Using HTTP for delivery in Delay/Disruption-Tolerant Networks
draft-wood-dtnrg-http-dtn-delivery-06

Abstract

This document describes how to use the Hypertext Transfer Protocol, HTTP, for communication across delay- and disruption-tolerant networks, by making every transit node in the network HTTP-capable, and doing peer HTTP transfers between nodes to move data hop-by-hop or subnet-by-subnet towards its final destination. HTTP is well-known and straightforward to implement in these networks.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 18, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Background and Introduction .......................... 3
2. Adapting the HTTP delivery mechanism for DTNs .......... 5
3. Other useful proposed additional HTTP headers .......... 7
4. Other suggestions on using MIME in DTN networks ....... 8
5. Security Considerations ................................ 9
6. IANA Considerations .................................... 9
7. Acknowledgements ...................................... 9
8. References ............................................. 10
   8.1. Normative References .............................. 10
   8.2. Informative References ............................ 10
Authors’ Addresses .................................... 12
1. Background and Introduction

Delay- and Disruption-Tolerant Networks (DTNs) are networks where conditions are such that links between nodes are not always permanent, may be of very long delay or exist only during very short contact periods where the link is up, and may change over time [RFC4838]. Some DTNs can be thought of as sparse ad-hoc networks, with nodes communicating intermittently only when they come into contact. Store-and-forward delivery of data is a useful way of communicating across these networks.

A specialised store-and-forward protocol for DTN delivery has been proposed in the IRTF DTN research group (DTNRG) - the Bundle Protocol [RFC5050]. Criticisms of the Bundle Protocol’s lack of reliability and its complexity have been made [I-D.irtf-dtnrg-bundle-checksum]. The Bundle Protocol is itself intended to be a routable data format, but the supporting architectures for node and application naming/addressing, automated routing, security, QoS, and resource discovery have not yet been agreed upon or in some cases even significantly worked on. These things already exist for the Internet Protocol, and can in many cases be easily leveraged for DTN networks [Wood09a].

This document outlines how the well-known Hypertext Transfer Protocol (HTTP) [RFC2616] can be used for store-and-forward communication across DTNs. HTTP is not used end-to-end as it is on the web. Instead, applications running on each node in the network communicate with their neighbours using dedicated hop-by-hop or subnet-by-subnet HTTP transfers to effect local data delivery. Additional HTTP header information adds context for onward forwarding and delivery to destination endpoints, and provides the reliability and support for error-detection currently missing from the alternative Bundle Protocol.

It must be stressed that this proposed use is distinct from proxy caching methods prevalent in the traditional web. Caching commands are not used; end-to-end HTTP requests are not intercepted by intermediate caches that attempt to fulfil them in the traditional web caching sense.

Although HTTP-DTN use as as a hop-by-hop message carrier between caches implementing some form of routing protocol between them, the distinction between client, server and proxy is replaced by peer intermediate caches using HTTP to communicate in separate sessions that together combine over time to make the full path between original source and final destination for the data.

HTTP is a session layer, running over a transport layer providing reliable delivery of the HTTP stream between hops. This transport
layer is commonly (and almost universally) TCP in the terrestrial Internet, although alternative transport layers, such as SCTP, can also be used under HTTP [I-D.natarajan-http-over-sctp]. For long-delay networks, or for network conditions where TCP or an equivalent is not suitable, an alternative transport layer such as Saratoga [I-D.wood-tsvwg-saratoga] can be used under HTTP instead in hop-by-hop communications between nodes. HTTP requires only reliable streaming that can be used to provide ordered delivery; how that reliable streaming is provided is up to the local transport layer in the local subnet, and multiple different transport layers can be used across the multiple hops between nodes to transfer data from source to final destination.

