, EIDs are uniform resource identifiers (URIs), [BE05] and each can refer to one or more bundle nodes, i.e. one EID could refer to a set of bundle nodes. (That last property of EIDs is essentially future proofing in order to support potential multicast-like modes [SY07m] of operation for the BP.)

Clearly then some component of each DTN must map from EIDs to lower-layer addresses when deciding how to forward bundles. The DTN architecture calls for this to follow the late-binding principle [FA03] so that the URI, (or in particular its DNS hostname part), should be resolved into a lower-layer address as close to the final destination as possible.

How that late-binding is eventually resolved, is not currently specified in any detail in either the architecture or the BP specification.

The next concept to consider is the "contact," which tries to capture the idea that not all nodes will be contactable at any given moment. According to Fall, [FA03] "The DTN architecture is targeted at networks where an end-to-end routing path cannot be assumed to exist. Rather, routes are comprised of a cascade of time-dependent *contacts* (communication opportunities) used to move messages from their origins toward their destinations. Contacts are parameterized by their start and end times (relative to the source), capacity, latency, endpoints, and direction."

This is in contrast to the trend in the Internet where we more and more consider that addressable entities are online all the time. In DTNs however we need to explicitly consider that communication is only possible at certain times, and perhaps also with additional time-varying constraints. As we'll see, the handling of those constraints is a major differentiator between DTN and other protocols.

Since bundles have to traverse lower-layer networks, they are ultimately subject to whatever restrictions exist on those networks in terms of maximum packet sizes. For example, on most IP networks it is safest to assume that single packets should be less than 1500 bytes long. [MO90] Other DTNs may be able to support forwarding of much larger bundles but may be subject to disruption of the lower-layer connections. Of course, in many such cases, e.g., where the lower layer runs over TCP, then the bundle will not in fact be fragmented (at the bundle layer) thanks to the retransmissions of lost IP packets being handled by the TCP layer. But in any case, the BP has to include support for fragmentation and re-assembly of bundles which it does in two forms – proactive and reactive fragmentation. In the former, the bundle node basically deliberately re-fragments due to knowledge about the outbound path for the bundle.

The reactive fragmentation option is intended to cater for cases where a link is unexpectedly broken after some significant amount of a bundle has already been transmitted and received. The idea is that the receiving bundle node can in any case forward on the fragment(s) it has already received. Whether or not this feature sees much use will be interesting to see – reactive fragmentation creates a number of potential problems, (e.g. with security when only some fragments are encrypted), so it may not see widespread adoption.

The BP's lower-layer protocol specifics are handled via so-called convergence layers (CLs). CLs for TCP, UDP, and other lower-layers (e.g. USB stick) have been implemented in the reference BP implementation, although only the TCP CL has so far been fully documented. [DE06] When speaking of TCP as a CL, we of course include a small "shim" layer required to glue together the BP and the TCP API (presumably the sockets API).

DTNs can clearly be subject to extreme constraints in respect of bit rates. For example, if some nodes near the edge of the network are running over an extremely low bit rate radio link, as was the case with say the Galileo probe to Jupiter where (due to a malfunction) bit rates in the tens to hundreds of bits per second were common, then a highly bit efficient encoding scheme may be required. For the BP, there is, in fact, a compressed header format [BU07c] defined for exactly this reason.

Another BP concept that needs to be understood is "custody." DTNs using the BP are essentially store-and-forward networks that don't require contemporaneous end-to-end connectivity. If, at a given moment in time, a bundle is only stored on one router somewhere in the "middle" of the network, then should anything go wrong, only that node is in a position to re-transmit the bundle. The idea of custody is that a node accepts a bundle for storage, with a commitment to keep the bundle until the bundle either expires or else is successfully delivered to the next custodian on the route to the destination. Custodians are therefore the nodes that can re-transmit the bundle if something has gone wrong.

3.3 The Licklider Transmission Protocol

LTP tackles delay tolerance and disconnection in a point-to-point environment with an emphasis on operation over single, but typically very high-delay, links. The canonical use case for LTP is a single-hop, deep-space link typically between a remote spacecraft and an Earth station. Such links suffer from long light-trip times, occultations and Earth-station scheduling restrictions.

For example, if an orbiter is about to be eclipsed behind its planet it may still send a block of LTP data, and, knowing that an acknowledgement cannot be received until the orbiter is no longer eclipsed, the orbiter can freeze all the timers that drive the operation of the protocol for the duration of the eclipse. Once out of eclipse, the spacecraft may once more restart these timers. This concept of frozen or "punctuated" timers is a crucial aspect of our LTP implementation (see Chapter 5 for details).

From the above, one might think that LTP is only useful for space communications. However, this mode of operation can also be very useful for terrestrial applications where disruption is highly likely. [MC07] Applications dealing with disruptive environments can be conventionally structured so that the application handles the expected errors, or else, using a protocol like LTP we can essentially isolate the application from all of this complexity, by, e.g., having a communications daemon that handles all disruptive events, in this case perhaps retransmissions required after a host reboots. In the case of a sensor network the resulting application layer sensor code can be simple, yet with reliable transmission over disrupted connections.[9]

It is important to understand that LTP is a point-to-point protocol, so there are no routing or congestion issues to consider for LTP itself – bytes are simply transferred between two peers with no intermediaries considered.

LTP is designed to be a potential CL to support the bundle protocol, although it can also be used in other contexts. So BP/LTP is a valid scenario, and has been implemented in the NASA JPL "ION" code mentioned in Section 3.4 below. In other contexts we have seen uses for LTP/UDP, e.g. between nodes in a sparse sensor network. LTP/UDP was also the variant used in DTNRG interoperability tests. [DT06] Once we begin to consider LTP/UDP, then, of course, some congestion issues do arise, for example, one should

[9] The SeNDT sensor service provider interface (SPI) allows the sensor code to interact with whatever device is in use and then simply call "ltp_sendto()" – the resulting sensor readings will be cached until the next available contact, even if that contact is subsequent to a reboot.

consider how to ensure that an LTP/UDP flow doesn't try to monopolize the available bandwidth to the detriment of TCP flows sharing some parts of the UDP path.

As we've seen, to operate in a DTN, a protocol cannot be chatty like TCP, since requiring any round trips before application data flows is just not an option. As a result, LTP effectively has no negotiation and all parameters required for interoperability have to be established on both sides before a contact occurs. LTP is consequently highly stateful, requiring relatively large amounts of information about previous and upcoming contacts.

While LTP uses a fairly standard set of protocol primitives for handling timeouts, re-transmissions, data integrity, origin-authentication, reliability and other on-the-wire issues, all of this is done without requiring multiple (or any) roundtrips prior to sending application data. Where LTP really differs, due to its deep-space heritage, [CC07] is in its concept of lower layer cues supporting scheduled communications. One can think of an LTP implementation sitting on top of a separate "layer" that knows the network state sufficiently well to tell each peer when and how much to receive and transmit. As it turns out this is also a very nice way to handle a sensor network using data mules. [MC07]

In addition, if a sender can only communicate with a receiver once each hour for one minute, and the sender expects an acknowledgment message from the receiver within two minutes of sending a message to him, then the appropriate timer to use is one that will take two hours to expire. The two minutes represent what might be called punctuated time and not elapsed time, where punctuated time is the continually-being-interrupted duration of the scheduled contacts with the peer in question.

Once such punctuated timers are maintained independently for each LTP peer, and in each direction, then a fairly high degree of delay tolerance has already been achieved. High-latency cases are thus handled by ensuring that the lower layer cues reflect the in-transit or scheduled latency of the communication, which in the case of a deep-space contact will mainly consist of the LTT between the peers in question.

3.3.1 Some LTP Details

We now introduce the most important LTP details that will be needed as we go forward. The set of bytes that an application transfers in a single LTP session is termed a "block."

When a block is bigger than the local maximum transmission unit (MTU), then it must be fragmented, into LTP "data segments."

LTP supports a model of partial reliability, where some or all of the data segments constituting a block can be retransmitted in the event of errors. We call the part of the block that is reliably transferred the "red part" of the block and the part that is transferred subject only to best effort the "green part." In LTP the red part of the block must be transmitted first, that is, the red part consists of the initial set of data segments of the block that are marked "red," and the green part consists of the remaining data segments up to the end of the block (EOB).

The red/green marking is encoded in the segment type, for example an initial red data segment will have a type of '0x00' and a green data segment will have a type value of '0x04.' The last red data segment is marked as a checkpoint (e.g. segment type '0x03') in order to allow the receiver to detect the boundary. This checkpoint also triggers the receiver to send a report segment as described below so that the sender can re-transmit any segments that may have been lost, thus providing reliability.

This feature allows, e.g. the control information (location, time, camera orientation, codecs, filters etc.) associated with a deep-space image to be transferred reliably since that information is all required in order to make any use of the image. Specific image pixels however can be green since their loss is less significant. A block can be partly or fully red or green. Data segments (DS) that are in the red part are selectively acknowledged using "report segments" (RS) that can each specify which range of red-part bytes have been successfully received to date.

In addition to including the EOB flag in the last DS, LTP also allows the sender to nominate any DS as a "checkpoint" (CP). On receipt of a checkpoint DS, the receiver must generate an RS reflecting its current state and enqueue that for the sender. The idea is that if one were dealing with a large block that this may be beneficial in terms of lessening the sender's need to store already transmitted bytes of the block. Upon receipt of the RS replying to the CP, the sender can release storage associated with bytes already successfully received. The CP mechanism has also been found to be helpful in some error cases. [FA06e]

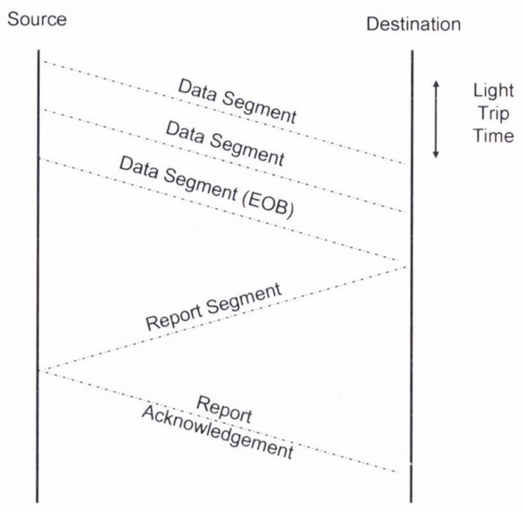


Figure 3.3 – Basic LTP operation.

To handle the potential loss of RS segments, "report acknowledgements" (RAs) are defined and there are also a few other segments types related to session cancellation. LTP control segments are those that are related to LTP session control, i.e. everything except the DS. The LTP specification requires that control segments be prioritized over data segments when de-queuing segments for transmission.

The extensibility mechanism of LTP essentially allows the addition of a set of type-length-value (TLV) tuples to each segment, into which whatever extensions are required can be placed. In order to provide some security, there are data integrity and cookie extensions defined for LTP, which essentially try to make it harder to succeed with a denial-of-service (DoS) attack on an LTP node.

Figure 3.3 shows a simple "red" LTP session in the style of a normal TCP interworking diagram. The LTP sender is transmitting an LTP block (which is probably a higher layer PDU) using a number of data segments, typically with each DS attempting to "fill" an MTU.

In most cases, when all data segments have been sent the last data segment is marked as the EOB so that the receiver knows that all data segments should have been received. After the receiver has seen the EOB, then it will usually respond with an RS that contains a "map" of the parts of the block that were successfully received so far (some data segments may be missing or corrupt).

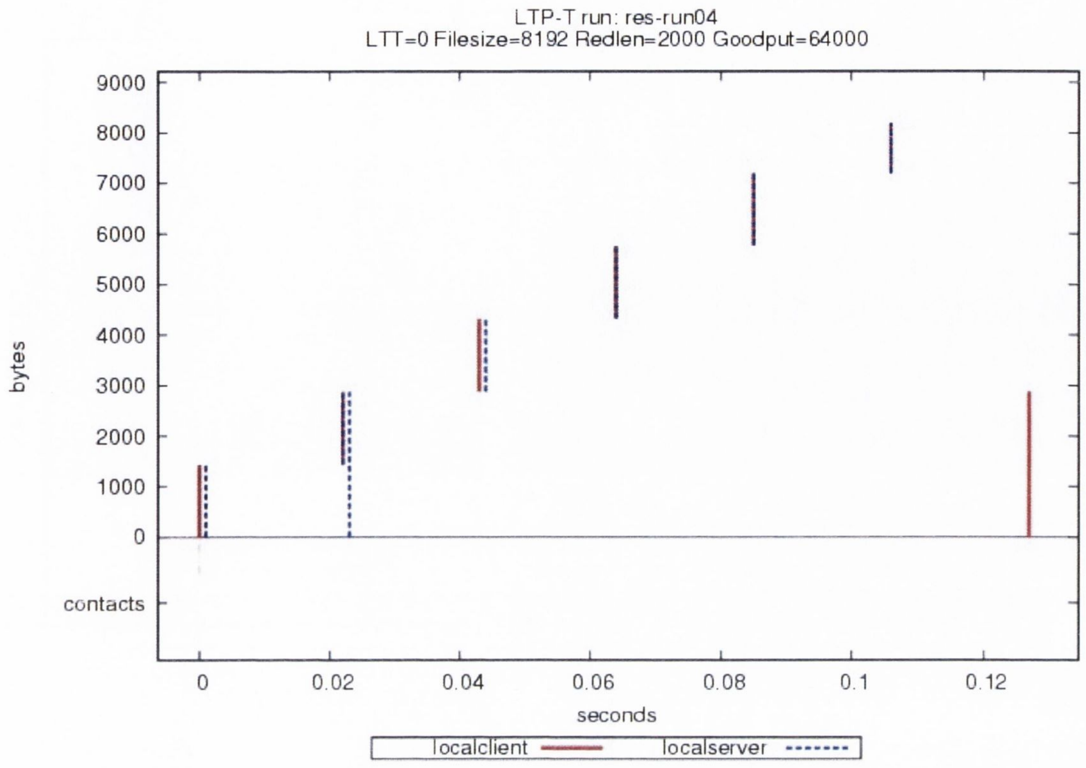


Figure 3.4 – A Small LTP Session.

If the EOB is lost then a timer is used to control its re-transmission. The RS (which may itself span a number of MTUs) is sent back to the sender who can then decide which bytes to re-transmit. The sender also acknowledges the RS with an RA segment.

3.3.2 Example LTP Sessions

In this section we give a few simple examples of LTP session flows and introduce the type of diagram that will be extensively used in later chapters. Additional examples can be found in Appendix B.

Figure 3.4 shows an example LTP session where a small file ("Filesize" is 8K) was transferred between two LTP instances, ("localclient" and "localserver"), running on the same host, so the light-trip-time was zero ("LTT") in this case. The number of red bytes requested to be sent ("Redlen") in this case was 2000 bytes. The "Goodput" figure represents the number of payload bytes transferred over the duration of the run. Full details of its calculation can be found in Chapter 6.

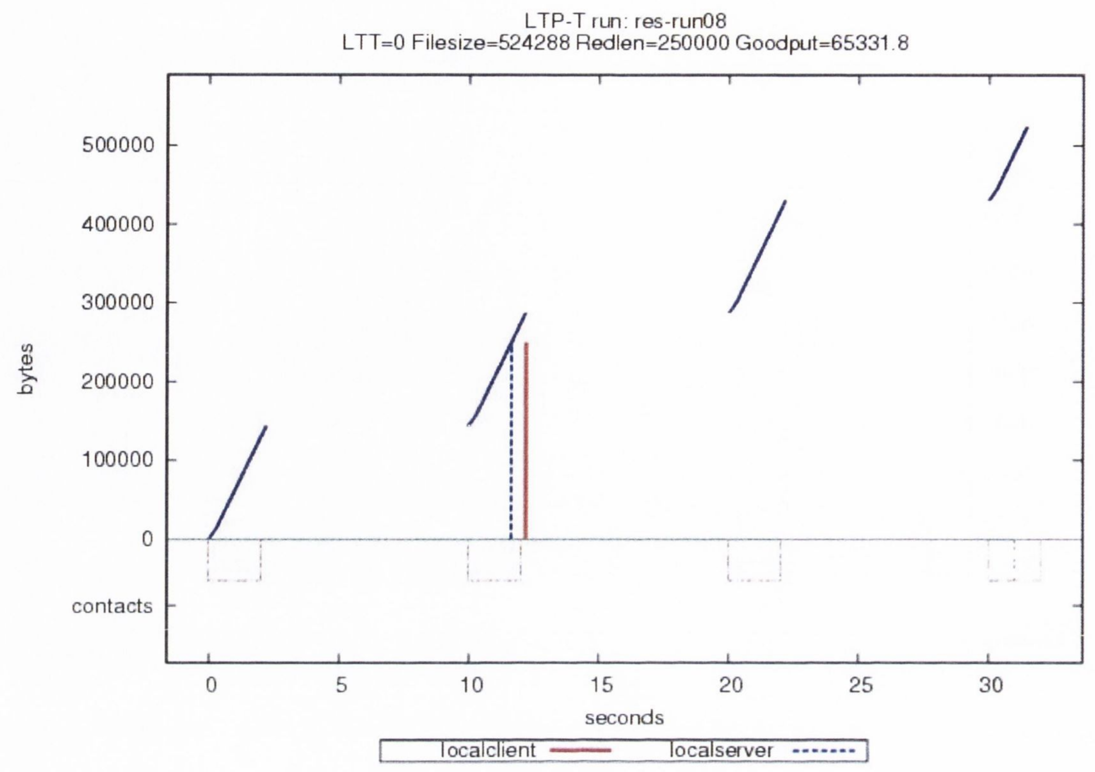


Figure 3.5 – A Medium Sized, Scheduled, LTP Session.

The x-axis shows the seconds elapsed since the first event of the run. This is calculated based on timestamps extracted from log files generated by our LTP implementation. The positive y-axis represents the bytes from the block to be transferred. The negative y-axis represents information about contacts, as can be seen in Figure 3.5 above.

The bars in the Figure 3.4 represent individual LTP segments. These appear as points in Figure 3.5 since the file size is much larger. For data segments, the bar represents the range of bytes sent in that data segments. Note that some of the client and server bars overlap in the diagram – in the Figure 3.4 run the last data segment was received less than one millisecond after transmission. For report segments, the vertical bar represents the upper and lower bounds of the range of bytes being reported upon. Report acknowledgements are not represented in these diagrams.

In Figure 3.4 the first two data segments are marked as red – the implementation rounded up the red length to 2880 bytes in order to make better use of the lower layer (UDP) MTU. The server generated a report segment, (at ~23ms), covering those. Subsequent data

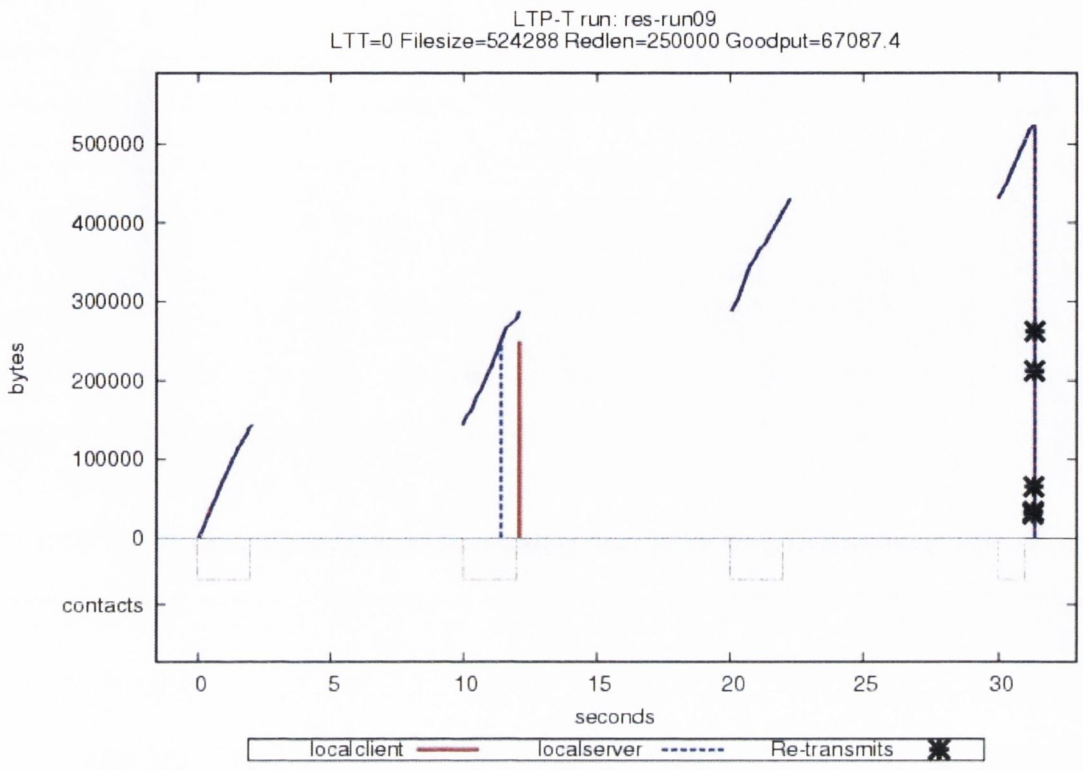


Figure 3.6 – An LTP Session with Packet Losses.

segments are sent by the client about every 20ms. Finally, (at ~127ms), the client processed the report segment and generated a report acknowledgement (not visible).

Figure 3.5 shows another LTP session, this time for a 0.5MB file with the red length set to 250,000 bytes, but this time with a schedule so that each node is "on" for 2 seconds out of each 10 second period. The contacts section of the diagram indicates this. As stated earlier, the data segments, (each of which fits into a single UDP packet), are too small to be seen, but the report segment (sent at ~11s, noticed at ~12s) can be seen.

Finally, Figure 3.6 shows the same setup as in Figure 3.5, but this time with some packet losses. Re-transmitted data segments are marked ("Re-transmits") and one can see the additional report segment following the last re-transmitted data segment. (Note that the goodput is higher here since the packet losses were generated by removing rate controls, so that the client was effectively transmitting too quickly for the underlying stack.

Going through the events in Figure 3.6 in more detail, from 0-2s the client transmitted 100 data segments, from 10-12s, another 100 segments were transmitted, of which 75 were red, and 25 green. The remaining green segments were transmitted as contacts allowed. At about 11.5s, the server responded with a report segment, indicating 4 missing red data segments and the client generated a report acknowledgement for that which was sent at about 12s (the delay being due to data segments that were already in the lower layer transmission queue).

At roughly 31.3s, the client re-transmitted the missing data segments (shown by bold crosses in the Figure) and the server responded with a new report segment indicating that the entire red part had now been received. The client then responded with a report acknowledgement for that, however, with the horizontal scale used in the Figure, these exchanges cannot be distinguished.

3.3.3 LTP History

LTP is essentially revisiting a problem space that has previously been examined by the Consultative Committee for Space Data Systems (CCSDS). In particular, LTP provides many of the functions provided by the CCSDS File Delivery Protocol (CFDP) [CC07].

One could therefore ask whether or not the IRTF approach to this protocol is significantly different to what the CCSDS has done, or whether it would be better to leave development of such protocols to the CCSDS. The CCSDS (it seems) generally takes a more OSI-style[10] approach to protocol development – specifying both the protocol and the service interfaces that are required for conformance.

For example, at one point the CFDP specification [CC07] requires an implementation to free up some buffers in response to receipt of a message. If we ignore the interfaces for the moment, there are clearly other protocol implementations possible (e.g., based on garbage

[10] OSI is the open systems interconnection model of networking promoted in the 1980s and 1990s, whose component protocols have largely been superseded by Internet protocols. The fact that the most recent version of the OSI reference model dates from 1994 perhaps shows that it is a somewhat outmoded approach.

collection), and the tradition in the Internet community is to be less prescriptive about implementations compared to the OSI approach.

In more closed communities, such as those building and operating spacecraft, following the OSI model more strictly is a reasonable approach, but one that arguably doesn't scale up to applications where there are millions of developers spread throughout the world with some being volunteers and others professionals. [BE02] The Internet approach of specifying protocols without interfaces has been demonstrably more successful over the last two decades.

The closed community aspect is perhaps also demonstrated by the lack of security mechanisms in CFDP. Equivalent Internet protocols (e.g., FTP) [PO85] have included significant security considerations [HO97] for at least a decade and one would imagine that no new Internet Standard protocol would be considered, which mandated that implementers, for example, unconditionally provide directory listings. Again, for space missions, this security model (or lack thereof) can be justified to an extent, but it is clearly unacceptable for a protocol intended to have wider applicability.

Some CCSDS protocols also exhibit layering violations for reasons that are, unsurprisingly, specific to spacecraft. For example, the CCSDS Proximity-1 [CC06p] protocol has primitives that allow a communicating peer to reset the clock on a spacecraft. In a space-mission context that is probably reasonable, but as it stands such a feature could be to the detriment of the host if deployed in many terrestrial environments. Since such features are actually necessary in some contexts, the LTP extension mechanism allows such features to be defined, but as extensions, they would only affect a subset of LTP implementations – presumably that subset where the utility of the feature in question outweighs whatever risk is inherent in its deployment.

It may also be the case that CCSDS, as a de jure standards organization, cannot as easily create the type of experimental protocol that is the main point of IRTF research groups like DTNRG. Given that LTP is an experimental protocol, implementers are aware that they should expect changes over time. In contrast, presumably CCSDS would only define new protocols in response to immediate mission requirements. The inherent backwards compatibility requirements in such cases can also tend to stifle innovation to at least some extent.

Having said all that, one can clearly see the CFDP heritage of LTP. Considered from an Internet protocol point-of-view, many of the more innovative aspects of LTP (e.g. partial reliability, no negotiation) are actually directly derived from CFDP equivalents.

3.4 Implementations

With protocols as complex as the BP or LTP there is clearly the potential to specify many seemingly sensible protocol features that turn out to be over complex, useless or simply broken. In the absence of running code, protocol designers will in fact include all three types of error in any reasonably complex protocol. Recognising the above, DTNRG members have produced "reference" implementations of both protocols, which has a number of useful consequences. Firstly, as pointed out, it keeps the protocol designers honest by acting as a reality check whenever paper-only plans are getting out of control. Secondly, a reference implementation makes it easier for people to experiment.

The BP reference implementation is freely available for download[13] and is currently maintained by a small team. There are various other BP implementations, some only of specific CLs, others more general, but those have not been as widely used in experiments. As far as we know, there are currently four LTP implementations, though only two have been released to date. The Ohio University LTP implementation is available for download[15] and is a Java implementation of LTP. The author's LTPlib 'C' library is also available.[16]

In November 2006, at the DTNRG meeting in San Diego, successful interoperability testing sessions were held for both the BP[17] and LTP[18]. [DT06] During the LTP testing,

[13] Look for "code" below http://www.dtnrg.org/

[15] http://masaka.cs.ohiou.edu/ocp/

[16] http://dtn.dsg.cs.tcd.ie/ltplib/

[17] http://www.ietf.org/proceedings/06nov/slides/DTNRG-1/sld1.htm

basic operation for both small and large (10MB) blocks was validated, in both directions. As the Java code didn't support specific extensions, all that could be validated was that the extensions produced by LTPlib were correctly skipped over, and in the case of the cookie extension, LTPlib correctly dropped the session once the responses didn't contain a correct cookie. During testing we also turned on some of the error generation in LTPlib, basically randomly dropping segments. For a large block, this resulted in reports that spanned more than one report segment, thus validating what might otherwise seem to be a relative corner-case for the LTP specification. This mechanism is fully explained in the base LTP specification. [RA07]

In general this event[19] was very successful (as was the BP testing[20]) and both were reported as highlights of the meeting for the IRTF[21] in the IETF meeting proceedings[22].

More recently, during the spring 2008 IETF in Philadelphia[23], the LTPlib implementation was successfully tested with the JPL ION implementation of LTP. This necessitated the addition of a new mode of operation for LTPlib, using e.g. single-byte identifiers, in order to match the (deep-space related) restrictions of the ION code. While only a very basic test was carried out, it does mean that LTPlib has now been interoperated against two different LTP implementations.

3.5 Summary

In this chapter we described DTNRG and the two protocols currently being specified by the DTNRG. In the next chapter we will look at the reasons to design a DTN-friendly transport protocol, which is intended to complement the BP and LTP.

---

[18] http://www.ietf.org/proceedings/06nov/slides/DTNRG-2/sld1.htm

[19] http://www3.ietf.org/proceedings/06nov/slides/DTNRG-2/sld1.htm

[20] http://www3.ietf.org/proceedings/06nov/slides/DTNRG-1/sld1.htm

[21] http://www3.ietf.org/proceedings/06nov/slides/plenaryt-1.pdf

[22] http://www3.ietf.org/proceedings/06nov/index.html

[23] http://maillists.intel-research.net/pipermail/dtn-interest/2008-March/003086.html

Chapter 4

DTN Transport

In this chapter we describe and justify the use of a DTN transport-layer solution for delay- and disruption-tolerant networking (DTN). In particular we demonstrate that there are various situations in which a DTN transport is preferable to the overlay network approach exemplified by the bundle protocol (BP). As stated earlier, the particular DTN transport protocol we have designed is called LTP-T - its details are set out in the next chapter. This chapter compares and contrasts the DTN transport approach with the overlay approach.

4.1 Terminology

It is, of course, important to be clear on the terminology that we use. This is especially important since there are cases where a DTN transport can look very like an overlay network and vice versa. And in this particular case, many of the terms about which we need to be precise, are terms that are in broad use in networking and are not specifically DTN terms. This section defines the meaning of such terms as used in this thesis.

We refer to a DTN transport when making generic arguments, and to LTP-T when dealing with arguments that are specific to our protocol design. Similarly, we refer to overlay networks when making generic arguments and to the BP when making arguments specific to the current BP. The protocols considered here use different terms to describe their protocol data units (PDUs) - "bundles" for the BP and "segments" for LTP & LTP-T; we will use the (in any case better) term "**packet**" when making protocol-independent arguments.

A "**host**" is a computer system that routes packets at some layer, though perhaps not using DTN protocols. A "**node**" is a host that runs a DTN protocol, regardless of whether that node is part of an overlay network running the BP or running a DTN transport protocol such as LTP-T. Not all hosts are nodes, for example, with the BP or LTP-T layered over the Universal Datagram Protocol (UDP), there can be a set of Internet Protocol (IP) routers (hosts) in between adjacent DTN nodes. DTN nodes are "**adjacent**" when there is a (possibly delayed or disrupted) communication path, consisting of one or more links, between them that involves no other DTN node.

A DTN "**hop**" refers to the delay-tolerant transmission of packets between two adjacent DTN nodes. A "**link**" is a connection between two hosts, which can be the same as, or different from, a DTN hop. For example, if some packets are sent via a satellite hop, with the ground stations being nodes, but with the satellite not implementing any DTN protocol, (and thus not being a node), then this single DTN hop involves two separate links (uplink and downlink).

The "**source**" for a packet is the node that creates that packet and initially forwards it to the next-hop node. Equally, the source is also the host that forwards the packet on the first link. The "**destination**" is the final node on the path, for which the packet is intended. Where we need to discuss packets originated by a non-node host, (e.g. when considering denial-of-service), then we specifically highlight that fact.

When talking generally, we will talk about "**store and forward**" in preference to "**custody**" which is a BP-specific term at this stage – even though the concepts are almost the same. One difference however is that with BP custody, the current custodian should inform the previous one once it accepts a bundle into custody, whereas in general (and in LTP-T) this is not the case. When we talk about "**storage**" we mean non-volatile storage on a node, such that the to-be-forwarded packets will continue to be available for forwarding for longer than the expected disruption of the next hop.

4.2 A Summary in Advance

Since there are many arguments presented below, it may help to summarise most important points in advance. Each of these is discussed and justified in detail later in this chapter.

We will see that handling delay or disruption as close as possible to the delayed or disrupted link is, in general, better. Protocols, or protocol deployments, that deal with such delays or disruptions "further away" will not do as well as those that are closer to the site of delay or disruption. For example, if a packet is dropped due to disruption, the sooner that a DTN node notices the packet loss, the sooner it can retransmit the packet (if retransmission is required). If the re-transmitting node is multiple hops away from the disrupted link, then it will take longer for it to become aware of the packet loss.

In a similar vein, we will see that the "density" of an overlay, in terms of the node-vs-host ratio is significant. As the node-vs-host ratio approaches one, then most distinctions between an overlay network and a DTN transport disappear or at least become less significant. In fact, more precisely, the ratio of interest is between nodes and hosts that are adjacent to disrupted links, though we make no numerical claims about specific values of this ratio (other than "1")[1]. This implies that we can (actually quite easily) construct scenarios involving a relatively sparse overlay with all nodes running over a single transport layer, such that a DTN transport should be preferable for that scenario.

We will also see that store-and-forward overlays that use multi-hop "backwards" signalling between nodes are problematic, for management, security and complexity reasons. A DTN transport involves no such signalling, whereas the BP uses this approach for custody acknowledgements.

However, before getting to the detailed justification for the above, we must define what we consider to be a DTN transport.

4.3 What is a DTN Transport?

In the OSI model [IS94] and the Internet a transport layer sits above the network layer and provides end-to-end connectivity upon which session and/or application layer services can be built. According to these basic models, the session and/or application layer in particular resides on the same host as the transport layer. The defining characteristic of a DTN however, is the general lack of contemporaneous end-to-end connectivity - so one might be justified in being puzzled by the very concept of a DTN transport.

[1] Presumably at some point when the density of the overlay passes some threshold it no longer makes sense to regard the approach as being an overlay. Say if 95% of hosts are running the overlay protocol – is that then really what we are calling a transport here? In any case, we assume overlays are generally relatively sparse in terms of the node-to-host ratio.

Our goal is to be as close to a standard transport layer as is possible. This means that a DTN transport is a protocol that must be able to provide transport services in a “normal,” non-challenged, network and that sits above a network (or lower) layer. In situations characterised by a lack of end-to-end connectivity, a DTN transport must be able to provide something similar to the BP’s custody concept – otherwise the packets will not reach their destination. And provision of anything like custody requires that some intermediate host store packets for some period until the next hop becomes usable.

So, a DTN transport node should, where possible, forward packets as “far as possible” before the next node stores the packets. How far packets can travel before being stored depends essentially upon the routing scheme in force and then upon whatever random/accidental/known delays and disruptions occur as the packets traverse their path to the destination. For many routing schemes, and certainly in the general case, this argues for DTN-transport code to be present on all hosts adjacent to links that may experience delay or disruption. Put another way, the DTN transport approach calls for, though doesn’t strictly require, all hosts adjacent to potentially disrupted links to be DTN nodes.

Figure 4.1 illustrates how a DTN transport differs from a more typical transport. With a normal transport protocol, such as the Transmission Control Protocol (TCP), only the source and destination need to implement the transport layer – all routers in between need only implement the Internet protocol (IP). The bold line in the figure shows the path taken by packets flowing from node 1 to node 15, with one link on the path, (between nodes 10 and 9) broken to indicate delay or disruption. That link, plus the disrupted link between nodes 5 and 9 would be sufficient to prevent traffic flowing in this case since there is no way to re-route around the disruption.

For the DTN transport Figure 4.1 shows that nodes that are adjacent to potentially disrupted links implement the DTN transport and are able to store packets for later forwarding should that be necessary. The result is that, including source and destination, there are 6 DTN-transport nodes shown on the path used in Figure 4.1 – whether each would in fact receive and (store and) forward traffic can depend on the (lower layer) routes that are configured for the various nodes and hosts involved and on the types of delay and disruption experienced as packets flow.

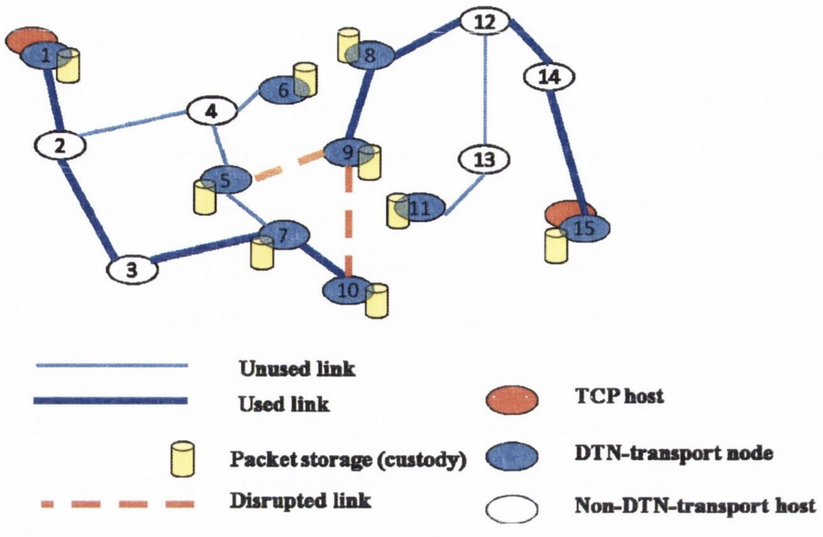


Figure 4.1- Normal and DTN transports

As a consequence of all the above, a DTN transport will have a high level of homogeneity. Since the DTN transport aims to have packets travel as far as possible, every hop above the "thin waist" network layer has to use the same protocol. In contrast, a DTN overlay can use different transport layers at each of its hops.

This homogeneity requires that we select some network layer over which to run our DTN transport, and following the IP hourglass argument [CR99] there is only one, generic, reasonable choice here which is to run over IP. In order to allow for easier implementation however, we also allow layering our DTN transport over UDP (over IP), so that kernel modifications are not required on many platforms.

Since our packets must, in general, be sent via more than one DTN hop in order to handle the disrupted links, we clearly need a way to release storage at each hop once data has been sent. With our DTN transport model, all signalling to handle this release of stored data is handled in a single-hop manner. That is, there is no multi-hop "backwards" signalling involved in the release of stored data. Put another way, DTN transports, as defined here, handle Automatic ReQuest for retransmission (ARQ) [LI84] on a DTN-hop by DTN-hop basis.

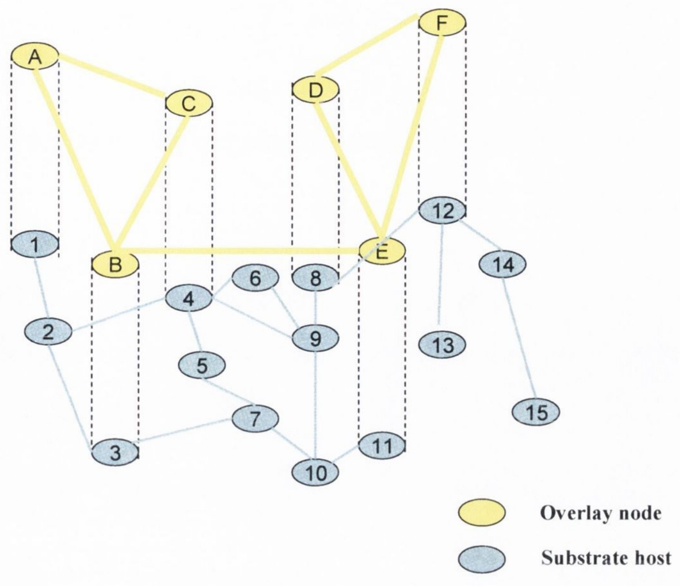


Figure 4.2 - An overlay network

Our definition of a DTN transport is therefore:

A DTN transport protocol is a DTN protocol that runs over IP or UDP, where nodes (preferentially located close to potentially disrupted links) can store packets for subsequent forwarding once the disruption event is over, and where only single-hop signalling is used to implement ARQ.

Having defined what we mean by a DTN transport, we now move on to consideration of when such a protocol will be more beneficial than the other approaches envisaged for handling DTNs.

4.4 Transport vs. Overlay

In this section we compare and contrast the overlay and DTN transport approaches. Note that for the purposes of this section, unless stated otherwise, we restrict ourselves to considering the case where the overlay runs over a single transport. Cases where this is impossible of course favour the overlay approach.

Figure 4.2 depicts a generic overlay network [TO01] where a set of substrate hosts are connected via various links and a subset of those hosts acting as nodes form the overlay network with its DTN hops depicted. In the figure hosts “1”-“15” represent the physical or lower layer view of the network, whereas nodes "A" to "F" represents the overlay network. In the diagram node "A" is in fact exactly the same machine as host "1", "B" is the same as "3" etc. We will use variations of this diagram to represent the different topologies we consider below.

Table 4.1 gives an overview of the factors that differentiate between a DTN overlay approach and the DTN transport layer approach. Essentially these are the specific factors related to the location of functionality in a DTN that we referred to in the problem statement in section 1.4. We examine each factor in detail below.

Factor	Highlights	Argues for
Custody location	Not clear that selective location of custody is beneficial since it may cause significant additional delay if custodians are located far from disrupted links.	Transport (strongly)
Deployment considerations	New overlays are, in general, easier to deploy than transports (particularly web services overlays), however the BP is no easier to deploy than a DTN transport.	Neutral
Topological considerations	Different topologies can favour either.	Neutral
IPN considerations	The IPN, as the poster-child DTN, in fact doesn't favour either approach due to the nature of the specific deployment context.	Neutral
Routing	DTN routing is still such an open topic that it is hard to tell, but overlays being less dense make for easier routing.	Overlay?
Management	Overlay leaves more open and therefore makes network management harder.	Transport
Future proofing	Models like the convergence layer allow much easier support for new experiments.	Overlay
Security	Overlay is more heterogeneous (complexity) and has harder-to-meet AAA[2] requirements.	Transport
Wireless sensor network density	More capable nodes are more likely with less need for switching transport layers.	Transport
End-to-end-ness	A DTN transport is closer to the end-to-end principle.	Transport
Late-binding and DNS	Late-binding is part of the overlay architecture, but not the DTN transport	Overlay

Table 4.1 – Overlay vs. Transport overview.

[2] Authentication, Authorization and Accounting (AAA) is a standard abbreviation used in network security.

4.4.1 Custody Considerations

A DTN overlay network must also do store and forward but in contrast to a DTN transport, with an overlay (like the BP) one nominates the storage points, (in the BP, the custodians), as opposed to requiring that every node be able to store and forward. In either case, it is quite acceptable for a DTN node to be temporarily unable (due to storage congestion) or unwilling (due to policy) to store packets (take custody), but with our DTN transport model every node must at least be *able* to store as well as forward. With the overlay approach, a node implementation with no non-volatile storage could be considered compliant, whereas in the DTN transport case, such an implementation would require a convincing, and unusual, rationale.

The overlay model certainly seems attractive, since it allows a network planner plenty of flexibility to decide where non-volatile storage is distributed amongst DTN nodes, and how DTN nodes are distributed amongst hosts. There is however, a problem with this flexibility. In a DTN-overlay network, packets are stored at a node when some previous node asks for that, and the custodian's storage is released when the next custodian (or final destination) sends some kind of acknowledgement. Furthermore there is no requirement that these nodes are adjacent (i.e., on the same DTN hop), so that multi-hop signalling will be required.

To take an example, Figure 4.3 shows a network with two overlay custodians and one disrupted link. Imagine that packets should nominally flow along the path shown, that is through nodes 4, 2, 3, 7, 10, 9 and then 8, with the link from 10 to 9 being unexpectedly disrupted so that packet forwarding fails there.

In the DTN-overlay, assume node C is the current custodian and is in the process of transferring custody to node D. The problem is that when the 10 to 9 link is disrupted, node C knows nothing about that and so must await a re-transmission timer expiry before retransmitting the packet, which will now traverse the 4, 2, 3, 7, 10 path needlessly.

In the DTN-transport case, where each of the hosts adjacent to a potentially disrupted link is a DTN node, the disruption is handled at node 10, which will re-transmit the relevant packets to node 9 at the next opportunity. In this case, even if node 10 is a storage constrained device, or equally is suffering congestion, then node 10 can, in principle,

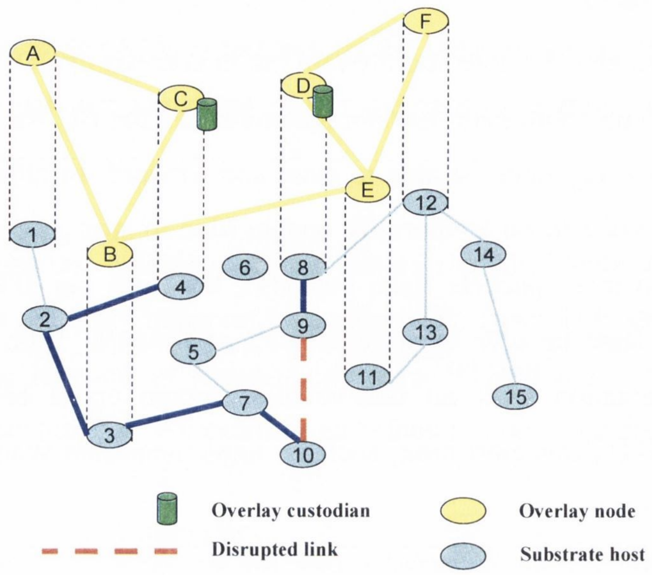


Figure 4.3 –Custody in DTN Overlays

announce this fact to node 7, which might be able to route around the congested node, say having packets follow a path from 7 to 5 and then 9.

In this example, so long as other factors are equal, the DTN-transport approach outperforms the DTN-overlay approach. Again though – this is because the DTN-transport nodes are closer to the problematic link – with a sufficiently dense overlay similar performance will result.

There is also a routing issue to note here. In a DTN overlay, since custody acceptance has to be acknowledged in order to allow storage to be released, this introduces a requirement for custody signals to travel backwards along the path. In addition to creating additional traffic, this also introduces a requirement for a multi-hop backchannel, which sometimes will not be possible to meet. For example, in many ad-hoc scenarios such custody acknowledgements will fail to arrive because the previous custodian will no longer be routable. Since the DTN transport approach requires nodes to always store and forward, there is no need for a new multi-hop backwards acknowledgement and so the difficulty does not arise.

Finally, multi-hop backwards signals introduce a few hard security problems. The first is that nodes must now have a way to determine which other nodes are authorized to request that they accept custody, and such an any-to-any authorization infrastructure is complex. Simpler and more practical alternatives, (e.g., only accepting custody for known nodes), could act to partition a DTN if they make it impossible to get packets past some disrupted link.

The second security problem is that most protocols (including the BP) give the new custodian no hint as to what kind of security services ought to be applied to the backwards signals. For example, if the old custodian were to release the storage on the basis of an unauthenticated custody acknowledgement signal then this creates a simple denial-of-service (DoS) vector. If, on the other hand, the old custodian does require authentication of the signals, (say for accountability reasons), then there is no way for the new custodian to know that this is the case. (However, the BP is not alone in this respect; protocols going all the way back to X.400 [WI91] have had this problem.)

Lastly, even were the DTN overlay protocol's packets to indicate the security requirements for the backwards signalling, (which is itself hard to specify, unless the new custodian is known in advance), this creates an additional key management burden. The reason is that there is a directional aspect to the use of the cryptographic keys that supply the security mechanism to implement whatever service is required. For example, if we require a digital signature on the signals, then the old custodian will probably have to have a copy of a root-certification authority (CA) public key for the new custodian's domain, and perhaps even access to a certificate status checking mechanism, even across the presumably disrupted "divide," and there are currently no simple ways to provide such access. The bottom line here is that even if the new custodian can decide that some security service is required for the backwards signals, provisioning the network so as to make that work is often hard.

The contrast with the DTN transport approach is fairly stark here. With the DTN transport, storage and release of stored packets is always a single-hop issue. And since those two nodes are in any case in contact, and perhaps have other security associations, all of the above is much simpler.

So overall, custody handling fairly strongly favours the DTN-transport approach.

4.4.2 Deployment

One of the main benefits of overlay networks is the ability to deploy such a network on top of an existing substrate – this is one reason why peer-to-peer networks of various kinds and content-delivery networks [BY04] are designed as overlays. Typically the substrate in question is the terrestrial Internet. Deploying a transport on top of IP, or even UDP, [PO80] also requires no change in the underlying network and so, in one sense, is no harder than an overlay.

