Abstract

This document describes a simple extension of the Uniform Resource Identifier (URI) format that allows preferred transport mechanisms, including protocols, ports and interfaces, to be specified as
parseable additions to the scheme name. This explicit configuration is beneficial for separation of the HyperText Transfer Protocol (HTTP) from underlying transports, which has been increasingly recognised as useful when a variety of ways of transporting or configuring use of HTTP are available and a choice of mechanism to use must be indicated.

Table of Contents

1. Background and Introduction ........................................... 3
2. Extending the URI scheme to indicate transports and interfaces ........................................... 4
3. Relevant work .......................................................... 6
4. Security Considerations .............................................. 7
5. IANA Considerations ................................................. 7
6. Acknowledgements .................................................... 7
7. References .............................................................. 7
   7.1. Normative References ............................................ 7
   7.2. Informative References ........................................... 8
Author’s Address ......................................................... 9
1. Background and Introduction

Desire to separate the Hypertext Transfer Protocol (HTTP) [RFC2616] from its traditional transport of the Transmission Control Protocol (TCP) is increasing.

There are environments where TCP is not suitable, or absent, yet HTTP can still be used as a method to transfer data. Being able to indicate the desired transport and interface to use in the URI for a program to interpret when executing HTTP GETs or PUTs is useful when a choice of mechanisms and interfaces are available, and infrastructure such as DNS cannot be queried for advice.

This document outlines how the desired transport and interface can be indicated in the Uniform Resource Identifier (URI) format [RFC3986] by a simple extension to that format using existing syntax.

This syntax is useful for carrying HTTP over different transport protocols. HTTP can be thought of a session layer, running over a transport layer providing reliable delivery of the HTTP stream. This transport layer has commonly (and almost universally) been 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. This has been described in detail [I-D.wood-dtnrg-http-dtn-delivery].

HTTP requires only reliable streaming that can be used to provide ordered delivery to the application; how that reliable streaming is provided is up to the local transport layer in the local network. In the examples given above, TCP or SCTP are used to carry HTTP over the congestion-sensitive public Internet, while Saratoga would be used for HTTP across dedicated private links.

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 even have to run over a TCP/IP stack, but could even be made to run directly over something else entirely. Given the prevalence of IP in many networks, it is likely that two popular waists (layers that other layers interface to) exist: IP and
HTTP. The transport protocol and physical environment that IP and HTTP are used with will vary more, depending on local conditions and needs.

Being able to specify how HTTP or other schemes are carried is useful when a variety of methods are available to choose from. The syntax described here is useful for local configuration, e.g. in a scripting language that is aware of the local host and remote host's shared support of a given transport protocol. It is less useful on the public world-wide web, because users (and web page designers) are not generally capable of determining which transport protocol(s) are supported by their web browser, operating system, or network. However, the option to explicitly choose a communication method is useful.

2. Extending the URI scheme to indicate transports and interfaces

Before describing this URI scheme syntax, it is worthwhile to lay out the foundations on which this syntax is based.

HTTP is not explicitly tied to use over TCP. To quote [RFC2616], section 1.4:

"HTTP communication usually takes place over TCP/IP connections. The default port is TCP 80, but other ports can be used. This does not preclude HTTP from being implemented on top of any other protocol on the Internet, or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used; the mapping of the HTTP/1.1 request and response structures onto the transport data units of the protocol in question is outside the scope of this specification."

The URI format syntax ([RFC3986], section 3.1) defines the scheme as:

scheme = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )

The period (.) is in use in a number of scheme names; the other punctuation characters appear unused.

To quote [RFC2718], section 2.2.2:

"When a scheme is associated with a network protocol, the specification should completely describe how URLs are translated into protocol actions in sufficient detail to make the access of the network resource unambiguous. If an implementation of the URL scheme requires some configuration, the configuration elements must be clearly identified."
With this foundation, this draft proposes that schemes can be extended to include configuration elements that indicate transport and interface. These modify use of the http request, and are in the format:

```
scheme = scheme [ "--" port behaviour ] [ "++" transport ] ["+-" interface ] [ "--" dynamic configuration]
```

where the optionally-included --port is a description mapping to the default IANA-assigned port number, or the equivalent name, indicating the desired behaviour over a transport. This information is specified locally as service names in e.g. /etc/services on unix machines.

The optionally-included ++transport is the transport name or IANA protocol identifier number that that name maps to. As we aim to separate HTTP from TCP and place it over other transports, nicknaming this proposal ‘http++’ obviously follows.

The optionally-included +interface can contain a locally-meaningful specifier identifying the local interface to use; useful on multi-homed devices. The local interface identifier may identify a virtual or physical interface or one that is only capable of using IPv4, IPv6, or another network protocol (in the case of HTTP being run over something other than a TCP/IP stack.) Defining virtual interfaces limited to one network protocol allows the choice of network protocol to be made.