Steve Deering has often described IP as 'the waist in the hourglass' [Deering98] - what is above and touching on IP can be changed, what is below and touching on IP can be changed, but provided the new elements continue to interface to and work with IP, the hourglass remains complete and the network stack remains functional. Here, HTTP is the waist in this particular hourglass; applications can use HTTP to communicate, provided HTTP runs over a reliable transport stream. The applications can vary. The transport stream can be changed; HTTP does not have to run over TCP/IP, but could even be made to run directly over e.g. HDLC or a CCSDS reliable bitstream. Given the prevalence of IP in many networks, it is likely that two waists exist; IP and HTTP are likely choices, but the transport protocol and physical environment will vary more. An expansion of this argument is given in [Wood09b].

Separation of HTTP from the underlying transport layer to make HTTP a layer in its own right is increasingly likely to happen; this is analogous to the use of different "convergence layers" under the Bundle Protocol. Being able to set what transport layer to use depending on conditions is useful, and one simple configuration approach to this, able to support HTTP-DTN, was outlined in [I-D.wood-tae-specifying-uri-transports].

HTTP use here relies on the three P’s – Persistence, Pipelining and the PUT directive. These are all present in the HTTP/1.1 specification.

This document contains an overview of how HTTP can be simply adapted to the DTN environment by the use of HTTP/1.1 with persistence and pipelining, the PUT and GET directives, and some trivial extra HTTP headers needed to indicate e.g. a destination in the DTN network.

The remainder of this specification uses ‘file’ as a shorthand for ‘binary object’, which may be an HTTP ‘object’, file with an associated MIMEtype, or other type of contiguous binary data.
A significant benefit to use of HTTP is that the well-known MIMEtype mechanism, integral to HTTP, provides hints on what received files are, and what applications should do with them [RFC2045]. The Bundle Protocol does not support MIMEtypes, or any similar mechanism. HTTP/1.1’s use of MIME is specified in [RFC2616] rather than in the separate MIME documents.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. [RFC2119]

2. Adapting the HTTP delivery mechanism for DTNs

Here, HTTP is used as a peer-to-peer protocol in the sense that multiple files may be transferred in both directions simultaneously between two communicating nodes using HTTP for DTN use. There is not intended to be a strict client/user-agent to server relationship as there is in the web. Instead, sending data across a path of six nodes, four nodes between source and destination, will require a minimum of five separate per-hop HTTP transactions between each pair of nodes to move the data onwards to the next node. This breaks the traditional end-to-end control loop and transfer into separate control loops and transfers suitable for the DTN environment.

When two nodes come into contact across a local hop or a subnet, a request for files to be copied, stored, and carried onwards can be made by the receiving node issuing an HTTP GET request. Alternatively, the sending node can simply issue a series of HTTP PUT requests once a connection is established, if it believes that putting the data to the receiving node moves it closer to its eventual destination. The receiving node can always reject transfers with error codes.

HTTP-DTN is a superset of HTTP/1.1. HTTP/1.1 pipelining and persistence permits multiple PUTs to be made in sequence. Support for these in implementations is crucial to the mechanisms outlined here. (Note that [I-D.natarajan-http-over-sctp] also takes advantage of HTTP pipelining and persistence.)

The key to enabling HTTP use for DTN networking is an added Content-Destination: header, which specifies the final destination of the file, and can be used by routing in the HTTP-using applications to decide over which available links the file should be sent. Content-* headers are special, in that they may not be ignored (section 9.6 of [RFC2616]). Recipients not understanding Content-Destination: will generate a "501 (Not Implemented)" error code. This separates HTTP use in DTNs described here from normal end-to-end HTTP web use. HTTP
DTN nodes MUST support Content-Destination:. Files that are PUT are cached and then relayed onwards by intermediate peers to the Content-Destination:. GET requests for files can be forwarded by intermediate peers to the Content-Destination:.

The information provided in Content-Destination: identifying the destination may be an IP address, DNS name, Bundle Endpoint Identifier (EID) or other text-string identifier useful to the local DTN routing mechanisms being used.

Similarly, a Content-Source: header provides a textual identification of the original source of the data. HTTP-DTN nodes MUST support Content-Source:.