Since neither approach requires that all hosts be nodes, they are equivalent in terms of not requiring that all hosts are changed in order to allow deployment.

If the transport protocol is run as part of kernel code and not in userland, then it will, however, be more difficult to deploy since modifying kernels must be done more carefully.

Since both the BP and LTP-T are experimental protocols where the user community has little or no experience, whatever deployments occur in the near term are likely to be fairly small-scale and limited, e.g. in terms of how they could affect other flows, so neither approach outscores the other in terms of relative maturity.

Web services type overlays in particular are easier to deploy, due partly to the way in which firewalls have been deployed and configured over the last decade. An overlay based on the BP however, faces the same deployment problems as LTP-T – new holes have to be punched in firewalls in order to allow packets to flow. Even though both the BP (4556) and LTP (1113) have officially assigned port numbers, one would not expect firewall vendors to view opening those as being a high priority.

However, there are currently no web services based DTN overlay proposals, nor DTN overlay proposals based on a convergence layer (CL) that would have existing holes in firewalls, so though this factor could have favoured the overlay approach, with the current overlay proposals, it does not. So our conclusion is that there is, at present, no practical advantage for the overlay approach in terms of deployment considerations.

4.4.3 Topology

We next examine whether and how a network's topology favours a DTN overlay or a DTN transport approach. It is worth starting with a note of caution here - in the following we must bear in mind that network topology changes over time, and, particularly with a DTN, those changes may be more significant than the structure of any snapshot of the network topology. Conclusions drawn from static views of network topology may give a misleading picture.

Topology type	Favours
Internet-in-the-middle	Overlay (very slightly)
Long-fat-networks	Transport
Data-mules	Transport (very very slightly)
Deep-space	Overlay (very very slightly)
Random badness	Transport
DTN-DMZ	Overlay (for now)

Table 4.2 – Different DTN topologies favour different solutions.

We now consider various representative topologies to see if any of them favour the overlay or transport approaches. Table 4.2 lists the topologies considered and our conclusions as to which approach each favours. The topologies listed are the set of (in the author's opinion) distinct DTN topologies for which good DTN reference examples exist. As can be seen from the table, overall, we believe that the network topology is relatively neutral between the overlay and transport approaches.

Internet-in-the-middle Topology

The Internet-in-the-middle topology is one where the Internet is seen as a set of sources and sinks for flows that are targeted to/from sets of challenged, (e.g. sensor/actuator), networks that are connected to the Internet via one or more gateways. For example, some military tactical networks are designed in this manner [SC05, RI05]. A more common example might be a set of environmental monitoring nodes like the Sensor Networking with Delay Tolerance (SeNDT) nodes [FA06i] connected to a data sink on the Internet.

Generally with this topology, there is little benefit in having a DTN node that is in the interior of a well-connected part of the network, (i.e., where all other nodes that are only one hop away are also well-connected). One would instead expect that the DTN nodes would be at the edge of the well-connected Internet and not in the middle. (With perhaps a few well connected nodes acting as data sink or control sources.) This applies to both overlays and transports.

If a challenged network is using some unusual lower layer (e.g. using data-fusion [IN00] or network coding [WI05]), then clearly the transport approach will be problematic since, by definition, the transport only runs over IP. However, there are sensor networks (like SeNDT) where the nodes are not extremely resource challenged other than in terms of power and contact opportunities, and so can make use of standard Internet protocols while in contact. In such cases, a DTN transport can make excellent sense.

Overall though the Internet-in-the-middle topology very slightly favours the overlay approach on the basis that it more easily allows for the use of non-standard lower-layers in the challenged parts of the network.

Long-fat-network Topology

The archetypal long-fat-network (LFN) is an otherwise standard Internet flow that involves a geosynchronous Earth orbit (GEO) satellite [MO00, GL98]. Such a (pair of) link(s) involves a distance of approximately 35,786 km up and back with each leg involving an additional LTT (compared to a normal IP flow) of ~120ms in either direction, thus adding a total of a little less than one quarter of a second to all flows involving these links. In fact, with the addition of queuing time in the spacecraft, the RTTs are even worse than this, being typically of the order of 700ms.[12]

Regardless of how the LFN is handled it remains the case that the overlay approach requires there to be some CL for these links that is also DTN capable. Without such a layer, it would not be possible to operate the LFN, since LFNs typically have two delayed or disrupted links (the uplink and downlink). So arguably the most natural thing to do is to implement a CL that is actually a transport protocol[13]. Doing so allows the satellite itself to be addressed and so also allows other scenarios (e.g. constellations of LEO satellites) to be

[12] http://bnrg.eecs.berkeley.edu/~randy/Daedalus/BARWAN/DBS.html
[13] In fact SCPS-TP is frequently used as this "inter-PEP" protocol (see Chapter 2).

supported more easily. At that point (when a new transport is defined to support the satellite hop), one would begin to question the benefits of the overlay since all delay and disruption is being handled in the DTN transport which could probably relatively easily[14] be extended to run end-to-end.

Summarising, LFNs demonstrate few benefits for overlays which seem to require a DTN transport as a CL. So overall, we see the DTN transport layer approach being preferable to the DTN overlay here.

Data Mule Topology

The data mule [SH03] approach to DTNs involves one or more so-called data mules, (sometimes known as "message ferries" [ZH04]), that physically move amongst other network nodes establishing connections to each node according to a schedule; or whenever proximity allows, or following some other discipline. In many scenarios, the data mule also occasionally contacts a node, (here called a base station, but the term is irrelevant), that is well-connected to the Internet, and thus the data mule provides a moving bridge between the Internet and the isolated nodes.

Since the data mule itself will generally experience highly disrupted communications as it enters into, and looses, contact with other hosts, the data mule itself really has to be a DTN node. So in this topology we have a wandering DTN node that connects to other hosts. The next question is whether those mule-to-host connections really have to use a DTN protocol? In fact, in many data mule scenarios, one could use TCP between the mule and the other node so long as the contact duration is sufficient.

So one could structure a data mule driven DTN to use a home-grown application layer protocol (e.g. achieving end-to-end connectivity via a sequence of FTP connections), or one could use either a DTN overlay or a DTN transport. This level of flexibility, and the

[14] "Easily" here is comparing the DTN overlay and DTN transport approaches – deploying either to the endpoints is hard but has been discussed already.

fact that there appear to be many applications[15] [BR04, FA06i, ZH04] that already provide "natural" data mules makes this topology attractive for many scenarios.[16]

However, the data-mule topology doesn't really help decide between a DTN overlay and a DTN transport. The fact that all, or almost all, nodes are using the DTN protocol makes the overlay and transport very similar. However, other things being equal, there is no clear benefit to using different transport layers for one data mule, at which point the overlay protocol only represents additional overhead, so we consider this topology to very slightly favour the DTN transport approach.

Deep Space Network Topology

Figure 4.4 shows an example of the type of network topology one might encounter on a deep-space mission [BU03]. Essentially there are five types of node that can be involved each with different properties as described below[17]:

Landers are on-planet nodes that suffer extreme power challenges, have very (frequent and long duration) disrupted communications with all peers and extreme delays back to Earth (from 4 to 20 minutes in the case of Mars). Landers can communicate direct-to-Earth at low bandwidths and by consuming quite an amount of hard-to-generate power. A Mars lander will generally be visible from somewhere on Earth for roughly half the time, however, the lander will not have power for such extended data transmission, and, in fact, the DSN Earthstations (see below) will be busy doing other things most of the time. The result is that the practical availability of the lander for direct-to-Earth communications is highly constrained. In order to save power and increase the scientific data return landers

[15] As already described, lake water quality and noise monitoring, bus-based networks like DieselNet [ZH06d] and Zebranet [ZH04z].
[16] In fact, there is a partly separate research community working on what they call "opportunistic networking" dealing specifically with networks that make use of existing physical entities as network nodes. Arguably, this is a subset of the problems addressed by DTN, but of course, an opportunistic networking researcher might justifiably argue the opposite!
[17] While this description is overly detailed for this part of the argument, we will be using an emulation of a setup like this in our evaluation so the detail is useful for understanding that section.

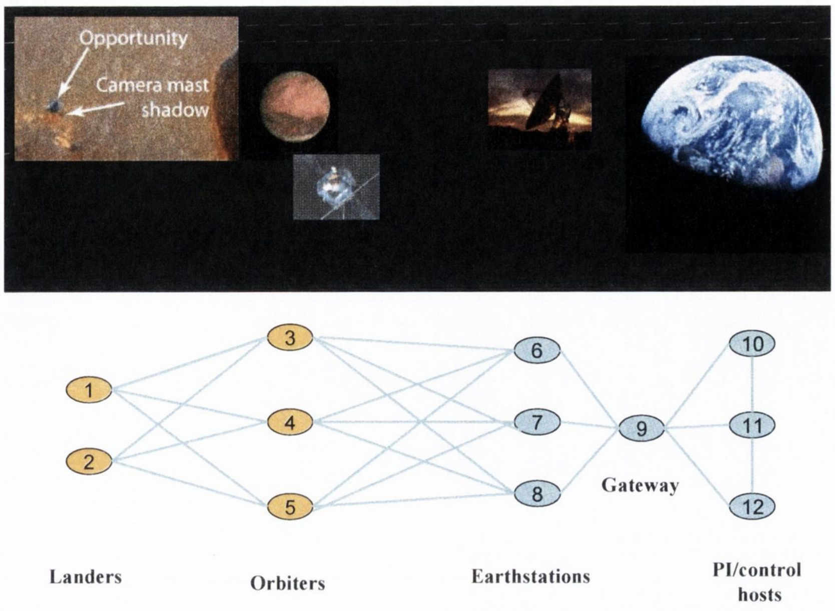


Figure 4.4 – A deep-space network topology; images courtesy of NASA and ESA.

will, where possible use orbiters, as data relays; such orbiter passes will only last minutes but can use much higher data rate communications.

Orbiters as the name implies are in orbit around the target planet and perform as science instruments in their own right as well as (and even sometimes primarily) as data relays for landers. Typical Mars orbiters will tend to have roughly 90 minute orbits during perhaps 2/3rds of which the orbiter will be visible from Earth.

The **Deep Space Network (DSN)**[18] is a NASA network of Earthstations spread around the globe so that the DSN can, in principle, offer coverage to any deep space mission. The DSN is however almost always oversubscribed; so that a fairly complex scheduling system is used to allocate contacts during which a particular mission is allocated use of a DSN antenna. The actual DSN has an IP based network of its own for managing antennae and

[18] http://deepspace.jpl.nasa.gov/dsn/ There is also an emerging European equivalent called ESTRACK but as yet all deep-space missions have required NASA DSN support.

distributing science data. While this network is itself complex (mainly due to the need to capture as much of the physical layer information[19] as possible for later analysis), for our purposes it is sufficient to imagine there being one or more DSN **gateway nodes** between the DSN and the Internet.

Finally, there will be a number of **principal investigator** (PI) hosts and **control hosts** scattered around the Internet that are authorized to take part in the deep-space mission network for various purposes that are indistinguishable from our point-of-view.

Before contrasting the overlay and transport layer approaches in this context we should also very quickly note some legacy protocol issues that arise in this environment. The main point is that (from an Internet point of view) atypical lower layer protocols are common in deep-space networking – Forward Error Correction (FEC)[20] [HO01] is considered well worth the effort and even corrupted bit streams can have a value.[21]

So, how do the overlay and transport approaches compare here? First, even the term overlay is somewhat odd in this context. There are, and for the foreseeable future will be, so few hosts involved in deep-space missions that essentially every space-side host will take part in any overlay[23]. Given that there are, in this case, likely to be very few Internet connected hosts that are (or should be) authorized to be part of this application; we end up with an overlay here that in fact includes most every host involved, exactly as we would were we to apply the DTN transport approach to this topology.

[19] The DSN is also used for space science – in which case there is (generally) only physical layer information recorded.
[20] The use of FEC means that links are either perfect, or else totally broken (when FEC fails), and also that link throughput may depend on the level of FEC applied.
[21] As indeed do retained analog measurements. In many cases the DSN is part of the science effort as well. For example analysis of the changes in radio frequency propagation as a spacecraft is occulted can provide information about planetary atmospheres. Another example would be a partial image which can still have science value.
[23] For those few mission proposals that involve significant numbers of nodes (e.g. NetLander) the probability is that the less capable nodes should be more properly treated like spacecraft instruments and not as if they were independent spacecraft.

However, the heterogeneous lower layers seen here do make it more likely that the overlay approach is more suited for deep-space networking.

Randomly Bad Topology

We now move to consider a topology where essentially every link has some non-negligible probability of being disrupted (for simplicity, we will ignore delay in this part of the argument). In this case, we assume some standard network topology is being used and would like to consider whether or not an overlay differs from a transport when link disruptions happen at random.

This is clearly reminiscent of the type of argument about the Internet and the web being examples of scale-free networks [AL00] which have excellent resilience in the face of random failures. It is unclear whether or not there will be any difference between a DTN transport and an overlay in these terms. Will the fact that a DTN transport is denser make it less likely to resemble a scale-free network? Or, will the relative sparsity of an overlay make it less like a scale-free network? Those are questions that will become interesting as DTN deployments grow, but for the present are not resolvable.

One indicator may be that one peer-to-peer network - Gnutella [RI02] – has been found to create traffic patterns that conflict to some extent with how the IP network that is its substrate is typically provisioned. For example, Gnutella creates lots of additional, and perhaps unexpected, cross Autonomous System (AS) traffic. Since deployed peer-to-peer topologies are inherently hard to investigate[24] this may or may not be a significant point, but it at least indicates the potential for what is, in this case, a scale-free overlay on top of a scale-free network leading to possibly suboptimal routing.

While one could probably build a mathematical model differentiating between overlays and transports for each of the standard network topologies, for the purposes of this work that is unnecessary. In order to see this, we note that a DTN transport approach, being

---

[24] Since they have few, if any, central points where statistics can be gathered.

inherently "closer" to disrupted links, can react more quickly to disruptions and with better knowledge of the actual state of affairs. The reason for this is simple: in an overlay, there may be disrupted links between hosts that are not part of the overlay, and in that case, detecting and recovering from the disruption must take longer than in a DTN transport that involves "more" participating nodes that are closer to the disrupted links. As we saw above when discussing custody (in Section 4.4.1), distance from such disrupted links is problematic.

So random disruptions essentially favour the lower layer protocol, which in this case is the DTN transport.

DTN-DMZ Topology

In this section, we discuss how the two DTN approaches fare in the light of the fact that many current enterprise networks isolate themselves from the less-trusted Internet via a so-called De-Militarized Zone (DMZ) [SH03d] in which various security, privacy and performance-enhancing services tend to be located. For example, firewalls control which flows are allowed in or out and perhaps manage Network Address Translation (NAT) as well; scanners may examine flows and email traffic for dubious patterns; split-DNS servers distinguish how name/address mappings are seen on the "inside" and "outside" and load balancers may hide multiple nodes behind what appears to be a single service interface in order to meet busy-hour requirements.

Many of these services are widely considered to breach the end-to-end principle, [CA02] (as do DTNs[26]). They are however extremely widely deployed, and used, on today's Internet. It is therefore interesting to consider how or whether our two DTN approaches differ in respect of how they address, subvert or otherwise affect these kinds of DMZ services.

[26] See Section 4.4.9 below.

We would expect that wanting to have DMZ-like services in a DTN would favour the overlay approach because DMZ services depend on packets being funnelled towards special hosts (that provide DMZ services) which is quite reminiscent of an overlay. And in fact, the bundle security protocol (BSP) specification [SY07] is in some way tailored towards this view. While the same services can be implemented with a DTN-transport (since they are largely not visible to the endpoints), an overlay is still a more natural fit since a DMZ itself is a kind of sub-network overlay. This is analogous to the way in which the HTTP protocol is more DMZ "friendly" than end-to-end protocols like IMAP.

Having said that however, the DMZ model for enterprise networks may be beginning to break down, for example, because it doesn't necessarily suit when an enterprise deploys an internal wireless LAN – at that point many access points are physically attached to the "internal" network but clearly allow connection establishment attempts from less-trusted environs (for example, the pub across the road). In addition, the fact that many hosts are now mobile (and possibly connected to untrusted networks at home) also means that the inside/outside distinction is much more blurred than used to be the case. So there is already a trend away from the DMZ model and towards boundary-less security, in which case presumably being able to embed DMZ-like services into any DTN-transport node would become the more natural fit. Whether or not this so-called "de-perimeterisation" [HA06, BL05] will become widespread however, is not at all clear at this stage.

However, since the DMZ model is in widespread use and is closely analogous to the overlay approach, this factor favours the overlay approach.

Concluding on Topology

We have considered a range of DTN topologies, and found some that favour each of the approaches. In the end our analysis has essentially produced an overall neutral conclusion. This is both because there seems little to choose between the two approaches, but also because of the caveat noted at the start of this section – for some DTNs there may be no fixed topology at all, which further re-enforces our overall neutral stance when considering how network topology affects the DTN overlay and transport approaches.

4.4.4 The IPN

The fact that the DTNRG have mainly focused on the overlay approach represented by the BP is, in part, due to the fact that this work had its origin in the InterPlanetary Network (IPN). [BU02] In the original IPN, networks on, and around, each planet were termed "regions" - and the terrestrial Internet was treated as simply one more region. From that point of view an overlay approach was eminently sensible, since each region was assumed to be autonomous, including possibly running different lower-layer protocols. So it is no wonder that the BP, which started out as an IPN protocol, should take the overlay approach.

However, given the evolution from IPN to DTN described earlier, one could ask whether that evolution indicates more than just the problem of solar-system scale experiments being inordinately expensive, or are there some networking reasons why we're better off with a DTN rather than an IPN and if so, does that tell us anything about the overlay versus transport approaches?

Firstly, the IPN was originally spoken about as being the one and only overlay network covering the solar system. If we assumed that, then we would no longer be discussing the pros and cons of different networking approaches, but of two different architectures for one network, which is a different discussion. So, we will treat the original IPN as being one amongst many solar-system scale networks for the remainder of the argument[27].

The question then is whether an overlay is a good or a bad way to create one among many (possibly interconnecting) solar-system scale networks? Since the main benefit from an overlay is that it allows for the use of heterogeneous transport protocols in different parts of the communications path from source to destination, this is clearly a plus in the IPN context.

---

[27] In fact, the original IPN concept was largely the product of one space agency (NASA). Were numerous different space agencies involved it is not at all clear that there would still be a single network.

The late-binding [FA03] feature that the overlay naturally provides is also demonstrated at its clearest in this context. If a Mars-based node uses a name of some form for a destination on Earth, then it would seem to be wise not to depend on an up-to-date name/address mapping (for Earth) being accessible to the source on Mars at the time of original transmission of a message. This is a good argument, but of course, only for a network of a certain scale – the deep space networks envisaged for the next two decades will not really benefit from late-binding since there will not be a sufficient number of nodes involved either on Earth or Mars for name mapping to become a real problem.

One exception to this could be a cislunar network, where one could envisage a large scale network being built in the next few decades. However, the latencies involved here are not really problematic for management traffic. So again, late-binding doesn't really seem to be that significant for cislunar networks.

It would appear from the above that the late-binding principle is actually not very beneficial for the overlay approach when considered in the context of a realistic IPN being deployed at some time in the next two decades. Our conclusion is the IPN view of overlay vs. transport is actually neutral rather than favouring the overlay as one might expect.

4.4.5 Routing

Routing is perhaps the main open research topic in DTNs – we simply do not know at this time how best to handling routing issues in DTNs in general. [JA04] As a result, any conclusions to be drawn from arguments offered in this section must certainly be considered extremely tentative.

One could make a good argument that no DTN protocol ought hard-code one (or more) routing scheme(s), on the basis that a DTN protocol should be usable in as broad a range of environments as possible, and that the range of environments is so broad that no one-size-fits-all routing scheme will work everywhere.

For example, the mostly deterministic deep-space networking use case differs hugely from cases where opportunistic contacts are the norm.

Since the overlay approach cannot really constrain routing very well[28], such approaches must inherently be less performant in some situations. There will always be some networks where additional knowledge of the flows would allow better routes to be derived; in fact Jain et al. [JA04] show that some schemes of theirs that make use of oracles perform better than those that don't. On the other hand, the same paper describes such oracles as perhaps being more trouble than they were worth!

The density factor also works in favour of the overlay here to an extent – with fewer nodes for which routing has to be managed, the routing problems become easier. However, the counter argument is that someone has to manage the lower layer routes in any case and if some of those involve delay or disruption prone links, then we have not solved the routing problem simply by distributing routes to overlay nodes. We must also provide some lower layer hosts with the information necessary to handle the DTN links to which they are adjacent (assuming such links are in the "middle" of paths between overlay nodes).

The only current publicly-specified[29] routing scheme for the DTN overlay is PRoPHET [LI06] which is a probabilistic routing scheme where forwarders attempt to increase the probability of delivery by forwarding each bundle to a group of nodes that have previously been successful next-hops. Clearly, whatever CL is in use will also affect routing however, so PRoPHET alone doesn't give a full picture, but is intended to benefit from whatever "best effort" the lower layers can manage, e.g. there has been some consideration of how PRoPHET and MANETs might interact [OT06]. For DTN transports, we only have the LTP-T protocol where routing is a mixture of IP routing and static tables (with wildcarding) that are used at each LTP-T node. So while we do have some examples of DTNs with defined routing, in no case do we have any real deployment experience at a scale that is useful.

---

[28] Since the overlay DTN cannot in general make end-to-end assumptions about e.g. performance (due to the diverse CLs involved) it equally cannot be very deterministic about routes.
[29] "Specified" here is meant in the sense of having a (relatively) stable, published specification that different implementers could code from – and get the same result. In the case of PRoPHET, this took the form of an Internet-Draft.

So for routing we currently prefer the overlay approach, based on the fact that it may lead to simpler solutions if the overlay is sparse, though recognising that this is an extremely tentative conclusion.

4.4.6 Future Proofing

As DTN matures, various new and variant protocols will emerge at different layers attempting to better handle aspects and causes of delay and disruption. An overlay protocol like the BP is more robust when faced with such changes. For example, it may well be that there will be a period where many DTN-transports (e.g. LTP-T and others) and/or DTN lower-layers (e.g. USB Sneakernet[30]), [QU03] are defined that each form the basis of separate experiments. In that scenario, an overlay protocol can remain the focus of interoperability, so long as there is some good model for integrating new lower layers into the overlay. For example, the BP implementation does this with its CL concept[32] which is robust enough to handle TCP, UDP, raw Ethernet and various other layers right down to Sneakernet.

There is however, an argument to the contrary which is basically that the hourglass model[33] encourages us to make IP the thin waist of the hourglass [CR99]. In a sense the CL model for the BP and (potentially) other overlays encourages the development of a fatter (or split) waist for the hourglass. To the extent that one believes the hourglass argument is telling, one may consider that layering DTN-transports directly on IP (or perhaps on UDP) provides a safer basis on which to proceed and will therefore be better future-proofed, e.g. as new IP-over-foo combinations are designed, and more importantly, actually get used.[34]

Of course, the first argument above only applies when more than one CL is in use. For DTN use cases that use a single CL, the overlay and transport are equally future-proofed.

---

[30] http://doi.acm.org/10.1145/864056.864078

[32] A downside of this is that before the overlay can interoperate, each party has to define/select and implement some common CL. With a DTN-transport, it is more likely that the transport protocol specifies those details natively.

[33] http://www.cs.virginia.edu/~cs757/slidespdf/deering-hourglass-london-ietf.pdf

[34] Many IP-over-foo options are in widespread use today, while only a small number of BP/foo options have yet been implemented in the BP reference implementation.

However, at present, we would have to favour the overlay approach given the flexibility inherent in the CL model.

4.4.7 Management

From the network management point of view, an overlay network must be inherently more complex to manage since each lower layer technology requires specific knowledge and imposes its own operational burden. In many real networks, this knowledge problem is the most critical [CU05] since each new technology requires operator training and the need to deal with diverse technologies makes handling operator staff turnover more problematic. Overlays will also be harder to monitor, diagnose and debug since the administrator will almost certainly have less knowledge about the current state of the nodes themselves, given that they are connected by a potentially large range of technologies. Managing homogeneous networks is simpler.

With a DTN-transport that has a well-defined binding to its lower layers, many of the other management issues are ameliorated – so long as the DTN-transport does, in fact, make specific choices about lower layer binding that is – were some putative DTN-transport to support a range of network layers, then we would be in almost the same position as with an overlay. In particular a DTN-transport that binds to IP or UDP can leverage a lot of existing management know-how and tools.

Considering the management of overlays a little more, there has been at least one attempt to rein in the complexity of managing overlays, namely the X-Bone, [TO01, TO05] which aims to be a generic management layer for any overlay network on top of IP. However, in the case of a DTN overlay, one of the main justifications of the overlay approach is the potential for using non-IP lower layers, in which case the X-Bone would not be directly usable, at least without additional effort.

The X-Bone also provides a nice example of a generic difficulty with managing DTNs, that is, node or service discovery. The X-Bone uses techniques like IP multicast in order to discover hosts that are to be managed.

Good handling of node discovery is a crucial feature for any network management scheme [BR04d]. Unfortunately, a DTN, (whether served by an overlay or transport protocol), inherently makes discovery difficult since the discovery process will probably be significantly hindered by timeouts due to delayed and/or disrupted links. The net result here is that a system like the X-Bone is not easily able to manage a DTN, even though the X-Bone was designed specifically for overlay management.

One could presumably extend the X-Bone[35] and whatever DTN protocol one uses so that DTN nodes would notify some nearby X-Bone resource daemon or overlay manager component of the existence (and perhaps "connectedness") of other DTN nodes – this could for example be done using a collection of values of the "via" extensions in LTP-T,[36] or reporting information from the BP. So, while the X-Bone is not currently suited for DTN node discovery, one can envisage a way to solve that problem, in a way that doesn't really distinguish between the overlay and transport approaches. In the end, the more homogeneous nature of the transport approach means that we strongly prefer it when considering network management.

4.4.8 Security

From the cryptographic-based security service point of view overlays are somewhat difficult, since, in general, an end-to-end security analysis will have to consider combinations of lower layer, CL, and DTN-overlay security. With the current Internet it is often hard to know when a flow is secured using IPSec, [AU05] and with an overlay the difficulties are increased with each new security option added. Achieving an end-to-end picture of security from an overlay is inherently harder.

To give an example, the BP security specification (BSP) defines a way for a node to digitally sign a bundle, for example, in order to provide data origin authentication. Many people however envisage DTNs with sensor nodes that are so challenged in terms of CPU

[35] There would undoubtedly be additional changes required, e.g. the X-Bone makes use of SSL, which uses TCP, which won't work in some DTN contexts.
[36] See Chapter 5.

and power that signing is not possible on the sensor nodes themselves. For this reason, the BSP defines a way for an intermediate node to digitally sign the bundle, the main intent being to allow the application of CPU-intensive security functionality at the first node on the path that is not computationally challenged. The result is that due to the nature of the overlay, it is less predictable as to what entity has signed a bundle, which could lead to a node (or more likely a designer/developer), making a false assumption that the sensor data arrives from the sensor with integrity assured. This is just one of a number of examples of ways in which the overlay approach causes increased complexity from a security point of view.

With a DTN transport, since there is at least one degree of freedom less, the end-to-end security analysis is easier, even if it remains complex if one allows consideration of lower-layer security, as one should. Basically, we no longer have to consider CL security features and are less likely to implement security (or other) features in the "middle" of the DTN.

The absence of DTN key management schemes means however that end-to-end confidentiality in particular is problematic for both approaches. With the BSP, we have definitions of how to apply end-to-end confidentiality though so far without any key management specification. For LTP-T, we have yet to define any confidentiality service, though would plan to provide similar functionality to the BSP, once some DTN key management specification exists.[37] Both the BSP and LTP-T have (relatively) well-defined hop-by-hop and end-to-end data integrity services. So, for cryptographic security services, the transport is preferred since it is simpler.

The area of authentication, authorization and accounting (AAA) [HA02] is one that has been, and continues to be, problematic on the Internet [FA00, TS05]. AAA might also be expected to be similarly problematic for DTNs when/if they move beyond experiments and become something for which service is purchased or for which service consumption has to

[37] Hopefully, whatever key management is done for the BSP can be re-used for LTP-T, though probably requiring a different encoding scheme.

be accounted. We have also yet to see any serious consideration of AAA for DTNs so much of what is below is speculative, though defensible.

With a DTN transport, where custody is not a choice but a duty, many AAA difficulties simply go away, or are handled via ingress controls that might be built in to a firewall node. Since each DTN transport node must accept custody for each packet that it receives, there is only one decision to be made – whether or not to accept a packet from the adjacent node. A DTN overlay has two different decisions to make, whether to accept the packet and separately whether to accept custody of the packet. The former decision is the same as in the DTN transport case, but the latter decision may involve the previous custodian which will generally not be an adjacent DTN node. Making decisions about potentially any DTN node (the previous custodian) is inherently harder since the decision point may have no relationship with the node making the decision.

Moreover, with an overlay where custody[38] is selective, and where off-path signalling (of custody or reporting) is allowed, the protocol must contain flags and values to indicate the selections, which means that external hosts can more easily target a custodian with e.g. DoS attempts. Such signalling also presents a challenging authorization problem since one has to decide whether or not to forward packets (bundles in this case) that are only loosely associated with in-band traffic, which is surely a harder decision to make (or manage).

Current AAA protocols, in the main, could fairly easily be extended [RI00e] to supply the kinds of new information that a DTN AAA infrastructure would require. So long as the administrative information changes sufficiently slowly then standard solutions like RADIUS [RI00], Diameter [CA03] and SAML [GR03] could be used. However, these would not necessarily be typical deployments of those protocols since, e.g. the Policy Enforcement Point (PEP) and Policy Decision Point (PDP) might both have to be on the

[38] A similar point could be made here about the reporting scheme in the BP, where the packet inspector would have to decide whether to allow in a packet that calls for reports to go outside its domain – not at all an easy decision to make well.

same side of a set of disruption-prone links. There would be work to be done to ensure that those protocols were practically usable for an overlay.

With a DTN transport, since all our signalling is in-band and on-path, AAA problems are generally less severe. The overlay approach is also more vulnerable in that complexity is the enemy of security, so that the ability to mix and match different lower layers will inevitably create new vulnerabilities, simply due to the combinatorics of the situation making analysis harder.

In summary then security considerations favour the transport approach given the reduced complexity and the fact that the AAA requirements flowing from a transport layer are simpler than those associated with an overlay.

4.4.9 The End-to-End Argument

We have already stated a number of times that the DTN transport approach is preferred due to the fact that it involves no off-path signalling. The fact that this has cropped up a number of times indicates that there is perhaps really a more generic argument that can be made. Essentially, that is the end-to-end design principle, [SA84] which basically says that functionality should be implemented in the application layer, where there is no clear benefit from implementing the same functionality inside the network. This principle is widely credited with being highly important in the success of the Internet versus the at-one-time favoured Intelligent Networking (IN) or Open-Systems Interconnection (OSI) approaches.

One consequence of the end-to-end argument is that it is easier to introduce new services if those don't require changes to the middle of the network, so, for example, with the Internet, one can become the main search engine without having to ask permission of any ISP.

Both the DTN overlay and transport approaches fairly clearly fail to honor the spirit of the end-to-end argument since they are inherently dealing with passing traffic where there is no contemporaneous end-to-end connection. However, the transport approach is also fairly clearly more in tune with the end-to-end argument, since it at least keeps functionality on-path and minimizes the number of new functions and entities that are present in the middle of the path from source to destination.

4.4.10 Late Binding and the Domain Name System

Any scheme that touches the Internet but is simple to deploy from the perspective of the domain name system (DNS) will generally be able to spread much more easily. This doesn't simply mean being able to use the DNS protocol to map names to addresses, but (with the current state of the DNS) also implies not requiring any new resource record (RR) types, and effectively requires use of RRs to be limited to certain types [AL94]. Basically, if a DTN protocol (whether overlay or transport) requires something new or unusual from the DNS then its deployment will be problematic.

Equally, if a DTN protocol ignores the DNS and attempts to replicate some DNS data, then its deployment may also be problematic since this leads to the DNS and local version of the data eventually differing. Once one approaches the scale of some of the larger DNS zones then such data replication is highly problematic.

The above are also issues when considering handling the late-binding approach[40] of DTN overlays [FA03]. If the destination node name is looked up, (e.g. for routing purposes), at an intermediate node, then the late-binding advantage has been given up if the protocol involved in doing the look up is liable to be disrupted. For example, if a DNS lookup is used from a node that is liable to be disconnected from the Internet, then we will lose the advantage of late binding. In particular, LTP-T can involve such lookups and so does not benefit from late binding.

So as long as a DTN does not involve too many names, then late-binding is preferential. However, presumably this also creates a (thus far unmet) need for a name resolution

[40] In this context late-binding is where each router along a path can separately resolve the destination address, so as a result the source doesn't need to be able to resolve the destination address from the destination name, it only needs to know which forwarding interface to use.

service that is usable from the middle of the network, which is not a trivial matter. Even so though, the late binding feature favours the overlay.

4.4.11 Summary of Overlay vs. Transport

In the above we examined a wide range of overlay vs. transport issues and were we to simply aggregate our conclusions from Table 4.1 those would be: overlay-favoured (3), transport-favoured (5) and neutral (3). This provides a credible argument that the transport approach is at least tenable and deserves detailed investigation when compared against the currently mainstream overlay approach.

4.5 Design Pragmatism

In addition to the above there are also a few additional, more pragmatic factors worth mentioning. Some of these issues really have more to do with protocol implementations, but such pragmatic issues also need consideration when developing a design for a DTN protocol and specifically for a DTN transport.

4.5.1 The Sockets API

Firstly, many developers are currently familiar with programming applications on top of a transport layer via the sockets API [GI99], so the ability to use such an API is pragmatically quite useful. While developers are also beginning to become more familiar with developing applications based on overlays (mainly in a web services/web2.0[41] context), at this point the sockets API is still the more widely understood.

In particular, most applications that run on top of sockets could be relatively easily ported to use a DTN transport. While they might still require application layer changes to handle operating in a connectionless and highly-delayed manner those will be easier if the sockets interface is maintained.

[41] http://www.ariadne.ac.uk/issue45/miller/

Actually doing this is relatively simple, especially if the application ran over UDP sockets previously (i.e., calls to `sendto()` rather than `send()`). In fact the BP reference implementation could also provide a socket-like API so the differences here needn't be huge, but developers are more used to a single library against which they link their applications and with the option to statically link or else to only distribute a shared object library or DLL with the application.

The BP reference code currently has more onerous requirements than this – whether this is an inherent aspect of the BP reference implementation[42] is not clear. Use of separate processes and a database for bundle storage is a fairly natural approach to take for an overlay but is not really what application developers would expect from a communications stack.

The fact that there is a well-known transport layer API that is known by many developers and that overlay development is less mature means we consider this to favour the DTN transport approach.

4.5.2 Always-On Operation

A DTN protocol implementation that insists upon the use of complex configuration and/or routing data before traffic flows will be much harder to use than one that offers an "always-on" mode of operation when no configuration is present. While this is perhaps largely an implementation issue, it can make a significant difference in terms of acceptance, and might perhaps constrain protocol design to some extent. For example a protocol that requires more than one node to be available before application data can flow is harder to experiment with[43] – and experimenting is how most developers begin to work with new things. As a special case of this, the protocol should also be able to work in a loopback mode so that it can be used even on a single host.

---

[42] http://www.dtnrg.org/wiki/Code

[43] An erasure coding protocol would be one such – there is no benefit in using such a protocol in a 1:1 situation.

This slightly favours the DTN transport again, on the basis that it is easier to know which always on configuration to use as the default since there are fewer degrees of freedom.

4.5.3 Real-world Wireless Sensor Network Density

While this section partly repeats some earlier arguments, the relative importance of wireless sensor networks for DTNs justifies the overlap. In addition since the argument here, if correct, may be significant outside the context of DTNs, it is worth having a separate section.

The basic argument is that when one considers the reality, as opposed to the theory, of sensor networking, one sees far less need for the use of different, or unusual, transport layers. That in turn, means that there is less need to develop an overlay network protocol to handle DTN requirements and that a DTN transport that works well with current Internet transports is an option that warrants examination.

We already saw (in Chapter 2) that there is a relationship between conserving power and radio range with realistic sensor networks favouring devices with larger radio ranges. Increased radio range will also commonly involve external antennae for better signal quality. Such antennae are not miniature, but tend to be of the order of 6 cm long or more depending on power available and the desired radio range[44].

Similarly, realistic wireless sensor networks will favour devices that are easy to find, when the time comes to re-deploy (or harvest, or re-configure) nodes. Many proposals [CE04, CO01] assume that it is beneficial to have very many tiny devices placed in a measurement field without ever considering that those devices, if left for too long, become very effective pollutants (batteries are not environmentally friendly [RY03]).

---

[44] http://www.tonystrains.com/technews/nce-hg-antenna-test.htm

Lastly, nodes that themselves harvest and store power will be much more practical to use than those that can only use batteries or fuel cells – the robustness offered by being able to recover from low-power situations is a significant benefit for real systems.

The net effect of all these points is to favour larger, more capable nodes for real-world sensor networks, which runs counter to many of the current approaches being taken to sensor networking which concentrate, to an extent, on miniaturisation and dense distributions of devices each (supposedly) cheap, with very short-range radios (O(10m)) [AK02s].

We can also see a few economic problems with the dense sensor network approach. Firstly, unit costs, even if reduced by an order of magnitude, i.e. from O(€50) to O(€1), are not nearly cheap enough for the densities involved, for any widespread application. Even €1/node with a 10m range requires O(€1million) to cover a field of 400 sq km – for some applications that is not a very large area at all – roughly the size of the island of Saint Helena[45].

And that of course ignores the deployment and retrieval costs which will likely be an order or magnitude more than the presumed O(€1) equipment cost. In fact, it is probable that there will be a balance between deployment and retrieval costs – cheaper deployment (scattering) will tend to mean more expensive retrieval (searching) and vice versa. An estimate of €10 cost/node incl. deployment and retrieval would mean our €1million limits us to roughly 4 sq km - the size of Helgoland which is the smallest island in Schleswig-Holstein[46].

So our conclusion here is that practical wireless sensor networks will tend to involve more capable nodes that are therefore likely to be able to support standard IP stacks. This, in turn, means that networks of such nodes are less likely to require changes in the transport

---

[45] https://www.cia.gov/library/publications/the-world-factbook/rankorder/2147rank.html
[46] http://www.schleswig-holstein.de/Portal/EN/Portal__node.html__nnn=true

layer. This favours the DTN transport approach over the DTN overlay one, or at least calls into question some of the motivation for developing an overlay for this application.

4.5.4 Middlebox Visibility

Middleboxes [SR02] are systems within a network that carry out some function requiring "application intelligence," typical examples include web proxies and caches in a content delivery network. Both approaches to DTN protocols involve the use of new middleboxes in order to handle the lack of end-to-end connectedness. There has historically been some concern [CA02] about this approach generally. The main concern being that middleboxes undermine the end-to-end aspects that have been crucial to the success of the Internet. [DA01] One of the conclusions reached (in [CA02]) is that it is preferable that the actual endpoints (i.e., source and destination) be able to be made aware of the locations and behaviour of whatever middleboxes are present in the path between them. So, for example during the setup of a voice-over-IP (VoIP) call, various session initiation protocol (SIP) proxies may be used and it was argued that there are good reasons why a caller might want to know which middleboxes have been involved in setting up the call.

The counter argument to this is that any such control or signalling creates potential vulnerabilities whether to the source (potential DoS if flooded by reports) or to the middleboxes (whatever hosts can signal to them, will be able to exploit whatever vulnerabilities exist in the middlebox). In addition there are some unsolved problems with such reporting, for example, if an outbound packet makes use of some security features (e.g., authentication, integrity, confidentiality), then it is very hard to know which security services to apply to reports back to the source. In such a case, if a source encrypts a packet, is that a signal that reports returned to that source should also be encrypted? If not, then there is a potential information leakage. But if so, then this imposes a fairly extreme burden for key management or else creates very stringent restrictions on where traffic can flow.

So how does this impact the proper layering for a DTN protocol? Well, in one sense not at all since one can include middlebox reporting and control features at either layer, so there is no architectural difference there. However, the BP does include reporting back to the source (or to a "Report-to" endpoint), whereas LTP-T does not.

At this point it is not clear which is the better approach – the additional information provided by the BP reporting scheme has been found useful in interoperability testing, but generally such features have created real vulnerabilities in deployed services (e.g. McGann considered this for the session initiation protocol (SIP) [MC05]), so, for the present, we count this as being neutral for our argument.

4.6 Summary

In this chapter we defined what we mean by a DTN transport, provided a detailed analysis of how this compares with the currently mainstream overlay network approach to DTN protocols and considered some additional pragmatic issues.

The overall conclusion is that a DTN transport as defined here is a worthwhile prospect and one that warrants further investigation since it offers advantages over the overlay approach in some contexts. Note that we do not conclude that a DTN transport is generally *better* than an overlay, only that it is sufficiently interesting to justify further investigation.

In chapters to come we will describe one such DTN transport protocol, its implementation and validation.

Chapter 5

The LTP-T Protocol

So far we have derived a set of DTN protocol requirements, defined the concept of a DTN transport and compared and contrasted that with the overlay networking approach exemplified by the Bundle Protocol (BP). In this chapter we specify a DTN transport protocol as the basis for the further investigation of the concept of a DTN transport.

This protocol also provides the basis for the evaluation called for in the problem statement in Section 1.4. The goal of that evaluation will be to determine whether or not our DTN transport protocol can usefully serve as a DTN protocol, and whether or not our DTN transport protocol can outperform the BP.

We have already described the Licklider Transmission Protocol (LTP), which is a DTN point-to-point protocol with an extensibility mechanism. [FA08] We note that as a point-to-point protocol, LTP has none of the off-path signaling traffic of the BP. We also know that LTP can be used in at least some DTN contexts, (e.g., deep-space), and that there can obviously be situations where a sequence of DTN hops each use LTP. We therefore use the LTP extensibility mechanism to create an end-to-end capable DTN transport protocol, aimed at meeting the requirements we laid out in Chapter 2. We call that protocol the "LTP Transport" or LTP-T.

LTP-T was initially described in a TCD Computer Science Technical Report. [FA05] The version described here is largely the same, though some minor changes have been made as a result of implementation experience. In the remainder of this chapter we first give a quick overview of LTP-T, specify the protocol, and then discuss issues arising and some implementation factors.

5.1 LTP-T Overview

Since the base LTP specification [RA08] specifies LTP, including a full state machine, our specification of LTP-T mainly consists of the definition of a number of extensions and the specification of how those are processed.

We define an LTP-T session to consist of a sequence of DTN hops, each of which is itself an LTP session. For LTP-T, all LTP sessions are layered over UDP. At an intermediate LTP-T node, the outbound LTP session can be started before the inbound LTP session has completed. In this way LTP-T attempts to get as many bytes of a block as close to its destination, as rapidly as possible.

Figure 5.1 is an interworking diagram for an LTP-T session involving one intermediate router as well as the source and destination. The sequence of events is as follows (taking the labels A-G from the figure):

A: The source is ready to send out a block consisting of three data segments (DSes). At this time, a contact (shown by the thick double-ended arrow) opens with the next hop and the source sends the block as three red data segments with the last one being the end of block.

B. One light-trip time (LTT) later, the router starts to receive the data segments. However, the router cannot yet forward anything since, at this time, it has no open contact with a suitable next hop (in this case that will be the destination).

C. A contact opens with the destination so the router can now forward bytes from the block. In this case the MTU between the router and the destination is larger than it was between the source and router, so only two data segments will be required to transfer the block. The first such data segment is now sent from the router to the destination containing most of the bytes from the first two data segments sent from the source to the router.

D. The full block has now been received by the router which sends a report segment back to the source indicating that the entire block has arrived. At the same time the router has now sent both of the data segments to the destination and is awaiting a report segment from the destination.

E. By this time the destination has received and acknowledged the block by sending a report segment back to the router and has now also received a report acknowledgement from the router. Note that since the LTT between the router and

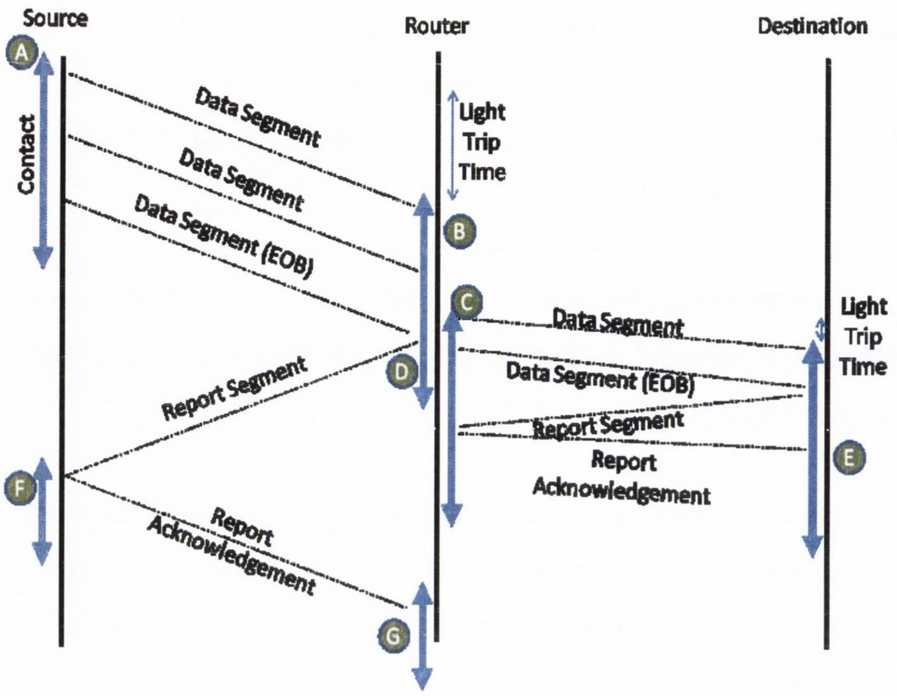


Figure 5.1 – An inter-working diagram illustrating an LTP-T session.

destination is less than between the source and router, the LTP session between the router and destination is completed before that between the source and router.

F. The source now receives the report segment indicating that all data segments arrived at the router and responds with a report acknowledgement. Note that the source knows nothing about whether or not the block has arrived at the destination.

G. The router receives the report acknowledgement from the source and at this point the LTP-T session is completed.

Further details, (including pseudo-code), of the forwarding algorithm are given later (Section 5.4) when we describe our LTP-T implementation.

As shown in the example above, since the MTUs for the inbound and outbound links may differ, LTP-T has to define a way to handle the re-fragmentation of the block. Essentially each LTP-T node can fragment the block into data segments in whatever way it selects with the only restriction being that the size of the red part of the block cannot decrease - it can increase. Since the red part is defined in terms of data segments (DSes), and not bytes, we may not be able to preserve the exact size of the red part due to re-fragmentation and/or the addition/deletion of extensions. For example, as will be seen below, LTP-T defines a "via" extension that is useful for debugging and tracing and is added at each hop. Adding that extension in general causes the block to be fragmented into a different set of data segments on each hop (since the "via" extension value grows larger at each hop). This can cause additional segments to be marked as red. The essential requirement is that each red byte from the block must always be red regardless of how nodes re-fragment the block.

Comparing LTP-T to the BP, one might say that each LTP-T node is taking custody for the red part of the block. However, given that this act of taking custody is implicit in the protocol, there is no need for specific protocol data units (PDUs) to control or report on this, representing a perhaps significant simplification compared to the BP.

All information required for end-to-end operation (e.g., the final destination) is carried by means of LTP extensions that are (mostly) preserved across all the LTP sessions required to deliver the block to the final destination.

