Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 4, 2007

N. Gershenfeld
CBA/MIT
D. Johnson
Sun Microsystems, Inc.
T. Snide
Schneider Electric
K. Lynn
Cisco
October 2006

Trivial Hypertext Transfer Protocol (THTP)
draft-gershenfeld-thtp-00.txt

Status of this Memo

Copyright Notice

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 leveraging HTTP's URI naming scheme. This document describes the THTP protocol.

Table of Contents

1. Introduction

The HTTP protocol [RFC1945] introduced a powerful naming construct that simultaneously identifies and locates resources. The anticipated proliferation of smart, tiny, networked devices require a standards-based naming scheme [ieee-i0], [i0], but often cannot tolerate the overhead necessitated by the current tight coupling between HTTP and TCP [RFC0793]. Specifically, small, inexpensive 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.

For example, the application of a networked light switch sending an "on" instruction to a 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.

This document 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 widen the range of possible environments where HTTP-like semantics are used. For example, a similar but more complex scheme is used by UPnP [upnp]; THTP codifies a simple, portable, standards-based means of extending HTTP to UDP.

2. THTP Design

THTP is designed to be lightweight and easy for applications and application designers to implement. THTP uses HTTP's URI naming scheme. THTP differs from HTTP in several important ways. These differences are conscious design decisions based on THTP's intended environment of limited computational power or bandwidth and single-packet exchanges. This section details critical aspects of the THTP design.

2.1. Protocol Relationships

THTP uses UDP [RFC0768] as its transport layer. Figure 1 shows the relationship between THTP and various Internet protocols.

Protocol Relationships

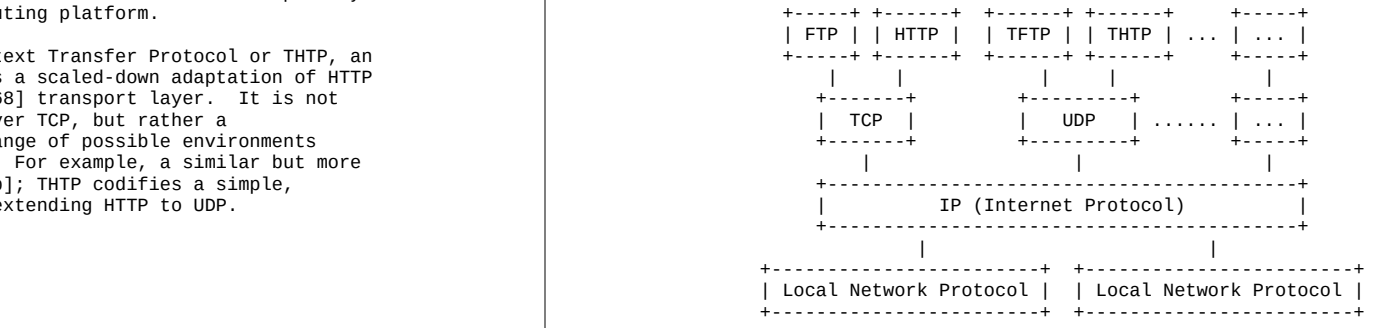

Figure 1

2.2. URI Format

THTP defines a naming scheme analogous to HTTP's Uniform Resource Locator (URL) which has been subsumed into the more abstract notion of a Uniform Resource Identifier (URI) [RFC3986]. THTP leverages the URI naming and large base of existing implementations to provide an efficient means of addressing and communicating with devices in a range of environments.

A THTP URI is semantically identically to those in HTTP, but with UDP as transport. The THTP URI format is defined as follows:

```
THTP_URI = "thtp:" "//" host [ ":" port ] [ abs_path [ "?" query]]
```

THTP URIs are processed identically to HTTP URIs. THTP uses UDP port 80 by default, but may use other UDP ports via the optional port specification in the URI.

2.3. Request Model

In contrast to the request/response paradigm of HTTP, responses in THTP are not required. Responses may be either unnecessary or implicit via an out-of-band channel. Note that THTP does not preclude request responses, however it explicitly separates the two functions. A user may query the state of the light bulb remotely, receive a notification that the filament is burned out, or require a device to periodically report its status via THTP. The separation of request and response in THTP has several design implications, most notably on reliability and message size.

A THTP message must fit entirely in a single UDP packet. THTP cannot transfer data larger than a single packet in a single request. Applications that require ordered delivery, large messages, flow control or congestion control should use HTTP (over TCP). Multiple messages cannot be placed within a single THTP packet. As such, "chunked" transfer-coding is not allowed.

2.4. Reliability

THTP uses UDP as its transport layer. Since UDP is an unreliable transport protocol and THTP does not include reliability, THTP makes no guarantees of packet delivery. For example, light switches and home appliances, the user receives immediate feedback: the room illuminates, therefore the request over the network was successful. If the packet is lost and the light bulb does not light, the user can actuate the light switch again.

However, THTP does not preclude reliability at other layers where necessary to support specific applications. Consider for example an application that requires lightweight, reliable single-message passing but not ordered delivery. Such an application could use THTP in conjunction with reliability at the physical or application layer. In addition, automated systems without an out-of-band feedback mechanism require additional verification, either by actively querying the state of the remote device or by adding application layer reliability.

2.5. Multicast

Because THTP uses UDP, it is possible to send a THTP message to a multicast group address. 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 receive the message and switch on. While Internet multicast deployment is limited, THTP multicast messages are useful in many local area networks.

3. HTTP Features

THTP must implement a minimum subset of HTTP's features, but is not required to implement all of HTTP. The subset of features THTP must support are a natural consequence of using UDP and maintaining simplicity. Additional HTTP mechanisms may be implemented on an application specific basis by prearrangement. However, a THTP server may always legally respond with the status code 501 (not implemented) as needed. Status codes are documented in Section 3.3.

At a minimum, THTP clients and servers must support the HTTP GET and POST 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.

3.1. 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

3.2. POST Method

THTP must also 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.

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

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

3.3. Status Code Definitions

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

3.3.1. 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.

3.3.2. 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.

3.3.3. 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.

3.4. 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.

3.5. 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. Caching is not explicitly forbidden, but THTP's request model anticipates all requests to be carried through to the final recipient,

4. 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 must rely on provider ingress filtering [RFC2827]. Instead, THTP may use appropriate lightweight encryption [sea] and/or authentication.

5. References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[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/ .

Appendix A. Acknowledgments

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

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, Inc
1414 Massachusetts Avenue
Boxborough, MA 01719
US

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

Acknowledgment

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