

Network Working Group
Internet-Draft
Intended status: Informational
Expires: June 13, 2008

N. Gershenfeld
CBA/MIT
D. Johnson
Sun Microsystems, Inc.
T. Snide
Schneider Electric
K. Lynn
Cisco
December 11, 2007

Trivial Hypertext Transfer Protocol (THTP)
draft-gershenfeld-thtp-01.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on June 13, 2008.

Copyright Notice

Copyright (C) The Internet Society (2007).

Internet-Draft

THTP

December 2007

Abstract

THTP, the Trivial Hypertext Transfer Protocol, is an implementation of HTTP over UDP transport. THTP is designed for environments with limited computational power or bandwidth and single-packet exchanges. As such, THTP is best suited for the emerging class of applications running on embedded devices and sensor networks. THTP decouples HTTP from TCP and provides a subset of HTTP's functionality, in particular URI naming. This document describes the THTP protocol.

Table of Contents

- 1. Introduction 3
- 2. Motivation 4
- 3. THTP Design 5
 - 3.1. Protocol Relationships 5
 - 3.2. URI Format 5
 - 3.3. Request Model 6
 - 3.4. Reliability 6
 - 3.5. Multicast 7
- 4. HTTP Features 8
 - 4.1. GET Method 8
 - 4.2. POST Method 8
 - 4.3. Status Code Definitions 9
 - 4.3.1. Successful 2xx 9
 - 4.3.2. Client Error 4xx 9
 - 4.3.3. Server Error 5xx 9
 - 4.4. Proxies 9
 - 4.5. Caching 9
- 5. Security Considerations 10
- 6. References 11
- Appendix A. Acknowledgments 12
- Authors' Addresses 13
- Intellectual Property and Copyright Statements 14

1. Introduction

This memo details Trivial Hypertext Transfer Protocol or THTP, an application-layer protocol. THTP is a scaled-down adaptation of HTTP designed to run over the UDP [RFC0768] transport layer. It is not intended as a replacement of HTTP over TCP, but rather a complementary scheme to expand HTTP-like semantics to a wider range of possible environments.

2. Motivation

HTTP [RFC1945] provides a powerful naming construct that simultaneously identifies and locates resources. However, the tight coupling between HTTP and TCP necessitates TCP's protocol overhead and state machine complexity. Decoupling HTTP from TCP enables a well-standardized protocol to operate in a wider range of environments and devices.

For example, the anticipated proliferation of smart, tiny, networked devices require a standards-based naming scheme [ieee-i0], [i0], but often cannot tolerate large amounts of overhead. In addition the communication requirements and patterns of small, inexpensive devices and sensors may differ from traditional network participants. Specifically, embedded devices and sensors receiving or sending single-packet commands and responses require neither mandated reliable network transport nor packet sequencing at the transport layer.

Consider the application of a light switch using IP to communicate with a light bulb over a network. The light switch sending an "on" instruction to the networked light bulb in the same physical room does not need the overhead of a TCP full three-way handshake. In addition, implementing TCP or even T/TCP [RFC1644] is prohibitively expensive in terms of the communication and state machine complexity on such a resource constrained computing platform.

Rather than requiring constrained devices or environments to implement a new lightweight protocol, THTP codifies a simple, portable, standards-based means of extending HTTP to UDP.

3.5. Multicast

Because THTP uses UDP, THTP MAY use multicast group addresses. Assuming the underlying data-link layer network supports broadcast or multicast transmission, a single THTP message could be sent to, and received by, multiple nodes. An immediately practical application is to assign locally scoped multicast group addresses to a set of nodes. For example, a single light switch might send a THTP "on" instruction to multicast group M. All light bulbs assigned to group M would process the message and switch on. While Internet multicast deployment is limited, THTP multicast messages are useful in many local area networks.

4. HTTP Features

THTP implements a minimum subset of HTTP's features, but is not required to implement all of HTTP. The subset of features THTP supports are a natural consequence of using UDP and maintaining simplicity.

THTP clients and servers MUST support the HTTP GET and POST methods and MAY support other methods. A server MUST always respond to a GET request or provide the appropriate status code error. A server MAY respond to a POST request or provide the appropriate status code error. A server MUST respond with an error on an error event regardless of the method type. A THTP server MUST always legally respond with the status code 501 (not implemented) for requests it cannot process as needed. Status codes are documented in Section 4.3.

4.1. GET Method

THTP and HTTP's use of the GET method are identical. THTP MUST support the GET method. A THTP GET request from a client expects a response and the server MUST send the requested object identified by the Request-URI or an error. If the Request-URI refers to a data-producing process, the data from that process is returned.

To send information from the client to the server process, a client MAY imbed that information in the HTTP URI and use the GET method. The user agent appends a '?' to the action URI along with the data set in 'application/x-www-form-urlencoded' format.