At each LTP-T node, incoming segments that are not destined for this node are forwarded (or dropped) based on whatever routing scheme is used to determine the next LTP node. LTP-T itself does not define a routing scheme, nor how routing information is distributed. Although one could envisage a delay-tolerant equivalent of the Border Gateway Protocol (BGP) [RE06] being used for that purpose the definition of such a protocol is not part of this work. BGP itself could not be used since it runs over TCP and also generally involves long-lived TCP sessions; however, many of the other features of BGP could be directly relevant here since LTP-T also forwards based on the destination.

LTP-T provides all the features of LTP, including partial reliability. LTP-T also provides end-to-end data integrity and congestion control. An end-to-end confidentiality service is not provided at present, since as previously discussed, such a service is much less valuable in the absence of a working DTN-friendly key management scheme.

Our design goals here include taking account of the pragmatic arguments discussed in Section 4.5, for example, that an existing LTP implementation ought to be able to simultaneously work in either point-to-point or transport mode. LTP-T should also be relatively easy to implement, given an LTP implementation. We also require that the protocol should be implementable below a sockets API.

5.2 LTP-T Protocol Specification

In this section we define the LTP-T protocol. This is followed (in section 5.3) by some discussion of specific design issues. Our implementation is described in Section 5.4.

5.2.1 LTP-T Extensions

In this section we describe the set of LTP extensions used by LTP-T.

As a point-to-point protocol LTP need not carry a destination identifier, does not have to consider (re-)fragmentation and has no need to consider congestion at intermediate nodes on a path. A transport protocol must of course consider these issues and in LTP-T these are handled using the LTP extension mechanism, which was originally defined to handle the addition of authentication fields to LTP.

The LTP extension mechanism allows for the addition of both header and trailer extensions, up to a maximum of 16 (of each). All LTP extensions consist of a one-byte tag, a self-describing numeric value (SDNV[1])-encoded [ED06] value-length field and a set of octets containing the value. In the base LTP specification [RA08], extension numbers 0xC0-0xFF are set aside for private use extensions and are used here since the LTP-T extensions are not (yet) formally recognised.

We therefore start by defining the following set of extensions to LTP (the tag used in our implementation for each extension is given in parenthesis).

Source address (0xC0): the value of this header extension is the address of the originating LTP node (aka the LTP engine ID). This extension need not strictly be present for the first

[1] An SDNV is a self-describing numeric value, basically an encoded positive, multi-precision integer. The SDNV encoding scheme is defined in LTP and is also used by the BP.

hop, where the LTP session ID contains the required value, but should be included. This information would be required when propagating congestion information upstream[2].

Destination address (0xC1): this header extension contains the address of the final LTP node and is required for forwarding and routing. It is encoded identically to the source address header.

Estimated block size (0xC2): this header extension contains up to 4 network byte ordered octets (i.e. a 32 bit unsigned integer) that specify (an estimate for) the number or bytes in the block to be transmitted, so as to allow for congestion control at intermediate nodes. Note that this extension is an estimate since the exact size may not be available, for example, if the application deals with streaming content. It is not an error for the actual block to be bigger or smaller than this value.

Port (0xC3): this header extension contains network byte ordered octets that represent the port at the destination to which this data is targeted. It is analogous to the client service identifier currently defined in LTP. Using more than two bytes for the value will mean that many nodes will not be able to translate that into a TCP/UDP port number which could be troublesome, so we recommend that the value be less than 65536.

Hop count (0xC4): this header extension has as its value one byte containing the number of further hops allowed before segments from this session should be dropped. When a packet arrives when the session hop count is zero then, if the current node cannot communicate directly with the destination, the segment must be dropped. This extension is used to mitigate undesirable looping.

Via (0xC5): This header extension should contain a sequence of LTP address/time pairs representing the path taken by the data so far. The encoding is as a single octet string containing UTF8 characters. Any local time format may be used. If a node is configured to accumulate tracing information as segments traverse a path, then it may include a via

[2] The upstream direction is the direction from which data segments arrive, i.e. towards the source, and the downstream one is (hopefully) towards the final destination for the block.

extension in any LTP-T segment being forwarded. This first via must have one entry representing the sender and the time the segment was transmitted (rather than the time at which the segment was enqueued). Subsequent nodes should include an updated via extension in all segments (including fragments) containing data which was found in that first segment. Updating the via extension value is done by adding this node's identifier and timestamp to the end of the previous value, that is, the value of the extension lists the nodes in the order in which they were traversed.

A node may include more than one entry for itself in the via extension string, perhaps adding one at the time of receipt of the segment and another at the time of forwarding. Note that this may become hard to interpret if re-fragmentation has occurred, which will be common if all nodes add via segments. Care should be taken if any conclusions are drawn from the value in a via extension. This extension is for testing and debugging and should be reminiscent of the Received: mail header field.

End-to-end authentication (0xC6): This field defines how to calculate an authentication field, similar to that defined in LTP, but that can be validated end-to-end. We use almost the same syntax as the basic LTP authentication extension, the differences being that we do not need both a header and trailer here[3], and that the input bytes consist only of those bytes of the block that will be the same end-to-end.

In contrast to LTP's hop-by-hop authentication, in this case there is no need to use a trailer extension since the authenticator covers the entire block, and not just a single segment. So this extension must be carried as a header extension with the ciphersuite, the optional KeyID and the authenticator value all present.

Only the final destination node is required to verify this extension's value, intermediate nodes may verify the value if the ciphersuite allows and if they cache/digest all the required bytes of the block.

[3] The base LTP authentication extension authenticates segments, whereas here we are authenticating blocks, so that the memory buffering considerations that motivate the use of a trailer extension do not apply.

The input bytes for the end-to-end authentication signature or message authentication code (MAC) calculation comprise the following, in the order stated:

- The number of extensions included in the calculation (encoded in a single byte, a value between three and five[4]).
- The ciphersuite and optional KeyID from the end-to-end authentication header
- The source address, even though this need not be explicitly present in the first hop session
- The destination address
- The estimated block size, if present
- The port number, if present
- The bytes of the block, preceded by its length, encoded as an SDNV

When canonicalising the above, the entire encoded set of extension values (source, destination etc.), but excluding the tag and length, are included in the input to hashing. Note that the input bytes do not include hop-by-hop extensions, nor the hop-count, or the via extension, which all change at each hop.

The structure of this extension is the ciphersuite, followed by the optional KeyID, followed by the authentication value. Where the KeyID is not supplied we include a zero-byte value instead in the encoding (but not in the input to digesting) in order to make it easier to decode the structure. The contents of the end-to-end authentication header are:

- A single octet containing the ciphersuite (same values used as in LTP)
- The optional KeyID. If no KeyID is supplied, the next byte is 0x00. If a KeyID is supplied, then the subsequent bytes contain the SDNV-encoded length of the KeyID, followed by the bytes of the KeyID itself.
- The remaining bytes of the header value contain the actual authentication value, as determined by the ciphersuite.

---

[4] This count is for future proofing. For now, 3 means the source, destination and the end-to-end authentication header, 4 means either port or estimated block size was present and 5 means both were.

Congestion notification (0xC8): The value of this extension is a sequence of pairs: the first element of each being an encoding of the LTP node address to which the congestion information pertains, the octets of this element are preceded by a two byte unsigned length field, and then the octets representing the node address itself. The second part of each pair is a 4 octet (network byte order) value specifying the number of (real, not punctuated) seconds from the time of transmission of the segment during which it is believed the node in question will be suffering from storage congestion. This value may be calculated in various ways in different circumstances – see Section 5.3.1 below.

In order to support various different orders of magnitude of delay the leftmost byte of this value is treated as the exponent of a power of 10 that is to be multiplied by the rightmost 24 bits. So, if the value is 0x00000001 this represents 1 second, a value of 0xff000001 represents one tenth of a second, and 0x02000001 represents one hundred seconds. If the appropriate time value is unknown (e.g., if the congestion timer described below expires) then the time field should be zero.

Note that receivers should exercise caution in reacting to receipt of such a notification – there is nothing to prevent a bad actor using this notification to try to control how and when segments are transmitted, for example, a greedy implementation could try to use this to decrease the latency of its data transmission.

Table 5.1 below summarises the set of LTP-T extensions.

Extension	**Justification**
Source address	Useful for logging and debugging.
Destination address	Required for forwarding.
Estimated block size	Required for congestion handling.
Port	Useful for de-multiplexing.
Hop count	Useful for logging and debugging Allows handling erroneous routing loops.
Via	Useful for logging and debugging.
End-to-end auth.	Required to provide data integrity.
Congestion notification	Requried for congestion handling.

Table 5.1 –LTP-T extensions summary.

5.2.2 LTP-T Protocol Operation

LTP-T basically consists of a sequence of more-or-less independent LTP hops, with the differences/additions as described below.

Other than the congestion notification extension, all other LTP-T extensions are carried in data segments; they must not be present in report segments, report acknowledgements or cancel-related segments. The congestion notification extension can be carried in any segment (since it is not a part of any session).

The originating LTP node determines the red/green parts of the block and in order to meet the originator's expectations for partial reliability, this distinction must be honoured by all intermediate nodes, which can, however extend the red part of the block. Such red-part extension will typically be done to ensure efficient mapping of data segments to MTUs. In this respect, the source trusts the intermediaries to carry out the protocol correctly, which leaves it vulnerable to an intermediary that causes delivery of a smaller red part than requested. For example, if the source nominated the first 1K bytes of a 10K block as red, then the destination could end up receiving only the first 0.5K bytes as red if some intermediary cheats. In principle, this could lead to unexpected or even dangerous application behavior.

Other then applying end-to-end authentication there is no way to detect this specified by LTP-T. The justification for this decision is that, in the absence of end-to-end authentication, the intermediary can in any case modify the values of the bytes of the block so any improvement upon simply trusting the intermediaries requires some use of end-to-end authentication. One could argue that we should re-define the end-to-end authentication extension so that it only covered the red part of the block, however, that would remove the ability to extend the red part (which is specified on a per-segment, not a per-byte basis) which can be useful when re-fragmenting.

Alternatively, one could argue that the end-to-end authentication extension itself should specify which bytes of the block it covers. While that appears to be a reasonable proposal, for now, we have chosen the simpler option that the end-to-end authentication extension covers the entire block. Future experimentation may of course indicate a need to revisit this decision.

LTP-T handles errors and Automatic Retransmission reQuest (ARQ) on a hop-by-hop basis, so reports are used just as in LTP. Effectively this corresponds to the custody concept for bundles, with the difference being that each LTP node is required to accept custody of all blocks it successfully receives. Custody is therefore passed when a block is successfully received by the next node; however, whether or how many times a node will re-transmit a stored block is a local matter.

In some cases, an entire LTP-T block will be successfully transferred between two peers prior to any of the block's bytes being forwarded to a subsequent peer on the path to the destination. However, there will also be many cases where some of the block will be forwarded before the entire block has been received. In such cases, even bytes which have yet to be acknowledged to the originating peer may be forwarded to the next hop.

An intermediate node must re-fragment the block into segments if required to do so. For example, if the MTU on the next hop is too small for segments received on the inbound hop. Fragmentation must not shorten the red part of the block, but may extend it; otherwise nodes are free to re-fragment in any way.

Since each node takes custody of each segment's bytes, an LTP node may well suffer from congestion in terms of storage space. In order to reduce the likelihood of this occurrence, the estimated block size extension may be used to reserve space. A node that does not have sufficient space to handle an incoming block should cancel the session indicating that congestion has occurred. Congestion is discussed further below.

Cookie handling and hop-by-hop authentication are handled on a per-hop basis – there is currently no end-to-end signaling related to either.

Since LTP-T is built on LTP, and since both protocols make use of LTP extensions, we have somewhat of a clash when it comes to handling unsupported extensions – effectively we have to route around the fact that we have two namespaces but only one extension-tag space. For now we adopt the rule below, but in future, one could simply pre-allocate a range of LTP extension tags for use with LTP-T which would help simplify the rules below significantly.

Where a node is forwarding an LTP segment that contains an unknown or unsupported extension it should not include that extension in any corresponding outgoing segment, unless specially configured to do so. A node may be configured with a list (or equivalent) of extension tags that are in fact to be forwarded. When an unsupported extension instance is to be forwarded, then each inbound instance of the extension must be included in exactly one outbound segment. Note that where the outbound segment size is significantly bigger than the inbound segment size, then it can happen that one outbound segment could include more than one inbound unsupported extension. The reason is that since LTP is a point-to-point protocol, then in general, it makes no sense to forward unknown extensions. If additional LTP-T extensions are defined later on then there will be an issue with upgrading old nodes, however for an experimental protocol at this stage of development, such a flag-day is acceptable, especially since we expect implementations to include support for the list of to-be-forwarded extension tags.

We use the following terminology when discussing the use of these extensions: The "initial segment extensions" are the source (where necessary) and destination address, the block size, the port and the hop-count. We have the following rules for handling initial segment extensions:

- The first segment of a session must contain the initial segment extensions; subsequent segments of the session may include them. LTP-T receivers must be able to handle the case where the initial segments for a session do not contain these extensions (or are not correctly received, at first). Including the initial segment extensions in a number of segments may be useful if there is a high probability of the first segment of the session being corrupted or lost.
- Initial segments should be marked as red. It basically makes no sense (in general) to risk that the initial segment extension-containing segments get lost.
- The same values for the initial segment extensions must be used for all of the initial segments, that is, a node may not vary the values in these extensions within a single session.
- If end-to-end authentication is to be used, the initial segments should also contain the end-to-end authentication header extension. The reason for the should above is that if a source were to be supporting a streaming application, then it will not be

able to calculate the authentication header until the last bytes of the stream have been seen. However, this is an unusual case, especially end-to-end.

Note that there is no requirement to retransmit, or request retransmission for, data that fails end-to-end authentication checking. Such failure should be recorded and made available to a receiving application, but the data may still be delivered. The reason for this is that the end-to-end authentication covers both red and green parts of the block, and of course green segments may go missing. So long as the red part has arrived then the block should be delivered irrespective of end-to-end authentication. If an application requires fully end-to-end authentication then this can be easily achieved by making the entire block consist of red segments. Essentially, we leave it up to the higher layers to decide how to properly deal with such cases.

LTP-T clearly requires some routing scheme in order to work, however we do not define how to route blocks at this level, since that effectively has to be handled in the same way as lower layer cues, e.g. via a scheduling module. This is required, for example, if routes should be selected on a first-available basis in an attempt to optimize "progress." We therefore assume (for now) that the scheduling module (see below) also provides routing tables or equivalent. Characterising routing schemes that are generally suitable for LTP-T is for future study. We will however see what the LTP-T implementation does later in this chapter.

5.3 LTP-T Design & Implementation Discussion

Now that we have seen the protocol we can discuss some of the more interesting aspects of the LTP-T design, some of which are generic and some implementation specific.

5.3.1 Congestion Handling

The scheme for handling congestion is currently quite simple and has not yet been implemented, so should be considered provisional.

We assume that each LTP implementation has a (punctuated) congestion timer for to-be-forwarded blocks, that is, for the entire LTP session (perhaps based on the estimated block size extension). The idea is that the entire block has to be transmitted and its red part acknowledged before this timer expires. The congestion timer is started when the first outbound segment for the block is transmitted (not when it is enqueued). This timer applies regardless of the route taken, so that the timer is not reset if a fall-back route is attempted following a failure of the first attempted outbound LTP session. If this timer expires, the entire block is deleted and all state information associated with the block is also deleted (e.g., an inbound session might be cancelled).

If the congestion timer expires on an intermediate node, then that node should, at the next opportunity, signal the congested state to upstream nodes by using the congestion extension. Similarly, a node suffering storage congestion should signal this to relevant peers by including the congestion notification extension in a cancel segment.

Nodes further upstream should further report the congested state to peers whenever they are aware of it and where segments from that peer may transit the congested node. If a node has no reason to believe that this information will be useful, (for example, if it is known for policy reasons that the upstream peer will never route data via the congested node), then the information should not be forwarded. So LTP-T piggybacks congestion notifications and has no equivalent to an explicit congestion notification. [RA01]

Nodes must keep account of the time elapsed and the scheduling involved and must not include an entry in a congestion notification if that would be useless to the recipient. In a deep-space context, it is not unreasonable to expect nodes to be able to calculate an estimate of the time for which they will continue to be congested, since the set of upcoming contacts is generally known. If the expected congestion time remaining is less than the light trip time from the notifier to the peer and then back to the congested node there is no point in informing the peer, since the congestion event will be over before the information arrives at the peer. In this way, congestion notifications are bounded, in time and (equivalently) in space.

5.3.2 Re-fragmentation

LTP-T segments will arrive at an intermediate node according to the prevailing schedules or whatever conditions determine successful segment arrivals. The node will of course carry out the LTP protocol with the upstream node but may also begin transmitting segments to the next downstream node, even before the entire block has been received.

When this happens there are a number of new considerations, mainly due to the potential for re-fragmenting the block.

If the MTU size differs in the upstream and downstream directions then re-fragmentation is almost certainly necessary, at least if efficient transmission is a goal. Re-fragmentation can also be caused for other reasons, in particular if the node is on some kind of security boundary, the upstream segments may use cookies and hop-by-hop authentication, but perhaps the downstream direction is inside a "secure" zone and these extensions can be dropped. In this case re-fragmenting would be more efficient in terms of requiring fewer segments, and so an implementation may choose to take advantage of this.

The current implementation handles re-fragmentation without requiring the block to be copied, or requiring that there be no "gaps" in the inbound block – essentially the implementation keeps track of the "scopes" separately for the inbound and outbound LTP sessions, even though only one relay socket is used.

Whenever re-fragmentation does occur, then we have to consider how to handle the LTP-T extensions. We could for example require that the value of some extension be the same in all incoming segments and treat changes as an error. However, we also need to allow some extension values to change (e.g., the via segment values will differ due to different timestamps). And if re-fragmentation resulted in three data segments becoming two then we have to decide which value to include in the forwarded segments.

At this stage our general approach is to include the most recently received value for any given extension (over the entire block, so far) in all of the outbound segments. Our implementation currently does this even if no re-fragmentation is required – so that we can maintain a single value per LTP-T extension per block which simplifies the data structures required. It may be worth noting that, for the via extension, this has the effect that the outbound segments will contain a via extension value that reflects the most recently arrived inbound segment. For example, if the entire block is received before any segment is forwarded, then the via extension on the first forwarded data segment will reflect the via extension value of the last data segment arriving from that block. One consequence of this approach is that if ever a new LTP-T extension type is defined, that definition must allow instances from individual segments to be dropped[5], since there is no LTP-T equivalent of "do not fragment." [PO81i]

5.3.3 Forwarding

In this section we describe some details of how LTP-T forwarding works. These details are perhaps somewhat implementation dependent, since different implementations might involve different trade-offs.

Forwarding is relatively straightforward to envisage though somewhat cumbersome to implement. The basic idea is that each time we have a contact opportunity with a next hop, we check whether or not there are any data segments to send to that hop (LTP control segments[6] will be forwarded by the normal hop-by-hop LTP code).

There is one notable difference between this forwarding and that normally done by an LTP node. The LTP-T node must only prepare segments when the communications contact is imminent, whereas an LTP source or destination can often prepare segments for transmission by following the state machine, but without reference to when the next contact arises[7].

If we assume for a moment that an LTP-T router were to behave in the same way as a normal LTP node, then we can see why this is the case. Were the router to do this, it would receive some data segments, decide that these need to be forwarded, create new data segments (for the 2nd, outbound, LTP session) and queue those for later transmission. However, if some data segments from the inbound session were initially lost, but then arrive prior to the outbound session contact, then the router will, at best, be likely to be sending data out of order, but is, in fact, more likely to send data segments with unnecessary gaps. For the router to do as well as it can, it must not calculate the segments

---

[5] This will complicate key management for an end-to-end confidentiality mechanism, e.g. making it quite hard to use a counter-mode of operation.

[6] Recall that LTP control segments are the RS, RA and session cancellation segments.

[7] This does assume that the MTU will not vary on the relevant time-scales and that there are some time-and-size dependent extensions to be added etc. that might affect the fragmentation.

to transmit until (perhaps just before) the next outbound communications contact, when additional retransmitted data from the originator may be available.

In fact one can envisage a number of forwarding strategies that might differ in how they perform in the face of various operational aspects of the system. In the limit, one might not forward any data until the entire block has successfully arrived and any red-parts ACKed etc. Though this will clearly be less performant than more aggressive forwarding in many cases, there may be cases where it makes no difference if one attempts to forward aggressively – say if the time between contacts is much longer than a session duration which is, in turn, shorter than a single contact. In such cases, there is little point in attempting to be aggressive in forwarding.

The LTP-T implementation currently includes all configured LTP-T extensions in all DSes. While this is the simplest thing to do, one could tune the implementation to take better account of bandwidth availability and loss characteristics of specific LTP hops, for example, if the next hop were known to have a much smaller MTU than the one following that, then one can safely omit some instances of extensions like end-to-end authentication.

5.3.4 Routing

LTP-T routing requires a router to select a next-hop destination for each outbound block. In principle one could use many different schemes to select the next hop, but for now, our implementation simply does a wild-carded IP address table lookup based on the destination address. Whenever there is a contact open, we check which of the currently not-yet-forwarded blocks should be sent via that contact's peer. Currently this involves a simple address mapping table, mapping from a netmask (e.g., "127.0/16") to a particular next hop IP address.

The first time we encounter a netmask that matches the destination and where the next hop is currently contactable then we have identified the next hop for that block, and so begin the outbound LTP session. One consequence of that is that we are selecting the route for the entire block – different DSes cannot be routed differently in our model.

Since LTP-T inherently involves taking custody at each hop, this creates a limit on how "dynamic" routing can be. Essentially since each LTP session is heavily stateful, we require that the routes remain stable for about the duration of an LTP session. Were the underlying routing tables to change more frequently, then it is likely that many LTP sessions would fail, and so, eventually would the LTP-T session. One consequence of this is that one might want to use some other routing scheme were one to want to use LTP-T over a mobile ad-hoc network where routes will commonly change. On the other hand, with careful selection of netmasks, one can actually support many situations and this method of routing is well known and studied. [DO00]

With LTP-T, the combination of routing and fragmentation works differently from the BP. In LTP-T once we select the next hop, then the entire block is forwarded to that next hop and any re-fragmentation is carried out as required for that hop. Ignoring fall-backs, LTP-T never sends some bytes of a block on one route, and others on a different route. BP fragmentation, in contrast, splits a bundle into separate bundles that can each take a different route towards the destination.

Basically, the LTP-T scheme is simpler, and as a result less flexible, but generates no overhead traffic and so is also more predictable. From at least the security point-of-view such additional complexity and overhead traffic is highly undesirable. However, this simplicity/flexibility trade-off is one that recurs in protocol design in many places and there is never a "right" answer in general – only extended experience with operating the protocol will really select the right approach here.

5.3.5 Routing Loops

Most protocols properly treat routing loops as error cases that are to be avoided where possible, and mitigated when they occur. In a DTN however, there are some scenarios where a route containing a loop is the best route to follow.

For example, take the data mule scenario shown in Figure 5.2, where the data mule traverses a field of six nodes on the outward leg of its traverse, and then passes the same set of nodes, in reverse order, on the return leg of its traverse. As shown, node 2 deposits packet A with a destination on the Internet. However, when node 3 deposits the much larger packet B, with a destination of node 5, the data mule's storage is starting to become full. So, knowing (somehow) that it will contact node 4 on the return leg, the data mule temporarily deposits packet A with node 4. Having delivered packet B and now with

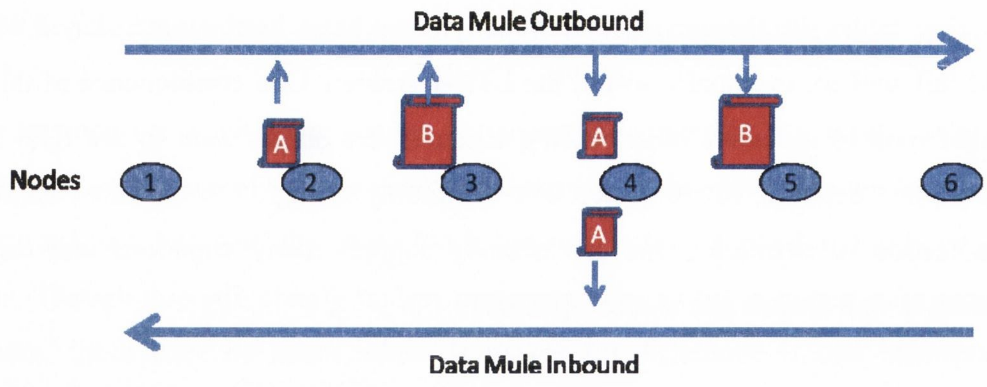


Figure 5.2 – A desirable routing loop.

sufficient storage, the data mule can again receive packet A on the return leg and finally successfully deliver the packet on its return to somewhere well-connected. In this example, packet A has followed the path, {node-2, data mule, node-4, data-mule, ..., destination}, which would normally be seen as containing a routing loop by the data mule the second time it receives the packet.

From the above we can see that we do not want to avoid all routing loops in a general DTN, but clearly we also do not want to allow a packet to loop endlessly. For this reason one can justify the inclusion of some form of loop control. That loop control could take the form of including a via-like field or some other form of tagging, or could (as in the case of the BP) be based on an expiry field. For LTP-T, we chose a simple hop-count, to be decremented at each hop, and where the packet is dropped if the hop-count reaches zero before the packet reaches its destination. The reasons for this are that it requires no clocks and hence is simpler and more robust than the BP's expiry method, and secondly, that it is much easier to implement compared to any tagging based scheme.

5.3.6 UDP Binding

In the design of LTP-T, we basically assume that we are running LTP and LTP-T over UDP or IP. For ease of implementation however, we have chosen to concentrate on layering LTP-T on UDP since this means we do not need to modify kernel code and can more easily port the code to various platforms. An IP binding would be trivial to derive though would of course require that a new IP protocol number be assigned for LTP, which is perhaps premature. LTP-T can therefore be considered to be layered only on UDP for the present.

The UDP binding for LTP is fairly straightforward and basically involves using the IP address and optionally UDP port number as the LTP engine ID in LTP PDUs. The IP address is carried in network byte order and the port is carried similarly in two octets. If no port number is supplied then the standard[8] LTP port number (1113) is used.

5.3.7 A 3rd Party Analysis of LTP-T

Based on the initial definition of LTP-T, [FA05] Fahad [FA06e] carried out a study of LTP-T and also ran a simulation based analysis and found that:

- The omission of LTP-T extensions in "initial" segments can be problematic – the example was the omission of the LTP-T destination in the 2nd and subsequent DS, when the 1st DS is lost. Such cases are problematic, and probably justify inclusion of at least the destination in all DSes.
- More checkpoints help with lossy hops - LTP supports data segments that effectively request an early report segment – a so-called checkpoint segment. Whenever a received LTP segment contains the checkpoint (CP) flag, then the receiver prepares and sends a report segment specifying the current state of reception of the LTP block.
- There was an ambiguity in the description of LTP-T [FA05], so that it wasn't clear that segments could be forwarded before the entire block had been received. That has been clarified in the description above; as such forwarding was always intended to be allowed.
- The more red segments there are, the higher the end-to-end delay (and hence the lower the goodput[9]), if there are "significant" losses on individual hops.

[8] This port is registered with IANA as 1113 ltp-deepspace for both TCP and UDP. The TCP port will presumably not see much use:-) http://www.iana.org/assignments/port-numbers

[9] Goodput here is defined as the number of bytes successfully received compared to the duration during which one could possibly receive something, e.g. time in "suspend" would not be counted when calculating goodput.

- The changes in response to increased propagation delay, increased losses, increased number of red segments and other factors were linear – there were no cases (in the study) of exponential decrease in performance[10].

Fahad's overall conclusion is that an implementation should add more CPs as it sees more losses on the next hop. Generally that conclusion, and the linear decreases in performance are interesting validations of the design of LTP and LTP-T.

5.4 LTP-T Implementation

In this section we describe the LTP-T implementation, how it is configured and aspects of the implementation that may be of interest.

5.4.1 Preamble – Implementation Overview

In order to simplify later parts of this section we begin with a quick description of the structure of our LTP and LTP-T implementation[11]. Both protocols are implemented in the same source base, with the default being to operate in LTP mode. The implementation involves about 20,000 lines of C/C++ code, using the GNU development tools (gcc, autoconf etc) and has been ported to various linuxes and Cygwin[13]. The build starting script is called "bootstrap" and calls autoconf, autoheader, automake and runs the resulting "./configure" script.

The main library is called `LTPlib` and is built as both a shared object and a static library. The library offers the afore-mentioned sockets API to applications but with the functions being preprended with the string `ltp_`, i.e., `ltp_sendto(), ltp_recvfrom, ltp_setsockopt()` etc. Each application socket can be used for multiple LTP (or LTP-T) sessions, both inbound and outbound.

---

[10] Fahad interestingly characterized different applications based on the percentage of segments that are "red". Whether that is a generic distinction worthy of note is not quite clear, but it is at least worth considering.

[11] The source code is available from: http://dtn.dsg.cs.tcd.ie/sft/ under the Mozilla/GPL/LGPL 3-way license.

[13] An earlier version of the LTP code (called "perfume") was fully integrated with the main SeNDT code and was also cross-compiled to run on arm-linux which is the OS on the SeNDT hardware platform.

```
$ ltpd -?
ltpd is the LTP and LTP-T ltp_daemon.
ltpd [-2 l2mtu] [-b buffersize] [-c crypto-cfg] [-d] [-g [fname]] [-i input] [-l log]
     [-m mode] [-o output] [-r routes-cfg] [-s] [-t LTP-T-cfg] [-v] [-w sleeptime]
     [-A ciphersuite] [-C cookielen] [-D dest[:port]] [-G cookie-grace-period]
     [-L listener[:port]] [-K keyid] [-S source[:port]] [-R redlen] [-I]
Local parameters:
     [-2 l2mtu] sets the layer 2 maximum transmission unit (packet size) via a sockopt()
        call (min=500,max=1048576)
     [-b buffersize] specifies the size of receive buffer for the application to allocate
        (def=65536)
     [-c crypto-cfg] sets the crypto-configuration file name to crypto-cfg [ltpd.crypt]
     [-d] instructs ltpd to sync state information to disk and re-load state on start-up
     [-g [file]] instructs ltpd to act as a file server or to ask the server for a named
        file. [def: ltpd.fetch]
     [-i input] sets the input file name [ltpd.in]
     [-l log] sets the log file name [ltpd.log]
     [-m mode] the operational mode - Client, Server or Router [Client]
     [-o output] sets the output file name(s) [ltpd.out]
     [-r routes-cfg] specifies where LTP-T routing configuration file [ltpt.routes]
     [-s] instructs ltpd to spawn a process to run as a daemon - the initial process
        exits leaving one behind [false]
     [-t [LTP-T-cfg]] use LTP-T & optionally specify configuration file to use
        (default: ltpt.cfg)
     [-v] requests verbose output (to stdio) [false]
     [-w [sleeptime]] when operating as a client, instructs ltpd to exit without waiting
         for the user to hit a character [10] - a value of "b" makes the sending socket
         block until the session is complete, i.e. "-w b"
Protocol parameters:
     [-A ciphersuite] specifies which LTP-Auth ciphersuite to use
        ["Off"/0; "MAC"/1, "Signature"/2, "NULL",3 ] default is none
     [-C len] turn on cookies of len bytes (cookies are off by default, same as len=0)
     [-D dest[:port]] sets the destination IP address or DNS name [127.0.0.1:1113]
     [-G cookie-grace-period] specifies the number of seconds before cookies must be seen
         in responses [5]
     [-I] forces the stack to use very short numbers, suitable for interoperating with
        the ION LTP code [false]
     [-K keyid] specifies the key id to use if some LTP Auth stuff is wanted.
     [-L listener[:port]] sets the host IP address and more commonly, port on which to
        listen as a server
     [-R redlen] sets the red/green length (default:allred=-1;allgreen=0,max=16777216)
     [-S source[:port]] sets the source IP address or DNS name
Notes:
- The "Router" mode of operation is only meaningful for LTP-T of course.
- Options using uppercase (e.g. D,G,S) relate to protocol parameters,
  others are local
- Protocol parameters all default to 127.0.0.1:1113
- See the ltp extension document for ciphersuite descriptions but note that NULL is not
  "off" its a test mode
- If you set a source (-S) while in server mode (-m S) then all responses will be forced
   to go to that address:port
```

Figure 5.3 – ltpd `usage()` output.

There are various test programs included in the source and one main binary, the LTP daemon (ltpd) that can act as either a file put/get client[14] or server or as an LTP-T router. The `usage()` information for ltpd is shown in Figure 5.3. As can be seen the code

[14] In the following we refer to the LTP sender as a client and the LTP receiver as a server.

```
Fix/detect lower layer status as necessary via "cues" SPI

if (Transmitting) then
    listeners=whos_listening()
    trickle_forward(listeners) ; create new segments for listeners based on stored data
    for each listener for which we have a segment enqueued (control segments first)
        while I haven't sent too much
            for each segment
                check if extensions to be added
                encode segment
                transmit segment
                update LTP protocol state (incl. timers if necessary)

if (Receiving and soething new in inbound queue) then
    decode segment
    associate with session or create new session (if possible)
        if no session found/created drop segment and we're done
    update inactivity timer
    update LTP protocol state (e.g. scopes)
    generate and enqueue control segments (e.g. RSes) as necessary

if (Transmitting or Receiving) then
    update punctuated timers
    handle any timer expiry events (e.g. enqueue a segment for a re-tx)
```

Figure 5.4 – main loop Pseudo-code.

supports all of the features of LTP and almost all of the features of LTP-T (the main missing features being congestion-related).

The LTPlib library runs in three threads, one for interfacing with the application, one for listening for incoming segments and one for handling queued segments and timers. The library generates a log file with detailed logging information; by default this will be called ltp.log, though there are APIs to control its name, location, verbosity etc.

The build environment allows the developer to turn on in-built error generation that randomly creates errors when sending segments. Features here include segment deletion, truncation, corruption and re-ordering, each happening with a compile-time configured probability.

5.4.2 Main Loop Pseudo-Code

The main loop of the LTPlib is implemented in the `ltp_ping()` function. Figure 5.4 presents a pseudo-code version of this function which is called repeatedly inside the main thread. If the function returns more than a threshold number of errors with no successes then the error state will be propagated to the application.

The `whos_listening()` function returns the names of the LTP peers to whom this node can currently send segments. For "always on" type cases, there are some wildcard names that can be returned. The `trickle_foward()` function is where each current intermediate LTP session is checked to see if new data segments can be created to be sent to the relevant next hop.

5.4.3 Routing

In implementation terms the LTP-T router is enabled via a command line option to the LTP daemon (ltpd). The library itself, when operating in router mode, creates a batch of internal relay sockets (currently limited to 8) that are used to process inbound LTP-sessions for which this node is not the destination. This is purposely not visible to the application since we wish to be able to use any LTPlib application as a router, once the correct socket options and/or configuration are established. In other words, it is possible for an application to act as an LTP-T router without writing any application code. While this design option is useful for testing, a more scalable LTP-T router would properly take a different approach, in particular embedding the router code into the kernel.

When a new inbound LTP session is detected, and some conditions (see below) are met, then that session is allocated to a relay socket and bytes from the DSes are buffered in the same manner as with any inbound session for this node. However, once there are any bytes in a relay socket buffer then those are available for retransmission following the normal scheme. In other words, once some bytes from the inbound session have arrived, they are immediately available for forwarding. The idea here is to be optimistically driven by the lower layer cues so as to push the bytes towards the destination as soon as possible.

There are two conditions to be met before a new inbound LTP session will be allocated to a forwarding socket. Firstly, there must be a free relay socket to use. When both the inbound and outbound LTP sessions on a relay socket are finished with, then the relay socket is closed and a new one opened.

The second condition is that the LTP-T destination is not the current node. However there is currently no check that the LTP-T configuration provides some next-hop for that destination. This leaves open a fairly problematic potential DoS attack since a bad actor could insert segments for spurious or unreachable destinations and thereby use up all of the relay sockets, at least until the inactivity timer (see below) expires.

Routes for the LTP daemon are set by configuring a list of IP network address prefixes. Whenever an inbound LTP session is determined to be for a different node, then the ltpt_gateway() function is called and returns the name of the next hop. This function looks up the routing table and takes the first matching network address as the next hop.

For example, if the destination is "10.0.0.1" and a routing table entry specifies "10.10.10.10" as the next hop for "10.0/16" then the function will find a match and the outbound LTP session will use "10.10.10.10" as its peer. There is also some wildcarding for UDP ports, with the explicit value ":0" being the wildcard, so "10.0.0.1:1999" will also match "10.0/8:0". At present only IPv4 dot-separated format addresses are supported by the routing code, but this could be easily extended since all of the matching logic is in a single function called ltpaddr_cmp().

5.4.4 Garbage Collection Model and Handling Reboots

In order for LTPlib to work in many DTN contexts, it is necessary for an LTP session (whether standalone or as part of an LTP-T session) to be preserved across a reboot cycle. This might happen for example in a power constrained node (like a SeNDT node) where the application sends data (calls ltp_sendto()) but the next data mule pass won't occur until a subsequent power off/on cycle. So some sensor code might generate sensor data, send that using LTP and then the system is put into standby mode, say for 10 hours, until the next data-mule pass occurs. It is only at this point that the LTP data segments can be sent.

Handling this clearly requires that the block be synchronized to non-volatile storage, but that could of course be done by the application, or by LTPlib[15]. It is easy to see that its better for LTPlib to handle this storage – since LTP deals with the communications "cues" (see below), this is where the knowledge of when the next contact might occur resides.

[15] A lower layer could not really do this, since LTP-T calls for at least the via extension to be encoded at the time of transmission, not when the application calls the ltp_sendto() function.

Having the application handle this would, unless carefully done, represent a layering violation since the application would be forced to deal with a protocol-specific detail ("cues") that are more properly handled by the communications stack.

The non-volatile storage used is simply the file system. When LTPlib is operating in "disk mode," then each internal state change results in a re-synchronization to disk. LTPlib maintains a state variable that stores the state of all LTP sessions outstanding at any given moment. All LTP-T specific state is also accessible from this data structure.

Once any application process is running that links LTPlib running in disk mode, then it will start by reading the state from disk and continue thereafter. Clearly this could lead to various forms of livelock and/or deadlock, however the current LTPlib implementation does not cater for this problem since it won't arise in the initial deployments considered by the SeNDT project, which all involve a single main application process running for the duration of the boot cycle on the SeNDT node. Were LTPlib to be used in a multi-user system, then either the code would need to be ported to the kernel, or, more likely, some additional code would be required to handle properly synchronizing LTP state across multiple processes. The current implementation of this scheme can be found in the ltp_file.cc source file where the walkstatetree() function implements most of the functionality.

Another issue with this approach is that it brings with it a requirement for garbage collection since, for example, some timers may have expired, so that when the data structure is loaded from disk it may contain obsolete data. The current implementation however, only prunes this when it comes time to write the data structure to disk again.

All of this actually creates an interesting benefit for application developers – the application can operate in a fire-and-forget mode where that makes sense, thus greatly simplifying the application's use of the sockets API. For a sensor application this can reduce the amount of code required (at the application layer) significantly – a very short function can create a socket, call sendto() and then close() the socket relatively secure in the knowledge that LTPlib will take care of transmission even if that only occurs after a re-boot.

5.4.5 "Cues" Service Provider Interface

This `ltp_ping()` function described above uses the ltp_cue_*() service provider interface (SPI), which can be initialized by the calling application or can use the built-in cue function implementations.

The main functions from the cues SPI that are used in the main LTP code are ltp_cue_whoami(), ltp_cue_whos_listening() and ltp_cue_whos_talking(). The first is provided since cue implementations will often involve schedule files or other configuration settings that will include the local identity. The ltp_whos_listening() function is used to determine when to forward segments to a peer and ltp_whos_talking() is used to decide when to decrement punctuated timers for those peers who should currently have the opportunity to be sending segments to me. The use of punctuated timers is the essential DTN component of the protocol stack.

Depending on configuration, the default for these functions is to use localhost (127.0.0.1) as the whoami() result and a wildcard node name for whos_listening() and whos_talking() which results in an "always-on" mode of operation for any IP address.

The other standard implementation is based on the SeNDT project and its schedule files which are described in the next section.

5.4.6 Schedules

The current version of LTPlib supports three types of schedule – absolute time, relative time and ephemeris-based, each of which determine whether connectedness with given addresses is "on" or "off" at a given moment. At any time an instance of LTPlib can be running according to a combination of one or more of each type of schedule.

The **absolute time** type schedule is the simplest and maps to the ltp_cue_*() SPI most obviously – the schedule consists of a list of on/off times for each wildcarded IP address, e.g. saying that 127.0/16 is "on" between 200703120056Z and 200703120100Z.

The **relative time** schedule sets a schedule dutycycle during which the on/off events of the schedule occur, and they recur after each dutycycle. The start time of the schedule can be given as an absolute time (or defaults to the most recent midnight). Given a start time and a

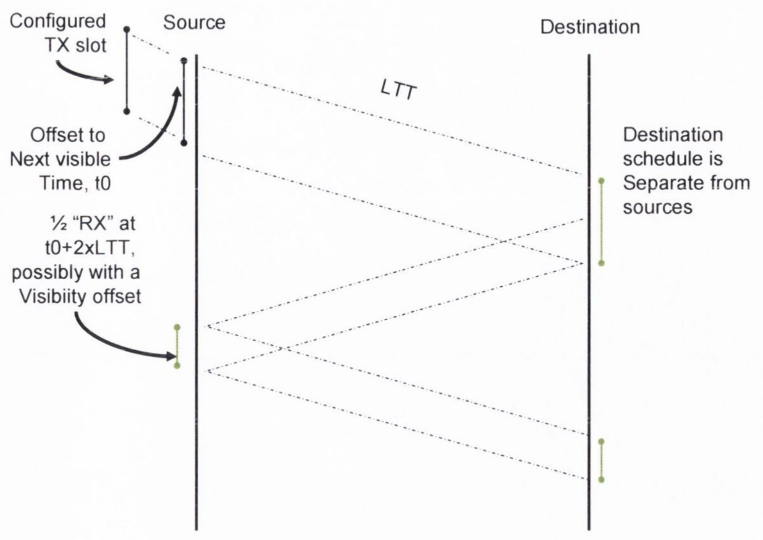

Configured TX slot
Source
Destination
LTT
Offset to Next visible Time, t0
Destination schedule is Separate from sources
½ "RX" at t0+2xLTT, possibly with a Visibiity offset

Figure 5.5 – An inter-working diagram for a DTN setup illustrating an ephemeris-driven schedule.

dutycycle we can easily see where we are in a schedule in order to determine the connectedness at that time.

In addition to the on/off events, the **ephemeris** type schedule also has a separate list of the visibility times for the connection in question. This information will generally be derived from real ephemeris data (as we'll see in the next chapter). During the process of calculating such visibility, the LTTs of the various connections are also handled, by yet another list.

The various lists are combined in order to meet the ltp_cue_*() SPI requirements – basically the scheme is to take a configured list of relative on/off times and to move those further forward until they are consistent with the visibility constraints. After that, new on/off events are generated after another RTT. The idea is that the configured on/off period is the "prime" period, but subsequent periods are invariably needed to handle e.g. segment loss and re-transmission etc. There are clearly potentially many variations on this base+RTT scheduling scheme – which will prove effective in the real world is a matter for future study. Figure 5.5 shows one such set of schedule events.

SOCKOPT	Comment
LTP_SO_AUTH	set/get LTP auth parameters to use - parameter is an ltp_auth_so
LTP_SO_BLOCK	calls to sendto will block until the LTP session has succeeded
LTP_SO_CLIENTONLY	turn off "layer 2" listening
LTP_SO_COOKIE	requests/reports cookies are on for this socket
LTP_SO_COOKIE_GRACE	set/get grace period during which we don't insist on seeing cookies
LTP_SO_INACTIVITY	identifies the sockopt() as the inactivity timer option
LTP_SO_L2MTU	set the l2 MTU from the application, ignoring lower layers
LTP_SO_LINGER	LTP sockopts identifies the sockopt() as the linger option.
LTP_SO_RED	set/get red/green-ness for the socket
LTP_SO_T_AUTH	recipient indicates it requires good LTP-T e2e auth
LTP_SO_T_CFG	feed in an LTP-T configuration
LTP_SO_T_ROUTE	feed in an LTP-T route
LTP_SO_SOP	Getsockopt() returns a structure describing the state-of-play with a socket

Table 5.2 – SOCKOPTs for LTPlib.

In addition to these types of schedule an additional XML schema[17] based approach was used in earlier iterations of the SeNDT code. That approach has been (temporarily) abandoned due to the dependency it introduced on the XML parser – it was felt to be more important at this stage to have an easy to compile, install and use library. The XML schema approach may be revisited if and when backend integration with planning or graphical information systems (GIS) software becomes a requirement - for the purposes of this thesis, it is not.

5.4.7 Configuration

LTP and LTP-T configuration is done via either files or calls to ltp_setsockopt(). We will document each separately though there is some overlap.

SOCKOPTS

Table 5.2 shows the set of socket options that are supported by LTPlib. With the exception of those described below, the meaning of these should be fairly obvious. Basically, the idea is to allow the application to control anything that can be configured via files.

The LTP_SO_BLOCK option is used to make the ltp_sendto() call block until the LTP session is completed – this is used mostly when generating timing information. Setting LTP_SO_CLIENTONLY has the effect of pretending that the layer 2 listener thread can't

[17] Those schemata are published at: http://dtn.dsg.cs.tcd.ie/schemas/

produce anything and so pauses any punctuated timers (as well as meaning the library will miss any inbound segments). LTP_SO_T_CFG allows the application to feed in an LTP-T configuration file name. See below for a description of this file format. LTP_SO_T_ROUTE allows the application to add a new route to the list of LTP-T routes – routing controls here use the same structure as in the route file described below. Finally (for now), LTP_SO_SOP can be used with ltp_getsockopt() to access detailed information about the state-of-play (SOP) of the socket. This information includes for example, the size of the red part of the block and information about remaining gaps in the green part – the structure returned is the ltp_sop structure defined in ltp.h.

Configuration Files

Table 5.3 lists the configuration files that may be used with an LTPlib installation. Recall that none of these files are required, in their absence LTPlib will act as an LTP client or server but will turn off LTP-T.

First, some preamble – all configuration files treat lines beginning with "#" as comments. Whitespace at the start of (almost) all lines is ignored. Whitespace, can however, be significant inside individual records in a file. Some of the file handling code is also a little brittle in how it handles, e.g. comma-separated lists, but following the examples found in the source tree should work fine.

The ltp.names configuration file is straightforward and simply acts as a string mapper, so that names can be given to nodes, node:port combinations, subnets etc. All of the other files unwind these names when they are encountered. No recursion is performed – each name is mapped once and once only. After mapping, any DNS names will be resolved in the normal manner. So if the configuration (after mapping) uses names and not IP addresses, LTPlib requires a working DNS, or at least a working name resolver (e.g. /etc/hosts file). This use of DNS is the same whether the installation is a LTP client or server or LTP-T router.

The ltpd.crypt file contains settings for use by the LTP authentication extension. Each record here specifies some peers, and the ciphersuite and optional key identifier to use for that peer. The file can, of course, contain any number of records. The first matching record

File name	**Comment**
ltp.names	Maps strings to strings, e.g. "landers" to "10.1.1.1:1234" or earth to "10.2/16"
ltpd.crypt	Contains LTP authentication key settings, including possible links to OpenSSL key files (e.g. mycert.pem, mykey.pem, or binary AES key files (e.g. mac-127.key)
ltp.sched	Contains one or more LTP schedules, which can in turn refer to one or more visibility files and ephemeris files
ltpt.cfg	Controls the LTP-T configuration
ltpt.routes	Specifies the LTP-T routes
Visibility file	Specifies the periods during which nodes are visible to one another
Ephemeris file	Specifies the LTTs between nodes at a given moment

Table 5.3 – LTPlib configuration files – the examples shown are close to those to be found in the LTPlib/client directory of the LTPlib source code.

is used for doing security on inbound or outbound segments. Note that the contents of this file can also be used for LTP-T's end-to-end authentication.

