Postellation: an Enhanced Delay-Tolerant Network (DTN) Implementation with Video Streaming and Automated Network Attachment

Marc Blanchet, Simon Perreault and Jean-Philippe Dionne
Viagénie, Québec, Québec, Canada

Delay-Tolerant Networking (DTN) technology is based on the Bundle Protocol (BP). Inherent to the underlying store-and-forward architecture, current DTN implementations do not support real-time streaming of data such as video, especially in the context of using current IP-based applications and devices over a DTN network. Moreover, current DTN implementations are configured statically with a list of peers, preventing the construction of a dynamic network of devices to join the network. This paper describes Postellation (http://postellation.viagenie.ca), a highly portable DTN stack running on Windows, Linux, Mac OS X, as well as embedded real-time operating systems such as RTEMS. The paper shows the key differences of this implementation, such as how real-time streaming is achieved, as well as the automated network attachment mechanism. A use-case scenario is to have IP devices on board spacecraft with a DTN node carrying IP traffic from the spacecraft to the Earth through the DTN space network. A DTN node on Earth then forwards the traffic to the IP network on Earth.

I. Introduction

DELAY-TOLERANT networking (DTN) is being proposed as the standard building-block for space networking, much like Internet Protocol (IP) has been the building block of the Internet. At the core of DTN is the Bundle Protocol[1] (BP), a store-and-forward protocol for generic data transfer.

In the DTN architecture, bundles are sent, forwarded, and received by DTN nodes. The lower-layer technologies and protocols over which bundles are transported are called “convergence layers”. A minimal DTN stack includes a BP engine and at least one convergence layer. Convergence layers exist for sending bundles across the Internet (i.e., TCP and UDP convergence layers) as well as over various space transmission technologies (e.g. Saratoga, Licklider Transmission Protocol (LTP), CCSDS space link protocol).

Routing is an important component of a DTN stack, however there is no single standard routing method or protocol. In the context of space networking, there are two main approaches: scheduled routing, where one knows in advance the positions and connectivity windows of DTN nodes, or dynamic routing using heuristics when such contact information is unavailable.

The interface between the DTN stack and its applications, the API, is not currently standardized.
Postellation (http://postellation.viagenie.ca) is a DTN stack with extra features aiming to enable fast application deployment by reusing tools and techniques that are currently deployed on the Internet. This lowers the cost of DTN adoption.

II. Use of Existing Internet Protocols and Applications

At the base of the Internet protocol suite is IP, the Internet Protocol. Transmission protocols such as TCP and UDP are carried in IP packets, being logically situated right on top of IP in the protocol stack. Application protocols, of which the most popular is undoubtedly the Hypertext Transfer Protocol (HTTP), come after, on top of the transport protocols.

Custom applications are developed on top of the standard application protocols. For HTTP in particular, a myriad of frameworks, development tools, programming languages, XML dialects, JSON REST APIs, etc. are available off the shelf and are being used to build the Internet we know. Countless programmers are being trained with the Web technologies as primary focus.

Making these Web technologies, or at least a subset of them, available over DTN would be an excellent way of “bootstrapping” DTN. It could have the potential to speed up adoption tremendously.

III. Use of Web Technologies over DTN

The cornerstone is HTTP. HTTP has request/response semantics that can be adapted very naturally to DTN[2]. Each HTTP request or response is encapsulated in bundles and sent over the DTN network. Furthermore, a gateway between DTN and IP can act as an HTTP proxy.

Carrying HTTP payload over DTN presents multiple challenges:

- TCP Congestion Control

  TCP, the protocol over which HTTP is transported, contains mechanisms to cope with congestion on the Internet. Congestion control works by maintaining a window of in-flight, unacknowledged data between the sender and the receiver. This window automatically grows as data is acknowledged by the receiver and shrinks as data is retransmitted by the sender. This mechanism is not suited for very long delays such as those found in space.

- Multiple Round-Trips

  Rendering a typical web page requires making multiple HTTP requests, often to different servers. Some HTTP responses may in turn trigger other HTTP requests. Overall, this means that the set of requests that are necessary to render a web page is partially ordered, making it impossible to completely parallelize all the requests. Thus, multiple round trips are necessary. With long delays it is imperative to minimize the number of round trips to maximize performance.