The optionally-included "--" gets configuration information and transport/services to use dynamically via some method, e.g. DNS records, but can be partly overridden by other locally-provided configuration from the other three modifier types described above. This is described more in the ‘relevant work’ section below.

Use of these modifiers would permit http++sctp:// or http++saratoga++udp:// for the uses outlined in [I-D.natarajan-http-over-sctp] and [I-D.wood-dtnrg-http-dtn-delivery]. Port and internet protocol numbers assigned by IANA are accepted as equivalent to assigned names for these underlying protocols, so http++7542++17:// specifies HTTP over Saratoga over UDP. http++132 is equivalent to http++sctp in specifying HTTP over SCTP. As is usual, these are case insensitive, so that http++sctp, HTTP++sctp, and HtTp++ScTp are all equivalent.

Adding these optional transport indicators to the scheme name does not change the namespace in any way; the URI should still be treated as if it began simply http: and is the http namespace. Similarly, ‘https++’ shares the https namespace, only indicating services,
transports and interfaces for the https request to use.

This document deliberately does not state what ’http:’ by itself implies; local use may invoke dynamic configuration, or simultaneous attempts to use multiple available transports or interfaces. ’http:’ is NOT restricted to mean ’http++tcp:’.

If required, the port the scheme is actually run over, which the behaviour of any specified default port is mapped to, is still indicated later in the URI as :number, e.g. :80. When this is not specified the default port for that transport behaviour is used. For example, Saratoga runs over UDP and is assigned default port 7452 by IANA. Saying ’http--7452:’ means do the default behaviour for port 7452, i.e. use the Saratoga protocol. http--7452://blah:1024 indicates that the default behaviour on port 7452 -- that is, Saratoga -- should be carried over to port 1024, i.e. use the Saratoga protocol on port 1024 instead.

Knowing that a ’+-’ service port behaviour stated in the scheme is only associated with either TCP or UDP would mean that ++tcp or ++udp can be omitted when that --+behaviour is given. That is, --+saratoga is known to mean --+saratoga++udp.

Being able to specify the local interface to initiate a transaction on when a choice of interfaces is available on a multihomed device is useful, e.g. http--saratoga--serial0://.

3. Relevant work

[I-D.natarajan-http-over-sctp] proposes carrying http over sctp. This can be indicated with http++sctp:.

[I-D.wood-dtnrg-http-dtn-delivery] proposes separating HTTP from the underlying transport entirely, and running it over other transports, such as Saratoga [I-D.wood-tsvwg-saratoga]. This can be specified locally with http++saratoga++udp: or simply http++saratoga:.

We can also use ’--’ to indicate that information on resolving the URI must be sought from the network.

[I-D.jennings-http-srv] has proposed using DNS lookup of a SRV record to return a dynamic port value, as well as an address, and could indicate this using http--srv:// and http--srv:// in line with the use of a service name as indicated above. [Ed note: need to double-check Cullen’s current draft.]

Alternatively, some new DNS record type returning address, port and
other access information could be explicitly accessed via e.g. http--dns:// or some other indication of method. This could take advantage of DNS Name Authority Pointers (NAPTR) via S-NAPTR and U-NAPTR [RFC3958] [RFC4848], and encourage the use of and testing with those protocols before wider deployment. [I-D.faltstrom-uri] expands further on this approach. Both SRV and NAPTR can be used by SIP in allocating SIP servers [RFC3263].

Other work on evolving the URI format to enable service discovery with DNS for different transport protocols is in e.g. [Uruena05].

It should be possible to combine the static configuration in the parseable scheme format described here with getting other configuration information that is not explicitly given, but that is needed to access the URI dynamically, from DNS records when appropriate. Any static information explicitly provided should override information from dynamic configuration, just as explicitly indicating a port with e.g. :80 in the URI explicitly overrides the port returned by http--srv:.

4. Security Considerations

No additional security concerns have been thought of at this time.

5. IANA Considerations

No additional IANA considerations have been thought of at this time.

6. Acknowledgements

Thanks go to Fred Baker, Leslie Daigle, Cullen Jennings, Jonathan Leighton, Preethi Natarajan, Chip Sharp and Dan Wing for discussion on points of this draft.

7. References

7.1. Normative References


7.2. Informative References

[Deering98]

[I-D.faltstrom-uri]

[I-D.jennings-http-srv]

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

[I-D.wood-dtnrg-http-dtn-delivery]

[I-D.wood-tsvwg-saratoga]


[RFC4848] Daigle, L., "Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service
(DDDS), RFC 4848, April 2007.

[Uruena05]  

Author’s Address

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