A signature ciphersuite record specifies the public key file containing the public key (actually certificate) of the peer, and, if this is the signer's own installation, the name of the private key file containing the private key. These files are formatted according to the formats used by the openssl package, e.g., in the case of the public key this is the ASN.1 DER-encoded [IT97] RSAPublicKey as defined in RFC 3447 (see sections 3.1 and A.1.1). [JO03]

In the case of the MACing ciphersuite, the record contains the name of a file that is treated as a binary file, the first 128 bits (16 octets) of which contain the AES key to be used for the MACing operation. The file may contain ASCII or UTF8 or binary data – in all cases the first octets of the file are simply read and fed into the MACing algorithm.

The ltp.sched file contains one or more schedules of the types listed earlier, and in the case of an ephemeris-driven schedule, the name of an ephemeris file. Each schedule entry specifies its type (only "COMMS" are of interest here[18]), a schedule index, an "ON" or "OFF" setting, a schedule-relative time (in seconds), the name of the peer involved and whether the entry relates to transmission ("TX") or reception ("RX").

[18] The SeNDT XML schedules referred to earlier also use other types related to use of sensors, diagnostics etc. that can in fact be changed via an SPI specifically for this purpose.

Ephemeris-driven schedule entries are a little different – in this case each entry represents a "TX ON" type, and the duration of the prime ON period is specified as well as the number of iterations to follow that ON period[19]. For each of the iterations configured, both an "RX ON" period and a "TX ON" period are added one RTT after the end of the prime "TX ON" period. The basic idea is that we configure the initial TX (or RX) ON periods we want, and subsequent periods are added in order to handle potential segment drops. This kind of schedule calculation is depicted in Figure 5.5 above.

Currently, those subsequent periods are hard-coded to be half as long as the configured period. There are of course an infinite number of variations of the ways in which such schedules can be calculated. For example, in an earlier instance of scheduling [FA03e] we calculated a "filled" schedule that attempted to include TX and RX slots of a given size wherever allowed by the visibility constraints. The current approach was chosen for its relative simplicity, and also in order to be cognizant of the LTP punctuated timers – keeping those shorter should result in faster retransmissions for lost or corrupted segments but clearly also increases the potential for un-necessary retransmissions. While it is not really a part of this thesis to attempt to optimize this facet of scheduling, one can clearly see that it is an area where more work can be done.

5.4.8 File transfer via LTP-T

Mainly for test purposes the LTP daemon also supports a file transfer mode, where the initiator can request a file from the peer. This is done as a payload protocol, where the block from the initiator contains a specially formatted request for a file. Since this is a payload protocol, it also works with LTP-T and allows us to create new test scenarios via scripting on the originator side.

In order to act as a file server the LTP daemon must be given a command line option and must be compiled specially (with LTP_DODGY_FILESERVER set). This is because the current implementation is not secure, in the sense that one would require say from an FTP server. In particular, the current code only reads a file from the current working directory,

[19] This functionality is implemented in the eph_extend() function in the sendt_cues.cc source file.

but doesn't take into account potential international character based attacks when parsing the file request[20]. Basically, the requested file name must not contain any "/" characters, but this is insufficient on some operating systems. This is acceptable since the feature is only intended for test purposes.

5.4.9 Miscellany

This section contains a number of minor, but noteworthy points.

Segment Timers and Re-try limits

In the current implementation we set a default of 10 punctuated seconds[21] for various timers, e.g., receipt of an RS in response to a checkpoint DS. Once these timers expire, the relevant segment will be re-transmitted and the timer re-initialised. Clearly there is a need for a re-try limit so as not to loop endlessly, and this is currently set to 3 re-tries for the DS/CP and RS timers, but only 1 re-try for the session cancellation timers. Once the set number of re-tries has been done, then the socket is closed.

Each of these timers may be set independently in the ltp_mib_var setting. To date we have not added a SOCKOPT for passing this control setting, but doing so would be an obvious enhancement.

Inactivity Timer

Each socket in LTPlib also has an associated inactivity timer. This timer is punctuated but is decremented whenever this node can talk to the socket's peer[22]. When no segments have

[20] For a recent example see: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-6950

[21] We have seen tests where up to 6 punctuated seconds have elapsed before a DS/CP was processed. This is most likely due to implementation inefficiencies that are more problematic during stress tests, e.g. the replay cache gets quite large in such cases. However, the overall session still succeeds in almost all cases.

[22] In fact, the current implementation could be improved here – the inactivity timer is not decremented if the socket's next-hop peer is never contactable. There should really be a second non-punctuated timer to handle cases where the contact to that peer is never re-activated. There is also a related bug where only one such timer is maintained for a relay socket – the right thing to do would be to maintain two such and only delete the socket after they have all expired.

been sent or received and no ltp_*() API has been called for the requisite duration, then the timer expires and the socket is deleted (and any buffered data is lost). If an application subsequently makes a call on this socket (e.g. a new ltp_recvfrom()) call, then it will receive an error response. This mechanism is essentially a safety-valve for cases where either disruption is too great or a fire-and-forget application has left a socket hanging and a subsequent process has loaded that socket's state from disk.

Checkpoint Setting

Similar to the above, one might wish to vary the number of CPs included in DSes on the basis of the known, or historical, loss rates for the next hop. For example, one could improve performance, potentially significantly, if additional CPs were added as some function of the loss rate seen, though probably with some exponential decay factor to eventually return the sender state to normal if losses reduce. The current implementation makes no such adjustments.

Ltpd Output files

The ltpd binary produces a few output files – a logfile (default ltpd.log) contains detailed run data but can mostly be ignored other than for debugging. When run in server mode, the ltpd.out-<<node-port>> file contains the accumulated data received from the relevant node:port combination. Note that this file is accumulated over potentially many runs. The ltpd.out-<<node-port>>.sop file contains the state-of-play (i.e. data returned thanks to LTP_SO_SOP) information, and is again accumulated. These files contain timestamps that are used for the generation of the results in the next chapter.

5.5 Summary

In this chapter we specified the LTP-T protocol at a level of detail that an implementer familiar with LTP should be able to use as a functional specification. We also justified the set of LTP extensions used by LTP-T and provided an overview of our LTP and LTP-T implementation. Finally, we discussed how LTP-T is configured and managed as well as some other implementation considerations.

Chapter 6

Evaluation

This chapter describes our evaluation of the LTP-T protocol based on the implementation described in the Chapter 5. Our goal here is, as implied by the problem statement in Section 1.4, to demonstrate the following:

(a) That LTP-T works as a DTN protocol in a set of basic tests.
(b) That LTP-T works, and that our LTP-T implementation performs as expected in a complex DTN – in this case an emulation of a Martian network.
(c) That in some scenarios, LTP-T out-performs the bundle protocol (BP).

In order to provide evidence for the first assertion, we carried out some single hop tests to compare LTP and the BP against a standard TCP application - secure FTP. Secure FTP was chosen since it allows for the same kind of file transfers as our BP and LTP/LTP-T tests. These simple tests involve various delays and are described in section 6.1.

Perhaps the poster child DTN use case is the case of landers on Mars communicating back to Earth, sometimes via Mars orbiters. As a proof-of-concept for LTP-T, we built a Martian network emulation using a set of standard computers to model Mars landers and orbiters interacting with a set of terrestrial nodes. Section 6.2 describes this network and how LTP-T performs in this context.

Finally, section 6.3 demonstrates that LTP-T can outperform the BP in a set of two-hop tests due to the fact that LTP-T can begin forwarding data segments while the BP has to wait until the bundle is fully received at the intermediary before starting to forward packets.

All source code, test setup scripts, test results, and other data related to this evaluation may be found at http://dtn.dsg.cs.tcd.ie/sft - start with the README file in that directory.

6.1 Comparing LTP-T with other Protocols

In this section we describe a set of basic tests that verify that LTP-T is a viable DTN protocol. We describe a set of single-hop tests that provide this verification.

The starting point is that basic interoperability testing for LTP, (not LTP-T), and our LTP implementation, has been successful, as described in Chapter 3. We therefore already know that LTP is a proven delay-tolerant peer-to-peer protocol, and that our LTP implementation, (LTPlib), provides a working implementation of that protocol.

For the tests described in this section, we use the time required to deliver a set of files of various sizes as our main metric. For the scenarios considered here, this is sufficient, (for those cases where the given protocols can actually operate), since the different protocols can be tested with the same file sizes and under the same, rather simple, network conditions.

6.1.1 Test Setup

In this section we describe the setup for our single-hop tests, where the DTN hop is, in this case, a UDP hop. The goal is to establish how the baseline performance characteristics of the various protocols compare and also to be able to compare protocols in environments that don't occur in our "Martian" test scenario, e.g. an LTT of 1s. The protocols compared are LTP (all green), LTP (all red), LTP (1st 1KB red), BP/UDP and, representing TCP, SFTP (i.e., FTP/SSH).

Each test run consists of 5 iterations of a file transfer, for 13 different file sizes. Test runs were repeated with 7 different latencies, over and above normal (in this case negligible) system delays. The variations are chosen to span a large range (in exponentially growing steps) but where (most of) the protocols remain operational. The imposed LTTs used are 0ms, 1ms, 10ms, 100ms, 1s, 10s and 100s. Note that the round-trip time (RTT) is 2 x LTT. The file-sizes used are 1KB, 2KB, 4KB, 8KB, 16KB, 32KB, 64KB, 128KB, 256KB, 512KB, 1MB, 2MB and, lastly 4MB. These combinations give us 5 x 13 x 7 = 455 samples per protocol, though since not all protocols work with all latencies we sometimes had fewer samples in our results, though generally around 400.

In the tests, files are sent from the sending node to the receiving node, via routers (at the IP layer), which are running the Netem module to add the relevant delays. In all cases the nodes are "always-on," that is, there is no additional disruption. (Our use of the Netem module is described further in Section 6.2 below).

6.1.2 UDP Rate Control

All of the tests described below, (including those in Sections 6.2 and 6.3), involve LTP or the BP running over UDP. Transmitting UDP packets too frequently results in packets being dropped by the underlying IP network, so there is a need for some rate control in order to ensure successful packet delivery. For this reason, both the BP/UDP convergence layer and our LTP implementation include code to ensure that we do not send sets of UDP packets too quickly. Since the operating system for the test platform is not a real-time operating system; the minimum delay we can impose between packet transmissions may vary between 20 and 40ms (determined from logs of these tests).

This rate control scheme has two effects: it determines the maximum throughput for the various tests but it also ensures fairness in the comparisons between the BP/UDP and LTP/UDP implementations. For example, since UDP packets will only be transmitted (roughly) every 20ms, a 4MB transfer requires almost an entire minute, just for transmission, (i.e., regardless of the latency of the link). Though very slow, this is acceptable for our purposes, as the same rate-control code is used in the both BP/UDP and LTP code, so that comparisons remain fair.

6.1.3 Software Versions

In order to carry out the comparative evaluations described below we had to do some work on the BP implementation. We started from the dtn-2.3.0 release[1] (dated December 2006) which, at that time, did not support bundle fragmentation. In order to be able to test with large bundles (our test scripts use files of up to 4MB), we had to modify the UDP CL to support fragmentation at the UDP layer. This change also included the UDP rate control

[1] http://www.dtnrg.org/docs/code/dtn_2.3.0.tgz

scheme described above, and also used the same MTU size as used by our LTP code.[2] The LTP/LTP-T code used here is as described in Chapter 5 and was checked to our CVS repository on May 1st 2007. The version string reported by the "sftp –v" command is "OpenSSH_4.2p1 Debian-7ubuntu3.1, OpenSSL 0.9.8a 11 Oct 2005".

6.1.4 Test Results

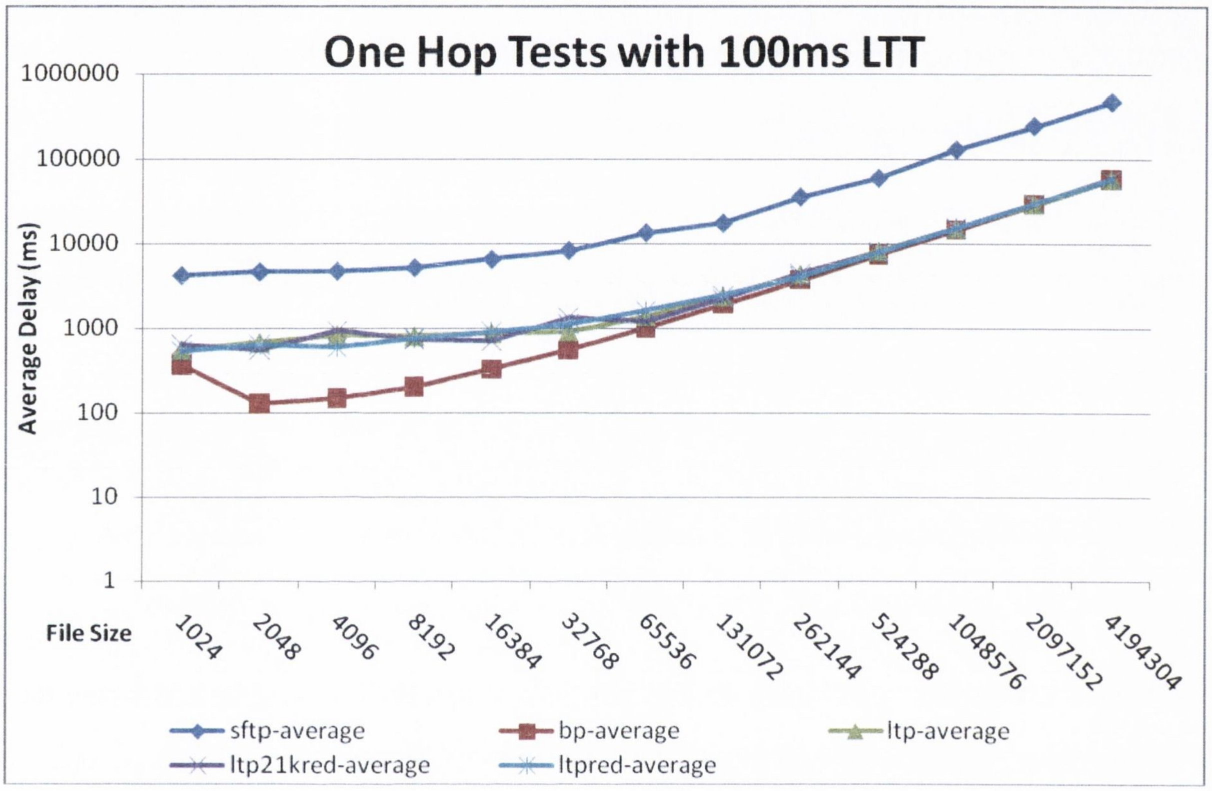


Figure 6.1 – One Hop Tests with 100ms LTT.

Figure 6.1 above shows the results for a comparison of the various protocols with a 100ms LTT. Each point represents the average of five runs of the relevant protocol. The results for other LTT values are presented in Appendix A. The standard deviations are relatively large for the BP and LTP protocols for small file sizes, (and hence clutter the plots), but are

[2] The change required is described in the dtn-users mail archive: http://mailman.dtnrg.org/pipermail/dtn-users/2007-March/000553.html

small relative to overall delay for larger file sizes and LTT (and hence invisible). For this reason, we only include error bars, (representing one standard deviation), for the data series in Figure A.1.

6.1.5 Discussion

The main things to note from Figure 6.1 and the graphs in Appendix A, are that SFTP fails shortly after 1s LTT, and so is not shown in Figure A.6, and that the BP and LTP perform almost identically for larger file sizes (i.e. over most of the range of tests).

The first two graphs (Figures A.1 and A.2) show that TCP outperforms the BP and LTP, especially when file sizes get bigger and TCP leaves slow start. This is however partly due to the UDP rate control scheme described above. At 10ms LTT TCP's advantage is diminishing, so that it is effectively unusable at 100ms LTT.

By 10s LTT, TCP is timing out, in this case due to a timer in the SSH daemon ("`LoginGraceTime`") that has a default setting of 120s. With the TCP handshake, and a few cryptographic and login RTTs, this limit is reached at the 10s LTT mark.

By 100ms LTT (200ms RTT) both the BP and LTP are outperforming TCP. This is consistent with the findings in another recent LTP-related PhD thesis [RA07t] where the author estimated that at around 50ms LTT, LTP will begin to outperform TCP.

The graphs for the BP and LTP options are very similar at higher latencies and for larger file sizes. At 100s LTT, the backlog in the IP router running the Netem module, becomes such that the router begins to drop packets and the unreliable variants are affected. In the very last figure (Figure A.6) we only include the BP and LTP (all green) tests and can see that they continue to perform as expected for the delay involved.

Our conclusion from this set of tests is that, as expected, both the BP and LTP outperform TCP at higher latencies, but that there is, so far, little to choose between LTP and the BP in terms of performance for larger files sent at higher latencies.

6.2 Martian Network Emulation

We now come to the second claim above: that our LTP-T implementation performs as expected in a complex DTN – in this case, an emulation of a Martian network.

Our Martian network emulates communications between the two Mars Exploration Rovers (MER), Spirit (MER-A) and Opportunity (MER-B); three Mars orbiters - Mars Global Surveyor (MGS), Mars Odyssey (ODY) and Mars Express (MEX); three Deep Space Network (DSN) Earth stations; Canberra (CAN), Goldstone (GOLD) and Madrid (MADR), and, as Internet endpoints: a DSN gateway and a set of three Principal Investigator (PI) nodes. The emulation uses one host for the landers, one for the orbiters and one for the terrestrial nodes. In addition, we also use two hosts, running Netem, to model the delay between Earth and Mars (e2m) and vice versa (m2e). The emulation runs using realistic contact information provided by NASA JPL[3] that reflects the situation for 45 days on and around Mars in early 2004, just after the Spirit and Opportunity rovers had landed.

6.2.1 Why Emulation?

As we cannot (currently) carry out tests on real spacecraft actually on or orbiting Mars, we must use some form of simulation of the deep space environment. There are however, two different choices here – one is to simulate the deep-space network using a network or space simulator, the other being to emulate the network, by setting up a network involving standard hosts running IP but with artificially managed delays between the hosts. Our conclusion is that the latter is preferable to the former. But in order to explain this we first examine the options. Table 6.1 lists the options that were considered for this evaluation.

We actually first planned to use SeNDT sensor nodes as the basis for evaluation. However, we revised this plan for a couple of reasons. The first was simple pragmatism – most of the operational SeNDT node hardware was being used for a noise monitoring pilot at the time

[3] Thanks to Charles Lee, Chad Edwards, Scott Burleigh, Leigh Torgerson and Adrian Hooke all of NASA.

Option	Comment
SeNDT [FA06b]	Instead of modeling an Earth-Mars system we could have used a network of SeNDT sensor nodes to compare use of the protocols on sensor nodes.
STK [MO00]	The "classic" space simulator, excellent spacecraft and ephemeris models, but no way to model network or higher layer protocols.
ns-2 [FA97]	The "classic" network protocol simulator. Excellent coverage in terms of protocols simulated. No standard way to model ephemeris.
OMNET++ [VA99]	Another, more modern, network protocol simulator, but with fewer protocols simulated and less well known. No standard way to model ephemeris.
DTN-VNUML [FE04]	Virtual Network User Mode Linux is yet another alternative whose scripting can allow simulation of quite interesting network scenarios. Again link delays are not well supported.
Netem [HE05]	Linux kernel module that allows one to add delays to a router. This is part of some standard distributions (e.g. Ubuntu). No ephemeris handling per se, but see below.

Table 6.1 – Options for Testing.

that the protocol implementations were ready for evaluation. The second reason is however, more telling in the end. If the SeNDT nodes were actually in the laboratory for tests, then the fact that we're running on an embedded processor running arm-linux doesn't really make any difference since we are, in this case, only emulating a network in any case. And since running repetitive tests in the field would be prohibitively expensive, we moved away from that plan.

Next we considered a space simulator, in particular the Satellite Took Kit[4] (STK), which is the pre-eminent space simulator today, roughly in the place where ns-2 is in network simulation. However, STK has been found not to have an interface that is usable for network simulation or emulation [BR06]. We had planned however to use STK to generate ephemeris data for our test scenarios, but that was no longer necessary once we had access to the NASA JPL contact data (described below) for our emulated network.

[4] http://www.stk.com/

The next option therefore was to use a network simulator. The obvious choice here would be either ns-2, or maybe less obviously, OMNET++. We didn't actually make any use of ns-2 since, during development, we routinely used both Linux and Cygwin platforms, and, at that time, ns-2 wasn't portable to the Cygwin environment. We did carry out some initial simulations based on extending OMNET++, (which does run on both development platforms), to support ephemeris driven delays, [FA03e] and while those were promising we ultimately rejected the entire idea of using a network simulator for reasons given below. Though there are a range of other open-source and commercial network simulators we didn't find any that were affordable and seemed like they would be significantly better than the two above.

There were three reasons for our rejection of network simulation in this case. Our first, though weakest, argument is that we share some of the skepticism [PA03, KU05] as to the usefulness and fidelity of network simulations, especially for a network setup like a DTN which is significantly different from typical simulated, or real, network settings. While the developer of the simulation undoubtedly gains insight from the work, our feeling is that readers of the results gain much less insight and, in fact, often appear to have trouble generating commensurate results. This can be due to issues with differing levels of detail, e.g., in wireless networks [HE01d] where different simulations embody different decisions as to what is interesting to represent faithfully. Simulation results also need to be presented so that they can be validated, something that isn't always the case. [HE01]

From our own work with OMNET++, one possibly general problem with network simulators may be a sort of "hidden variables" problem. In the simulation performed for one paper, [FA03e] the OMNET++ configuration file contained 174 separate numeric and/or string settings.[5] With so many fairly opaque settings, it is no wonder that simulation results can be hard to replicate. This problem is also discussed by Kurkowski et al. [KU05] While we do not make any claim that this argument represents a consensus within the networking community, our other arguments are stronger.

[5] The source for that simulation is available at: http://dtn.dsg.cs.tcd.ie/eslab37/

The second argument against using a network simulator for this work is that there are difficulties in using the same implementation for a simulation and in a real network stack. As part of our work on LTP, however, one our goals was to achieve interoperability with the University of Ohio Java implementation[6] and another was to be able to use LTP and/or LTP-T in SeNDT sensor nodes, so an implementation that was tied into a network simulator was not suitable for that aspect of the work.

In other words, we could only validate the basic protocol design via interoperating two different implementations, and that cannot readily be achieved via simulations. For example, Kalyan et al. [KA02] report that it was "extremely time consuming" to investigate TCP interoperability issues when using different network simulators, even though the relevant TCP models (in ns-2 and GloMoSim) had previously been "validated in separate projects" to theirs. With new protocols (like the BP and LTP/LTP-T) the situation would be worse and we would have given up the benefit of basing our tests on implementations that had successfully been interoperated.

Lastly, the test setups described in this chapter involve not just one, but two or more protocols (sftp, LTP, LTP-T, BP) being tested in sequence. While it might have been possible to implement (or re-implement) LTP-T in a network simulator, re-implementing an equivalent to the BP reference implementation would have represented significant effort and would have required yet another round of testing to validate the putative BP simulation against the BP reference code.

Another alternative would have been to use DTN protocol implementations running on a Virtual Network User Mode Linux (VNUML[7]). VNUML is essentially a way to simulate a network with a set of user-mode linux (UML) kernels running on a single host, and is typically used to simulate networks as part of Honeynet projects aimed at determining the behaviour of various forms of malware or bad actors. [GA06] Although DTN simulation in

[6] http://irg.cs.ohiou.edu/ocp/ltp.html

[7] http://mailman.dtnrg.org/pipermail/dtn-users/2006-June/000265.html

this context is, in itself, quite an interesting concept,[8] it is perhaps better suited to simulation of disrupted networks, rather than delayed links, since, to our knowledge, there is no obvious support for simulating large delays in such a virtual network.

In our case, we can afford the relatively modest hardware requirements that are needed to emulate the latencies involved while at the same time using real network stacks. So our final plan was to emulate the network, based on the Netem module[9] [HE05] that is part of some Linux 2.6 based kernel distributions – which can be done at little cost. Netem is designed to allow application and protocol designers to emulate wide area network (WAN) properties (e.g., delay and loss) in local area networks (LAN). Netem is useful because many WAN properties are simply not seen in LANs, which are quicker and more reliable and generally not subject to the same kinds of disruption. Netem is normally run on a separate host acting as an IP router that imposes WAN level delays (say around 100ms) and packet losses that wouldn't be seen on a LAN. While Netem is not specifically intended for emulating DTN scale latencies, in fact, 20 minute latencies imposed using Netem work well as is shown in the trace in Figure 6.3 below.

Details of our Netem setup for these experiments are specified below, but for now we want to consider why network emulation is a better approach. In the first place, emulations, while not perfect, are inherently better in respect of both of the main problems with simulations described above. With emulation, we get to use a real protocol implementation so that it can be interoperated and different combinations of protocols can be tested over the same wires. We also have less of a "hidden variables," problem, since the network traffic is, by definition, externally visible and can be recorded and monitored etc. which makes replicating/checking results much easier.

The network emulation scheme does of course have some issues, perhaps the main one being the lack of serendipitous problems. In a real network, rain will attenuate wireless

[8] The fact that the simulator has to hide its nature from the application is independently interesting.

[9] http://www.linux-foundation.org/en/Net:Netem

signals, operator mistakes will be made and other types of disruption will occur that will not necessarily occur during emulation. However, for our current purposes, (essentially LTP-T as a proof-of-concept and BP/LTP-T comparisons), modeling such events is not required since those events won't usefully differentiate between the protocols at their current stage of development.

There are however, a few issues with network emulation where the model is inaccurate. The MTU size and throughput of the emulated network have to work on the lower layers in use – in our case we have to live with an MTU size of 1500 bytes and fixed 100Mbps links (due to the Ethernet hardware we are using). While this does represent a departure from a true Earth-Mars network, it is a) fair to the various protocols, and b) quite reasonable, as shown below.

Firstly, one can envisage an Earth-Mars link approaching 100Mbps via the use of optical communications, with one study [BI03] forecasting data rates of 4-40Mbps, and while the variability in Earth-Mars distance does affect bandwidth, it is not likely to do so over the duration of a single LTP-T session or BP exchange.

Secondly, the Ethernet MTU isn't in itself unreasonable, since many of the proximity protocols [CC06p] used between spacecraft and orbiters use an MTU of roughly 1000 bytes.[10] For deep-space link protocols, like the telemetry space data link protocol, [CC03] the MTU can be a mission phase parameter, being set differently depending on other mission priorities, and so is harder to classify. However, the 1500 byte MTU is not very different from the current practice in deep-space communications.

So, our conclusion is that the best thing we can do is to evaluate LTP-T and other protocols using a network emulation scheme.

---

[10] Personal communication from Scott Burleigh of NASA JPL.

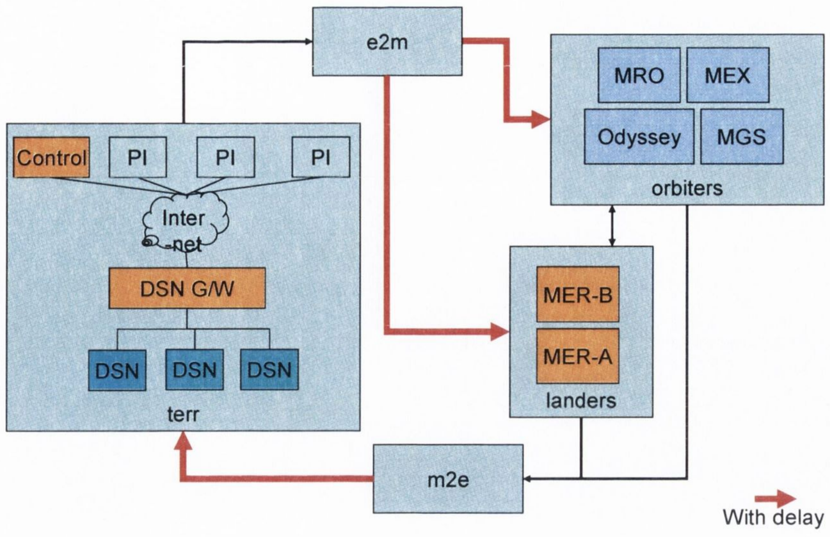


Figure 6.2 – The Martian emulation setup.

6.2.2 Hardware and Software

The hardware platforms used consisted of five Dell Latitude laptops (two model D410, three D300), each running a fresh installation of Ubuntu (version 6.10 server edition). The D400 systems have 1Ghz Pentium III processors, with 256 MB of RAM. The D410s have Pentium M processors running at 2GHz and have 512MB of RAM. All network traffic is over 100Mbps Ethernet. A single 8-port Ethernet hub (3Com OfficeConnect Dual Speed Switch 8 plus) connects the five hosts. The hub is also connected to the campus network, eventually via a switched port, so there is little extraneous traffic affecting the overall setup.

Figure 6.2 presents a network diagram for the setup we use, showing which hosts emulate which nodes. Essentially we allocate a specific host:port combination for each node in our emulation, e.g., the host representing MER-A (Spirit), listens for UDP traffic on the landers host at port 1301.[11] In addition to the DTN nodes, there are two additional hosts, e2m and m2e that are IP routers running Netem configured as described below to emulate the Earth/Mars and Mars/Earth latencies. Note that the landers-to-orbiters link in Figure

[11] For LTP and LTP-T the ltp.names file contains the mappings from strings to host:port combinations.

6.2 is not shown as delayed, but is run using an LTP ephemeris driven schedule (with a nominal 50ms delay for its LTP timers) since the link is disrupted.

All of the configuration settings, scripts and other instructions required to reproduce this and all other test setups are present at the location given above, in a directory called "test-setup". The test-setup/box-setup directory contains details of how the boxes were configured, e.g. specifying how NTP and SSH are setup. There are a few other details needed in order to make the setup repeatable, e.g. we have to set up IP routes so that packets from terr destined for landers go via e2m etc., and we also wish to be able to SSH into any of the boxes from outside the emulation without having that traffic delayed.

The LTP/LTP-T code used here is of course the LTPlib described in Chapter 5 with minor updates and bug-fixes added during testing, so this version is more recent, (and more stable), than that used for the tests described in Section 6.1.

6.2.3 DTN Goodput

The metric we use for these tests is goodput, [FL99] which essentially measures how many application layer bytes are successfully sent end-to-end, but taking into account the communications constraints inherent in the test scenarios. For example, periods when communications are impossible are not counted when calculating goodput.

The reasons we use goodput are as follows:

(a) It allows for fair comparisons between the various protocols and test scenarios.
(b) Packet delivery ratios, which have been used in other DTN tests, (e.g. [CH06]) are not relevant here, since in all our tests, we ensure reliability (or partial reliability) and all packets, (that are supposed to be received), are received.
(c) Bandwidth utilization (e.g. as used in [DE04]) is not used, since we are not attempting to "fill" a pipe here, but rather to produce a fair comparison of different protocols under the same conditions. We essentially ensure that each protocol makes use of the same bandwidth (see the section on UDP rate control above).
(d) DTN protocols and test scenarios are still at a relatively early stage in their development, so metrics related to fairness in the presence of other flows, routing and other factors, are for future study.

To quote a basic example of a goodput calculation: "Imagine that a file is being transferred using HTTP over a switched Ethernet connection with a total channel capacity of 100 megabits per second. The file cannot be transferred over Ethernet as a single contiguous stream, instead it must be broken down into individual segments, called packets. These packets must be no larger than the maximum transmission unit of Ethernet, which is typically 1500 bytes. Each packet requires 20 bytes of IP header information and 20 bytes of TCP header information, so only 1460 bytes are available per packet for the file transfer data itself. Furthermore, the packets are transmitted over Ethernet in a frame which imposes a 38 byte overhead per packet. Given these overheads, the maximum goodput is 1460/1538 * 100 Mbps which is 94.92 megabits per second or 11.866 megabytes per second." (quoted from Wikipedia[12]) In a DTN where connectivity is only available 50% of the time, the maximum goodput would be halved, i.e. 5.833 megabytes per second.

The DTN goodput is derived from goodput as defined above as the number of application bytes transferred, divided by the connected time, where the connected time is the time during which the node in question could usefully transmit data, so:

$$G = B / (E - D) \qquad \text{(Eq. 6.1)}$$

Where G is the DTN goodput, B is the number of payload bits (or bytes) transferred (i.e. not counting any headers such as LTP-T's "via" extension), E represents the elapsed time, and D is the "disconnected" time, representing the accumulated delay and disruption seen on the entire path.

We do include in this goodput calculation the time used by acknowledgements or the equivalent, and only consider a packet to have been delivered after whatever acknowledgments concerned have arrived. In the case of LTP for example, the time for the final report segment (RS) to arrive is included in the elapsed time for the goodput

[12] http://en.wikipedia.org/wiki/goodput - Quoted text correct as of 2008-09-15

calculation. As a consequence the DTN goodput for LTP sessions will depend to an extent on the red/green split, the block and segment sizing, the use of various optional headers (e.g. "via" and authentication related), the LTT and, of course, the error rate experienced.

6.2.4 DTN Goodput in the Martian Emulation

The actual calculation of DTN goodput for Martian LTP-T test runs requires some additional work to calculate the overall delay for the transfer. In our Martian emulation the delays involved are less easily determined due to the complexity inherent in handling the mixtures of LTT, ephemeris-driven visibility and scheduling explained below. For these tests, we use a scheme that derives the goodput from log-files generated during test runs.

For our DTN goodput calculation we extract all the sending and receipt times of LTP segments from the log files for the relevant LTP/LTP-T session. The session duration is the elapsed time between the first segment transmission time and the final segment receipt time. Note that the final segment receipt time can occur long after the payload bytes have been successfully delivered to the destination. For example, with a red file being sent from Mars to Earth, the DSN station that receives the file will have successfully transferred the entire file to its (terrestrial) destination before any LTP report segment has arrived back at Mars. In such a case, the last segment to be received will be the final report acknowledgement from Mars to the DSN station.

There are three different sources of delay in our Martian emulation – light trip time (LTT), visibility and scheduling delays. LTT and visibility are known, but scheduling delays are much less so, since the schedules must allow for a node to listen for a protocol response at more than one future contact. For example, if a DSN station sends a red EOB segment to Mars then it should be listening for a report segment two LTTs later, but since that report may be lost en-route, it must also schedule a contact for four LTTs later, which is when a re-transmitted report might arrive. And of course, all of those contacts must fit with the overall visibility of the nodes in question, so the actual secondary contact may occur significantly more that four LTTs later.

With 10 nodes operating schedules in our emulation, and with all but one of those schedules allowing for such re-transmission windows, within visibility constraints, calculating the expected delay in advance would be cumbersome. Once we allow packets to be randomly dropped en-route, it becomes impossible.

So, for our DTN goodput calculation, we scan the list of segment times (from the accumulated log files), and consider any period longer than a threshold during which no message transmission occurs, as being a “gap.” We use a threshold of 40ms since our UDP rate control (see below) ensures that we don’t send segments closer than 20ms apart, but due to operating system uncertainties, some segments will actually be transmitted more than 30ms apart. In tests, thresholds between 40ms and 100ms all produce the same DTN goodput output figures. (And a 40ms threshold also maps reasonably well to latencies seen on well-connected Internet paths.)

We then add up durations of all these gaps and consider the sum of the gaps to be the overall delay for the session. The usable time is then the overall session duration minus the accumulated delay for the session. The result is that our overall DTN goodput calculation is as specified in Equation 6.1.

6.2.5 Effect of UDP Rate Control on DTN Goodput

In this section we analyse the effect of our UDP rate control scheme, coupled with typical LTP-T overheads, on DTN goodput. Table 6.2 shows typical sizes for the various fields in an LTP-T segment, as used in some of our tests. Note that our LTP implementation is not currently optimized to properly fill an entire 1500 byte MTU, hence the odd overall packet sizes. In the table, the sizes for the via field are shown for segments as sent by the initial client (where only the via tag is sent), and for the next three hops – the via field increases in size at each hop as arrival and destination times are added.

Table 6.3 shows the effect this has on the maximum DTN goodput for error-free paths of 1, 2, 3 and 4 hops, with packets being transmitted every 20, 30, 40 or 50ms. Essentially, this varies from 66KB/s, which should be provided for a single-hop LTP-T session running on a single host with no delays imposed, down to ~10KB/s for a 4-hop path where operating system, network and/or implementation inefficiencies cause packets to be sent at a slower rate and thus lead to lower measured goodput values.

Field	**Size**			
LTP header fields	17			
LTP-T extensions	31	159	285	409
Source	8			
Destination	8			
Port	4			
Estimated Block Size	6			
Hop count	4			
Via (at hops 0,1,2,3)	1	129	255	379
Payload	1352	1265	1139	1013
Total	1400	1441	1441	1443

Table 6.2 – Typical Field Sizes in an LTP-T Segment.

Hops	**1**	**2**	**3**	**4**
Payload	1352	1265	1139	1013
goodput(KB) @ 1 packet / 20ms	66	61	55	49
goodput(KB) @ 1 packet / 30ms	44	41	37	32
goodput(KB) @ 1 packet / 40ms	33	30	27	24
goodput(KB) @ 1 packet/50ms	27	25	23	20

Table 6.3 – Expected Maximum DTN goodput.

In order to validate our LTP/LTP-T implementation versus the above DTN goodput analysis we performed some tests involving a single hop with no delays, and running on a single host. Appendix B.1 shows a pair of graphs from this test, demonstrating that the implementation achieved the predicted goodput figures when running with a schedule file where communications are turned "on" for two seconds from each ten second period. The goodput figures for the two runs shown were 62983.2 B/s and 65331.8 B/s respectively.

6.2.6 Emulating Delays

For the tests in Section 6.1, we only needed to be able to emulate a fixed delay (e.g. 10s), which can be easily done with Netem. However, for our Martian emulation, we need to use

```
stephen@TerrestrialInternet:~$ date
Tue Apr 24 19:06:51 IST 2007
stephen@TerrestrialInternet:~$ ping -i 60 landers
PING landers.dsg.cs.tcd.ie (134.226.36.242) 56(84) bytes of data.
64 bytes from landers.dsg.cs.tcd.ie (134.226.36.242): icmp_seq=1 ttl=63 time=1471369 ms
64 bytes from landers.dsg.cs.tcd.ie (134.226.36.242): icmp_seq=2 ttl=63 time=1471356 ms
64 bytes from landers.dsg.cs.tcd.ie (134.226.36.242): icmp_seq=3 ttl=63 time=1471354 ms
64 bytes from landers.dsg.cs.tcd.ie (134.226.36.242): icmp_seq=4 ttl=63 time=1471363 ms
64 bytes from landers.dsg.cs.tcd.ie (134.226.36.242): icmp_seq=5 ttl=63 time=1471352 ms

--- landers.dsg.cs.tcd.ie ping statistics ---
29 packets transmitted, 5 received, 82% packet loss, time 1680146ms
rtt min/avg/max/mdev = 1471352.260/1471359.404/1471369.808/948676.694 ms, pipe 25
stephen@TerrestrialInternet:~$ date
Tue Apr 24 19:35:52 IST 2007
stephen@TerrestrialInternet:~$
```

Figure 6.3 – Pinging with an Earth/Mars LTT.

a realistic LTT for Earth/Mars communications. The NASA JPL "Horizons" web site[13] provides a web-based interface where the Earth/Mars LTT can be retrieved for specified durations and at specified granularity. For our purposes we used this tool (and a little processing) to create a file ("deacde-2h.vec") that specifies the Earth/Mars LTTs with a two hour time step for the entire first decade of the 21st century.

Figure 6.2 above shows how the various machines are allocated responsibility for the various nodes in our emulation. Two of those hosts (e2m and m2e) act as IP routers, using the Netem module in order to enforce link delays. Those delayed links are shown as heavier lines in Figure 6.2. When a packet is sent from, e.g., terr to landers via e2m, the packet arrives normally at e2m (thanks to IP routes setup for that purpose) but is then delayed, at e2m, for the relevant LTT (by Netem) before finally being forwarded onto the landers host. Traffic in reverse is also appropriately delayed, by the m2e host.

With this setup, a ping from terr to landers can take for example 40 minutes to complete, but it does complete. Figure 6.3 shows the output of a ping from landers to terr, with an Earth/Mars scale LTT (in this case about 745 seconds). This is a nice demonstration that IP is, in fact, delay-tolerant, even though TCP is not.

[13] http://ssd.jpl.nasa.gov/horizons.cgi

All that remained was to find a way to set the correct LTT on the e2m and m2e hosts. This is done via an hourly cron job (since the LTT variation within one hour can be ignored) which runs a script that reads the decade-2h.vec file to produce the correct result for the current time. As the Netem command line interface is quite awkward to use, we developed scripts that wrap the Netem command line ("tc"), hide the details of the LTT file reading and set the desired LTT.

There are actually two different "desired" LTT settings that we may want to use – one is the LTT for the current moment in real time described above. For our emulation however, we want an LTT from within our test epoch. In this case, we simply deal with time "modulo" 45 days, starting counting from the epoch (in our case that is Jan 25th 2004). This last case matches the behaviour of the other nodes in the emulation which also deal with time "modulo" the 45 days in question via LTPlib scheduling.

One side-effect of using cron like this is that it makes it simple to switch off delays. Together with the fact that LTPlib reverts to "always-on" mode when there is no ltp.sched file present, this provides us with a nice way to turn off both delay and disruption – something that was very useful for debugging at many stages of development.

The decade-2h.vec file also becomes a part of our schedules, for example, being quoted from inside the various visibility-configuration files which are in turn referenced from the ltp.sched schedule file. We use a similar file decade-2h-50ms.vec for schedules between landers and orbiters. As the name implies, this imposes only a fixed 50ms delay regardless of the time of the contact.

6.2.7 Spacecraft Visibility Information

The contact information supplied by NASA JPL consisted of a set of files, each of which represents durations during which pairs of spacecraft, or spacecraft and DSN nodes, can communicate. Essentially, this information represents the pair-wise visibility of the spacecraft and DSN nodes involved. The data were produced using the Telecom Orbit and Analysis Tool (TOAST) [LE06] that is used to generate and evaluate orbits for telecommunications orbiters.

There is, or course, a certain irony here, TOAST is itself a simulator, though not one that was considered above, since it is not widely available. We can however, justify the use of this simulation output on a number of grounds.

First, it represents visibility which is largely determined by physics, and secondly it is a tool developed by the owners of the spacecraft concerned – so they have plenty of motivation for the outputs to be a good basis for further work; and thirdly, TOAST is not a network simulator, but rather an STK-like simulator but just more specialized – and our original plan was to use STK to generate ephemeris data – in this case, we just got one step closer to the spacecraft hardware than we envisaged.

The files from TOAST have the following parameters: a duration spanning 2004-025T00:00:00 until 2004-070T00:00:00; a step size of 1 minute (totaling 64,801 steps); DSN mask angle and Mars surface mask angle both of 10^o (the mask angle is the angle above which the object must be before being considered visible).

Two slightly differing file formats were used in the data supplied. For Mars to DSN communications, there was one file per spacecraft, with each line representing the visibility situation for one minute and containing a "1" when the spacecraft and DSN station are mutually visible, so for example, the 447th line of the MERA2CanGoldMadr.txt file supplied by NASA has the value "1 0 0" representing the fact that the Canberra DSN station becomes directly visible to MER-A 447 minutes after the start of our epoch.

The lander/orbiter files have a slightly different format, due to the fact that the windows of visibility between lander and orbiter are both much shorter and more variable than those that span the Earth/Mars distance. In the case of lander/orbiter visibility, the duration of the contact will be a few minutes, with significantly different data rates seen when comparing the early and late parts of the contact with those in the middle. For this reason, the file format specifies the data rate for the minute concerned so our sample record (the 240th) from the MERA2MGSODYMEXUHFRATE.txt file is "128 0 0" representing the fact that this MER-A/MGS contact supports a data rate of 128 Kbps.

As can be seen these file formats involve one line per minute, that involves 64,801 lines per file for the 45-day duration. Clearly, that much file processing isn't ideal for running anything like a stress-test, and in fact, that file format contains a lot of redundancy. So, for the purposes of this work, we reduced these "raw" formats, to our LTPlib schedule format where records contain relative-time specifiers and each is a delta from the last.

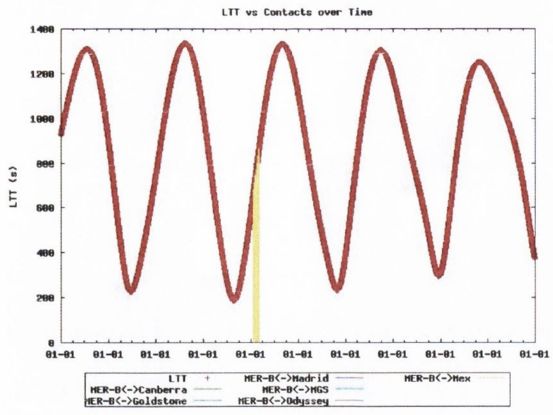
This reduces the MER-A to DSN file from 64801 to 233 lines. In this file format each line specifies a time (in seconds) and a "1" for visible, and a "0" for not visible, a set of conditions that apply until the time specified in the next line. The initial entry in the file names the nodes involved, specifying the name of the "from" node and a set of "to" nodes. For example, the 2nd line of the MERA2CanbGoldMadr.vis file is "VIS,26760,1,1,0,0" which means that 26,760 seconds from the epoch, the 2nd node is visible from the 1st but the 3rd and 4th nodes are not.

6.2.8 Communications Schedules

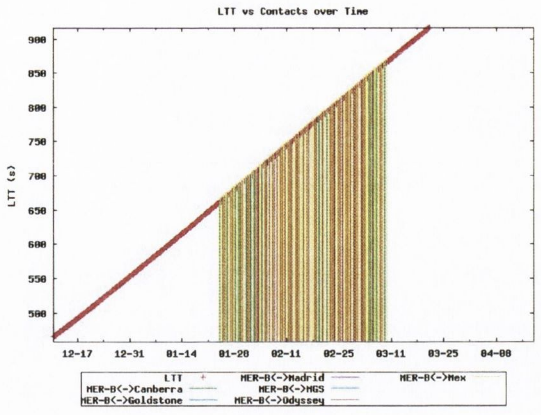
Communications with Mars are largely determined by LTT. Figure 6.4(a) shows a view of the LTT between Earth and Mars for the decade following 2000-01-01T12:00:00Z with our 45 day test epoch shown as a thick vertical line. Figure 6.4(b) is a closer view that shows the variation in LTT over our test epoch, from roughly 660 seconds at the start of the epoch, increasing to roughly 870 seconds at the end of the 45-day period.

Figure 6.4(c) shows the view for 5 days (from Feb 7-12 2004) – individual periods of visibility are now visible and are shown as coloured boxes lasting for the relevant duration (the heights of the boxes are offset from the LTT for clarity). Even at this level, one can see that the lander/DSN periods of visibility are much longer than lander/orbiter periods of visibility. Lastly, Figure 6.4(d) shows the visibility periods for one day (Feb 6th 2004), showing that on that day, MER-B could see MarsExpress three different times, Odyssey twice, and had extended duration visibility of the Goldstone and Madrid DSN stations.

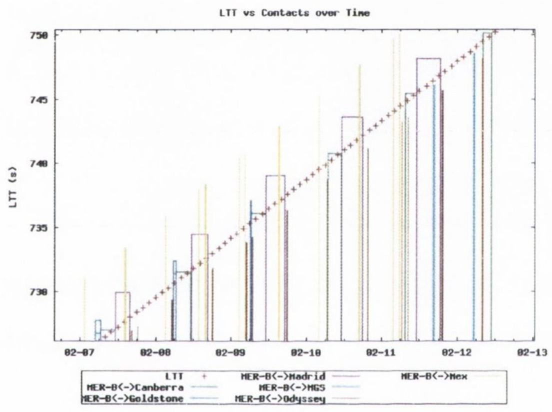
Of course, these periods of visibility cannot be fully utilized by the real MER-B, since additional considerations related to power management and daylight, scientific tasking (of all of the nodes, not just the lander), and general scheduling constraints, have to be taken into account – the DSN stations could not allocate all of this time to MER-B for the full 45 day period.