- Time-out mechanism at the application layer

  Some current web applications are designed with assumptions on what is a "reasonable" time for user interaction. Adjustment at the application layer is required when time-out delays are shorter than the expected DTN link delay. For instance, time-out mechanisms might be found in *asynchronous javascript* (Ajax) requests and in cookie based user sessions. Web applications built with DTN in mind need to relax those assumptions. We believe building DTN applications with web tools is not only feasible, but is a way to save on time, costs, and to minimize risk.

Using a local web cache alongside Postellation's HTTP-to-DTN proxy, it is possible to achieve a user experience similar to the one achievable on Earth on the majority of the popular web sites of the Internet. In one demonstration of Postellation's capabilities, we simulated a link delay of 1.3 seconds, equivalent to the Earth-Moon propagation delay.

DNS Resolution in a proxy context can be performed either in the client or server realm:

• When using an explicitly-configured HTTP proxy (i.e. non-transparent mode), browsers do not perform DNS resolution. They expect the proxy to do the DNS resolution based on the contents of the Host HTTP header. In that case, Postellation performs DNS resolution in the server realm, by the connector, after the DTN network has been traversed, as part of regular TCP connection establishment to the HTTP server.

• When using an implicitly-configured HTTP proxy (i.e. transparent mode), browsers will attempt to perform DNS resolution since, from their point of view, they are just establishing a regular connection directly to the HTTP server, unaware that a proxy is in the way. In this case it is necessary that a DNS infrastructure be made available in the client realm.

File-based Operation

File based operations are of significant importance in space missions. A file transfer protocol over BP have relatively simple semantics if we suppose the underlying layers are responsible for, relaying, custody transfer and data unit reassembly. Postellation provides two means to transfer files over DTN:

- Two trivial applications (*dtnsend* and *dtnrecv*) transmit an encapsulated a file in a single bundle
- Standard HTTP client and server combined with the HTTP-to-DTN proxy: The HTTP protocol in addition with the WEBDAV[3] extension is comparable in functionalities to other file transfer protocol, such as the CSSDS File Delivery Protocol (CFDP) and FTP. Files can be uploaded, downloaded and remote operation on the file system such as copy, move or delete can be performed.

IV. Video Streaming over DTN

The Postellation HTTP-to-DTN proxy can be used to tunnel any protocol carried over TCP. TCP tunneling is performed by using the CONNECT method of the HTTP standard. Streaming video or audio over the internet is characterized by a continuous flow of small packets. For this flow to continue over a DTN network, every packet must be encapsulated into an individual bundle.

V. Connecting DTN Nodes

Postellation nodes can connect to other DTN nodes via various convergence layers. Each node is named by an end-point-identifier (EID) and has a set of routes to reach other nodes. Auto-configuration of nodes is possible when nodes are organized in a tree topology. Nodes can generate a unique EID derived from the EID of the upstream node followed by a unique identifier (UUID). The structure of the EID reflects the relation with the upstream node and is used to determine the next-hop when forwarding bundles.

VI. Implementation

Postellation is written in C and complies with the ISO C99 standard. Emphasis has been on minimizing the memory footprint and portability to real-time as well as non real-time operating systems.

Postellation includes the Bundle Protocol daemon (bpd) and a suite of BP applications. Bpd and its applications run as separate processes and communication between components requires the BSD socket API and Libevent[4], an asynchronous event notification library. BP applications interact with bpd through a common Application Programming Interface (API), making easy the design of new applications.