For DTN use, DTN HTTP nodes MUST also implement and use Content-Length: and Content-Range: headers. These permit partial delivery of files and resends of missing pieces of files. The Content-MD5: header must be supported. This provides a simple end-to-end reliability check. The Content-MD5: header is intended to be generated by the source node first sending the data, and is not recomputed at other nodes.

DTN HTTP nodes MUST implement the Host: header, in line with current HTTP specifications. This header field MAY be left blank to request available files from the peer node, rather than identifying a desired file from a distant source by hostname matching the advertised Content-Source: header. A sender placing a new file into the DTN network for onward transmission MUST have the Content-Source: field of the data being sent match its Host: field.

Hop-by-hop HTTP headers MAY be implemented between peer nodes talking directly. The headers described in section 13.5.1 of [RFC2616] are available. New hop-by-hop headers MUST use the Connection: header approach described in section 14.10 of [RFC2616].

DTN HTTP nodes may optionally GET from and PUT to link-local IP multicast addresses when used over IP subnets. This permits efficient sharing of files on shared LANs, with recipients requesting resends via Content-Range: and checking assembly of file pieces using the Content-MD5: header. A GET to multicast can request a specific file from any available node that has it. The response to a multicast GET SHOULD be unicast, but a multicast HEAD MAY also be sent to inform other nodes that the sender has the file of interest. If other nodes also express interest in the file with GET requests to the sender, that file may later be PUT to a multicast address.

(Note that in the alternative Bundle Protocol, the Bundle Endpoint Identifier (EID) can identify a group of endpoints, rather than just
one; mapping the Bundle EID onto multicast IP addresses on IP subnets is possible. Placing textual EIDs directly in HTTP-DTN’s Content-Source: and Content-Destination: headers, or in a Host: field, would be possible to interwork HTTP-DTN and bundling.)

The utility of HTTP with multicast has been recognised previously as a method of simple service discovery later adopted for the universal plug and play (UPnP) protocol [I-D. draft-goland-http-udp] [I-D. draft-cai-ssdp-v1]. Rather than call out multicast and unicast separately as different protocols to be used by HTTP, recognising that a given destination or address indicates multicast or broadcast use should suffice.

Many existing HTTP/1.1 headers are directly useful with HTTP-DTN. For example, ETag: headers are useful for identifying unique copies of files in the network, and can be used to provide globally unique identifiers (GUIDs) for each version of a file. Age: headers are useful for estimating the amount of time a MIME object has been in the network - indicating both transmission and storage times. Last-Modified: times refer to the times on the origin server - that is, the Content-Source: - and should be preserved during onward forwarding. Max-Forwards: provides a TTL hop count and propagation limitation mechanism.

3. Other useful proposed additional HTTP headers

A number of other additional HTTP headers are proposed here, as likely to be useful. These SHOULD be implemented. These would benefit from being specified more completely, in line with the suggestions in [RFC2774].

An HTTP object is just one binary file; the ability to group objects together is useful (and is done in bundles by the Bundle Protocol). If we call a group of related objects sent from the same source to the same destination a ‘package’ (a name chosen to avoid any confusion with the Bundle Protocol specification), we can then define simple headers to be sent before each object:

Package-ID: - provides a unique textual identifier for the package

Package-Item: n of m (e.g. 1 of 7) - order of this HTTP file in the package

Package-MD5: - MD5 hash across all Content-MD5: headers added together in order of Package-Item: precedence.

A way to request missing Package-Items (from the previous node or
from the source) is likely to be very useful.

Precedence: headers could set importance of objects - very-high, high, normal low, very-low - to give simple quality of service and prioritization.

Some sort of header protection may be a good idea; Content-MD5: covers the message body (entity-body), but not the headers. Header-MD5: could cover some important HTTP headers. Header-MD5 could be preserved across hops if possible, avoiding unnecessary header reordering. Changing timestamps would invalidate the Header-MD5: end-to-end, however - this needs more thought, particularly on where timestamps are placed in HTTP headers.