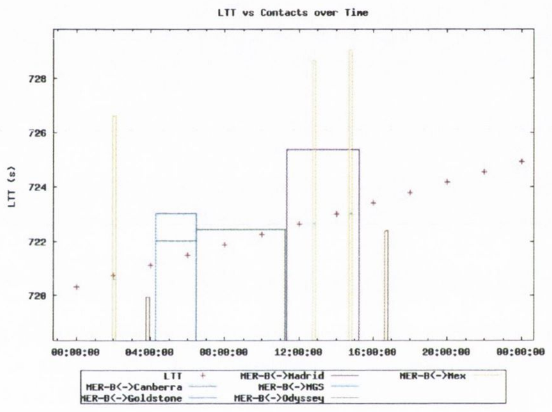
(a) Decade long View



(b) Three month view



(c) One week view



(d) Singe day view (Feb 6th)

Figure 6.4 – Views of MER-B Communications.

So, we need to overlay a communication's schedule of some sort on top of the visibility constraints. However, we do not have access to real scheduling data for this, since it is not easily available from the mission archives, though it could presumably be derived from DSN station schedules and mission image archives.

For the purposes of our emulation, we overlaid a communications schedule that called for a 30 second "primary" contact for each node, each hour (at 5 minutes past the hour), with diminishing-duration secondary contacts scheduled for 2 LTTs later, 4 LTTs later, etc. up to 10 LTTs after the primary contact. These secondary contacts allow for re-transmissions and acknowledgements, as stated earlier. While this schedule nominally creates a contact at five minutes past each hour, in fact all of the contacts are time-shifted so as to fit within the next relevant (and sufficiently long) visibility period.

While this scheduling scheme is certainly ad-hoc, it is sufficient for our purposes in that it creates primary contacts that are long enough that an entire block can be transferred in a single contact, but that are also sufficiently short that we can also test sending blocks that require more than one contact to complete. (An example of such an LTP-T session is shown in Appendix B.2.2.) It also means that tests can be run at any time during the epoch, and will complete, though requiring an uncertain duration.

Developing more realistic scheduling would have required us to include a model of application-layer traffic, which is essentially driven by scientific activity, and also to deal with schedule updates and their distribution, both of which are interesting aspects of a DTN, but are considered out of scope for now, since they are not required in order to establish our basic claim that LTP-T can work in a complex DTN environment.

Lastly, one should note that Figure 6.4 only shows the situation for one of the ten nodes in our emulation that have to deal with such schedules; however the overall level of scheduling complexity involved should be clear.

6.2.9 DSN Gateway Schedule

The LTP-T routes used for our tests are mostly obvious from Figure 6.2, for example, landers can send to Earth either via one of the orbiters or else via one of the DSN nodes. There is, however, an issue for the PI nodes - when they are originating traffic, they don't know which Earth station is best positioned to make an early contact with Mars, so the DSN Gateway (dsn-gw) node is used to solve this problem.

The dsn-gw node has its own schedule, (also generated from the JPL Horizons site), that reflects the visibility of Mars from the Earth stations during the period of the emulation. So, for example, the dsn-gw will forward to the Goldstone Earth station between 20040126T20:00:00 and 20040127T06:00:00 (emulated time), when Mars is visible (and more than 15 degrees above the local horizon). This dsn-gw schedule essentially overlaps with the orbiter and lander-specific schedules of the individual Earth stations.

```
PP: Pretty printed LTP message (at 2007-04-27 20:17:16.072).
PP: -----
PP: message type: Data segment                           [4]
PP: peer (session originator): PP: IP: 134.226.36.248:1404      [0xA1DCA2A7E08A7C]
PP: sno (session number): 0x0E764633                     [0xF3D98C33]
PP: extensions: 6 headers, 0 trailers                    [0x60]
PP:     ext 0: source: 134.226.36.242:1301
PP:     ext 1: dest: 134.226.36.248:1403
PP:     ext 2: port: 0x0D05
PP:     ext 3: est. block size: 524288
PP:     ext 4: hop count: 2
PP:     ext 5: via: (379)
VIA: 134.226.36.244:1102 at 2007-04-27 20:04:46.513 (arrival)
VIA: 134.226.36.244:1102 at 2007-04-27 20:04:46.514 (departure)
VIA: 134.226.36.248:1407 at 2007-04-27 20:17:15.680 (arrival)
VIA: 134.226.36.248:1407 at 2007-04-27 20:17:15.687 (departure)
VIA: 134.226.36.248:1404 at 2007-04-27 20:17:15.977 (arrival)
VIA: 134.226.36.248:1404 at 2007-04-27 20:17:15.978 (departure)
PP: csi (client service ID): 0x044E                      [0x884E]
PP: offset:  00000 (0x0000)                              [0x00]
PP: data length: 01013 (0x03f5)                          [0x8775]
PP: data (1st 4 bytes or less): 0x30313233...
PP:
PP: -----
```

Figure 6.5 – First Data Segment from an LTP-T session (on arrival).

While this scheme is somewhat artificial it does reflect a real issue with any DTN routing scheme – how much knowledge to represent at which nodes. In this case, the scheme represents coarse-grained ephemeris information at the intermediary. When blocks are to be sent from Mars to Earth, the DSN Earth station can send segments directly to the PI node.

6.2.10 Sample Segments

In this section we present some example LTP-T segments extracted from a single test run. Figure 6.5 shows an initial data segment that was sent from the MER-A, rover, (134.226.36.242:1301), to principal investigator #3, (134.226.36.248:1403), in this case via Mars Express (134.226.36.244:1102) which also having an open contact to Earth forwarded the segment immediately to Madrid (134.226.36.248:1407). The LTT at the time is reflected by the fact that the segment arrived in Madrid about 13 minutes after being sent from Mars Express. The segment was then forwarded via our putative DSN gateway (134.226.36.248:1404) and finally to its destination.

Figure 6.6 shows an end-of-block (i.e. the last) data segment for the same destination, but this time the segment was transmitted directly from MER-A to Madrid, since there was a direct-to-Earth contact open at the time, which is about 30 minutes after the previous example. Figure 6.7 shows yet another segment, but this time one where the destination is

```
PP: Pretty printed LTP message (at 2007-04-27 20:42:03.181).
PP: -----
PP: message type: Data segment                          [7]
PP: peer (session originator): PP: IP: 134.226.36.248:1404      [0xA1DCA2A7E08A7C]
PP: sno (session number): 0x75EF7759                    [0x87AFBDEE59]
PP: extensions: 6 headers, 0 trailers                   [0x60]
PP:     ext 0: source: 134.226.36.242:1301
PP:     ext 1: dest: 134.226.36.248:1403
PP:     ext 2: port: 0x0D05
PP:     ext 3: est. block size: 524288
PP:     ext 4: hop count: 3
PP:     ext 5: via: (253)
VIA: 134.226.36.248:1407 at 2007-04-27 20:41:26.334 (arrival)
VIA: 134.226.36.248:1407 at 2007-04-27 20:42:00.406 (departure)
VIA: 134.226.36.248:1404 at 2007-04-27 20:42:01.756 (arrival)
VIA: 134.226.36.248:1404 at 2007-04-27 20:42:01.828 (departure)
PP: csi (client service ID): 0x057F                     [0x8A7F]
PP: offset:  524256 (0x7ffe0)                                   [0x9FFF60]
PP: data length: 00032 (0x0020)                         [0x20]
PP: data (1st 4 bytes or less): 0x31323334...
PP:
PP: -----
```

Figure 6.6 – End-of-Block LTP-T Data Segment (on arrival).

the Mars Express orbiter. This sample, shows a delay of over 30 minutes at the Goldstone Earth station (134.226.36.248:1406) while waiting for the next orbiter contact.

6.2.11 Test Runs

In order to automate our Martian emulation test runs we developed scripts that clear all logs, re-start all LTP-T demons, send a single file for each run and continue running until the file is successfully transferred and all LTP sessions involved in the LTP-T session are completed. Once the LTP-T session is completed, logs files are gathered from the relevant hosts, and copied to a store for later analysis.

```
PP: Pretty printed LTP message (at 2007-04-27 21:17:32.658).
PP: -----
PP: message type: Data segment                                          [4]
PP: peer (session originator): PP: IP: 134.226.36.248:1406         [0xA1DCA2A7E08A7E]
PP: sno (session number): 0xBBFA4217                               [0x8BDFE98417]
PP: extensions: 6 headers, 0 trailers                              [0x60]
PP:     ext 0: source: 134.226.36.248:1401
PP:     ext 1: dest: 134.226.36.244:1102
PP:     ext 2: port: 0x0D05
PP:     ext 3: est. block size: 524288
PP:     ext 4: hop count: 3
PP:     ext 5: via: (253)
VIA: 134.226.36.248:1404 at 2007-04-27 20:27:56.994 (arrival)
VIA: 134.226.36.248:1404 at 2007-04-27 20:27:56.996 (departure)
VIA: 134.226.36.248:1406 at 2007-04-27 20:27:57.298 (arrival)
VIA: 134.226.36.248:1406 at 2007-04-27 21:05:02.728 (departure)
PP: csi (client service ID): 0x057C                                     [0x8A7C]
PP: offset:09112(0x2398)                                                [0xC718]
PP: data length: 01139 (0x0473)                                         [0x8873]
PP: data (1st 4 bytes or less): 0x41424344...
PP:
PP: -----
```

Figure 6.7 – Mid-Block LTP-T Data Segment.

In order to extract useful information from the log files, we use a set of scripts that produce the diagrams shown in Appendix B (using gnuplot) and that also calculate the DTN goodput (and other information) for the run, as shown in Appendix C. Those scripts can be found in the the test-setup/mars and test-results/mars-08 directories at the location given above.

This test scheme is somewhat limited at present, since each run only involves a single LTP-T session. In future, it would be interesting to test with multiple flows in parallel, however, that is not necessary for our current purposes, since, for now, our aim is only to demonstrate that LTP-T is a viable DTN protocol for such a complex environment.

6.2.12 Test Results

In this section we describe the results of various tests run with our Martian emulation. Our test result corpus consists of 75 LTP-T runs of Earth/Mars transfers, graphs of which are shown in Appendix B.2 to B.7. Appendix C is a full table of the results from each of those tests. Table 6.4 describes the different kinds of runs involved in our test set. While one could of course endlessly extend the test set, these tests are sufficient to show that LTP-T can operate in this complex environment. Appendix B also contains diagrams for some additional tests that are not counted towards these results – see the comments in the Appendix. The 75 tests from Table 6.4 are actually a subset of a larger set of 224 test runs carried out as the implementation was developed and debugged. Logs for all 224 runs can be found in the "mars-runs" directory at the URL given above.

Test type	**Section**	**#Tests**	**Redness**	**Sender**	**Dest.**	**Errors**
Earth → Mars	B.2	20	All green	PI1	MER-A	None
Mars → Earth	B.3	1	All red	MER-B	PI3	None
Mars → Earth	B.4	5	All red	MER-B	PI3	1% Mars->Earth packet loss
Mars → Earth	B.5	33	2KB red	MER-B	PI3	1% Mars->Earth packet loss
Mars → Earth	B.6	1	8KB red	MER-B	PI3	1% Mars->Earth packet loss
Mixed	B.7	15	Mixed	Mixed	Mixed	1% Mars->Earth packet loss
		75				

Table 6.4 – Types of Test Runs.

Factor	Average	Stdev	Min	Max
LTT	770s	35s	690s	825s
Test Duration	5,740s	10,402s	862s	64,902s
goodput	19KB/s	5KB/s	8KB/s	32KB/s

Table 6.5 – Overall Test Results.

The 75 runs were selected by excluding all runs with any "abnormal" behaviour. Abnormal behaviours seen included some crashes as well as some timer bugs where a session was cancelled before all reporting is complete, but after all application data has been delivered, that would have artificially increased the calculated DTN goodput. Essentially, we adopted a very conservative approach in selecting runs to use in DTN goodput calculations. That our selection is reasonable is shown by the fit between our measured results and the DTM goodput calculation provided in Section 6.2.5 as shown in Figure 6.8 below.

Table 6.5 above summarises our overall test results. Of the 75 tests, 47 involved sending a file of 47KB in size, 5 involved a file of 128KB, 22 involved a file of 0.5MB in size and 1 involved a file of 1MB. The total test duration for the entire set is 431,135s or 4 days, 23

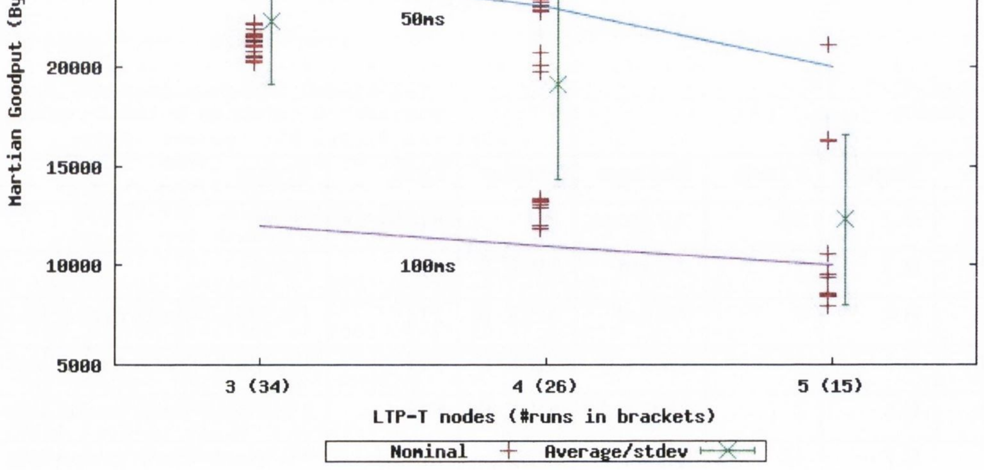


Figure 6.8 Measured DTN goodput vs. Number of Nodes.

hours, 45 minutes and 25 seconds (though in fact tests take significantly longer to run in real time) with a (modest) total of 15MB transferred between Earth and Mars. Of the 75 test runs 34 involved 3 nodes, 26 involved 4 nodes and 15 involved 5 nodes. The average number of nodes over all tests was 3.74. The 19KB/s average DTN goodput figure achieved compares well with the expected maximum DTN goodput were packets sent every 50ms, which would be about 23KB/s.

Figure 6.8 shows that the measured DTN goodput decreases as expected as the number of nodes involved increases. In the figure we also include lines indicating the maximum DTN goodput for various sending rates as calculated in Section 6.2.5.

Our network emulation shows that LTP-T could be used as part of a real Martian network, however, at this point our implementation remains a proof-of-concept, and we do not yet have reliable comparisons, for such complex scenarios, as to how LTP-T might perform against other DTN protocols (like the BP or elements of the CCSDS suite), or, perhaps more interestingly, against historic figures for data retrieval from the MER missions.

Given that LTP is a working delay-tolerant peer-to-peer protocol, and that LTP-T successfully delivered all blocks in the test cases above with a reasonable match to the maximum DTN goodput, we have now shown that LTP-T is, in fact, a viable DTN protocol for such environments.

6.3 Two-hop tests

Our final set of tests pit the BP/UDP against LTP-T and show, as expected, that, at least in the situation presented, LTP-T does in fact outperform the BP.

6.3.1 Test Setup

Here we use one intermediate node (MEX) as a relay between the sending (MER-A) and the receiving (PI1) nodes. The test setup here uses LTTs of 10ms 100ms, 1s, and 10s and file sizes of 32KB, 64KB, 128KB, 256KB, 512KB, 1MB, 2MB and, lastly 4MB. The high latency hop is the intermediate node (MEX) to destination (PI1) link, the sender (MER-A) to intermediate (MEX) has no delay introduced. Otherwise the test setup is as described in Section 6.1.

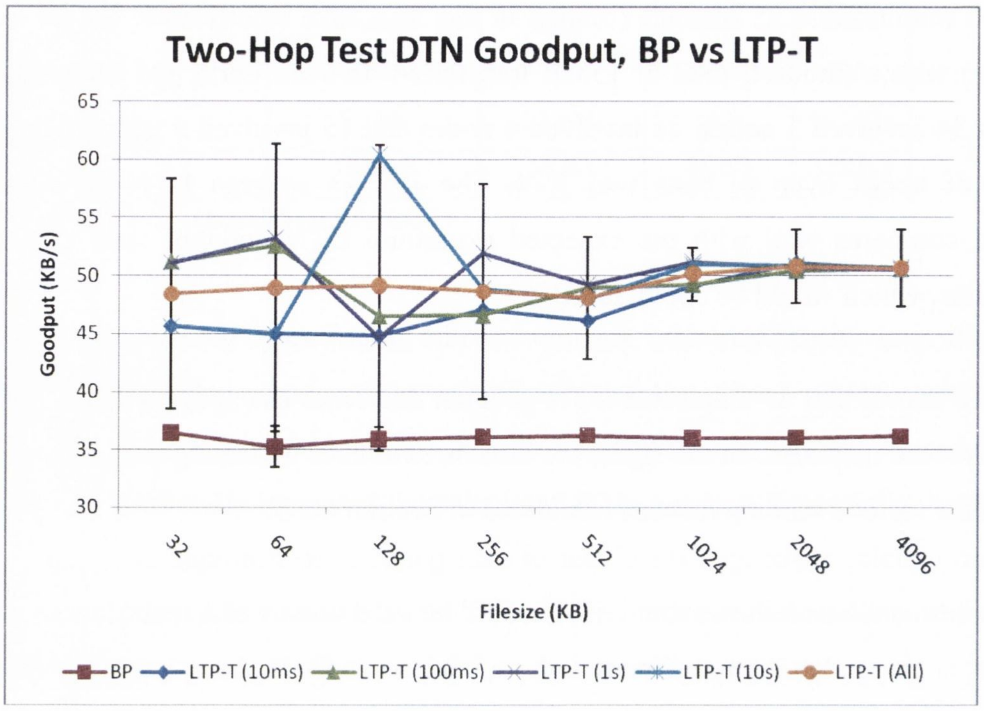


Figure 6.9 – Two-Hop Tests of BP vs LTP-T.

The BP software version was as in Section 6.1, the LTP-T version was as described in Section 6.2. All LTP sessions were run in all-green mode, and the BP made no use of custody, so both protocols are run in "unreliable" mode.

6.3.2 Test Results

Figure 6.9 above shows the DTN goodput achieved for the two hop tests for each of the file sizes tested. The BP data series represent the average for a single run of each latency, i.e., the first BP point represents the average DTN goodput for the BP sending a 32KB file over 10ms, 100ms, 1s and 10s. For LTP-T we show the average of 5 runs for each latency as a separate series, and also the overall average. In the case of the overall average (labeled "LTP-T (All)," each point represents the average over 20 runs, and for this data series we include error bars representing plus and minus one standard deviation.

Over all runs, the average DTN goodput for the BP is 36.0 KB/s, (standard deviation, 0.36), whereas for LTP-T the overall average is 49.3 KB/s (standard deviation 8.09).

6.3.3 Discussion

As can be seen from Figure 6.9 above, in this configuration, LTP-T outperforms the BP, essentially increasing DTN goodput by about one third. This is due to the fact that the LTP-T intermediary can start to forward LTP segments as soon as they are received, whereas the BP daemon only starts to forward parts of the bundle after the entire bundle has arrived. It would be non-trivial for the BP daemon to take the same approach, since its Convergence Layer (CL) model is built around the assumption that different CLs must be able to be used for the inbound and outbound contacts for a given bundle. This same benefit will accrue at each DTN hop for which a DTN node can forward packets as they arrive, so that for longer paths, LTP-T will also outperform the BP.

However, one should be clear – we are not here saying that LTP-T is "better" than the BP, but just that, as a DTN transport, it is easier for LTP-T to outperform the BP in some circumstances. Secondly, as an overlay, it is harder, though in principle perhaps still possible, for the BP to take advantage of CL specifics so as to perform similarly to a DTN transport.

In summary, then this section has shown that LTP-T can outperform the BP.

6.4 Conclusions

In this chapter we documented our evaluation of LTP-T, which used a proof-of-concept Martian network emulation and some specific comparative tests of LTP-T against the BP and other protocols. This allows us to conclude that LTP-T is very much a reasonable DTN protocol on which to base further DTN experiments.

Chapter 7

Conclusions and Future Work

In this closing chapter, we re-iterate the problem statement from Section 1.4, we summarise how this thesis has addressed that problem statement; we specifically consider how LTP-T meets the DTN protocol requirements elicited in Chapter 2, and we review the contribution and discuss near and longer-term areas for future work on DTN transports.

7.1 The Problem Statement Re-iterated

The main problems considered here are the consequences of choices related to the location of functionality (e.g. re-transmission) within a DTN and how those choices interact with various protocol features (e.g. reliability or security). The challenge faced is to explore this design space to develop protocols that meet a sufficiently broad set of requirements so as to warrant the wider-scale testing of those protocols with the ultimate goal of eventually incorporating DTN features and functionality into the standard suite of Internet protocols. This thesis represents yet another step along that road.

While there have been some attempts to examine meeting DTN requirements at the transport layer, there has yet to be a fleshed-out demonstration that this is possible and that, as an architecture, the result provides comparable performance to the overlay approach. This is the specific question that this thesis attempts to answer.

The methodology followed is to review the main causes of delay and disruption in order to derive a set of protocol requirements that are specific to the DTN context and to examine how existing protocols, and in particular the overlay approach as instantiated by the BP, meet these requirements.

Based on this review, we define our concept of a DTN transport protocol and analyse in detail how that compares with the overlay approach. We then describe one DTN Transport protocol (LTP-T), developed specifically in order to test the DTN transport concept. We describe a set of tests and results in order to demonstrate: a) that LTP-T is a viable DTN transport protocol, and b) that in some cases LTP-T can significantly outperform the BP.

We finally consider how well LTP-T meets our set of DTN specific requirements in order to point the way forward for future work in the area of DTN transport protocols.

7.2 Addressing the Problem Statement

In Chapters 4 and 6 we considered the significance of choices related to the location of functionality (in particular storage and re-transmission) within a DTN and how those choices interact with various protocol features (e.g. reliability or security). We explored this design space and developed a protocol (LTP-T) that meets a sufficiently broad set of requirements so as to warrant its wider-scale testing with the ultimate goal of eventually incorporating DTN features and functionality into the standard suite of Internet protocols.

We demonstrated that meeting DTN protocol requirements using the transport-oriented approach of LTP-T is feasible and also demonstrated that LTP-T can outperform the overlay approach as represented by the Bundle Protocol (BP). The demonstration of feasibility involved a complex Martian DTN emulation that justifies our claim that LTP-T is a viable DTN protocol. We also used a simple two-hop scenario to demonstrate that LTP-T can significantly outperform the BP when the intermediary can immediately begin forwarding packets.

We have thus addressed all of the issues called for by the problem statement in Section 1.4.

7.3 Meeting the DTN Requirements

We now consider how LTP-T meets each of the DTN requirements posed in Chapter 2.

R.1 DTN protocols must operate even in the face of the total absence of an end-to-end connection.

This is amply demonstrated for LTP-T by our Martian network emulation.

R.2 DTN protocols should be able to operate (relatively efficiently) even when link or path latencies are of the order of minutes, hours or days.

Again, the Martian network emulation demonstrates that LTP-T meets this requirement.

R.3 DTN protocols should support changes in the scheduling and/or contactability of nodes.

The Martian network emulation demonstrates that LTP-T meets this requirement, at least in a scheduled scenario. We have not demonstrated that LTP-T works where contacts are changing at a high frequency. However, our LTP-T implementation (LTPlib) includes a `whos_listening()` Service Provider Interface (SPI) designed for such cases. For example, if a Mobile Ad-Hoc Networking (MANET) routing protocol were used to support the IP layer, then, even without knowledge of how the specific MANET protocol works, the SPI could periodically send an Internet Control Message Protocol (ICMP) echo request message to determine which of a number of peers are currently contactable. In such a case, higher layers of LTPlib would then properly manage timers in the same way as in a scheduled scenario.

R.4 DTN protocols should be able to operate when the host hibernates or reboots in the "middle" of a "session."

LTPlib's "disk" mode of operation allows LTP-T to meet this requirement. Alhough we have not demonstrated this specifically in Chapter 6, this mode was demonstrated by the SeNDT project's lake water quality monitoring node. [MC07]

R.5 DTN protocols should be usable in systems where power conservation is the over-riding system-level requirement.

As a DTN protocol, LTP-T itself does not specifically address this system-level requirement. One could envisage the routing and forwarding implementation selecting next hops partly on the basis of power consumption, but it is not clear how well this might work. With the LTPlib "disk" mode of operation however, one can use LTP-T in a power-conserving device. This was also demonstrated as part of the SeNDT project. [MC07]

R.6 DTN protocols should generally attempt to move data "towards" the destination, even though some optimal routes may involve temporarily moving data "away" from the destination.

LTPlib depends on configured routes and/or schedules for this. The "towards" part is demonstrated by our use of the Deep-Space Network (DSN) gateway node schedule described in Section 6.2.9. The "moving away" part can also happen with the current forwarding scheme, since we select the next hop on the basis of the `whos_listening()` SPI, e.g. in the case of the run shown in Appendix B.2.18 we see segments being sent via an orbiter, resulting in that case in a significant increase in overall latency before the orbiter can contact the lander. However, to be clear, we should point out that neither LTP-T nor LTPlib currently have an explicit concept of "distance from the destination," nor have we done work on route optimisation as part of this thesis. Presumably some other DTN protocol could make explicit use of such a concept, but we have shown that LTPlib can meet the requirement for our Martian emulation even without such an explicit distance concept.

R.7 DTN protocols should not require simple, regular or strictly periodic nor cyclic patterns of connectivity, but should be able to benefit from such patterns where they exist.

LTP-T doesn't require any such regularity and LTPlib is able to handle both periodic and aperiodic patterns via our scheduling setup. By providing the SPI as discussed above, we also allow for future additions in this respect.

R.8 DTN protocols should be able to operate in situations where applications or the environment determine duty-cycles.

The set of socket options provided (see Section 5.4.7) allows LTPlib configuration and routes to be fully determined by the application with the same level of control as using the default file-based configuration and scheduling options. The fact that developers are familiar with this method of modifying communication stack behaviour is also relevant in meeting this requirement.

R.9 DTN protocols should not assume the same path is always used for application layer requests and responses.

LTP-T makes no such assumption. An application built on LTP-T can simply use different sockets, and hence LTP-T sessions, and different routes can be taken. While each LTP-T session does involve a single path, the path selected for an application layer request has no effect on the application-layer return path, which will consist of a different set of LTP sessions, and hence can use a different path through the DTN.

R.10 DTN protocols should be able to support a very wide range of data rates.

LTP is primarily designed to be used for deep-space communications and so is efficient in terms of bandwidth consumption. LTP-T inherits this efficiency, though care should be taken in how LTP-T extensions are distributed amongst LTP data segments. (LTPlib is currently wasteful in this respect.) While we have demonstrated LTP-T operating at low and medium data rates in the Martian emulation and other tests, we have not yet considered LTP-T operating at very high data rates (e.g. in the gigabit/second range). While there seems to be a clear potential for the use of DTN protocols in such high-performance networks, (where the bandwidth-delay product is increased due to the bandwidth available rather than delay or disruption), we currently make no claims as to the suitability of LTP-T for such environments.

R.11 DTN protocols must co-exist with, and be able to make use of, the existing Internet

LTP-T is layered on top of UDP and uses IP addressing and so can clearly make use of the existing Internet. With no specific configuration, LTPlib will operate in an always-on mode that will work well on the existing Internet. So, LTP-T could be used on the existing Internet without any requirement for a "flag-day." However, a study of how mechanisms like TCP Friendly Rate Control (TFRC) might improve LTP-T's Internet friendliness would be worthwhile and would increase one's confidence in respect of this requirement. Similarly, studies addressing ways to ensure fairness for different application layer flows in an LTP-T driven DTN would be required before we could confidently say that LTP-T fully meets this requirement.

R.12 DTN protocols should operate in the face of significant node mobility, even for infrastructure nodes.

LTP-T itself requires no new infrastructure nodes and all LTP-T nodes act as routers so LTP-T is well suited for use in scenarios with significant node mobility. Larger scale use of LTP-T however, may require use of the Domain Name System (DNS). For example, if routes, sources or destinations use DNS names rather than IP addresses. Such uses of DNS can be a challenge in some mobility cases.

R.13 DTN protocols should be designed so as to be highly robust in the face of DoS attacks.

For LTP-T, this mainly affects LTP, which has been specifically designed with Denial-of-Service (DoS) in mind, for example recommending random session identifiers in the security considerations part of the LTP specification. [RA08] The LTP cookie and hop-by-hop authentication extensions [FA08] allow the anti-DoS "paranoia" level to be ramped up, either generally, or in response to events. At the LTP-T level, since we have no multi-hop backwards signalling at all, it will be relatively hard to create DoS amplification attacks. This is in contrast to other protocols, like the BP, that have more complex signalling since which inherently increases the attack surface for implementations of the protocol in question.

R.14 DTN protocols should provide (or leverage) confidentiality and data integrity services.

LTP-T does provide end-to-end data integrity and, via LTP, hop-by-hop integrity. We have decided not to attempt to define an end-to-end confidentiality service until more work has been done on the general topic of DTN key management.

In summary then, the LTP-T protocol, and/or our LTPlib implementation substantially meets all of the DTN requirements posed in Chapter 2.

7.4 The Contribution

In this thesis we proposed the concept of a delay- and disruption-tolerant transport protocol and defined one such protocol, LTP-T. We have demonstrated that LTP-T and our LTPlib implementation address many of the issues identified in our problem statement and have shown that LTP-T meets the set of requirements we derived for DTN protocols, on the basis of our assessment of the background work on DTNs.

Our DTN transport concept, and LTP-T specifically, is proposed as an alternative to, but not as a replacement for, the overlay approach as exemplified by the BP. There are clearly deployment scenarios where the BP can work but where LTP-T cannot, for example, where different lower layers are in use on each DTN hop.

However, there are gains to be made from use of a DTN transport - both in performance and simplicity. For instance, with LTP-T one can sometimes forward segments more promptly than the BP can forward bundles. In terms of simplicity, the DTN transport model, with its restriction to on-path, single hop signalling is far simpler than the BP model of custodians, multi-hop backwards signalling and potential off-path reporting. That simplicity is beneficial in many respects, for example, such simplicity enables one to more easily have more confidence in the security of a DTN.

In summary, LTP-T offers a new option for use in the design of DTNs with lower layers that can all be accessed via IP. The DTN transport concept (which could be instantiated in other protocols) also results in a simpler model than the overlay approach and so may prove to be a better option when it comes to including DTN functionality into standard Internet protocols.

7.5 Future Work

Of course, we do not assert that the work on DTN transports, nor on LTP-T, is complete – a fact highlighted in our discussion of how LTP-T meets the set of DTN protocol requirements. In this section we suggest directions for further work on the DTN transport concept and LTP-T.

In the near term we can clearly work to enhance and further test the LTPlib implementation with a view to enhancing and extending the network emulation. For example, we could build a more realistic model of application layer traffic, and could include schedule distribution as part of the emulation. Separately, we could extend the emulation to better support the use of LTP-T in DTNs that are more dominated by frequent disruption (e.g. military tactical networks), rather than networks dominated by scheduling as in our Martian emulation.

There is also an opportunity to further develop DTNs generally, by making use of predictive methods (e.g. as described by Baliosian et. Al. [BA06]) for resource availability – the addition of such intelligence to DTN nodes may be required before larger scale DTN applications can be deployed.

We can also use our emulation setup to further analyse and the differences between the BP and LTP-T. For example, it would be interesting to investigate the effects of custody transfer on the performance of the BP and contrast that with the use of the red-part in LTP-T blocks. In a similar vein, we plan to investigate the effects of custody placement on DTN protocols along the lines of the topology discussion from Chapter 4.

Work on DTN key management and authorization is planned to begin in the near term and, once that has progressed, we plan to add an end-to-end confidentiality service to LTP-T. Similarly, work on DTN congestion is at a relatively early stage, and we plan to implement and investigate the LTP-T congestion control scheme described in Chapter 5.

In the medium term, we will determine whether there is broader interest in the concept of DTN transport, and specifically in LTP-T, in the DTNRG. If there is, we will provide a specification of LTP-T in the form of an internet draft so that others can also implement the protocol. Ultimately, there may be sufficient interest to begin work on some Internet standards-track DTN transport protocol(s), in which case, this work will form one amongst no doubt many, inputs to the ongoing development of delay- and disruption-tolerant networking.

References

[AB03] Abraham, I., et al, “A Generic Scheme for Building Overlay Networks in Adversarial Scenarios,” in Proceedings of the IEEE International Symposium on Parallel and Distributed Processing, Nice, France, 9pp-, April 2003.
doi: 10.1109/IPDPS.2003.1213125

[AK02] Akan, O., “Performance of TCP Protocols in Deep Space Communication Networks,” IEEE Communications Letters, Vol.6, Iss.11, p478-480, November 2002.
doi: 10.1109/LCOMM.2002.805549

[AK02s] Akyildiz, I., at al., “A Survey on Sensor Networks,” Communications Magazine, IEEE, Vol.40, Iss.8, p102-114, August 2002.
doi: 10.1109/MCOM.2002.1024422

[AK04] Akan, O., Fang, J., and Akyildiz, I., “TP-Planet: A New Transport Protocol for InterPlaNetary Internet,” IEEE Journal of Selected Areas in Communications, Vol.22, Iss.2, p348-361, February 2004.
doi: 10.1109/JSAC.2003.819985

[AK04i] Akyildiz, I, et al., “The state of the art in Interplanetary Internet,” IEEE Communications, Vol 42, Issue 7, p108-118, July 2004.
doi: 10.1109/MCOM.2004.1316541

[AL00] Albert. R., Jeong, H., Barabasi, A.., “Error and Attack Tolerance of Complex Networks,” Nature 406, p378-382, 27 July 2000.
doi: 10.1038/35019019

[AL94] Allocihio, C., et al, “Using the Internet DNS to Distribute RFC1327 Mail Address Mapping Tables,” Internet RFC 1664, August 1994.

[AL98] Albee, A., Palluconi, F., Arvidson, F. “Mars Global Surveyor Mission: Overview and Status”, Science, Vol.279,No.5357, p1671-1672, 13 March 1998.
doi: 10.1126/science.279.5357.1671

[AN01] Andersen, D., at al., "Resilient Overlay Networks," SIGOPS Oper. Syst. Rev., Vol.35, Iss.5, p131-145, December 2001.
doi: 10.1145/502059.502048

[AU05] Aura, T., Roe, M., and Mohammed, A., "Experiences with Host-to-Host IPsec," in Security Protocols, 13th International Workshop, Cambridge, UK, Springer LNCS 4631, p23-30, April 2005.
doi: 10.1007/978-3-540-77156-2_3

[BA04] US DARPA, BAA04-13: Disruption Tolerant Networking (DTN), May 2004.

[BA06] Baliosian, J. et al, "Self-configuration for radio access networks," 7th IEEE International Workshop on Policies for Distributed Systems and Networks, 4pp-, 5-7 June 2006.
doi: 10.1109/POLICY.2006.28

[BE02] Behrman, W., "Best Practices for the Development and Use of XML Data Interchange Standards," Center for Integrated Facility Engineering Technical Report #131, Stanford University, July 26, 2002.
http://www.stanford.edu/group/CIFE/online.publications/TR131.pdf

[BE03] Bell, J., et al, "Mars Exploration Rover Athena Panoramic Camera (Pancam) investigation," Journal of Geophysical Research, Vol. 108, No. E12, 8063, November 2003.
doi:10.1029/2003JE002070

[BE05] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax," Internet RFC 3986, January 2005.

[BE06] Bell, J., et al, "In-flight Calibration and Performance of the Mars Exploration Rover Panoramic Camera (Pancam) Instruments," Journal of Geophysical Research, Vol. 111, E02S03, January 2006.
doi:10.1029/2005JE002444

[BI03] Biswas, A., and Piazzolla, S., "Deep-Space Optical Communications Downlink Budget from Mars: System Parameters," The Interplanetary Network Progress Report 42-154, April–June 2003, Jet Propulsion Laboratory, Pasadena, California, p1-38, August 15, 2003.
http://tmo.jpl.nasa.gov/progress_report/42-154/154L.pdf

[BL01] Blumenthal, M., and Clark, D., "Rethinking the Design of the Internet: the End-to-End Arguments vs. the Brave New World," ACM Trans. Inter. Tech. Vol.1, Iss.1, p70-109, August 2001.
doi: 10.1145/383034.383037

[BL05] Bleech, N., "Visioning White Paper, What is Jericho Forum?" February 2005.
http://www.opengroup.org/projects/jericho/

[BO06] Boccaletti, S., et al., "Complex Networks: Structure and Dynamics," Physics Reports, Vol. 424, Iss. 4-5, p175–308, February 2006.
doi: 10.1016/j.physrep.2005.10.009

[BR00] Breslau, L., et al., "Advances in Network Simulation," IEEE Computer, Vol.33, Iss.5, p59-67, May 2000.
doi: 10.1109/2.841785

[BR04] Brooke, T., Burrel, J. and Beckwith, T., "Vineyard Computing: Sensor Networks in Agricultural Production," in IEEE Pervasive Computing Magazine, p38–45, March 2004.
doi: 10.1109/MPRV.2004.1269130

[BR04d] Breitbart, Y., et al., "Topology Discovery in Heterogeneous IP Networks: the NetInventory System," IEEE/ACM Trans. Netw. Vol.12, Iss.3, p401-414, June 2004.
doi:10.1109/TNET.2004.828963

[BR05] Brunner, M., et al., "Dagstuhl Seminar on Disruption Tolerant Networking," SIGCOMM Comput. Commun. Rev. 35, 3, p69-72, July 2005.
doi:10.1145/1070873.1070882

[BR06] Bruno, D., "The JBDS (Java Based Deep Space) Simulator: A New Approach," Trinity College Dublin Computer Science Technical Report, TCD-CS-2006-63, December 2006.
https://www.cs.tcd.ie/publications/tech-reports/reports.06/TCD-CS-2006-63.pdf

[BR94] Braden, R., "T/TCP – TCP Extensions for Transactions, Functional Specification," Internet RFC 1644, July 1994.

[BR97] Bradner, S., "Keywords for use in RFCs to Indicate Requirements Levels," Internet RFC 2119, March 1997.

[BU02] Burleigh, S., et al., "The Interplanetary Internet: A Communications Infrastructure for Mars Exploration," in Proceedings of the 53rd International Astronautical Federation Congress (2002), The World Space congress, Acta Astronautica, Vol.53, Iss.4-10, p365-373, August 2003. doi: 10.1016/S0094-5765(03)00154-1

[BU03] Burleigh, S., et al., "Delay-Tolerant Networking: An Approach to Interplanetary Internet," IEEE Communications Magazine, Vol.41, Iss.6, p128-136, June 2003. doi: 10.1109/MCOM.2003.1204759

[BU06] Burleigh, S., Jennings, E., and Schoolcraft, J., "Autonomous Congestion Control in Delay-Tolerant Networks," AIAA-2006-5960, SpaceOps 2006 Conference, Rome, Italy, June 2006. http://pdf.aiaa.org/preview/CDReadyMSPOPS06_1317/PV2006_5970.pdf

[BU07c] Burleigh, S., "Compressed Bundle Header Encoding (CBHE)," Internet Draft, draft-burleigh-cbhe, January 2007, work-in-progress. http://tools.ietf.org/internet-drafts/draft-burleigh-cbhe

[BU08] Burleigh, S., Ramadas, M., and Farrell, S., "Licklider Transmission Protocol – Motivation", Internet RFC 5325, September 2008.

[BU93] Bush, R., "FidoNet: Technology, Tools, and History," Communications of the ACM, Vol.36, Iss.8, p31-35, August 1993. doi: 10.1145/163381.163383

[BY04] Byers, J., et al., "Informed Content Delivery across Adaptive Overlay Networks," IEEE/ACM Trans. Netw., Vol.12, Iss.5, p767-780, Oct. 2004. doi:10.1109/TNET.2004.836103

[CA02] Carpenter, B. and Brim, S., "Middleboxes: Taxonomy and Issues," Internet RFC 3234, February 2002.

[CA03] Calhoun, P. et al., "Diameter Base Protocol," Internet RFC 3588, September 2003.

[CC03] CCSDS TM Space Data Link Protocol – CCSDS-132.0-B-1, Blue Book, Issue 1, September 2003. http://public.ccsds.org/publications/archive/132x0b1.pdf

[CC06] Space Communications Protocol Specification (SCPS) – Transport Protocol. Recommendation for Space Data Systems Standards, CCSDS 714.0-B-2, Blue Book, Issue 2, October 2006. http://public.ccsds.org/publications/archive/714x0b2.pdf

[CC06p] CCSDS Proximity-1 Space Link Protocol—Data Link Layer, CCSDS-211.0-B-4, Blue Book. Issue 4, May 2006. http://public.ccsds.org/publications/archive/211x0b4.pdf

[CC07] CCSDS File Delivery Protocol (CFDP) - CCSDS-727.0-B-4 Blue Book Issue 4, June 2007. http://public.ccsds.org/publications/archive/727x0b4.pdf

[CE04] Cerpa, A., and Estrin, D., "ASCENT: Adaptive Self-Configuring Sensor Network Topologies," IEEE Transactions on Mobile Computing, Vol.3, Iss.3, p272-285, July-August 2004. doi: 10.1109/TMC.2004.16

[CE07] Cerf. V., et al., "Delay-tolerant Networking Architecture," Internet RFC 4838, April 2007.

[CH02] Chandra, S., "Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats," in Proc. Multimedia Computing and Networking, San Jose, CA, p85-99, January 2002. doi: 10.1007/s00530-003-0089-0

[CH03] Christensen, P., et al., "The Miniature Thermal Emission Spectrometer for the Mars Exploration Rovers", Journal of Geophysical Research, Vol.108, No. E12, pROV5.1-ROV5.23 (1 p.1/4), 2003. doi:10.1029/2003JE002117

[CH06] Mooi-Choo, C., et al., "Store-and-Forward Performance in a DTN," Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd , vol.1, no., p187-191, 7-10 May 2006. doi: 10.1109/VETECS.2006.1682801

[CH05] Chaves, C., et al., "Honeynet Maintenance Procedures and Tools," IEEE Workshop on Information Assurance and Security, West Point, NY, USA, p252-257, June 2005. doi: 10.1109/IAW.2005.1495960

[CH81] Chaum, D., "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," Communications of the ACM, Vol.4, Iss.2, p84-90, February 1981.
doi: 10.1145/358549.358563

[CL03] Clausen, T., (ed.), "Optimized Link State Routing Protocol (OLSR)," Internet RFC 3626, October 2003.

[CO01] Conner, W., Krishnamurthy, L., and Want, R., "Making Everyday Life Easier Using Dense Sensor Networks," In Proceedings of the 3rd international Conference on Ubiquitous Computing (Atlanta, Georgia, USA, September 30 - October 02, 2001). G. D. Abowd, B. Brumitt, and S. A. Shafer, Eds. Lecture Notes In Computer Science, vol. 2201. Springer-Verlag, London, 49-55.
http://www.springerlink.com/content/u5m1yu1g91ereagg/?p=c9119a60e2f044f18bf74d6454a1739f&pi=3

[CO02] Coene, L., "Stream Control Transmission Protocol Applicability Statement," Internet RFC 3257, April 2002.

[CO03] Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., and Toshiba Corporation, Advanced Configuration and Power Interface Specification, October 10 2006..
http://www.acpi.info/spec.htm

[CR99] Crowcroft, J., "IP over Photons: How not to Waste the Waist of the Hourglass," Seventh International workshop on Quality of Service, London, p9-11, June 1999.
doi:10.1109/IWQOS.1999.766472

[CU02] Curran, K., and Parr, G., "A Framework for the Transmission of Streaming Media to Mobile Devices," Int. J. Netw. Manag. Vol.12, Iss.1 p41-59, January 2002.
doi:10.1002/nem.420

[CU05] Curran, K., and Parr, G. "Dynamic Reconfiguration of IP Domain Middleware Stacks to Support Multicast Multimedia Distribution in a Heterogeneous Environment," Journal of Computer Science, Vol.1 Iss.1, p7-18, 2005 ISSN 1549-3636.
http://www.scipub.us/fulltext/jcs/jcs117-18.pdf

[DA01] David, P., "The Beginnings and Prospective Ending of End-to-End – An Evolutionary Perspective on the Internet's Architecture," Stanford Institute for Economic and Policy Research, Discussion paper No. 01-04. http://www-econ.stanford.edu/faculty/workp/swp01012.pdf

[DE04] Demmer, M., et al., "Implementing Delay Tolerant Networking," Intel technical report, IRB-TR-04-020, Dec.28 2004. http://www.cs.berkeley.edu/~demmer/papers/dtn-irb-tr-04-020.pdf

[DE05] Devarapalli, V., et al., "Nework Mobility (NEMO) Basic Support Protocol," Internet RFC 3963, January 2005.

[DE07] Demmer, M., and Ott, J., "Delay-Tolerant Networking TCP Convergence Layer Protocol," Internet Draft, draft-irtf-dtnrg-tcp-clayer, February 2008, work-in-progress. http://tools.ietf.org/internet-drafts/draft-irtf-dtnrg-tcp-clayer

[DI04] Dingledine, R., Mathewson, N. and Syverson, P., "Tor: The Second-Generation Onion Router," Proceedings of the 13th USENIX Security Symposium, p21-38, 2004. https://tor-svn.freehaven.net/svn/tor/tags/imported-from-cvs/tags/tor-0_1_0_1_rc/doc/design-paper/tor-design.pdf

[DI06] Dierks, T., and Allen, C., "The TLS Protocol – Version 1.1," RFC 4346, January 2006.

[DO00] Donnelly, A., and Deegan, T., "IP Route Lookups as String Matching," Proceedings. 25th Annual, IEEE Conference on Local Computer Networks, p589-595, November 2000. doi:10.1109/LCN.2000.891104

[DT06] IETF-67 DTNRG Minutes, San Diego, November 2006. http://www.ietf.org/proceedings/06nov/minutes/DTNRG.html, http://www.ietf.org/proceedings/06nov/slides/DTNRG-1/sld1.htm, http://www.ietf.org/proceedings/06nov/slides/DTNRG-2/sld1.htm

[DU97] Durst, R., Miller, G., and Travis, E., "TCP Extensions for Space Communications," J. Wireless Networks, Vol.3, No.5, p389-403, October 1997. doi:10.1023/A:1019190124953

[ED07] Eddy, W., “Using Self-Describing Numeric Values in Protocols,” Internet Draft, draft-irtf-dtnrg-sdnv, September 2007. work-in-progress. http://tools.ietf.org/internet-drafts/draft-irtf-dtnrg-sdnv

[FA00] Farrell, S., et al., "AAA Authorization Requirements," Internet RFC 2906, August 2000.

[FA03] Fall, K., “A Delay-Tolerant Network Architecture for Challenged Internets,” in Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications (Karlsruhe, Germany, August 25 - 29, 2003). SIGCOMM '03. ACM Press, New York, NY, p27-34. 2003. doi:10.1145/863955.863960

[FA03e] Farrell, S., and Jensen, C., “A Flexible Interplanetary Internet,” Proceedings of the 37th ESLAB Symposium `Tools and Technologies for Future Planetary Exploration', Noordwijk, The Netherlands, 2-4 December 2003. ESA SP-543, p87-94, April 2004. http://articles.adsabs.harvard.edu/cgi-bin/nph-iarticle_query?2004ESASP.543...87F

[FA05] Farrell, S., and Cahill, V., “LTP-T: A Generic Delay Tolerant Transport Protocol,” TCD Computer Science Technical Report TCD-CS-2005-69, 7 December 2005. https://www.cs.tcd.ie/publications/tech-reports/reports.05/TCD-CS-2005-69.pdf

[FA05j] Fall, K., and McCanne, S., “You don't know Jack about Network Performance,” *Queue* 3, 4, p54-59, May. 2005. doi:10.1145/1066051.1066069

[FA06] Farrell, S., and Cahill, V., “Delay and Disruption Tolerant Networking,” ISBN 1-59693-063-2, Artech House, 2006.