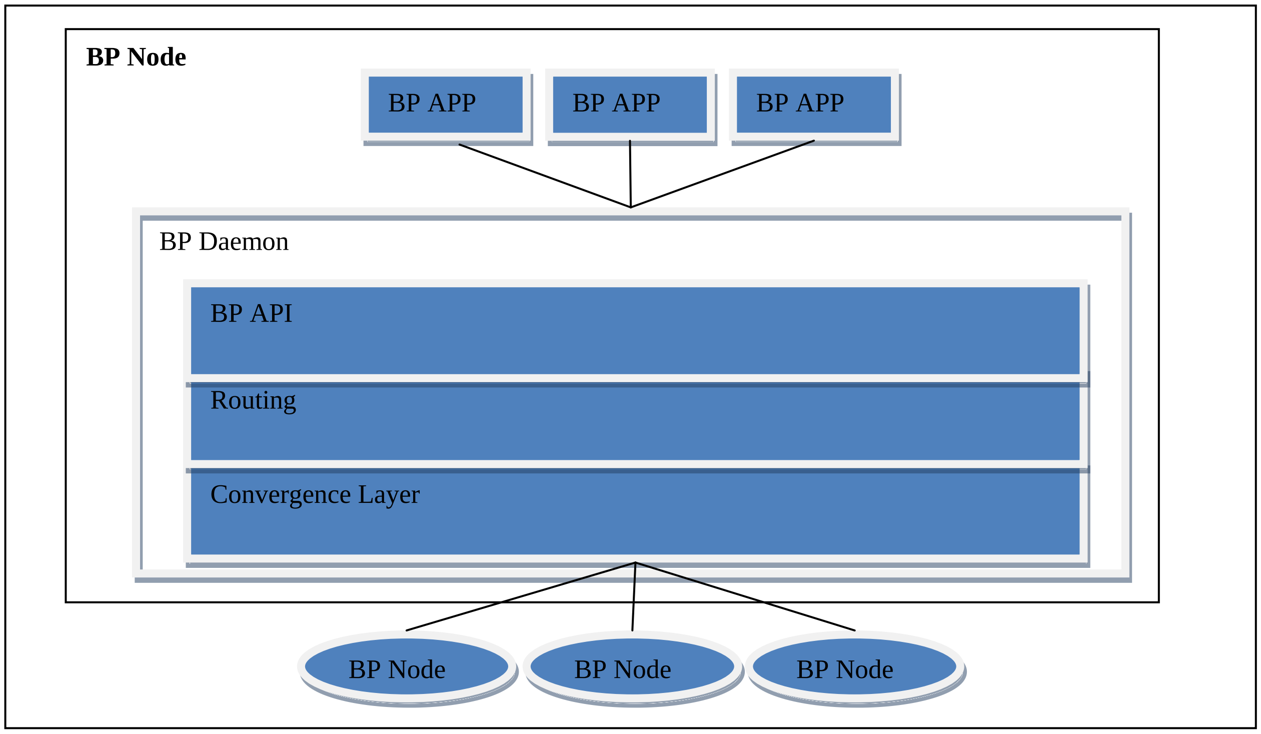

Figure 1: Postellation's components

The Postellation distribution includes the simple, de-facto standard applications *dtnping*, *dtnpong*, *dtnsend*, and dtnrecv. In addition, Postellation includes a DTN/HTTP proxy that is composed of two applications that use the Postellation API: the *acceptor* and the *connector*. They are named after the roles that they play in an HTTP connection establishment. When a regular HTTP client establishes a new HTTP connection, the *acceptor* accepts the connection over TCP. It converts the HTTP data into bundles that are sent over BP to the *connector*. Upon receiving the bundles, the *connector* will connect to the HTTP server over HTTP, completing the connection establishment process. From the point of view of the HTTP client and server, the *acceptor/connector* pair can be considered as an HTTP proxy. This proxy can work in transparent mode as well as non-transparent mode.

Postellation has been implemented and tested on Linux, Mac OS X, OpenBSD, Windows, and RTEMS.

The UDP and TCP convergence layers support both IPv4 and IPv6. A particular feature of this implementation is that UCP and TCP convergence layer link configurations do not specify a client or server role. Connection initiation is attempted from both sides and the first successful attempt "wins" while the other gets cancelled. In practice, on today's Internet, this facilitates mobility and Network Address Translator (NAT) traversal.

Postellation automatically fragments and reassembles bundles that are larger than the convergence layer MTU[*]. For both TCP and UDP, path MTU discovery[5] is used to determine the effective MTU to a DTN peer. It is also possible to statically configure the MTU per peer.

We extended the TCP convergence layer specification to include transport over a secure connection using Transport Layer Security (TLS). This non-standard but straightforward extension is very useful in delivering HTTP secure transactions.

Interoperability testing

Bundle forwarding between Postellation and nodes of other implementations has been tested. The applications used for sending and receiving bundles were *dtnping* and *dtnpong*.

Two testing scenarios were used:

- A Postellation node is used as an intermediary node between two nodes of foreign implementation (DTN2[6] and ION[7]).
- A foreign (DTN2 or ION) node is used as an intermediary node between two Postellation nodes.

No custody, report status or security features were used. Routing between nodes was statically configured.

Postellation is available for trial[†] and can be used to connect to a public DTN cloud, composed of nodes with artificial delays. The delay is introduced using *netem*, a Linux kernel module for emulating various connection characteristics. The UDP convergence layer is used between nodes where delay would make TCP impractical.

