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.
Table of Contents

1. Background and Introduction ........................................ 3
2. Adapting the HTTP delivery mechanism for DTNs ........................ 4
3. Other useful proposed HTTP headers .................................. 6
4. Other suggestions on HTTP for use in DTN networks .................. 7
5. Security Considerations ............................................. 7
6. IANA Considerations ................................................. 8
7. Acknowledgements ................................................... 8
8. References ..................................................................... 8
   8.1. Normative References .............................................. 8
   8.2. Informative References ............................................ 8
Authors’ Addresses ......................................................... 9
Intellectual Property and Copyright Statements .......................... 11
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 reliability and complexity have been raised [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.

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 error-detection missing from alternatives such as the 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. The distinction between client, server and proxy is replaced by peers using HTTP to communicate in separate sessions that together combine over time to make the full path between source and final destination.

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-httpbis-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 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 enviroment will vary more.

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.

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/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-httpbis-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 [RFC2119]). 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:

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. This MUST be implemented.

For DTN use, DTN HTTP nodes MUST also implement Content-Length:, Content-Range: and Content-MD5 headers. This permits partial delivery of files and resends of missing pieces of files. The Content-MD5: header 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 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 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 the Bundle EID can identify a group of endpoints, rather than just one; mapping the Bundle EID onto multicast IP addresses on IP subnets is possible.

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.

3. Other useful proposed HTTP headers

A number of other HTTP headers are proposed here, as likely to be useful. These SHOULD be implemented.

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’ 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: checksum across all Content-MD5 headers added together in order

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

Some sort of header protection is likely also a good idea. So, Header-MD5: could cover some important HTTP headers. Header-MD5 could be preserved across hops if possible, avoiding unnecessary header reordering. Timestamps prevent this, however - this needs more thought, particularly on where timestamps are placed in HTTP headers.

Timestamps and how they are handled needs to be examined here in greater detail. What if different machines have different notions of time?

For larger files, stronger checksums than MD5 should be looked at.

4. Other suggestions on HTTP for use 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 - this draft has outlined how HTTP can be used directly and independently in DTN networks without requiring the bundle protocol as a carrier.

5. Security Considerations

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

Because there is a need for each node to validate that a file has been received correctly, privately-keyed hashes that can only be checked at the destination should be avoided, and HTTP security mechanisms should be used instead.
6.  IANA Considerations

Despite the Content-* rule for rejecting unfamiliar headers, it may make sense 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

Work on the Saratoga protocol inspired some of the concepts that are reused here. We thank Wes Eddy and Kevin Fall for their review comments.

8.  References

8.1.  Normative References


8.2.  Informative References


[I-D.draft-goland-http-udp]
Goland, Y., "Multicast and Unicast UDP HTTP Messages",

[I-D.irtf-dtnrg-bundle-checksum]
Eddy, W., Wood, L., and W. Ivancic, "Checksum Ciphersuites for
the Bundle Protocol",
draft-irtf-dtnrg-bundle-checksum-03 (work in progress),
October 2008.

[I-D.natarajan-httpbis-sctp]
Natarajan, P., Amer, P., Leighton, J., and F. Baker,
"Using SCTP as a Transport Layer Protocol for HTTP",
draft-natarajan-httpbis-sctp-00 (work in progress),
October 2008.

[I-D.wood-tsvwg-saratoga]
Wood, L., McKim, J., Eddy, W., Ivancic, W., and C.
Jackson, "Saratoga: A Scalable File Transfer Protocol",
draft-wood-tsvwg-saratoga-02 (work in progress),
October 2008.

[Ott06] Ott, J. and D. Kutscher, "Bundling the Web: HTTP over
DTN", WNEPT 2006 Workshop on Networking in Public
Transport, QShine Conference Ontario, August 2006.

[RFC2660] Rescorla, E. and A. Schiffman, "The Secure HyperText

[RFC2817] Khare, R. and S. Lawrence, "Upgrading to TLS Within
HTTP/1.1", RFC 2817, May 2000.


[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst,
R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant

Authors’ Addresses

Lloyd Wood
Cisco Systems
11 New Square Park, Bedfont Lakes
Feltham, Middlesex  TW14 8HA
United Kingdom

Phone: +44-20-8824-4236
Email: lwood@cisco.com

Peter Holliday
Cisco Systems
Level 12
300 Adelaide Street
Brisbane, Queensland  4000
Australia

Phone: +61-2-6216-0604
Email: phollida@cisco.com
Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.