For larger files, stronger mechanisms than MD5 should be examined.

There may be a need to send HTTP-DTN transfers across paths that include hops with unidirectional one-way links with no return path, e.g. when a wireless sender knows that a receiver is available, but cannot hear it. Using: Connection: cannot-hear-response could be used across that hop to indicate that the sender cannot hear receivers.

Timestamps and how they are handled needs to be examined here in greater detail. HTTP has the same basic assumption as the Bundle Protocol - that all nodes are expected to know the current UTC time.

4. Other suggestions on using MIME in DTN networks

x-application-dtn has previously been proposed as a MIMEtype identifying Bundle Protocol bundles delivered by HTTP. This provides a way to support Bundle Protocol implementations in an HTTP infrastructure.

Moving HTTP transfers over DTN networks using the Bundle Protocol has already been proposed [Ott06]. By changing how HTTP is used - hop-by-hop rather than end-to-end - HTTP can be used directly in DTN networks without using the Bundle Protocol at all.

HTTP is a popular way to carry MIME, but support for MIME exists in other protocols, including email, SIP and BEEP. BEEP can be thought of as a more formalised and exactly-specified replacement for HTTP for machine-machine interaction - and this detailed formal specification makes BEEP complex [RFC3080]. BEEP provides an alternative to HTTP to support XML-RPC and SOAP. BEEP’s specification is formally intended to support multiple different transports, but only TCP transport of BEEP has been agreed [RFC3081].
HTTP’s simplicity of use and popularity appear to be compelling advantages over BEEP.

5. Security Considerations

Better-Than-Nothing Security [RFC5386][RFC5387] is likely to be useful here for ad-hoc communications without the availability of an existing authentication infrastructure.

Security considerations and detailed examination of HTTP over TLS (HTTPS) [RFC2817][RFC2818] and secure HTTP [RFC2660] are required here.

Many existing security mechanism for HTTP could be used unchanged for HTTP-DTN, if local conditions permit and the supporting infrastructure, e.g. DNS, is available. However, reusing these directly protects a single-hop transfer between peer nodes. To protect an end-to-end transfer, the security mechanisms would need to be applied using the information used in the Content-Source: and Content-Destination: headers, before applying the local security mechanism for the first peer-peer HTTP transfer.

6. IANA Considerations

Despite the Content-* rule for rejecting unfamiliar headers that separates HTTP-DTN peers from traditional HTTP servers, it may be desirable to use a non-standard port for DTN HTTP use over IP, rather than the well-known port 80. If so, such a port should be requested from IANA.

It may be necessary to request a dedicated IPv4 all-hosts multicast address and a dedicated IPv6 link-local multicast addresses for local HTTP DTN use, if local HTTP multicast is considered a desirable feature.

7. Acknowledgements

We thank Wes Eddy and Kevin Fall for their review comments.

Work on the Saratoga protocol inspired some of the concepts that are reused here, and we thank everyone involved in Saratoga’s development and implementation.

8. References
8.1. Normative References


8.2. Informative References

[Deering98]

[I-D.ietf-dtnrg-bundle-checksum]

[I-D.natarajan-http-over-sctp]

[I-D.wood-tae-specifying-uri-transports]


HTTP for delay- and disruption-tolerant networks with convergence layers", Workshop on the Emergence of Delay-/Disruption-Tolerant Networks (e-DTN 2009), St Petersburg, Russia, October 2009.

Authors’ Addresses

Lloyd Wood
Centre for Communication Systems Research, University of Surrey
Guildford, Surrey GU2 7XH
United Kingdom
Phone: +44-1483-689123
Email: L.Wood@surrey.ac.uk

Peter Holliday
Cisco Systems
Level 12
300 Adelaide Street
Brisbane, Queensland 4000
Australia
Phone: +61-2-6216-0604
Email: phollida@cisco.com