[*] MTU : Maximum Transfer Unit
[†] http://postellation.viagenie.ca

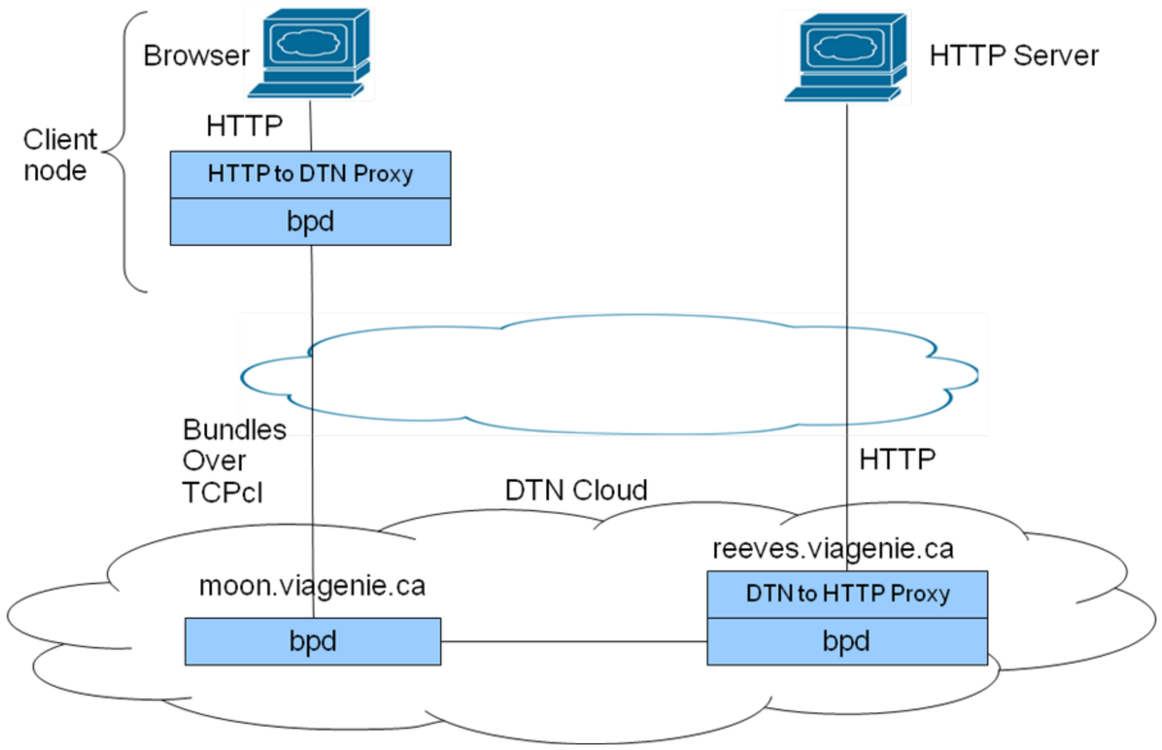

Figure 2: Test environment with artifical delay and the HTTP-to-DTN proxy

VII. Porting to Real-Time Operating System

Postellation has been ported to RTEMS[8]. Since RTEMS does not have a notion of processes, multiple threads are used instead. For example, to run an HTTP proxy on RTEMS, one would need to run bpd in its own thread and either the acceptor or connector application in another.

To give an idea of system requirements, the following memory footprint for an i386 target running bpd has been measured:

- Binary image size: 508 kB (full RTEMS OS + Postellation software)
- Heap size: 256 kB (bare minimum for enabling the RTEMS networking stack)
- Stack size: 4 kB (bare minimum on the i386 architecture)

This shows that Postellation makes it possible to deploy a full DTN stack in under one megabyte of memory.

VIII. Conclusion

Postellation is a lightweight and portable DTN (Bundle Protocol) implementation, featuring advanced HTTP and Video streaming capabilities, as well as automatic network attachments. It is available on various platforms such as Linux, MacOSX, Windows, OpenBSD and RTEMS. It can be found and tested at: http://postellation.viagenie.ca.

References

[1] Scott, K. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, November 2007.
[2] Jörg Ott, Dirk Kutscher: Bundling the Web: HTTP over DTN. WNEPT Workshop 2006. URL: http://www.netlab.tkk.fi/~jo/papers/2006-wnept-bundling-the-web.pdf
[3] Dusseault, L., Ed., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, June 2007.
[4] Libevent, http://libevent.org
[5] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.
[6] DTN2 URL : http://sourceforge.net/projects/dtn/
[7] ION, Interplanetary Overlay Network, URL: http://sourceforge.net/projects/ion-dtn/

[8] RTEMS, Real-Time Executive for Multiprocessor Systems, http://rtems.com