[FA06e] Fahad, S., “Transport Protocols for Interplanetary Communications”, Master recherche report, Ecole Nationale Supérieure des Telecommunications, 2006.

[FA06i] Farrell, S., et al., “When TCP Breaks: Delay- and Disruption-Tolerant Networking,” IEEE Internet Computing, Vol.10, No.4, p72-78, July/August 2006. doi:10.1109/MIC.2006.91

[FA06s] Farrell, S., and Cahill, V., "Security Considerations in Space and Delay Tolerant Networks," 2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06), p29-38, July 2006. doi:10.1109/SMC-IT.2006.66

[FA07s] Farrell, S., et al., "Delay-Tolerant Networking Security Overview", Internet-draft, draft-irtf-dtnrg-sec-overview, February 2008, work-in-progress. http://tools.ietf.org/html/draft-irtf-dtnrg-sec-overview

[FA08] Farrell, S., et al., "Licklider Transmission Protocol – Extensions", Internet RFC 5327, September 2008.

[FA08j] Fall, K.; Farrell, S., "DTN: an architectural retrospective," *Selected Areas in Communications, IEEE Journal on* , vol.26, no.5, p828-836, June 2008. doi: 10.1109/JSAC.2008.080609

[FA97] Fall, K., and Varadhan, K., "ns Notes and Documentation," Technical Report, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997. http://citeseer.ist.psu.edu/655233.html

[FE04] Fernández, D., Galán, F., and de Miguel, T. "Study and Emulation of IPv6 Internet Exchange (IX) based Addressing Models," IEEE Communications Magazine, Vol.42, Iss.1, p105-112, January 2004. doi:10.1109/MCOM.2004.1262169

[FI99] Finkenzeller, K., "The Radio Frequency Identification (RFID) Handbook", John Wiley & Sons, New York, 1999.

[FL07] Floyd, S., and Kohler, E, "TCP Friendly Rate Control (TFRC): The Small-Packet (SP) Variant," Internet RFC 4828, April 2007.

[FL99] Floyd, S., and Fall, K., "Promoting the use of End-to-End Congestion Control in the Internet," IEEE/ACM Trans. Netw. Vol.7, Iss.4, p458-472, August 1999.
doi:10.1109/90.793002

[FR04] Franklin, S, et Al., "The 2009 Mars Telecom Orbiter Mission," Aerospace Conference, , Montana USA, p456-, March 2004.
doi:10.1109/AERO.2004.1367627

[GA06] Galán, F., and Fernández, D., “Use of VNUML in Virtual Honeynets Deployment,” IX Reunión Española sobre Criptología y Seguridad de la Información (RECSI), Barcelona (Spain), September 2006. ISBN: 84-9788-502-3.

[GI99] Gilligan, R., et al., “Basic Socket Interface Extensions for Ipv6,” Internet RFC 2553, March 1999.

[GL98] Glover, D., and Kruse, H., “TCP Performance in a Geostationary Satellite Environment,” Sect. 6, Wireless. Annual Review of Communications 1998, International Engineering Consortium, April 1998. http://citeseer.ist.psu.edu/glover98tcp.html

[GR03] Gross, T., “Security Analysis of the SAML Single Sign-On Browser/Artifact profile,” Computer Security Applications Conference, 2003. Proceedings. 19th Annual, p298-307, December 2003. doi:10.1109/CSAC.2003.1254334

[GR90] Greengard, L., “The Numerical Solution of the N-Body Problem,” Comput. Phys. Vol.4, Iss.2, p142-152, February 1990. ISSN:0894-1866

[GU05] Gu, Y., and Grossman, R., “Optimizing UDP Based Protocol Implementations,” Proceedings of the Third International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet 2005), Lyon, France, February 2005.

[HA02] Hassel, J., “RADIUS: Securing Public Access to Private Resources,”, ISBN: 0-596-00322-6, O’Reilly, October 2002.

[HA03] Handley, M. et al., “TCP Friendly Rate Control (TFRC): Protocol Specification,” Internet RFC 3448, January 2003.

[HA06] Hayat, Z., Reeve, J. and Boutle, C., “Prioritisation of Network Security Services,” IEE Proceedings - Information Security, Vol.153, Iss.2, p43-50, June 2006. ISSN:1747-0722

[HA06t] Harras, K., and Almeroth, K., “Transport Layer Issues in Delay Tolerant Mobile Networks,” Proceedings of IFIP-TC6 Networking, p27-34, May 2006. doi:10.1145/863955.863960

[HE01] Heidemann, J., Mills, K., and Kumar, S., "Expanding Confidence in Network Simulations," IEEE Network, Vol.15, Iss.5, p58-63, Sep/Oct 2001.
doi:10.1109/65.953234

[HE01d] Heidemann, J., et al., "Effect of Detail in Wireless Network Simulation," USC/ISI TR-2000-523b, January 2001.
ftp://ftp.isi.edu/isi-pubs/tr-523.pdf

[HE05] Hemminger, S., "Network Emulation with NetEm," Linux Conference Australia, LCA2005, Canberra, April. 2005.
http://developer.osdl.org/~shemminger/netem/LCA2005_paper.pdf

[HE05a] Hebden, P., and Pearce, A., "Bloom Filters for Data Aggregation and Discovery: a Hierarchical Clustering Aapproach," Proceedings of the 2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing Conference, p175-180, 5-8 Dec. 2005.
http://ieeexplore.ieee.org/iel5/10633/33567/01595575.pdf

[HO01] Hogie, K., Criscuolo, E., and Parise, R.,"Link and Routing Issues for Internet Protocols in Space", 2001 IEEE Aerospace Conference, p2/963-2/976 vol.2, March 2001.
doi:10.1109/AERO.2001.931278

[HO02] Hong, X., et al., "Scalable Routing Protocols for Ad Hoc Networks," IEEE Network, Vol.16, Iss.4, p11-21, July/August 2002.
doi:10.1109/MNET.2002.102023

[HO97] Horowitz, M., and Lunt, S. "FTP Security Extensions," Internet RFC 2228, October 1997.

[IE99] IEEE 802.11, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification," August 1999.

[IN00] Intanagonwiwat, C., Govindan, R., and Estrin, D., "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks" Proc. ACM International Conference on Mobile Computing and Networking, p56-67, Boston, MA, 2000.
http://citeseer.ist.psu.edu/intanagonwiwat00directed.html

[IR99] Iren, S., Amer, P., and Conrad, P., "The Transport Layer: Tutorial and Survey," ACM Comput. Surv. Vol.31, Iss. 4 p360-404, December 1999.
doi:10.1145/344588.344609

[IS94] ISO standard 7498-1:1994, “Open Systems Interconnection – Basic Reference Model,” 1994.

[IT97] "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, 1997.

[JA04] Jain, S., Fall, K., and Patra, R., “Routing in a Delay Tolerant Network,” SIGCOMM Comput. Commun. Rev. Vol.34, Iss.4, p145-158, August 2004.
doi:10.1145/1030194.1015484

[JA97] Jaffe, L., and Herrell, L., “Cassini/Huygens Science Instruments, Spacecraft, and Mission,” Journal of Spacecraft and Rockets, Vol.34, Iss. p509-521, August 1997.
NASA no. 19990008058

[JO03] Jonsson, J, and Kaliski, B, “Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1,” Internet RFC 3447, February 2003.

[JO96] Johnson, D., and Maltz, D., “Dynamic Source Routing in Ad Hoc Wireless Networks,” In Mobile Computing, Imielinski, T., and Korth, H. (eds.), Chapter 5, p153-181, Kluwer Academic Publishers, 1996.
doi:10.1007/b102605

[KA01] Kasten, O., and Langheinrich, M., “First Experiences with Bluetooth in the Smart-Its Distributed Sensor Network,” Proc. Workshop on Ubiquitous Computing and Communications, Barcelona, Spain, September 2001.
http://citeseer.ist.psu.edu/article/kasten01first.html

[KA02] Kalyan, P., et al., “Experiences Applying Parallel and Interoperable Network Simulation Techniques in On-Line Simulations of Military Networks,” 16th Workshop on Parallel and Distributed Simulation, p88-95, 2002.
doi:10.1109/PADS.2002.1004205

[KA05] Kaufman, C., “Internet Key Exchange (IKEv2) Protocol,” Internet RFC 4306, December 2005.

[KA06] Kumazoe,K., et al., "Can High-Speed Transport Protocols be deployed on the Internet? Evaluation through experiments on JGNII," International Workshop on Protocols for Fast Long-Distance Networks, Nara, Japan, February 2006.
http://www.kyushu.jgn2.jp/research/r_060202/doc/full_k.pdf

[KA06f] Kaneki, K., "TCP-Fusion: A Hybrid Congestion Control Algorithm for High-speed Networks," International Workshop on Protocols for Fast Long-Distance Networks, Nara, Japan, February 2006.
http://wil.cs.caltech.edu/pfldnet2007/paper/TCP_fusion.pdf

[KE05] Kent, S., and Atkinson, R., "Security Architecture for the Internet Protocol," RFC 4301, December 2005.

[KI06] Kim, K., "Key Technologies for the Next Generation Wireless Communications," in Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis, Seoul, Korea, p266-269, October 22 - 25, 2006.
doi:10.1145/1176254.1176319

[KL01] Klensin, J., (ed.), "Simple Mail Transfer Protocol," RFC 2821, March 2001.

[KO06] Kohler, E., et Al., "Datagram Congestion Control Protocol (DCCP)," Internet RFC 4340, March 2006.

[KR02] Krishnan, R., et al., "Explicit Transport Error Notification (ETEN) for Error-Prone Wireless and Satellite Networks," Computer Networks Vol.46, Iss.3, p343-362, October 2004.
doi:10.1016/j.comnet.2004.06.012

[KU05] Kurkowski, S., Camp, T. and Colagrosso, M., "Manet Simulation Studies: The Incredibles," Mobile Computing and Communications Review, Vol.9, Iss.4, p50-61, October 2005.
doi:10.1145/1096166.1096174

[LE05] LeBrun, J., et al., "Knowledge-based Opportunistic Forwarding in Vehicular Wireless Ad-hoc Networks," 61st IEEE Vehicular Technology Conference, 2005. VTC 2005-Spring., p2289-2293 Vol. 4, 30 May-1 June 2005.
doi:10.1109/VETECS.2005.1543743

[LE06] Lee, S., et al., "Orbit Design and Optimization Based on Global Telecommunication Performance Metrics," IEEE Aerospace conference, Big Sky, Montana, USA, 11pp-, March 2006.
doi:10.1109/AERO.2006.1655772

[LI06] Lindgren. A., and Doria, A., "Probabilistic Routing Protocol for Intermittently Connected Networks," draft-lindgren-dtnrg-prophet, Internet-Draft, work-in-progress, February 2008.
http://tools.ietf.org/html/draft-irtf-dtnrg-prophet

[LI84] Lin, S., Costello, D., and Miller, M., "Automatic-Repeat-Request Error-Control Schemes," IEEE Communications Magazine, Vol.22, No.12, p5-17, December 1984.
ISSN: 0163-6804

[MA02] Reiff-Marganiec, S., "Runtime Resolution of Feature Interactions in Evolving Telecommunications Systems," PhD Thesis, University of Glasgow, Glasgow (UK), May 2002.
http://www.cs.le.ac.uk/people/srm13/publications/thesis.pdf

[MA05] Maki, J., et al., "Operation and Performance of the Mars Exploration Rover imaging system on the Martian surface," Systems, Man and Cybernetics, 2005 IEEE International Conference on, Vol.1, p930-936, October 2005.
doi:10.1109/ICSMC.2005.1571265

[MA05a] Martinez, K., et al., "Glacier Environment Monitoring using Sensor Networks," Proc. Workshop on Real-World Wireless Sensor Networks, Stockholm, Sweden, 2005.
http://www.sics.se/realwsn05/papers/martinez05glacial.pdf

[MC05] McGann, S., and Sicker, D.C., "An Analysis of Security Threats and Tools in SIP-Based VoIP Systems," Proceedings of the 2nd Workshop on Securing Voice over IP, Cyber Security Alliance, 2005.
http://www.colorado.edu/policylab/Papers/Univ_Colorado_VoIP_Vulner.pdf

[MC07] McDonald, P., et al., "Sensor Networking with Delay Tolerance," Proceedings of 16th International Conference on Computer Communications and Networks, ICCCN 2007, p1333-1338, August 2007.
doi:10.1109/ICCCN.2007.4318006

[MO00] Montenegro, G., et al., “Long Thin Networks,” Internet RFC 2757, January 2000.

[MO00s] Morisio, M., et al., “Investigating and Improving a COTS-Based Software Development Process,” icse, 22nd International Conference on Software Engineering (ICSE '00), p31-, 2000.
doi:10.1109/ICSE.2000.10046

[MO03] Möller, U., et al., “Mixmaster Protocol - Version 2,” Draft, July 2003.
http://www.abditum.com/mixmaster-spec.txt

[MO90] Mogul, J. and Deering, S., “Path MTU Discovery,” Internet RFC 1191, November 1990.

[NA06] Naumov, V., Baumann, R., and Gross, T., “An Evaluation of Inter-Vehicle ad hoc Networks based on realistic Vehicular Traces,” in Proceedings of the 7th ACM international Symposium on Mobile Ad Hoc Networking and Computing (Florence, Italy, May 22 - 25, 2006). MobiHoc '06. ACM Press, New York, NY, p108-119, May 2006.
doi:10.1145/1132905.1132918

[NE03] Newman, M., “The Structure and Function of Complex Networks,” SIAM Review Vol.45, Iss.2, p167–256, 2003.
doi: 10.1137/S003614450342480

[OT04] Ott, J., and Kutscher, D., “The ‘Drive-thru’ Architecture: WLAN-based Internet Access on the Road”, Proc. IEEE Vehicular Technology Conference VTC 2004-Spring, Milano, Italy, p 2615- 2622, Vol.5, May 2004.
doi:10.1109/VETECS.2004.1391395

[OT06] Ott, J., Kutscher, D., and Dwertmann, C., “Integrating DTN and MANET routing,” In Proceedings of the 2006 SIGCOMM Workshop on Challenged Networks (Pisa, Italy, September 11 - 15, 2006). CHANTS '06. ACM Press, New York, NY, p221-228. 2006
doi:10.1145/1162654.1162659

[PA03] Pawlikowski, P., “Do Not Trust All Simulation Studies of Telecommunication Networks,” Proc. International Conference on Information Networking (ICOIN) 2003, Jeju Island, Korea, LNCS 2662, p899-908, 2003.
doi:10.1007/b13389

[PA98] Padhye, J., et al., “Modeling TCP Throughput: a Simple |Model and its Empirical Validation,” SIGCOMM Comput. Commun. Rev. Vol.28, Iss.4, p303-314, October 1998.
doi:10.1145/285243.285291

[PE03] Perkins, C., Belding-Royer, E., and Das, S., “Ad hoc On Demand Distance Vector (AODV) Routing,” Internet RFC 3561, July 2003.

[PO80] Postel, J., “User Datagram Protocol,” Internet RFC 768, August 1980.

[PO81] Postel, J., “Transmission Control Protocol”, Internet RFC 793, September 1981.

[PO81i] Postel, J., (ed.), “Internet Protocol – DARPA Internet Program Protocol Specification”, Internet RFC 791, September 1981.

[PO85] Postel, J., and Reynolds, J., “File Transfer Protocol (FTP),” Internet RFC 959, October 1985.

[QU03] Interview: A Conversation with Jim Gray. *Queue* Vol 1, Issue 4, p8-17, June 2003.
doi.acm.org/10.1145/864056.864078

[RA01] Ramakrishnan, K., et al., “The Addition of Explicit Congestion Notification (ECN) to IP,” Internet RFC 3168, September 2001.

[RA08] Ramadas, M., et al., “Licklider Transmission Protocol – Specification”, Internet RFC 5326, September 2008.

[RA07t] Ramadas, M., “Study of a Protocol and a Priority Paradigm for Deep Space Data Communication,” PhD thesis, Russ College of Engineering and Technology, University of Ohio, June 2007.

[RE06] Rekhter, Y. et al, “A Border Gateway Protocol 4 (BGP-4),” Internet RFC 4271, January 2006.

[RE98] Reed, D., Saltzer, J. and Clark, D., “Commentaries on Active Networking and End-to-End Arguments,” IEEE Network, Vol.12, ISs.3, p66-71, May/June 1998.
doi:10.1109/65.690972

[RH05] Rhee, I., and Xu, L., “Limitations of Equation-Based Congestion Control,” SIGCOMM Comput. Commun. Rev. Vol.35, Iss. 4, p49-60, October 2005.
doi:10.1145/1090191.1080099

[RI00] Rigney, C., et Al., "Remote Authentication Dial In User Service (RADIUS)," Internet RFC 2865, June 2000.

[RI00e] Rigney, C., Willats, W, and Calhoun, P, "RADIUS extensions", Internet RFC 2869, June 2000.

[RI02] Ripeanu, M., Foster, I., and Iamnitchi, A., "Mapping the Gnutella network: Properties of large-scale peer-topeer systems and implications for system design," IEEE Internet Computing Vol 6, Iss.1, p50–57, January 2002.
doi:10.1109/4236.978369

[RI05] Rice, J., "Seaweb Acoustic Communication and Navigation Networks," Proc. International Conference on Underwater Acoustic Measurements: Technologies & Results, Heraklion, Crete, Greece, 2005.

[RO01] Rowstron A. and Druschel. P., "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," In International Conference on Distributed Systems Platforms (Middleware), LNCS 2218, p329-350, November 2001.
ISBN:3-540-42800-3

[RY03] Rydh, C., "Environmental Assessment of Battery Systems: Critical Issues for Established and Emerging Technologies," Doktorsavhandlingar vid Chalmers tekniska högskola, Ny serie nr 2030 ,ISSN 0346-718X, ISBN 91-7291-348-7, 2003.
http://homepage.hik.se/staff/tryca/battery/abstracts.htm

[SA02] Saroiu, S. et al., "An Analysis of Internet Content Delivery Systems," ACM SIGOPS Operating Systems Review, Vol.36, p315–328, Winter 2002.
doi:10.1145/844128.844158

[SA07] Sathiaseelan, A.; Fairhurst, G., "Performance of VoIP using DCCP over a DVB-RCS Satellite Network," IEEE International Conference on Communications, ICC '07, p13-18, June 2007.
doi:10.1109/ICC.2007.12

[SA84] Saltzer, J., Reed, D., and Clark, D. "End-to-end Arguments in System Design," ACM Trans. Comput. Syst. Vol.2, Iss.4, p277-288, November 1984.
doi:10.1145/357401.357402

[SC03] Schulzrinne, H., et al., "RTP: A Transport Protocol for Real-Time Applications," Internet RFC 3550, July 2003.

[SC05] Scott, K. "Disruption Tolerant Networking Proxies for On-the-move Tactical Networks," Military Communications Conference. MILCOM 2005 , Atlantic City, New Jersey, USA, p3226-3231, Vol. 5, October 2005.
doi:10.1109/MILCOM.2005.1606153

[SC05c] Scott, K., "CCSDS Concept Paper – Extended Signaling for SCPS-TP," CCSDS Concept Paper, June 2005.
http://www.ccsds.org/

[SC07] Scott, K., and Burleigh, S., "Bundle Protocol Specification", Internet RFC 5050, November 2007.

[SE06] Seligman, M., Fall, K., and Mundur, P., "Alternative Custodians for Congestion Control in Delay Tolerant Networks," in Proceedings of the 2006 SIGCOMM Workshop on Challenged Networks (Pisa, Italy, September 11 - 15, 2006). CHANTS '06. ACM Press, New York, NY, p229-236, September 2006.
doi:10.1145/1162654.1162660

[SH03] Shah, R., et al., "Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks," Ad Hoc Networks, Vol.1, Iss.2-3, p215-233, September 2003.
doi:10.1016/S1570-8705(03)00003-9

[SH03d] Shimonski, R., et al., "Building DMZs for Enterprise Networks," Syngress Publishing, 2003.
ISBN 1928994741

[SH04] Shukla, S., Bulusu, N., and Jha, S., "Cane-toad Monitoring in Kakadu National Park Using Wireless Sensor Networks," Proc. Network Research Workshop, as part of 18th APAN Meetings, Cairns, Australia, July 2004.
http://citeseer.ist.psu.edu/shukla04canetoad.html

[SP88] Spafford, E., "The Internet Worm Program: An Analysis," Purdue Technical Report CSD-TR-823, Department of Computer Sciences, Purdue University, West Lafayette, IN, 1988.
doi:10.1145/66093.66095

[SR02] Srisuresh, P., et al., “Middlebox Communication Architecture and Framework,” Internet RFC 3303, August 2002.

[ST00] Stewart, R., et al., “Stream Control Transmission Protocol,” Internet RFC 2960, October 2000.

[SU07] Su, J., et al., “Haggle: Clean Slate Networking for Mobile Devices,” University of Cambridge Computer Laboratory Technical Report, ISSN 1476-2986, UCAM-CL-TR-680, January 2007.

[SY07] Symington, S., et al., “Bundle Security Protocol Specification”, Internet-draft, February 2008, work-in-progress.
http://tools.ietf.org/html/draft-irtf-dtnrg-bundle-security

[SY07m] Symington, S., Durst, R., and Scott, K., “Delay-Tolerant Networking Custodial Multicast Extensions,” Internet-draft, May 2008, work-in-progress.
http://tools.ietf.org/html/draft-symington-dtnrg-bundle-multicast-custodial

[TO01] Touch. J., “Dynamic Internet Overlay Deployment and Management using the X-Bone,” Computer Networks, Volume 36, Issues 2-3, p 117-135, July 2001.
doi:10.1016/S1389-1286(01)00172-4

[TO03] Townes, S., et al., “Operational Demonstration of ka-band Telecommunications for the Mars Reconnaissance Orbiter,” Aerospace Conference, , Big Sky, Montana, USA, Vol.8, p3565-3580 March 8-15, 2003.
ISBN: 0-7803-7651-X

[TO05] Touch, J., et al, “A Global X-Bone for Network Experiments,” Proc. IEEE Tridentcom 2005, Trento Italy, p194-203, March 2005.
doi:10.1109/TRIDNT.2005.2

[TS05] Tsenov, T. et al., “Advanced Authentication and Authorization for Quality of Service Signaling,” 1st IEEE workshop on Security and QoS in communications networks, Athens, Greece, p224-235 September 2005.
doi:10.1109/SECCMW.2005.1588317

[VA99] Varga, A., "Using the OMNeT++ discrete event simulation system in education," IEEE Transactions on Education, vol.42, no.4, pp.11, November 1999.
doi:10.1109/13.804564

[WA05] Wang, R., et al., "The Digital Study Hall," Technical Report TR-723-05, Computer Science Department, Princeton University, March 2005. http://www.eecs.berkeley.edu/~mattkam/publications/PrincetonTR2005.pdf

[WA90] Waitzman, D. "A standard for the transmission of IP datagrams over avian carriers", Internet RFC 1149, April 1 1990.

[WI05] Widmer, J., and Le Boudec, J-Y, "Network Coding for Efficient Communication in Extreme Environments," Proceedings of ACM SIGCOMM Workshop on Delay Tolerant Networking and Related Topics, Philadelphia, PA, p284-291, 2005. doi:10.1145/1080139.1080147

[WI91] Wilkinson, C, "X.400 Electronic Mail," Electronics & Communication Engineering Journal, Vol.3, Iss.3, p129-136, June 1991. ISSN: 0954-0695

[WO03] Wong, K., et al., "BUSNet: Model and Usage of Regular Traffic Patterns in Mobile Ad Hoc Networks for Inter-Vehicular Communications", Proc. ICT 2003, Thailand, April, 2003. http://citeseer.ist.psu.edu/744836.html

[XI02] Hong, X., et al., "Scalable Routing Protocols for Mobile ad hoc Networks," IEEE Network, Vol.16, Iss.4, p11-21, July/August 2002. doi:10.1109/MNET.2002.1020231

[ZH03] Zhou, H., "A Survey on Routing Protocols in MANETs," Michigan State University Technical Report, MSU-CSE-03-08, March 2003. http://www.cse.msu.edu/cgi-user/web/tech/document?ID=532

[ZH04] Zhao, W., Ammar, M., and Zegura, E. "A Message Ferrying Approach for Data Delivery in Sparse Mobile ad hoc Networks," In ACM MobiHoc 2004, Tokyo Japan, p187–198, 2004. doi:10.1145/989459.989483

[ZH04m] Zhang, B., Sukhatme, G. and Requicha, A., "Adaptive Sampling for Marine Microorganism Monitoring," in IEEE/RSJ International Conference on Intelligent Robots and Systems, p1115-1122, Vol.2 2004. doi:10.1109/IROS.2004.1389546

[ZH04z] Zhang, P., et al., “Hardware Design Experiences in ZebraNet,” Proc. ACM Conference on Embedded Networked Sensor Systems, Baltimore, p227–238, 2004.
doi:10.1145/1031495.1031522

[ZH06] Zhang, Z., “Routing In Intermittently Connected Mobile Ad Hoc Networks And Delay Tolerant Networks: Overview And Challenges”, IEEE Communications Surveys and Tutorials, Vol. 8, Iss.1, 2006.
doi:10.1109/COMST.2006.323440

[ZH06d] Zhao, W., et al, “Capacity Enhancement using Throwboxes in DTNs,” In Proc. IEEE Intl Conf on Mobile Ad hoc and Sensor Systems (MASS), October 2006.
http://prisms.cs.umass.edu/brian/pubs/zhao.mass2006.pdf

Appendix A - Comparing LTP, LTP-T, the BP and SFTP

This Appendix presents additional results related to the one-hop tests described in Section 6.1. In all cases below, (except as otherwise noted), each data-point is the average over five runs of the relevant test.

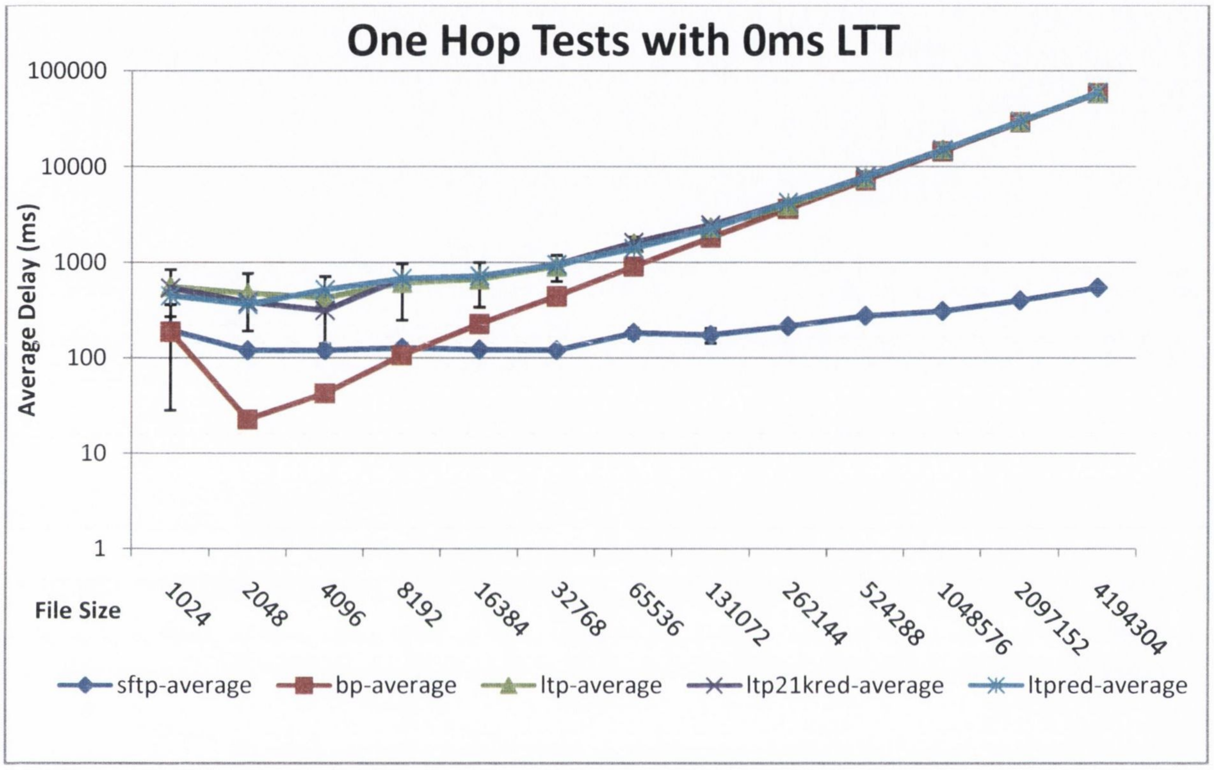


Figure A.1 – One hop tests with 0ms LTT (1 standard deviation error bars are shown for the SFTP and B-LTP data series)

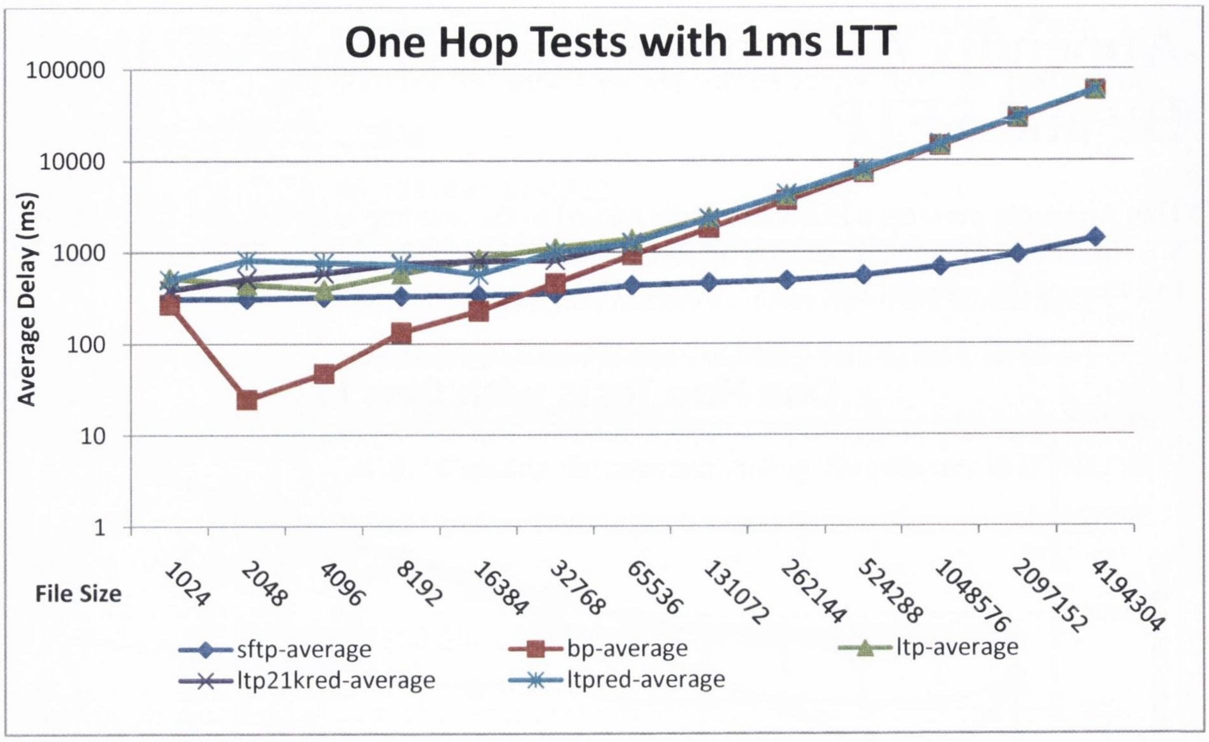


Figure A.2 - One hop tests with 1ms LTT

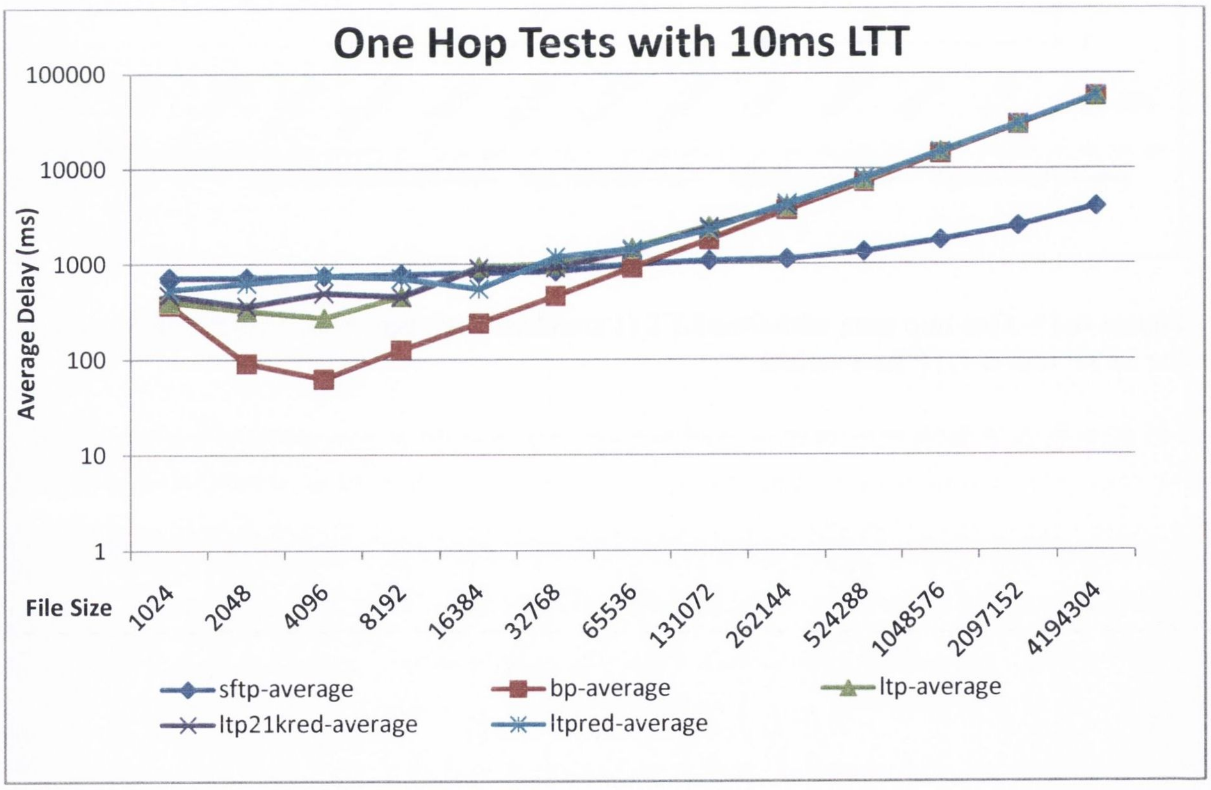


Figure A.3 - One hop tests with 10ms LTT

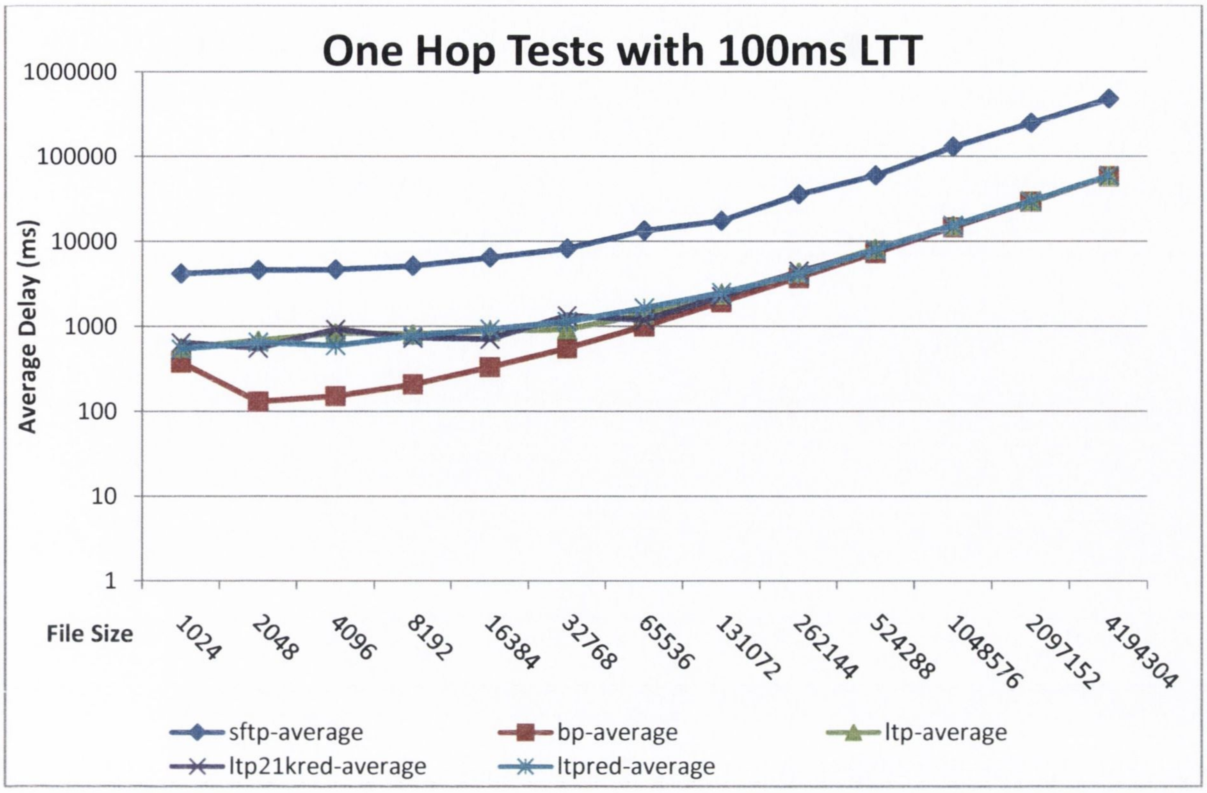


Figure A.4 – One hop tests with 100ms LTT

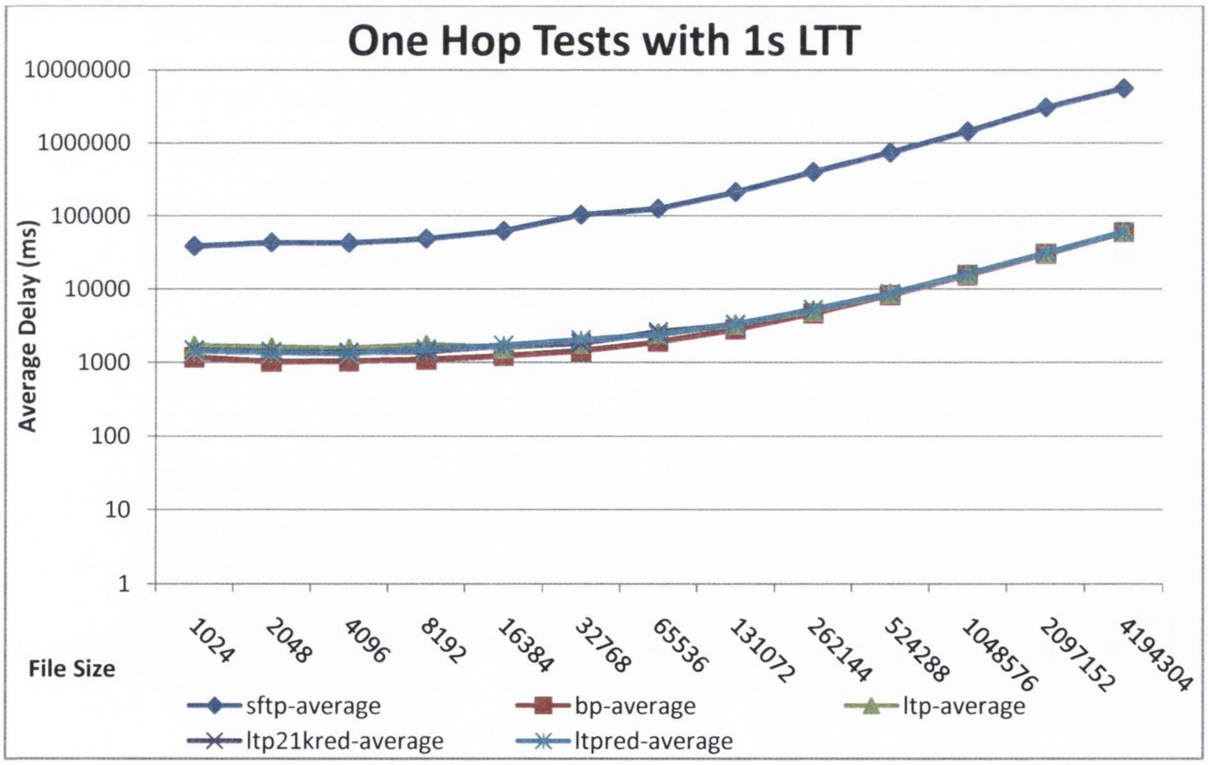


Figure A.5 – One hop tests with 1s LTT

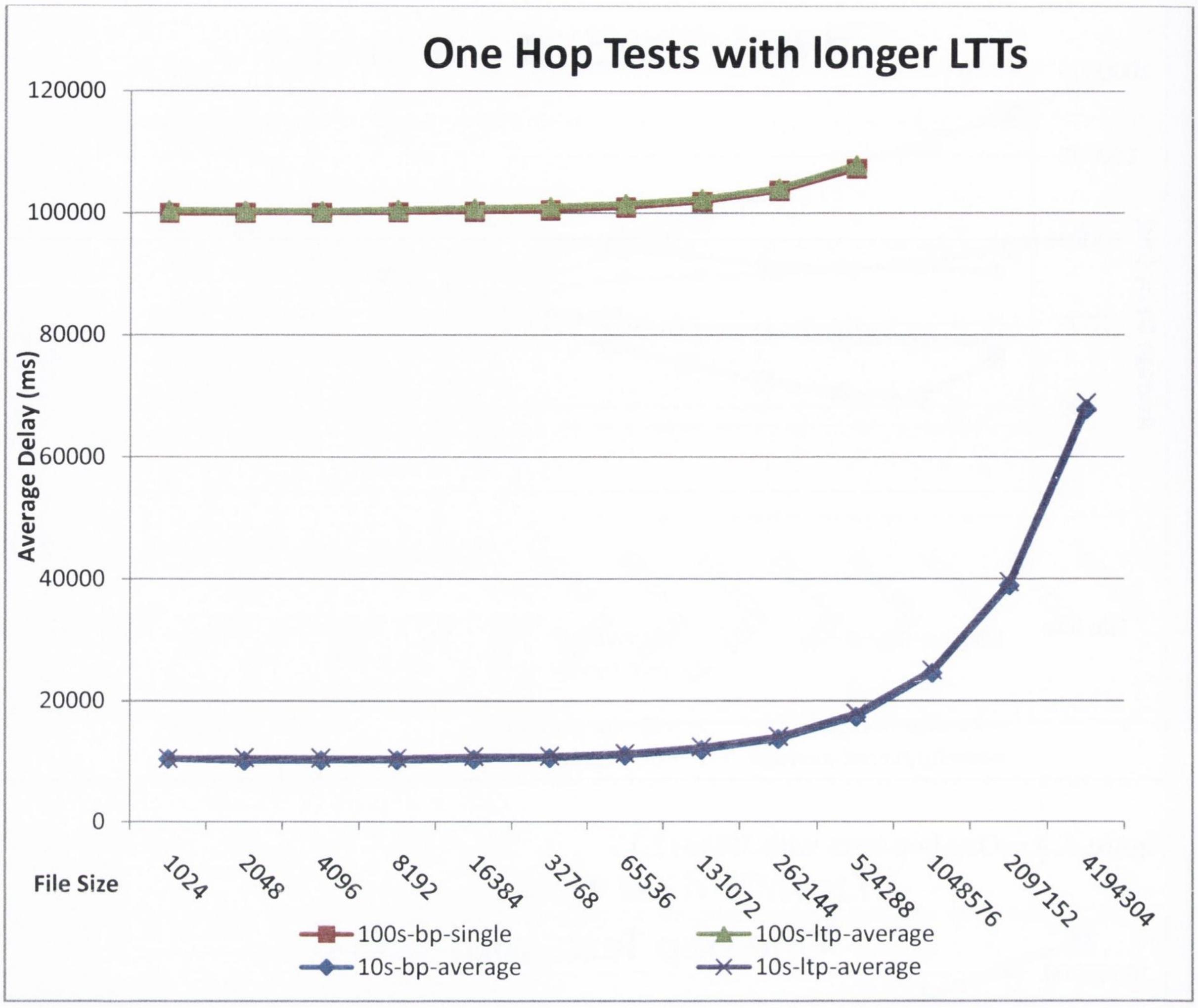


Figure A.6 – One hop tests with longer LTTs – only for BP and LTP (100s BP single run, not averaged)

Appendix B - Martian Emulation Runs

This Appendix contains diagrams of the test runs described in Chapter 6.2. Section 3.3.2 provides details of how to interpret these figures. Where the data transferred was an image, the image-as-received is also shown - in some cases where the test involved partial reliability and packet errors, these images can be seen to be corrupted.

In some cases, runs are included here which are of interest, but where, due to bugs (subsequently fixed, unless otherwise stated) in the LTP-T implementation, the runs were not used in the goodput calculations shown in Chapter 6.2. All such runs are commented. Appendix C presents the numeric data derived from the test runs that were used in the goodput calculations in Chapter 6.2 and also lists the relevant section number from this Appendix.

Figure B.1, (cropped from B.2.1 below), shows detail of the negative vertical axis of the graphics used below (the same in all cases). As can be seen, each scheduled node in our Martian network emulation is shown on the negative y-axis. Rectangles coming down from the x-axis indicate periods during which the node in question is availble to receive data, i.e. inbound contacts. In the figure, one can see that the dsn-gw has many such contacts, and that there is a "burst" of contacts around 400-500 seconds into the run. (While this aspect of the plots is perhaps of limited value here, they are automatically generated and are useful in other contexts, e.g. when debugging.)

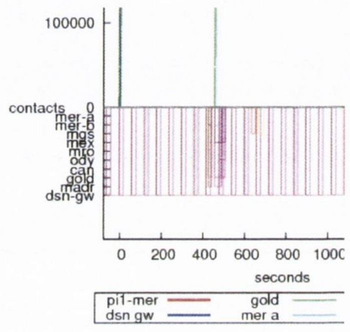


Figure B.1: Contacts details.

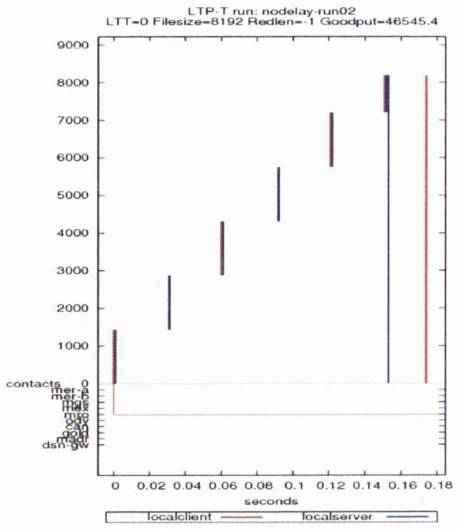
For traceability, the section titles below are taken from the log file names. The actual log files (and graphics) can be found below http://dtn.dsg.cs.tcd.ie/sft - see the README there for details.

B.1 Local Runs with no Additional Delay

There are no additional delays involved here, the sender and receiver are both on the same host, but rate control is turned on (sending segments at most every 20ms). In these cases, LTP was run with a schedule of 2 seconds on from each 10 seconds. (Ignore the contacts part of the diagrams for these runs.)