THTP MAY support "conditional GET" in instances where cached entities can be used without consuming unneeded network bandwidth

4.2. POST Method

THTP MUST accept the POST method, but uses POST in a slightly different manner than HTTP. A THTP server MAY respond to a successful POST, but is not required to do so. The optional response covers cases where the client is not expecting a response and where communication resources are scarce, for instance sensor nodes. In the case of a lightbulb and lightswitch, a POST request may be used to change the state of the light and does not require a response. However, a THTP server MUST always respond with an error status code on an error condition for POST.

The POST method requests that the destination server process the data within the request as a subordinate of the the Request-URI resource. To perform an action with a POST, the user agent uses the action URI

Internet-Draft THTP December 2007

and adds a message body of type 'application/x-www-form-urlencoded'.

4.3. Status Code Definitions

THTP uses the same status codes as defined by HTTP. THTP servers MUST implement the following status codes at a minimum:

4.3.1. Successful 2xx

200 OK The 200 successful status code differs slightly from that in HTTP due to it being optional for POST requests. Code 200 indicates that the request has succeeded. A successful GET request must result in response containing the 200 status code along the information queried. A POST request may return a 200 status code.

4.3.2. Client Error 4xx

400 Bad Request The request is malformed and rejected by the server.
404 Not Found The request cannot be honored by the server because the requested resource is not available.

4.3.3. Server Error 5xx

501 Not Implemented The request cannot be honored by the server because the server does not support the requested functionality. Such an error may occur for unsupported methods.

4.4. Proxies

THTP messages MAY be proxied. Proxies are a natural consequence of interconnecting a local area network, e.g. for home automation, with the larger Internet.

4.5. Caching

HTTP explicitly allows and makes provisions for content caching. A request may be honored by an intermediary other than the final recipient. When cached, the response comes from this intermediary. Clearly, in THTP's intended environments such as control and automation networks, caching is not expected. Because THTP's request model anticipates all requests to be carried through to the final recipient, THTP SHOULD NOT be cached.

Internet-Draft THTP December 2007

5. Security Considerations

Since THTP is implemented on top of UDP, many of the security issues inherent in UDP are inherited by THTP. By eliminating the minimal source validation afforded by the three-way handshake of TCP, THTP is vulnerable to source IP address spoofing. Without a stronger means of authentication, THTP MAY rely on ingress filtering [RFC2827] of source addresses. Enterprises MAY use transport layer packet filtering to keep all THTP internal to their private intranet. For example, a packet filter rule on the enterprise gateway would block UDP port 80 traffic from entering or exiting the intranet.

Applications that require strong encryption and/or authentication MAY employ appropriate lightweight encryption [sea].

Internet-Draft THTP December 2007

6. References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC1644] Braden, B., "T/TCP -- TCP Extensions for Transactions Functional Specification", RFC 1644, July 1994.
- [RFC1945] Berners-Lee, T., Fielding, R., and H. Nielsen, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [i0] Gershenfeld, N., Krikorian, R., and D. Cohen, "The Internet of Things", Scientific American , October 2004.
- [ieee-i0] Gershenfeld, N. and D. Cohen, "Internet 0: Interdevice Internetworking", IEEE Circuits and Systems , October 2006.
- [sea] Standaert, F., Piret, G., Gershenfeld, N., and J. Quisquater, "SEA: a Scalable Encryption Algorithm for Small Embedded Applications", CARDIS , April 2006.
- [upnp] Goland, Y. and J. Schlimmer, "Universal Plug N' Play Forum", <http://www.upnp.org/> .

Internet-Draft THTP December 2007

Appendix A. Acknowledgments

The authors gratefully acknowledge the contributions of (alphabetically): Robert Beverly, Danny Cohen, David Dalrymple, and Karen Sollins.

Internet-Draft THTP December 2007

Authors' Addresses

Neil Gershenfeld
Massachusetts Institute of Technology
Room E15-411
20 Ames Street
Cambridge, MA 02139
US

Phone: +1 617 253 0392
Email: gersh@cba.mit.edu

Douglas Johnson
Sun Microsystems, Inc.
Mailstop UBUR02-306
One Network Drive
Burlington, MA 01803
US

Phone: +1 781 442 0716
Email: douglas.johnson@sun.com

Todd Snide
Schneider Electric
1 High Street
North Andover, MA 01845
US

Phone: +1 978 975 9472
Email: todd.snide@us.schneider-electric.com

Kerry Lynn
Cisco Systems
1414 Massachusetts Avenue
Boxborough, MA 01719
US

Phone: +1 978 936 1342
Email: kelynn@cisco.com

Internet-Draft THTP December 2007

Full Copyright Statement

Copyright (C) The Internet Society (2007).

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 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.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).