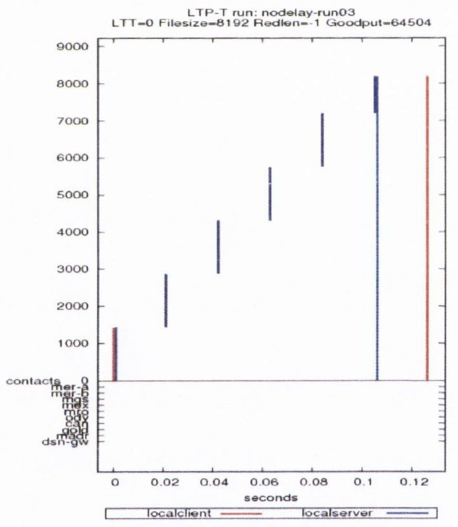
These runs are not used in the calculations in Chapter 6.2.

B.1.1 nodelay-run02



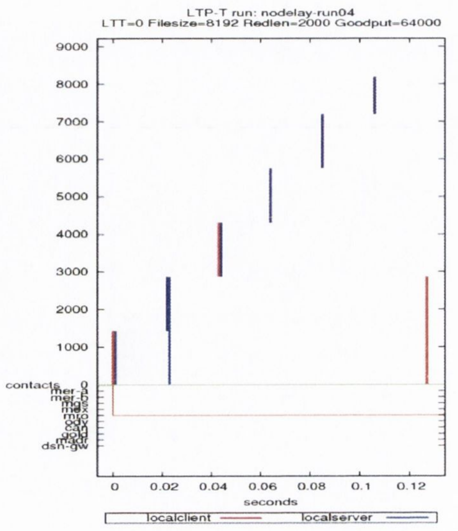
This is a nominal run.

B.1.2 nodelay-run03



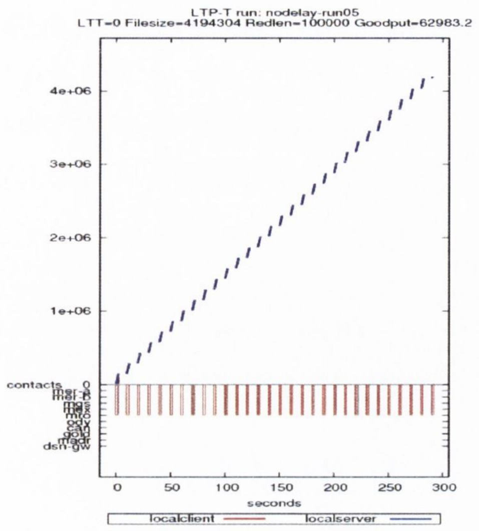
This is a nominal run.

B.1.3 nodelay-run04



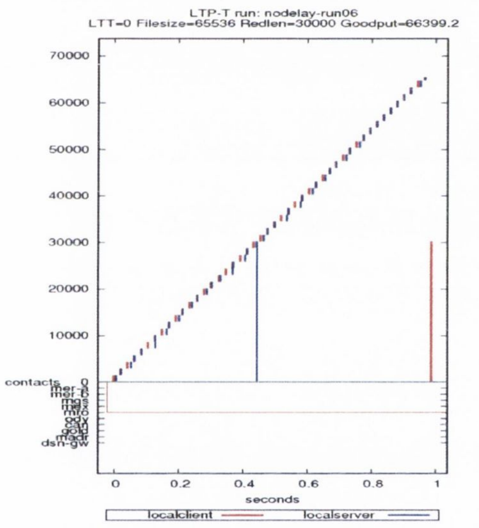
This is a nominal run.

B.1.4 nodelay-run05



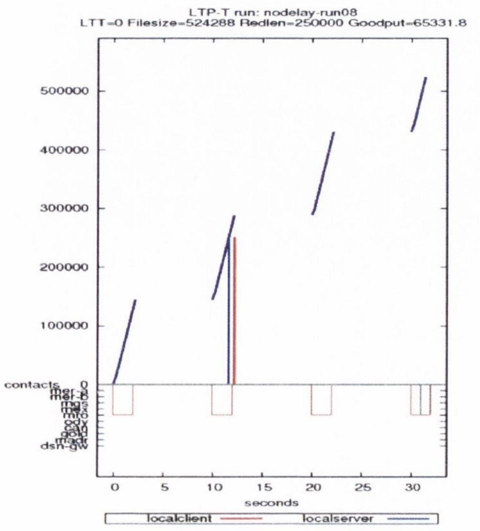
This is a nominal run. Although present, the report segment is not easily visible at this scale.

B.1.5 nodelay-run06



This is a nominal run.

B.1.6 nodelay-run08

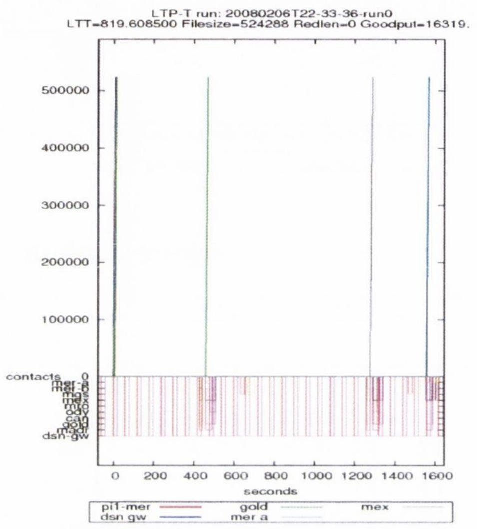


This is a nominal run.

B.2 Error-Free Runs from Earth to Mars

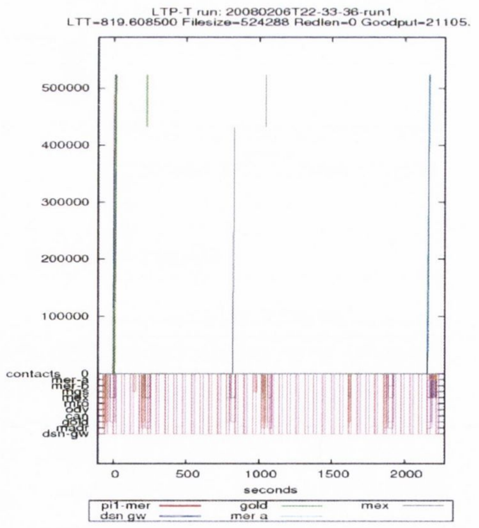
This Section presents a set of runs where node PI1 (on Earth) sends blocks to node MER-A (on Mars) via the dsn-gw node and whichever DSN Earth stations and/or orbiters are appropriate according to the prevailing schedule. The second run below shows the LTP block being split over two contacts between the Goldstone DSN station and the Mars Express orbiter.

B.2.1 20080206T22-33-36-run0



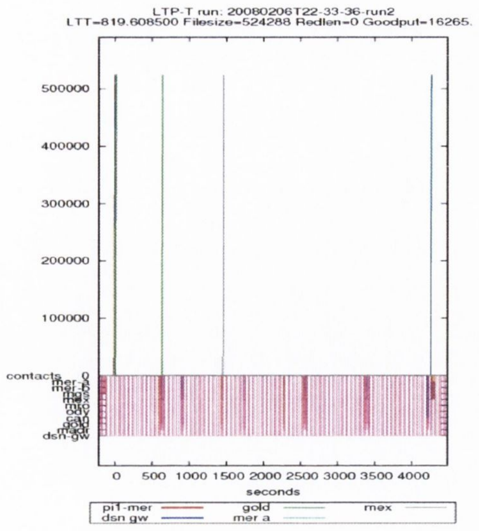
This is a nomimal all-green run.

B.2.2 20080206T22-33-36-run1



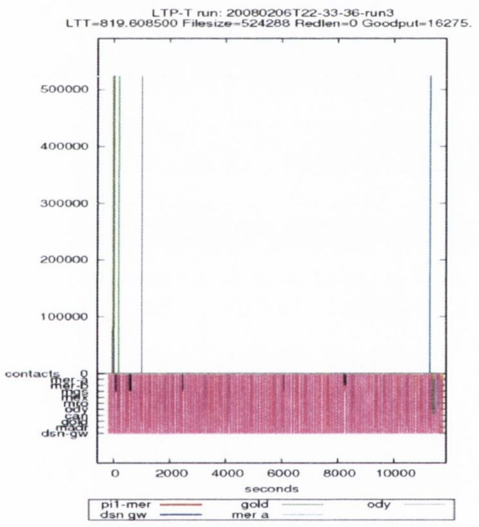
This is a nominal all-green run showing the use of a second contact.

B.2.3 20080206T22-33-36-run2



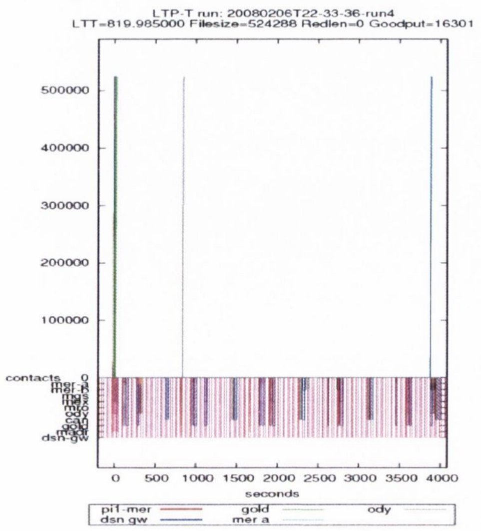
This is a nomimal all-green run.

B.2.4 20080206T22-33-36-run3



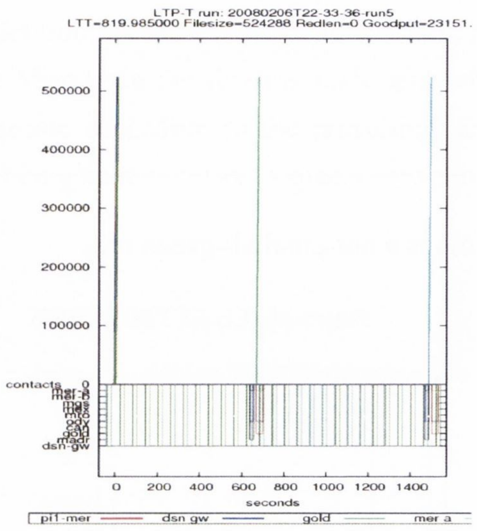
This is a nomimal all-green run.

B.2.5 20080206T22-33-36-run4



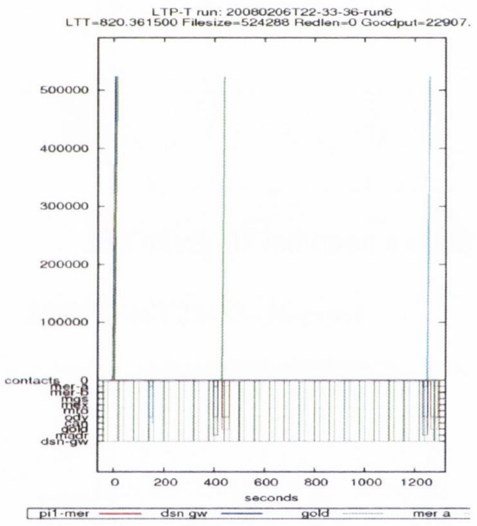
This is a nomimal all-green run.

B.2.6 20080206T22-33-36-run5

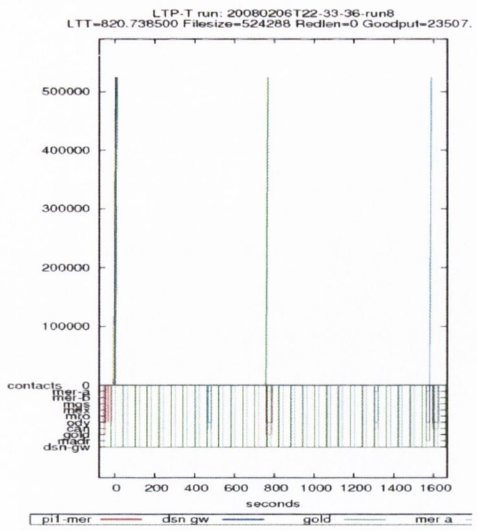


This is a nomimal all-green run.

B.2.7 20080206T22-33-36-run6

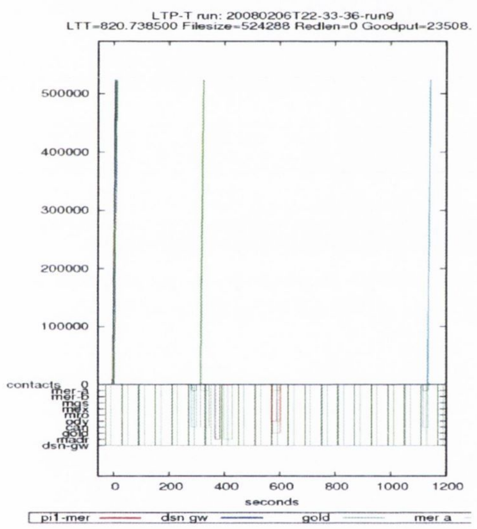

LTP-T run: 20080206T22-33-36-run6
LTT=820.361500 Filesize=524288 Redlen=0 Goodput=22907.
500000
400000
300000
200000
100000
contacts 0
mer-a
mer-b
mgs
mex
mto
ody
cap
gold
madr
dsn-gw
0 200 400 600 800 1000 1200
seconds
pi1-mer
dsn gw
gold
mer a

B.2.9 20080206T22-33-36-run8



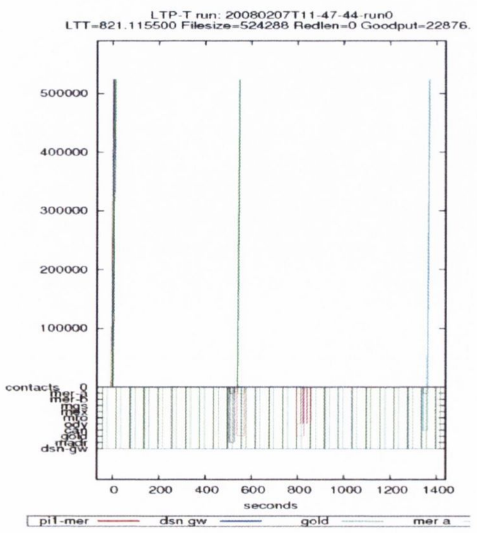
This is a nomimal all-green run.

B.2.10 20080206T22-33-36-run9



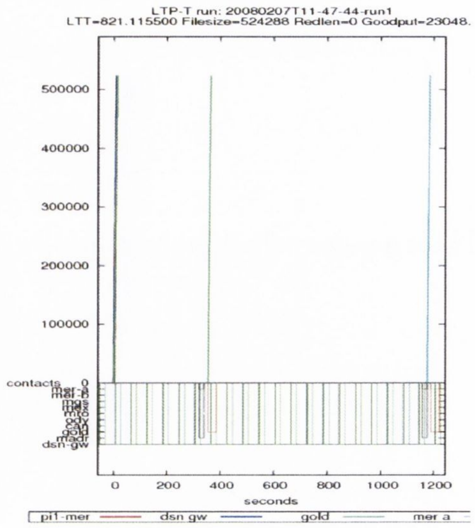
This is a nomimal all-green run.

B.2.11 20080207T11-47-44-run0



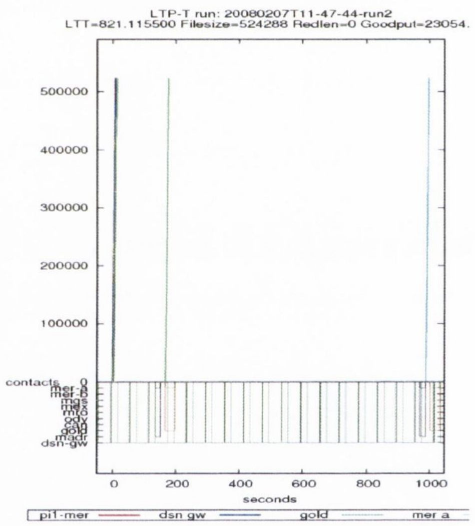
This is a nominal all-green run.

B.2.12 20080207T11-47-44-run1



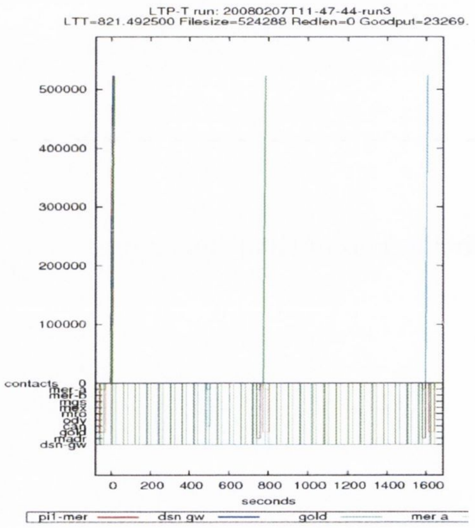
This is a nominal all-green run.

B.2.13 20080207T11-47-44-run2



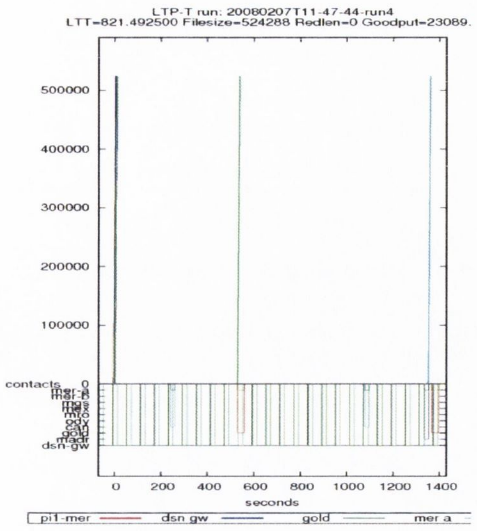
This is a nominal all-green run.

B.2.14 20080207T11-47-44-run3



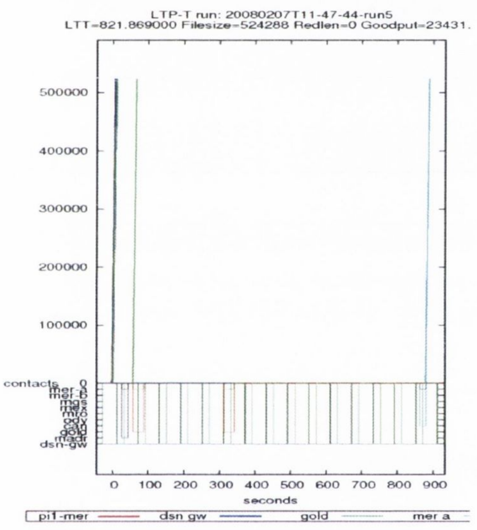
This is a nominal all-green run.

B.2.15 20080207T11-47-44-run4



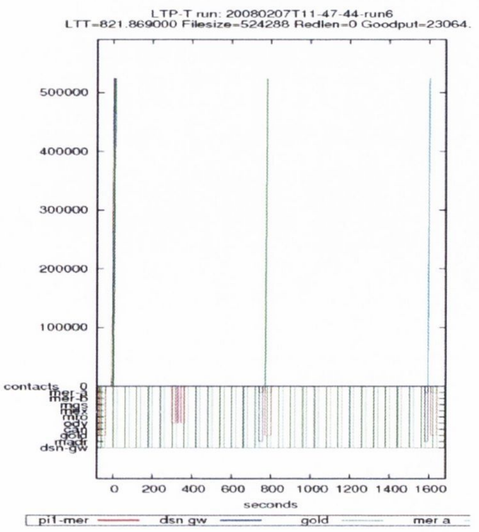
This is a nominal all-green run.

B.2.16 20080207T11-47-44-run5



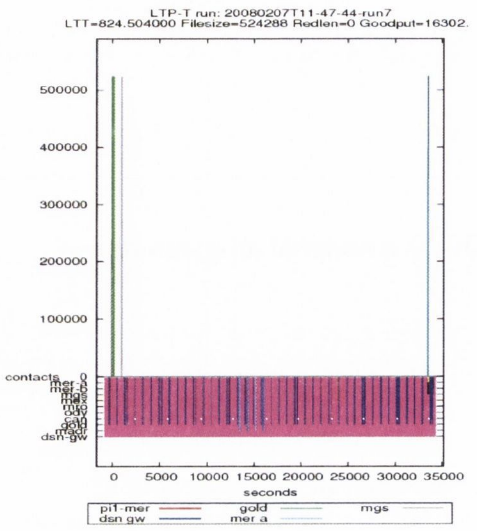
This is a nominal all-green run.

B.2.17 20080207T11-47-44-run6



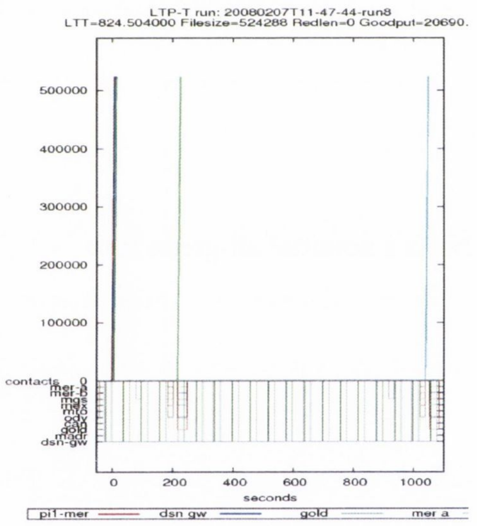
This is a nominal all-green run.

B.2.18 20080207T11-47-44-run7



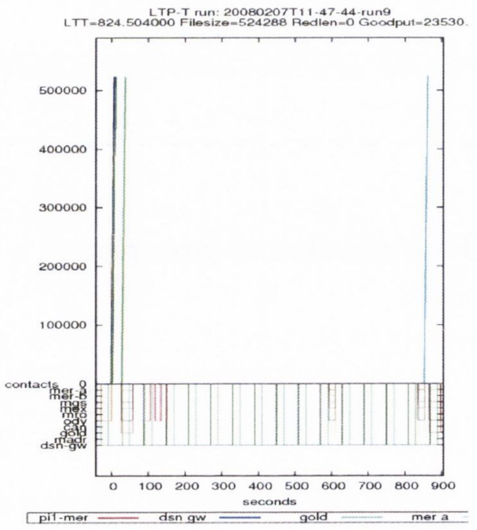
This is a nominal all-green run. (With a significant delay before MGS has a contact with MER-A.)

B.2.19 20080207T11-47-44-run8



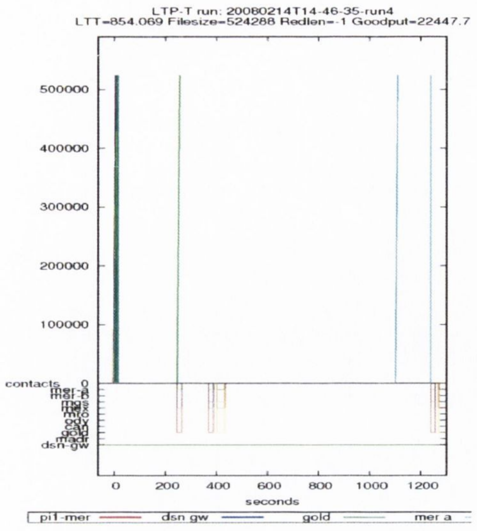
This is a nominal all-green run.

B.2.20 20080207T11-47-44-run9



This is a nominal all-green run.

B.2.21 20080214T14-46-35-run4

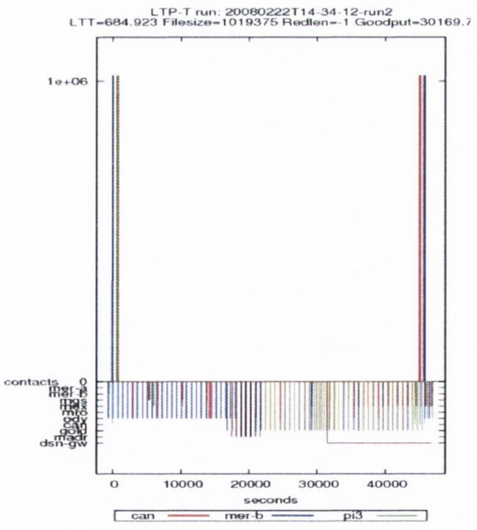


This is a nominal run, except that the test was terminated as the Report Segment from MER-A was in-flight. Transmission of the Report segment from MER-A was properly delayed as can be seen. (This is not used in Section 6.2.)

B.3 Error-Free Runs from Mars to Earth

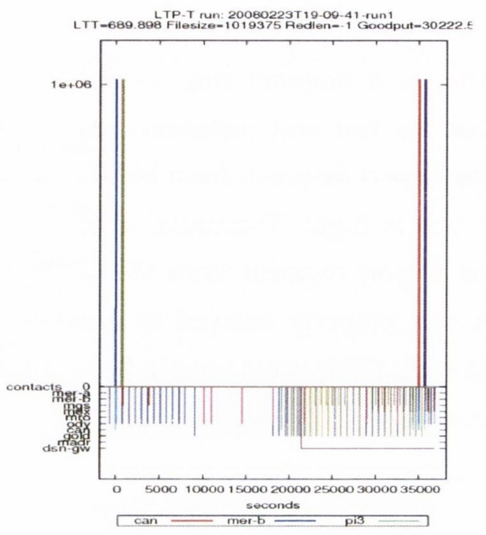
In these runs, blocks were sent from Mars (node MER-B) to Earth (node PI3), via whichever DNS Earth stations and orbiters are appropriate according to the prevailing schedule. There were no errors introduced and all blocks were fully red.

B.3.1 20080222T14-34-12-run2



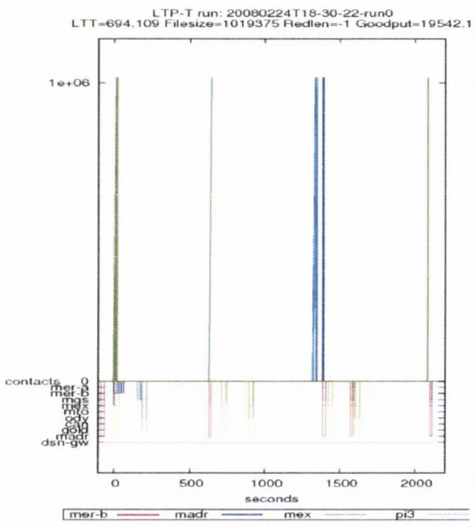
In this case, the Canberra node had to wait until the next mornnig to send the Report segment to MER-B (due to the schedule). This run is not included in calculations in Section 6.2.

B.3.2 20080223T19-09-41-run1



This is a nominal run.

B.3.3 20080224T18-30-22-run0

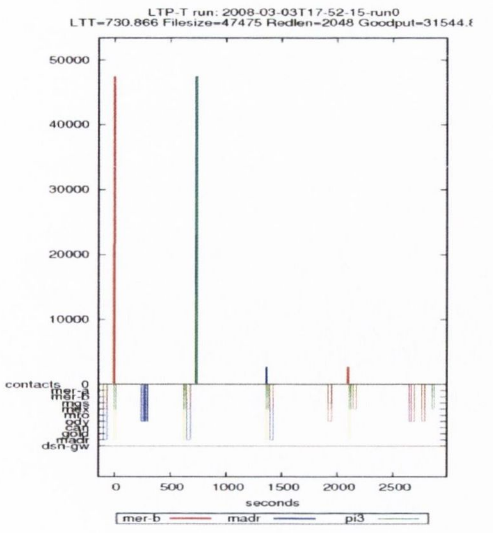


This is a nominal run, except the test was terminated as the last Report ackowledgement was in flight from MarsExpress to Madrid. This run is not included in the calculations in Section 6.2.

B.4 Mars to Earth: 2K Red with 1 Percent errors

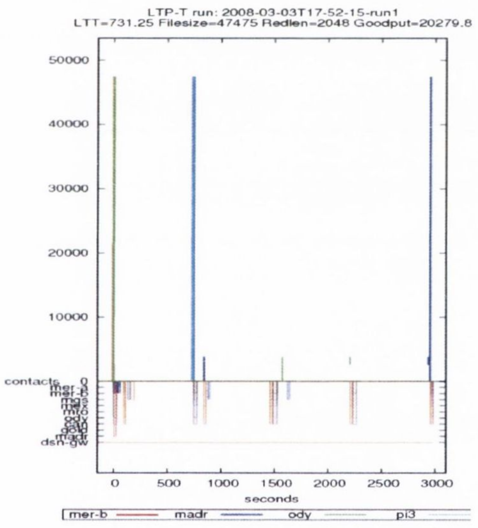
This section shows runs as in the previous secction but with the first 2K bytes of the block being red, and with 1 percent of packets from Mars to Earth being dropped (by the m2e router, using Netem). The initial report segments can be seen to only cover the 2K red part, and in some cases re-transmission is visible, followed by a final report segment covering the entire block.

B.4.1 2008-03-03T17-52-15-run0



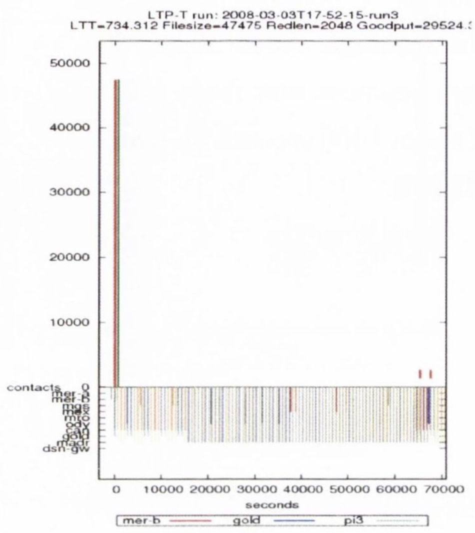
This is a nominal run. The last message shown is the Report Segment sent from Madrid at about 1400 seconds arriving at MER-B.

B.4.2 2008-03-03T17-52-15-run1



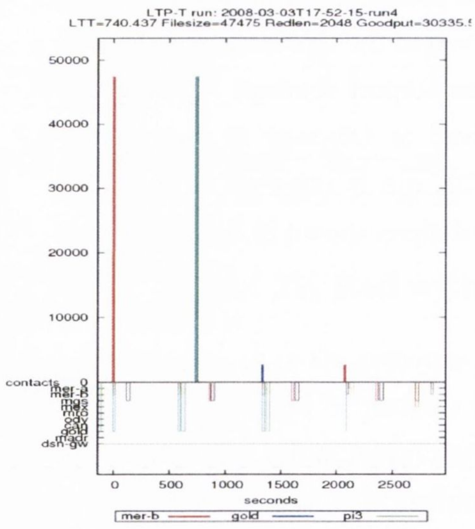
Here we see re-transmission of red Data segments from Odyssey to Madrid. The final Report segment shown is from Madrid. Since test runs are terminated approximately two LTTs after the initial set of blocks have arrived at the destination node the last Report segment had not arrived at Odyssey at that point. This run is not used in the calculations shown in Section 6.2.

B.4.3 2008-03-03T17-52-15-run3



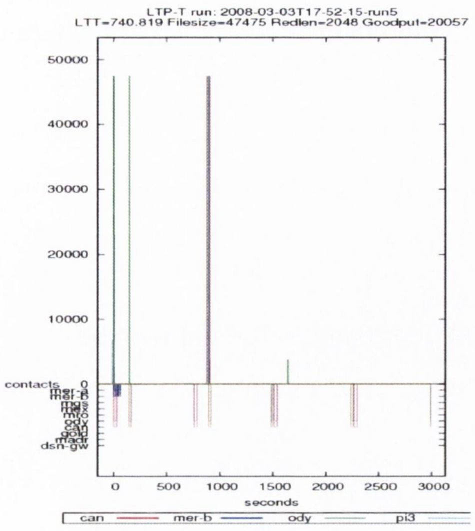
In this run the Goldstone node crashed after the block had been successfully delivered to the PI3 node, but before the Report segment had been sent back to MER-B. MER-B then retransmitted the last red Data segment twice before closing the LTP session. In this case, the schedule led to a significant delay before these re-transmissions could occur, hence the long duration here. (The scale of the diagram also leads to the Goldstone and PI3 nodes' being indistinguishable.) This run is not included in the calculations in Section 6.2.

B.4.4 2008-03-03T17-52-15-run4



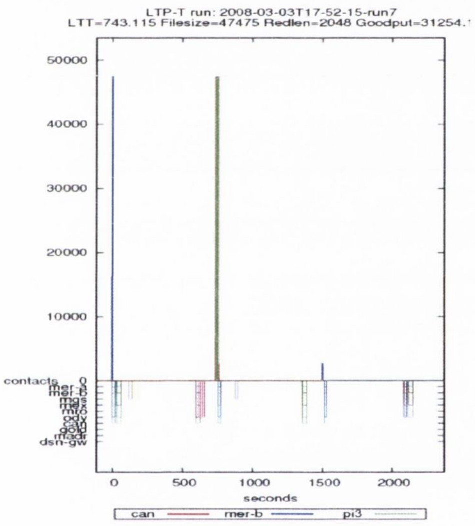
This is a nominal run.

B.4.5 2008-03-03T17-52-15-run5



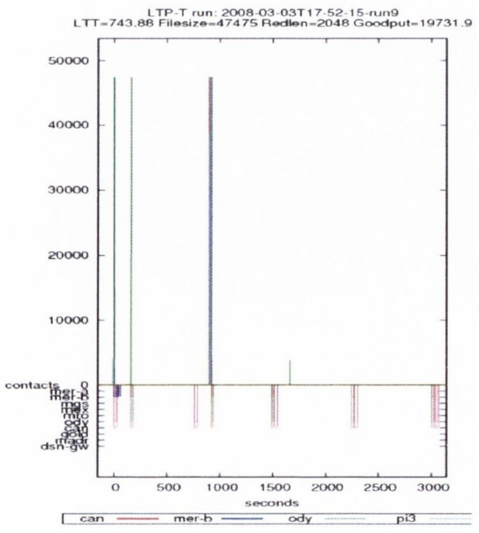
In this run, we see the segments being delayed on board the Odyssey orbiter before being forwarded to Earth. The red part at arrival was 4556 bytes long, due to red part extension. Contrast run7 below where the red part on arrival is shorter since there are fewer hops.

B.4.6 2008-03-03T17-52-15-run7



This is a nomimal run. The size of the red part at arrival here was 3795 bytes, which is shorter than in run5 above, where the additional hop resulted in the red part being further extended.

B.4.7 2008-03-03T17-52-15-run9

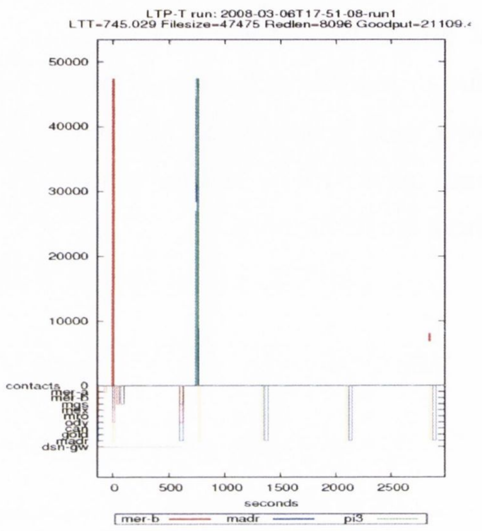


This is a nominal run.

B.5 Mars to Earth: 8K Red with 1 Percent errors

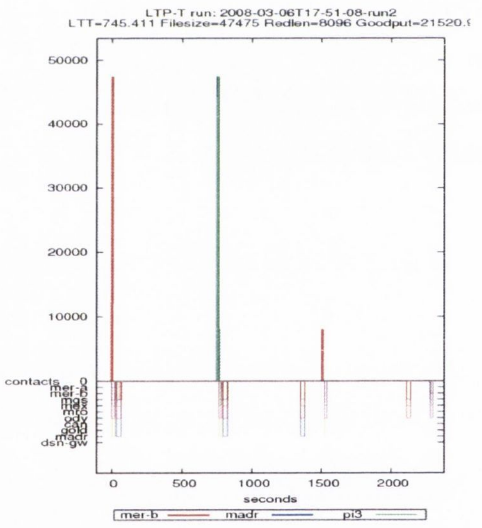
This section shows runs as in the previous section, but this time with the first 8K bytes of the block being red.

B.5.1 2008-03-06T17-51-08-run1



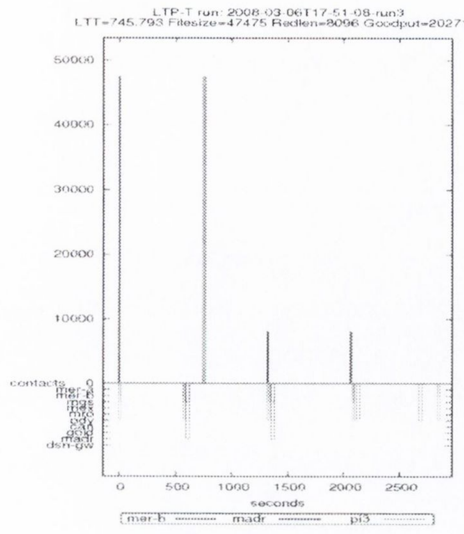
This is another run where the Madrid node crashed. However, we can see the re-transmission of the last red Data segment in this run. This run is not included in the calculations in Section 6.2.

B.5.2 2008-03-06T17-51-08-run2



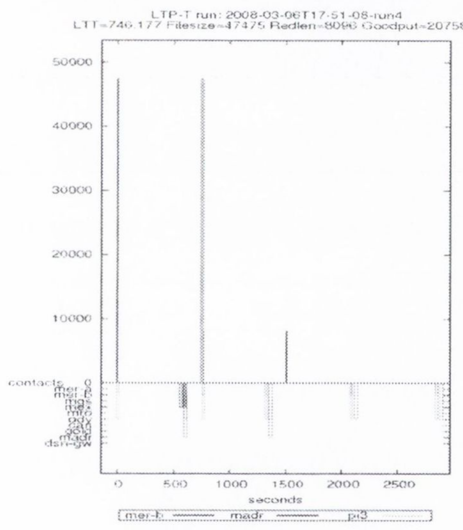
This is a nominal run. The red length on arrival was 8855 bytes.

B.5.3 2008-03-06T17-51-08-run3



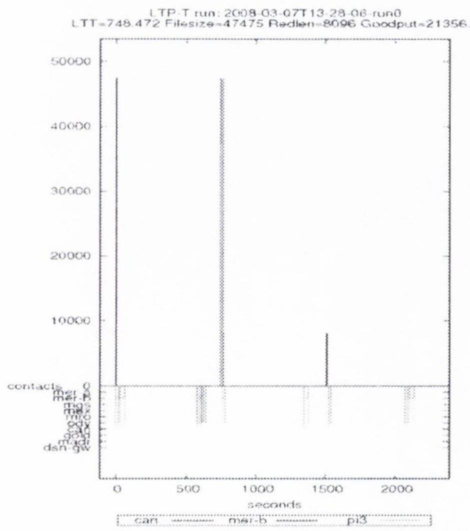
In this run, we can see the Report segment from Madrid is delayed until the next contact with MER-B allows for its transmission.

B.5.4 2008-03-06T17-51-08-run4



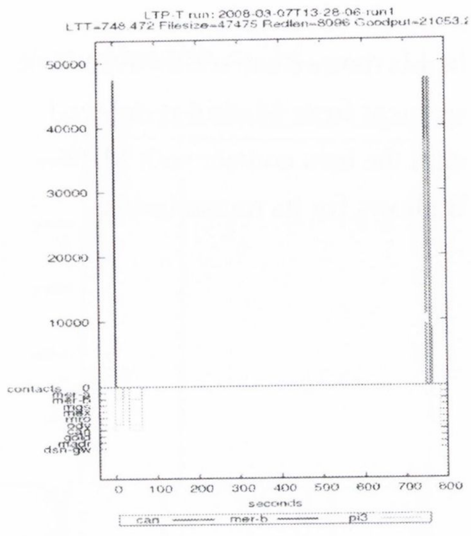
This is a nominal run.

B.5.5 2008-03-07T13-28-06-run0

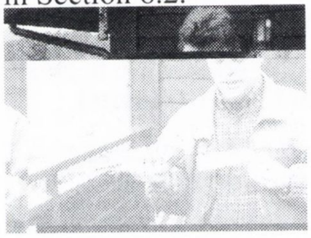


This is a nominal run. We can see the image returned in this test. In this case there no packet errors occurred.

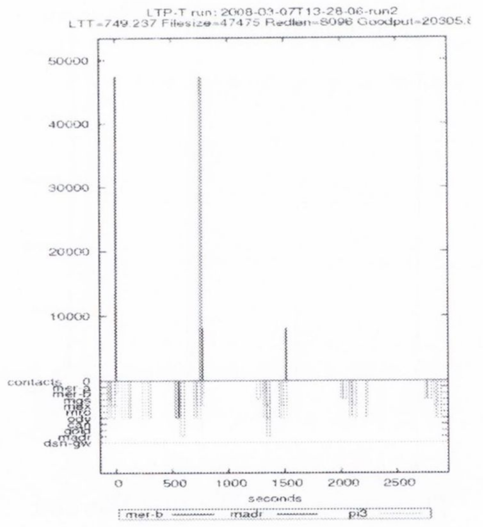
B.5.6 2008-03-07T13-28-06-run1



In this run, a packet in the green part was dropped (1351 bytes from offset 9465 to 10816 were missing) and we can see the resulting degradation of the image. The Canberra node also crashed in this case, so the graph doesn't show the Report segments arriving back at MER-B. This run is not included in the calculations in Section 6.2.

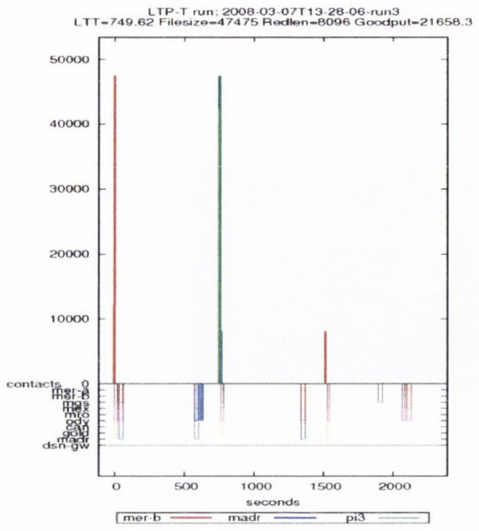


B.5.7 2008-03-07T13-28-06-run2



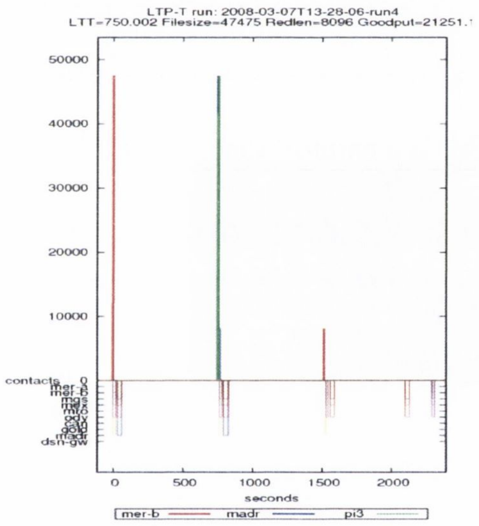
This is a nominal run.

B.5.8 2008-03-07T13-28-06-run3



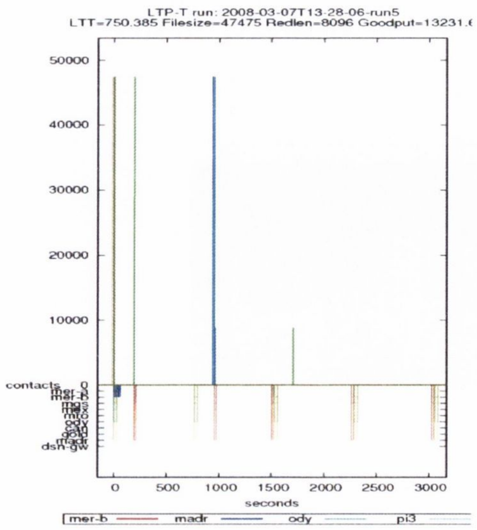
This is a nominal run.

B.5.9 2008-03-07T13-28-06-run4



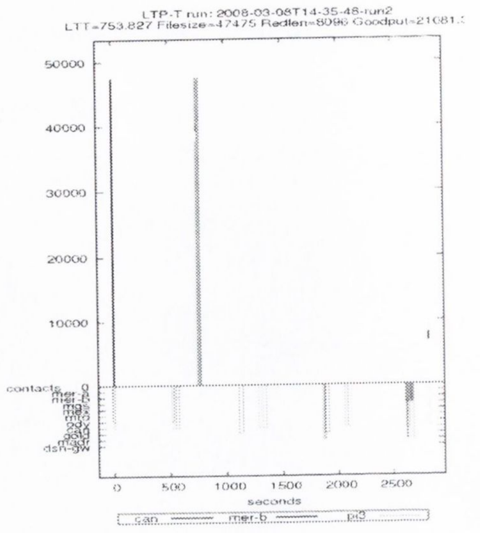
This is a nominal run.

B.5.10 2008-03-07T13-28-06-run5

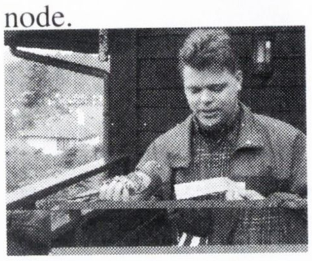


This is a nominal run.

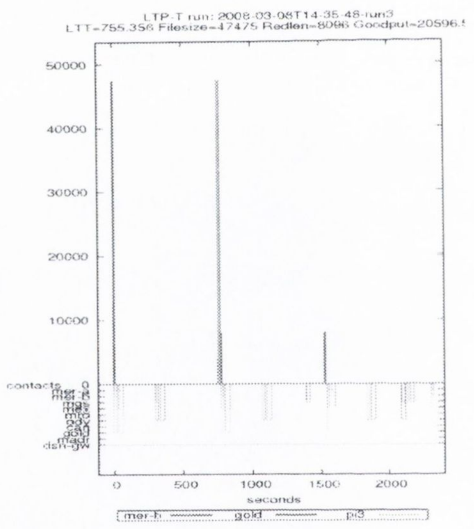
B.5.17 2008-03-08T14-35-48-run2



In this run, a green packet was dropped. The Canberra node also crashed, and so MER-B can be seen re-transmitting the last red Data segment. However, the image was delivered to the PI3 node.

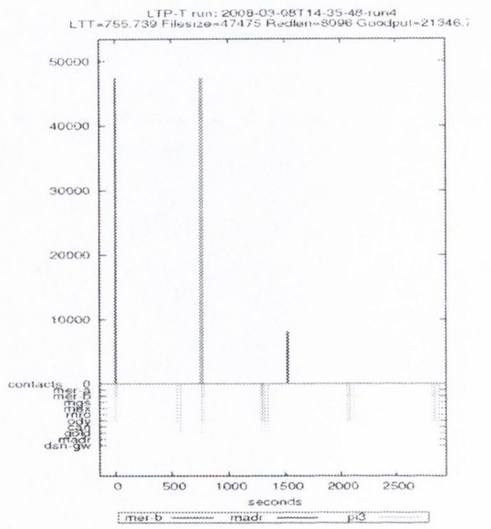


B.5.18 2008-03-08T14-35-48-run3



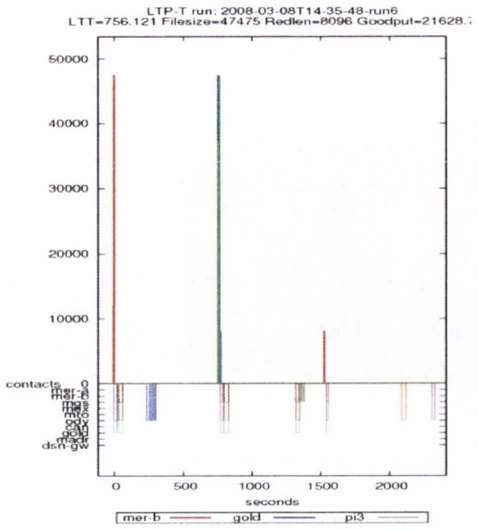
This is a nominal run.

B.5.19 2008-03-08T14-35-48-run4



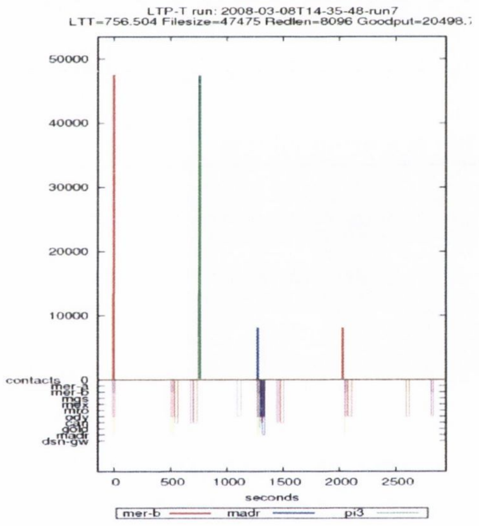
This is a nominal run, where a green segment (from 44616 to 45968) was dropped.

B.5.20 2008-03-08T14-35-48-run6



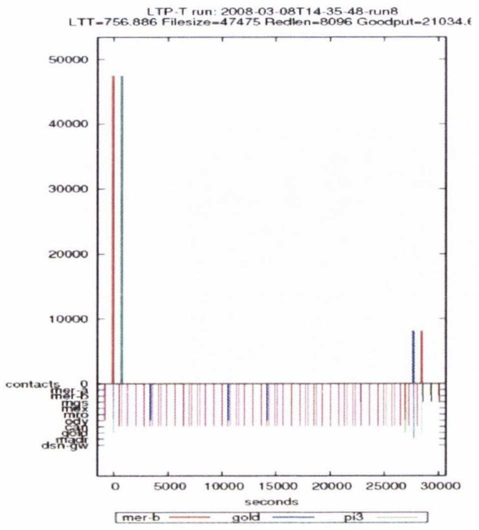
This is a nominal run.

B.5.21 2008-03-08T14-35-48-run7



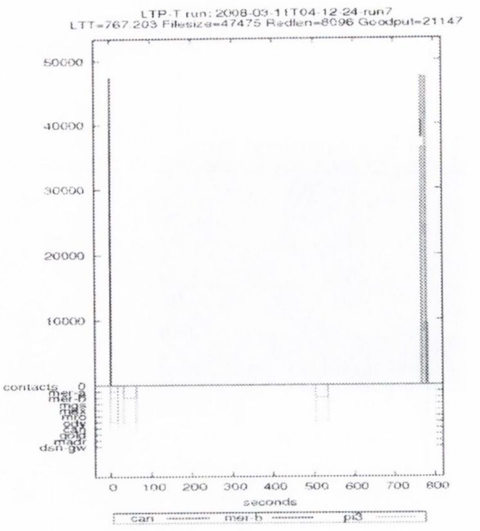
This is a nominal run.

B.5.22 2008-03-08T14-35-48-run8



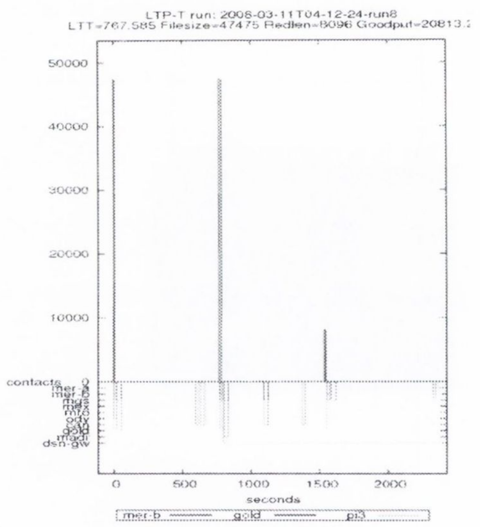
This is a nominal run, though one where the schedule resulted in a 7.5 hour delay between delivery of the block from Goldstone to PI3, and the time when Goldstone could transmit the Report segment back to MER-B.

B.5.29 2008-03-11T04-12-24-run7



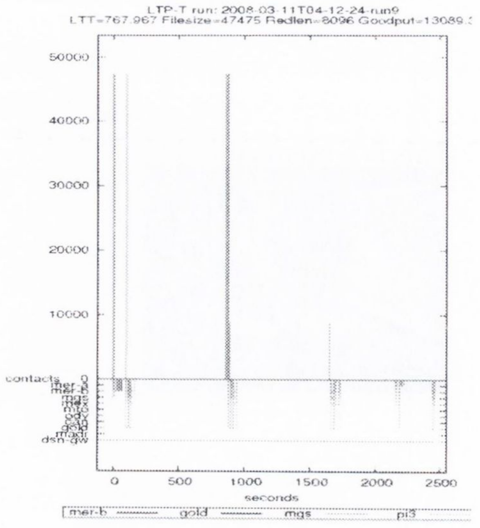
In this run, the Canberra node crashed. A green segment was dropped.

B.5.30 2008-03-11T04-12-24-run8

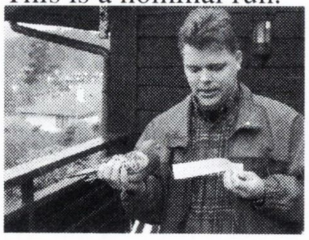


This is a nominal run.

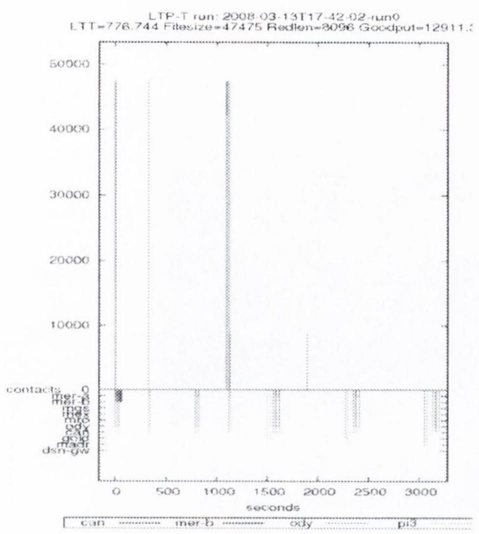
B.5.31 2008-03-11T04-12-24-run9



This is a nominal run.

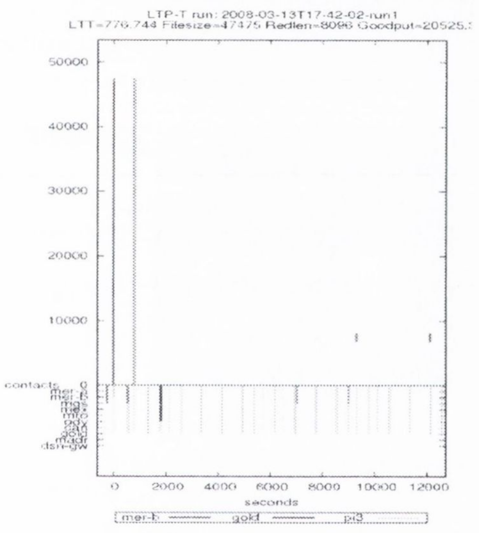


B.5.32 2008-03-13T17-42-02-run0

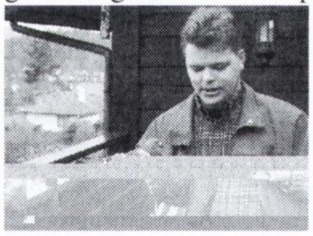


This is a nominal run.

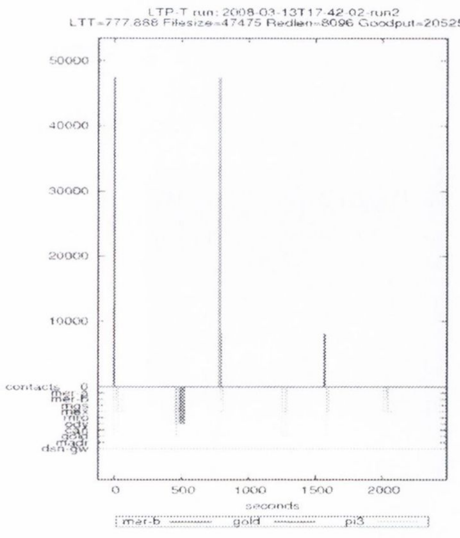
B.5.33 2008-03-13T17-42-02-run1



In this run, the Goldstone node crashed and we can see the two re-transmissions of the late red Data segment from MER-B. One green segment was dropped.

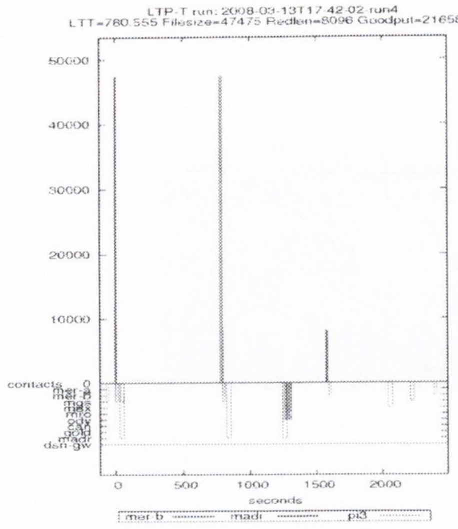


B.5.34 2008-03-13T17-42-02-run2



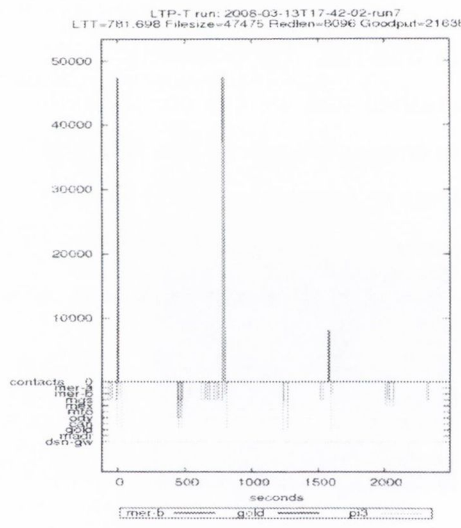
This is a nominal run.

B.5.35 2008-03-13T17-42-02-run4



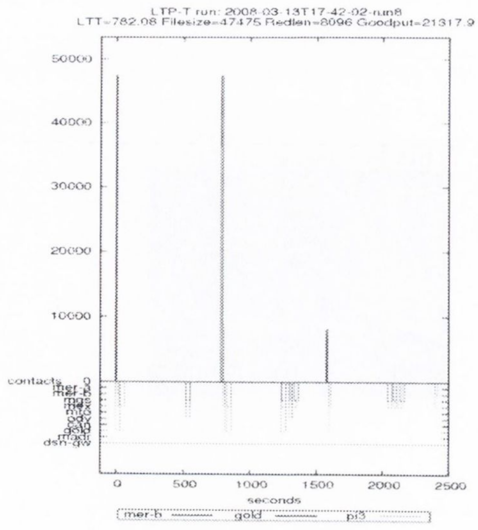
This is a nominal run.

B.5.36 2008-03-13T17-42-02-run7



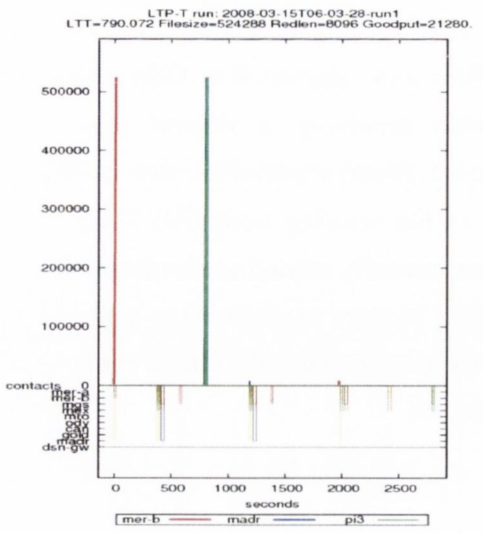
This is a nominal run.

B.5.37 2008-03-13T17-42-02-run8



This is a nominal run, with a green segment dropped.

B.5.38 2008-03-15T06-03-28-run1

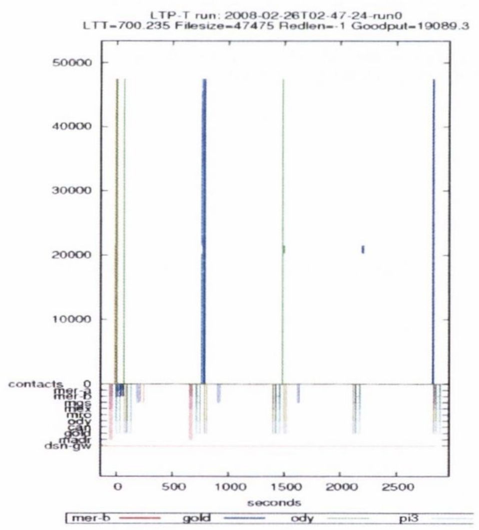


This is a nominal run.

B.6 Fully Red with 1 Percent errors

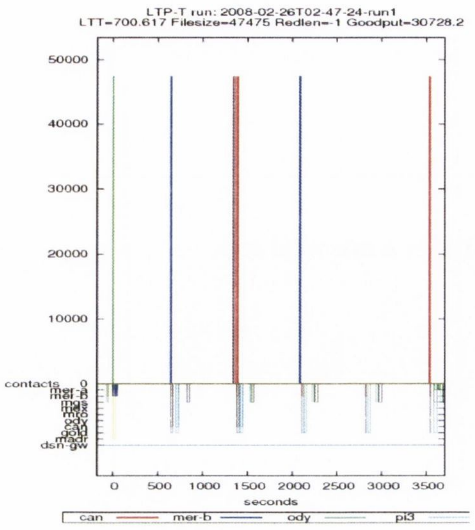
This section is as before, but with the entire block fully red.

B.6.1 2008-02-26T02-47-24-run0



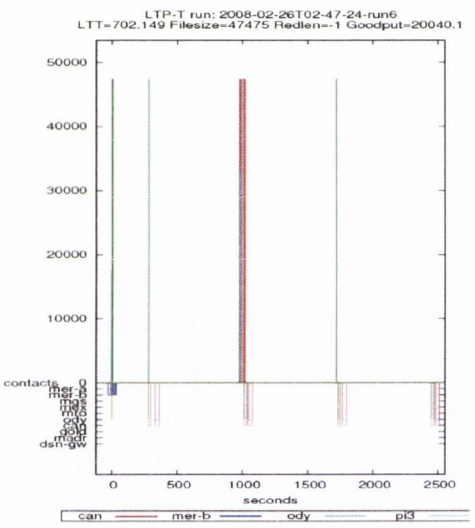
In this run, a Data segment was re-transmitted from Odyssey after being dropped. The last Report segment shown is received by Goldstone from PI3 after the successful receipt of the re-transmitted Data segment.

B.6.2 2008-02-26T02-47-24-run1



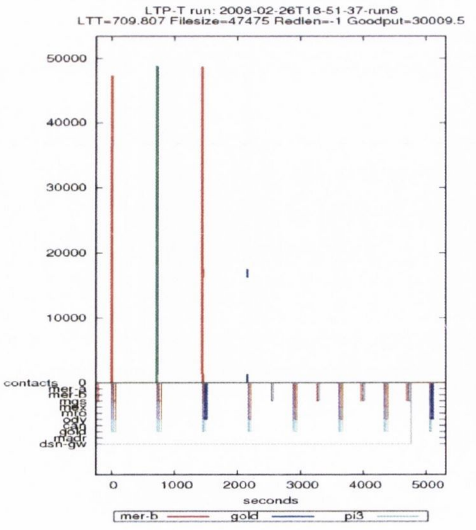
This run shows the Odyssey node receiving a Report segment from a previous run before the sending node (MER-B) had actually started transmitting. That Report segment was actually a re-transmission from run0 above!

B.6.3 2008-02-26T02-47-24-run6



This is a nominal run.

B.6.4 2008-02-26T18-51-37-run8

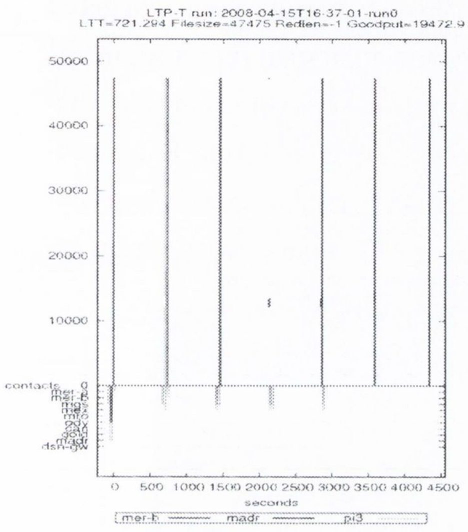


This run sees a re-transmission of four dropped data segments from MER-B. The Goldstone node however, crashed due to a subsequently fixed bug before forwarding.

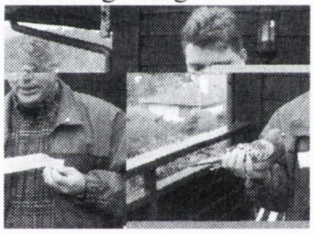
B.7 Miscellaneous Runs

This section shows a set of various runs, using the latest version of the LTPlib software. Some bugs previously noted are fixed in these runs.

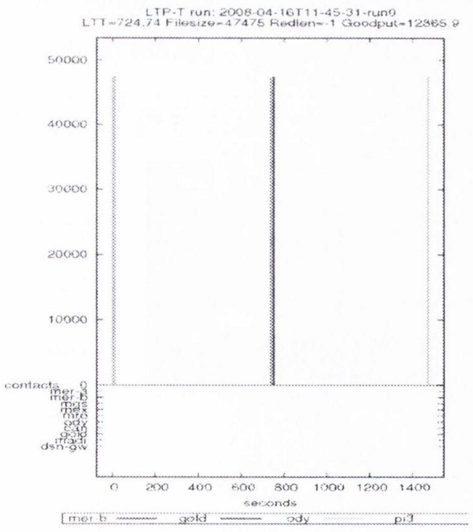
B.7.1 2008-04-15T16-37-01-run0



This run shows a still extant implementation bug where the Madrid node marked the block as being finished before the receipt of the re-transmitted segments from MER-B. As a result the PI3 node exited and so didn't receive the correct bytes after they eventually arrived at Madrid. The bug has no effect on Goodput, but its effect on the resulting image can be seen!

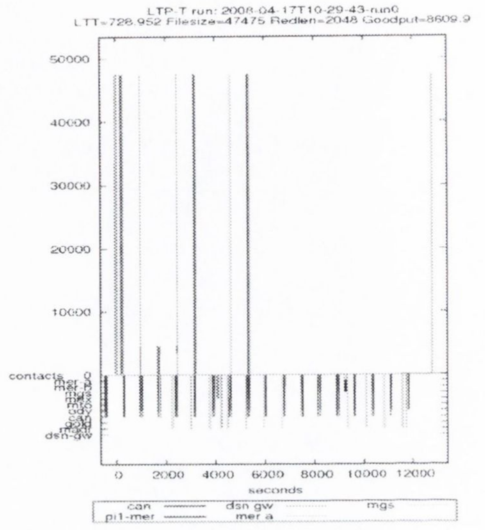


B.7.2 2008-04-16T11-45-31-run0



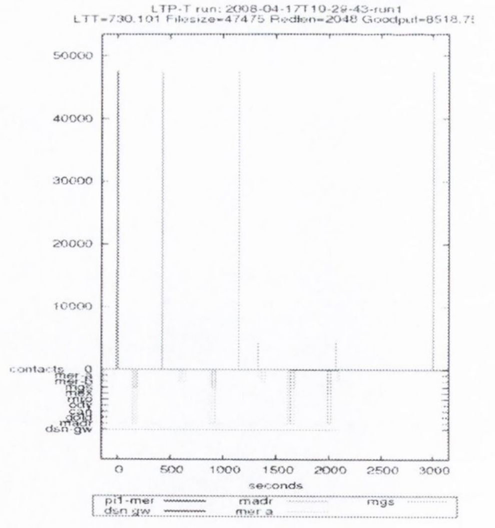
This is a nominal run.

B.7.3 2008-04-17T10-29-43-run0



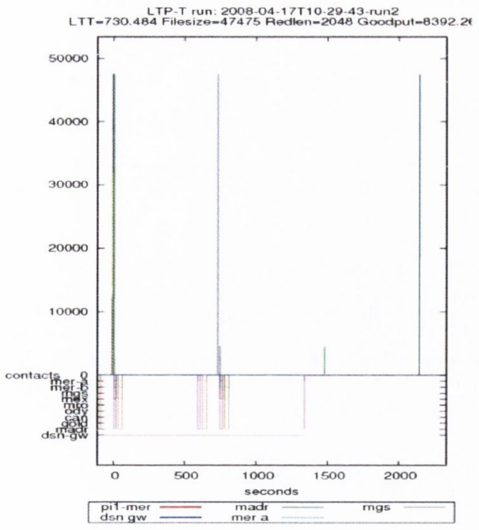
In this run, the Canberra node re-transmitted its last red data segment due to a timer-expiry. (This happens since timers have a 1-second granularity, so sometimes trigger just before receipt of the required Report.) This re-transmission triggered some additional report segments which are visible in the diagram, but the run is considered nominal since this bug simply adds unnecessary report re-transmissions.

B.7.4 2008-04-17T10-29-43-run1



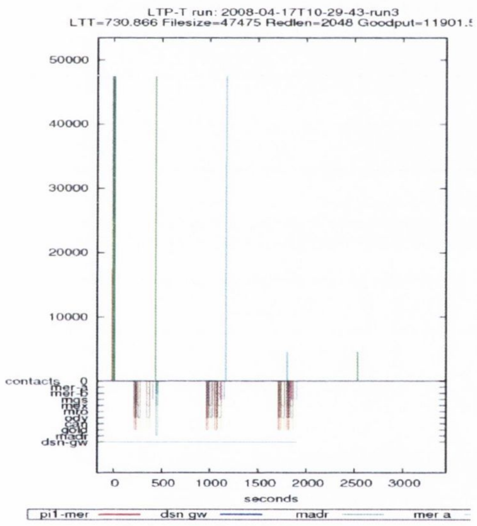
This is a nominal run. The last "spike" in the graph shows MGS forwarding the entire block to MER-A - the two cannot be distinguished in this view, but are distinct in the data.

B.7.5 2008-04-17T10-29-43-run2



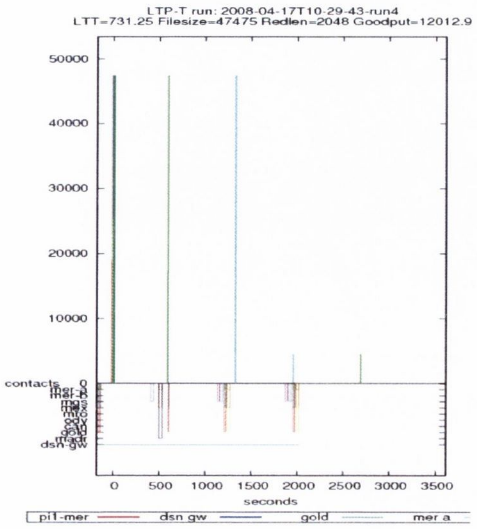
This is a nominal run. The last "spike" in the graph shows MGS forwarding the entire block to MER-A - the two cannot be distinguished in this view, but are distinct in the data.

B.7.6 2008-04-17T10-29-43-run3



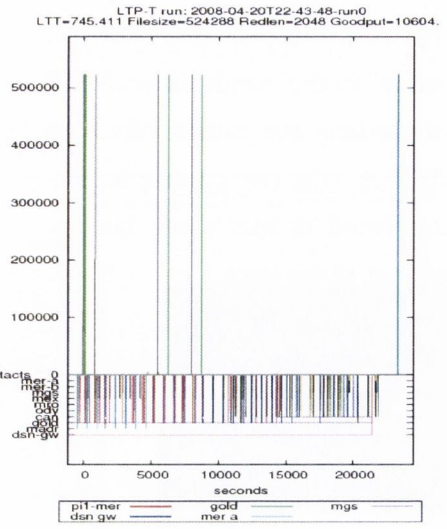
This is a nominal run.

B.7.7 2008-04-17T10-29-43-run4



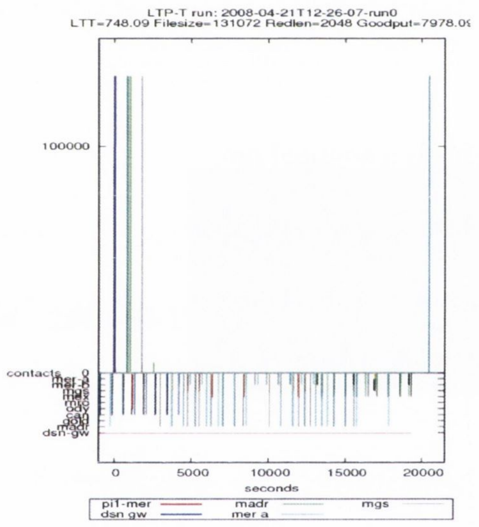
This is a nominal run.

B.7.8 2008-04-20T22-43-48-run0



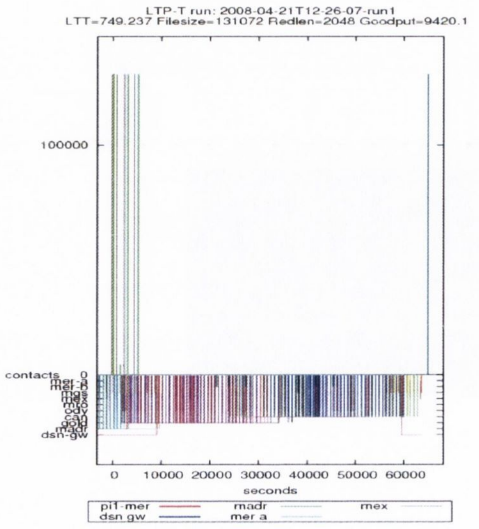
This run succeeded, though involved an unnecessary Report exchange between MGS and Goldstone due to the timer expiry granularity issue - a timer expired just before the expected Report acknowledgement was received at MGS from Goldstone causing a re-transmission of the Report segment.

B.7.9 2008-04-21T12-26-07-run0



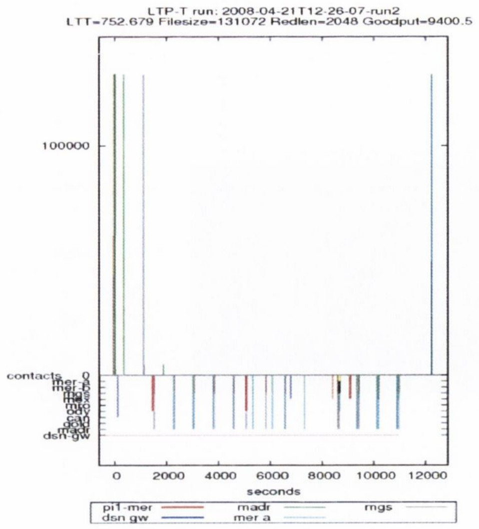
This is a nominal run.

B.7.10 2008-04-21T12-26-07-run1



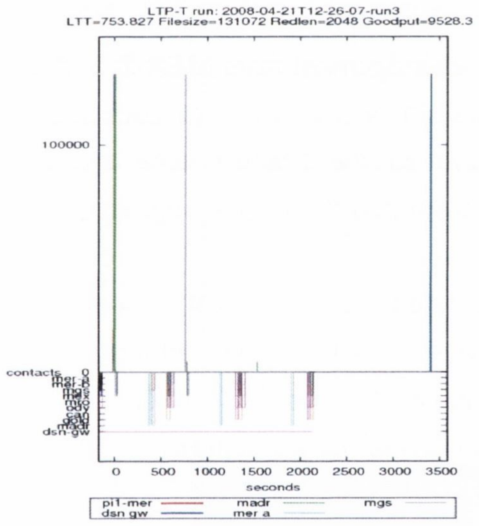
This run succeeded, though involved an unnecessary Report exchange between MarsExpress and Madrid due to the timer expiry granularity issue previously mentioned. This run is also notable for its duration - it began at 2008-04-21 19:28:21 and only finished at 2008-04-22 13:30:04, a total of 18 hours!

B.7.11 2008-04-21T12-26-07-run2



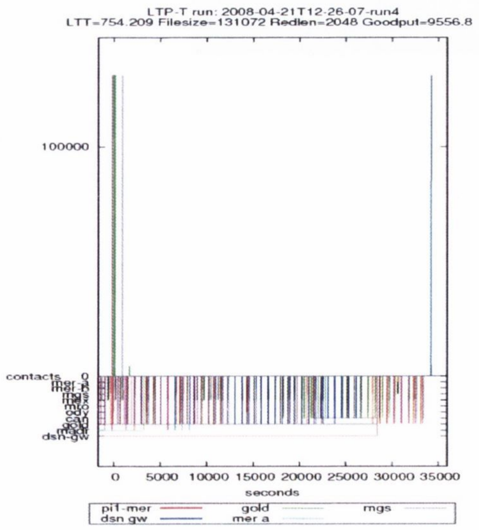
This is a nominal run.

B.7.12 2008-04-21T12-26-07-run3



This is a nominal run.

B.7.13 2008-04-21T12-26-07-run4



This is a nominal (though long) run.

Appendix C - Martian Emulation Test Result Details

Section	Title	ltt	filesize	redlen	goodput	duration	timeused	start	end	nodes	nodelist
B.2 1	20080206T22-33-36 run0	819.6085	524288	0	16319.7	1567.24	32.126	2008-02-06 22:39:03.127	2008-02-06 23:05:10.366	5	pi1-mer+dsn gw +gold +mer a +mex
B.2 2	20080206T22-33-36 run1	819.6085	524288	0	21105.8	2168.17	24.841	2008-02-06 23:29:02.180	2008-02-07 00:05:10.355	5	pi1-mer+dsn gw +gold +mer a +mex
B.2 3	20080206T22-33-36 run2	819.6085	524288	0	16265.6	4268.2	32.233	2008-02-07 00:50:02.148	2008-02-07 02:01:10.345	5	pi1-mer+dsn gw +gold +mer a +mex
B.2 4	20080206T22-33-36 run3	819.6085	524288	0	16275.7	11305.4	32.213	2008-02-07 02:24:45.126	2008-02-07 05:33:10.496	5	pi1-mer+dsn gw +gold +mer a +ody
B.2 5	20080206T22-33-36 run4	819.985	524288	0	16301	3887.22	32.163	2008-02-07 06:00:23.129	2008-02-07 07:05:10.349	5	pi1-mer+dsn gw +gold +mer a +ody
B.2 6	20080206T22-33-36 run5	819.985	524288	0	23151.5	1496.24	22.646	2008-02-07 07:35:37.209	2008-02-07 08:00:33.451	4	pi1-mer+dsn gw +gold +mer a
B.2 7	20080206T22-33-36 run6	820.3615	524288	0	22907.7	1258.79	22.887	2008-02-07 08:25:41.051	2008-02-07 08:46:39.838	4	pi1-mer+dsn gw +gold +mer a
B.2 8	20080206T22-33-36 run7	820.3615	524288	0	22830.9	1010.7	22.964	2008-02-07 09:15:54.303	2008-02-07 09:32:45.008	4	pi1-mer+dsn gw +gold +mer a
B.2 9	20080206T22-33-36 run8	820.7385	524288	0	23507.5	1589.16	22.303	2008-02-07 10:06:16.198	2008-02-07 10:32:45.360	4	pi1-mer+dsn gw +gold +mer a
B.2 10	20080206T22-33-36 run9	820.7385	524288	0	23508.6	1144.68	22.302	2008-02-07 10:55:27.510	2008-02-07 11:14:32.193	4	pi1-mer+dsn gw +gold +mer a
B.2 11	20080207T11-47-44 run0	821.1155	524288	0	22876.7	1370.47	22.918	2008-02-07 11:51:42.151	2008-02-07 12:14:32.619	4	pi1-mer+dsn gw +gold +mer a
B.2 12	20080207T11-47-44 run1	821.1155	524288	0	23048.7	1184.66	22.747	2008-02-07 12:40:54.002	2008-02-07 13:00:38.664	4	pi1-mer+dsn gw +gold +mer a
B.2 13	20080207T11-47-44 run2	821.1155	524288	0	23054.7	997.329	22.741	2008-02-07 13:30:05.494	2008-02-07 13:46:42.823	4	pi1-mer+dsn gw +gold +mer a
B.2 14	20080207T11-47-44 run3	821.4925	524288	0	23269.6	1605.88	22.531	2008-02-07 14:19:57.316	2008-02-07 14:46:43.195	4	pi1-mer+dsn gw +gold +mer a
B.2 15	20080207T11-47-44 run4	821.4925	524288	0	23089.3	1360.97	22.707	2008-02-07 15:10:06.130	2008-02-07 15:32:47.102	4	pi1-mer+dsn gw +gold +mer a
B.2 16	20080207T11-47-44 run5	821.869	524288	0	23431.9	889.119	22.375	2008-02-07 15:59:46.823	2008-02-07 16:14:35.942	4	pi1-mer+dsn gw +gold +mer a
B.2 17	20080207T11-47-44 run6	821.869	524288	0	23064.9	1604.02	22.731	2008-02-07 16:33:57.493	2008-02-07 17:00:41.515	4	pi1-mer+dsn gw +gold +mer a
B.2 18	20080207T11-47-44 run7	824.504	524288	0	16302.5	33506.7	32.16	2008-02-07 18:38:43.677	2008-02-08 03:57:10.425	5	pi1-mer+dsn gw +gold +mer a +mgs
B.2 19	20080207T11-47-44 run8	824.504	524288	0	20690.1	1048.73	25.34	2008-02-08 04:29:22.893	2008-02-08 04:46:51.626	4	pi1-mer+dsn gw +gold +mer a
B.2 20	20080207T11-47-44 run9	824.504	524288	0	23530.7	862.236	22.281	2008-02-08 05:18:30.443	2008-02-08 05:32:52.679	4	pi1-mer+dsn gw +gold +mer a
B.3 2	20080223T19-09-41 run1	689.898	1019375	-1	30222.5	36381.5	33.729	2008-02-23 21:03:12.030	2008-02-24 07:09:33.483	3	can +mer-b+pi3
B.4 1	2008-03-03T17-52-15 run0	730.866	47475	2048	31544.8	2842.27	1.505	2008-03-03 18:19:00.004	2008-03-03 19:06:22.269	3	mer-b+madr +pi3
B.4 4	2008-03-03T17-52-15 run4	740.437	47475	2048	30335.5	2831.84	1.565	2008-03-05 20:19:58.097	2008-03-05 21:07:09.932	3	mer-b+gold +pi3
B.4 5	2008-03-03T17-52-15 run5	740.819	47475	2048	20057	2978.79	2.367	2008-03-05 22:04:57.788	2008-03-05 22:54:36.579	4	can +mer-b+ody +pi3
B.4 6	2008-03-03T17-52-15 run7	743.115	47475	2048	31254.1	2259.46	1.519	2008-03-06 10:55:00.014	2008-03-06 11:32:39.473	3	can +mer-b+pi3
B.4 7	2008-03-03T17-52-15 run9	743.88	47475	2048	19731.9	2990.66	2.406	2008-03-06 14:04:57.754	2008-03-06 14:54:48.417	4	can +mer-b+ody +pi3
B.5 2	2008-03-06T17-51-08 run2	745.411	47475	8096	21520.9	2266.95	2.206	2008-03-06 22:42:29.020	2008-03-06 23:20:15.969	3	mer-b+madr +pi3
B.5 3	2008-03-06T17-51-08 run3	745.793	47475	8096	20271.1	2827.14	2.342	2008-03-07 00:20:30.038	2008-03-07 01:07:37.176	3	mer-b+madr +pi3
B.5 4	2008-03-06T17-51-08 run4	746.177	47475	8096	20758.6	2826.46	2.287	2008-03-07 02:07:51.025	2008-03-07 02:54:57.484	3	mer-b+madr +pi3
B.5 5	2008-03-07T13-28-06 run0	748.472	47475	8096	21356.3	2275	2.223	2008-03-07 13:55:20.098	2008-03-07 14:33:15.096	3	can +mer-b+pi3
B.5 7	2008-03-07T13-28-06 run2	749.237	47475	8096	20305.8	2820.67	2.338	2008-03-07 19:08:09.090	2008-03-07 19:55:09.758	3	mer-b+madr +pi3

Section	Title	ltt	filesize	redlen	goodput	duration	timeused	start	end	nodes	nodelist
B.5 8	2008-03-07T13-28-06 run3	749.62	47475	8096	21658.3	2278.92	2.192	2008-03-07 20:55:26.077	2008-03-07 21:33:24.992	3	mer-b+madr +pi3
B.5 9	2008-03-07T13-28-06 run4	750.002	47475	8096	21251.1	2280.56	2.234	2008-03-07 22:29:58.005	2008-03-07 23:07:58.570	3	mer-b+madr +pi3
B.5 10	2008-03-07T13-28-06 run5	750.385	47475	8096	13231.6	3017.45	3.588	2008-03-08 00:04:57.641	2008-03-08 00:55:15.087	4	mer-b+madr +ody +pi3
B.5 11	2008-03-07T13-28-06 run6	750.385	47475	8096	20262.5	2281.69	2.343	2008-03-08 01:29:58.076	2008-03-08 02:07:59.764	3	mer-b+madr +pi3
B.5 12	2008-03-07T13-28-06 run8	751.149	47475	8096	20543.1	2283.57	2.311	2008-03-08 04:30:00.106	2008-03-08 05:08:03.677	3	mer-b+madr +pi3
B.5 13	2008-03-07T13-28-06 run9	751.532	47475	8096	13317	2499.82	3.565	2008-03-08 06:04:57.815	2008-03-08 06:46:37.631	4	mer-b+gold +mgs +pi3
B.5 14	2008-03-08T09-45-25 run0	752.297	47475	8096	13373.2	3025.17	3.55	2008-03-08 10:04:57.872	2008-03-08 10:55:23.047	4	mer-b+gold +ody +pi3
B.5 15	2008-03-08T14-35-48 run0	753.062	47475	8096	13060.5	3028.4	3.635	2008-03-08 15:04:57.623	2008-03-08 15:55:26.026	4	can +mer-b+ody +pi3
B.5 16	2008-03-08T14-35-48 run1	753.445	47475	8096	21071.9	2290.73	2.253	2008-03-08 16:42:53.024	2008-03-08 17:21:03.751	3	can +mer-b+pi3
B.5 18	2008-03-08T14-35-48 run3	755.356	47475	8096	20596.5	2296.74	2.305	2008-03-09 02:30:08.015	2008-03-09 03:08:24.758	3	mer-b+gold +pi3
B.5 19	2008-03-08T14-35-48 run4	755.739	47475	8096	21346.7	2811.71	2.224	2008-03-09 04:08:45.646	2008-03-09 04:55:37.354	3	mer-b+madr +pi3
B.5 20	2008-03-08T14-35-48 run6	756.121	47475	8096	21628.7	2298.39	2.195	2008-03-09 07:43:01.006	2008-03-09 08:21:19.396	3	mer-b+gold +pi3
B.5 21	2008-03-08T14-35-48 run7	756.504	47475	8096	20498.7	2806.16	2.316	2008-03-09 09:21:44.607	2008-03-09 10:08:30.763	3	mer-b+madr +pi3
B.5 22	2008-03-08T14-35-48 run8	756.886	47475	8096	21034.6	29218.2	2.257	2008-03-09 11:08:54.483	2008-03-09 19:15:52.675	3	mer-b+gold +pi3
B.5 23	2008-03-08T14-35-48 run9	758.797	47475	8096	21550.2	2814.17	2.203	2008-03-09 20:21:48.011	2008-03-09 21:08:42.180	3	mer-b+gold +pi3
B.5 24	2008-03-11T04-12-24 run1	765.293	47475	8096	21938.5	2807.63	2.164	2008-03-11 06:22:27.060	2008-03-11 07:09:14.694	3	mer-b+gold +pi3
B.5 25	2008-03-11T04-12-24 run2	765.675	47475	8096	21638.6	2807.04	2.194	2008-03-11 08:09:29.015	2008-03-11 08:56:16.051	3	mer-b+gold +pi3
B.5 26	2008-03-11T04-12-24 run3	765.675	47475	8096	21015.9	2328.17	2.259	2008-03-11 09:56:30.150	2008-03-11 10:35:18.325	3	mer-b+gold +pi3
B.5 27	2008-03-11T04-12-24 run4	766.057	47475	8096	21034.6	2328.55	2.257	2008-03-11 11:43:31.045	2008-03-11 12:22:19.591	3	can +mer-b+pi3
B.5 28	2008-03-11T04-12-24 run6	766.821	47475	8096	21317.9	2803.03	2.227	2008-03-11 15:09:37.021	2008-03-11 15:56:20.049	3	can +mer-b+pi3
B.5 30	2008-03-11T04-12-24 run8	767.585	47475	8096	20813.2	2319.07	2.281	2008-03-11 18:46:44.062	2008-03-11 19:25:23.131	3	mer-b+gold +pi3
B.5 31	2008-03-11T04-12-24 run9	767.967	47475	8096	13089.3	2430.63	3.627	2008-03-11 20:28:57.489	2008-03-11 21:09:28.124	4	mer-b+gold +mgs +pi3
B.5 32	2008-03-13T17-42-02 run0	776.744	47475	8096	12911.3	3123.51	3.677	2008-03-13 18:04:57.421	2008-03-13 18:57:00.927	4	can +mer-b+ody +pi3
B.5 34	2008-03-13T17-42-02 run2	777.888	47475	8096	20525.3	2364.23	2.313	2008-03-14 00:57:19.002	2008-03-14 01:36:43.232	3	mer-b+gold +pi3
B.5 35	2008-03-13T17-42-02 run4	780.555	47475	8096	21658.3	2372.08	2.192	2008-03-14 15:44:14.005	2008-03-14 16:23:46.081	3	mer-b+madr +pi3
B.5 36	2008-03-13T17-42-02 run7	781.698	47475	8096	21638.6	2375.96	2.194	2008-03-14 20:57:34.080	2008-03-14 21:37:10.040	3	mer-b+gold +pi3
B.5 37	2008-03-13T17-42-02 run8	782.08	47475	8096	21317.9	2376.3	2.227	2008-03-14 22:44:19.104	2008-03-14 23:23:55.405	3	mer-b+gold +pi3
B.5 38	2008-03-15T06-03-28 run1	790.072	524288	8096	21280.5	2782.53	24.637	2008-03-16 16:24:56.075	2008-03-16 17:11:18.609	3	mer-b+madr +pi3
B.6 3	2008-02-26T02-47-24 run6	702.149	47475	-1	20040.1	2433.34	2.369	2008-02-26 12:11:28.033	2008-02-26 12:52:01.369	4	can +mer-b+ody +pi3
B.7 2	2008-04-16T11-45-31 run0	724.74	47475	-1	12865.9	1473.38	3.69	2008-04-16 12:05:56.281	2008-04-16 12:30:29.661	4	mer-b+gold +ody +pi3
B.7 3	2008-04-17T10-29-43 run0	728.952	47475	2048	8609.9	12812.2	5.514	2008-04-17 10:49:29.012	2008-04-17 14:23:01.255	5	can +pi1-mer+dsn gw +mer a +mgs

Section	Title		ltt	filesize	redlen	goodput	duration	timeused	start	end	nodes	nodelist
B.7 4	2008-04-17T10-29-43	run1	730.101	47475	2048	8518.75	3008.9	5.573	2008-04-17 15:31:52.434	2008-04-17 16:22:01.334	5	pi1-mer+dsn gw +madr +mer a +mgs
B.7 5	2008-04-17T10-29-43	run2	730.484	47475	2048	8392.26	2224.1	5.657	2008-04-17 17:29:14.275	2008-04-17 18:06:18.378	5	pi1-mer+dsn gw +madr +mer a +mgs
B.7 6	2008-04-17T10-29-43	run3	730.866	47475	2048	11901.5	3283.25	3.989	2008-04-17 19:11:38.843	2008-04-17 20:06:22.089	4	pi1-mer+dsn gw +madr +mer a
B.7 7	2008-04-17T10-29-43	run4	731.25	47475	2048	12012.9	3439.13	3.952	2008-04-17 21:09:04.464	2008-04-17 22:06:23.598	4	pi1-mer+dsn gw +gold +mer a
B.7 8	2008-04-20T22-43-48	run0	745.411	524288	2048	10604.7	23320.3	49.439	2008-04-20 23:03:35.331	2008-04-21 05:32:15.583	5	pi1-mer+dsn gw +gold +mer a +mgs
B.7 9	2008-04-21T12-26-07	run0	748.09	131072	2048	7978.09	20470.5	16.429	2008-04-21 12:45:53.663	2008-04-21 18:27:04.138	5	pi1-mer+dsn gw +madr +mer a +mgs
B.7 10	2008-04-21T12-26-07	run1	749.237	131072	2048	9420.15	64902.2	13.914	2008-04-21 19:28:21.891	2008-04-22 13:30:04.132	5	pi1-mer+dsn gw +madr +mer a +mex
B.7 11	2008-04-21T12-26-07	run2	752.679	131072	2048	9400.56	12256.7	13.943	2008-04-22 14:40:56.368	2008-04-22 18:05:13.029	5	pi1-mer+dsn gw +madr +mer a +mgs
B.7 12	2008-04-21T12-26-07	run3	753.827	131072	2048	9528.35	3413.06	13.756	2008-04-22 19:08:19.966	2008-04-22 20:05:13.024	5	pi1-mer+dsn gw +madr +mer a +mgs
B.7 13	2008-04-21T12-26-07	run4	754.209	131072	2048	9556.84	34218.4	13.715	2008-04-22 21:05:45.727	2008-04-23 06:36:04.082	5	pi1-mer+dsn gw +gold +mer a +mgs
B.7 14	2008-04-23T22-05-14	run0	758.797	47475	32000	22257.4	2307.03	2.133	2008-04-23 22:30:15.018	2008-04-23 23:08:42.051	3	mer-b+gold +pi3
B.7 16	2008-04-23T22-05-14	run3	760.708	47475	32000	21337.1	2312.12	2.225	2008-04-24 08:43:15.094	2008-04-24 09:21:47.209	3	mer-b+gold +pi3
B.7 17	2008-04-23T22-05-14	run4	761.091	47475	32000	22163.9	2809.13	2.142	2008-04-24 10:22:05.255	2008-04-24 11:08:54.389	3	mer-b+madr +pi3
Statistics		Average	770.2169533	205871.9467	5279.107	19214.46	5740.811	10.82075			3.747	
		Stdev	34.6041729	235369.9349	6588.205	5466.155	10402.39	11.57899			0.773	
		Min	689.898	47475	-1	7978.09	862.236	1.505			3	
		Max	824.504	1019375	32000	31544.8	64902.2	49.439			5	
		Total		15440396	Bytes		430560.8	811.556	Seconds			
		Total		15078.51172	KB							

Columns are as follows:

Section	is a pointer to the related part of Appendix B.	Duration	is the elapsed (wall) time for the run
Title	is the title of the relevant section and a pointer to the relevant directory below http://dtn.dsg.cs.tcd.ie/sft/	Time used	is the "non-gap" time for the run (see Section 6.2)
		Start	is the wall-clock time at which the run started (incl. Milliseconds)
		End	is the wall-clock time at which the run ended (incl. Milliseconds)
LTT	is the Light Trip Time for the run	Nodes	is the number of nodes involved in the run
Filesize	is the size of the file transferred	Nodelist	lists the set of nodes involved
Redlen	is the length of the red part as input to ltpd ("-1" means all)		
Goodput	is the measured Martian Goodput for the run		