

intarea
Internet-Draft
Intended status: Informational
Expires: January 1, 2018

B. Bruneau
Ecole Polytechnique
P. Pfister
Cisco
D. Schinazi
T. Pauly
Apple
E. Vyncke
Cisco
June 30, 2017

Discovering Provisioning Domain Names and Data
draft-bruneau-intarea-provisioning-domains-01

Abstract

An increasing number of hosts and networks are connected to the Internet through multiple interfaces, some of which may provide multiple ways to access the internet by the mean of multiple IPv6 prefix configurations.

This document describes a way for hosts to retrieve additional information about their network access characteristics. The set of configuration items required to access the Internet is called a Provisioning Domain (PvD) and is identified by a Fully Qualified Domain Name (FQDN). This identifier, retrieved using a new Router Advertisement (RA) option, is associated with the set of information included within the RA and may later be used to retrieve additional information associated to the PvD by the mean of an HTTP request.

Status of This Memo

This Internet-Draft is submitted 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 January 1, 2018.

Copyright Notice

Copyright (c) 2017 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. Introduction	3
2. Terminology	3
3. Provisioning Domain Identification using Router Advertisements	4
3.1. PvD ID Option for Router Advertisements	4
3.2. Router Behavior	5
3.3. Host Behavior	5
3.3.1. DHCPv6 configuration association	6
3.3.2. DHCPv4 configuration association	7
3.3.3. Interconnection Sharing by the Host	7
4. Provisioning Domain Identification using IKEv2	7
5. Provisioning Domain Additional Information	8
5.1. Retrieving the PvD Additional Information	9
5.2. Providing the PvD Additional Information	10
5.3. PvD Additional Information Format	10
5.3.1. Connectivity Characteristics Information	12
5.3.2. Private Extensions	12
5.3.3. Example	12
6. Security Considerations	13
7. IANA Considerations	14
8. Acknowledgements	14
9. References	14
9.1. Normative references	14
9.2. Informative references	15
Appendix A. Changelog	16
A.1. Version 00	16
A.2. Version 01	16
Appendix B. Connection monetary cost	17
B.1. Conditions	17
B.2. Price	18
B.3. Examples	19

Authors' Addresses	20
--------------------	----

1. Introduction

It has become very common in modern networks that hosts have Internet or more specific network access through different networking interfaces, tunnels, or next-hop routers. The concept of Provisioning Domain (PvD) was defined in [RFC7556] as a set of network configuration information which can be used by hosts in order to access the network.

In this document, PvDs are identified by Fully Qualified Domain Names called PvD IDs, which are included in Router Advertisements [RFC4861] as a new option and are used to identify correlated sets of network configuration data.

Devices connected to the Internet through multiple interfaces would typically be provisioned with one PvD per interface, but it is worth noting that multiple PvDs with different PvD IDs could be provisioned on any host interface, as well as noting that the same PvD ID could be used on different interfaces in order to inform the host that both PvDs, on different interfaces, ultimately provide identical services.

This document also introduces a way for hosts to retrieve additional information related to a specific PvD by the mean of an HTTP-over-TLS query using an URI derived from the PvD ID. The retrieved JSON object contains additional network information that would typically be considered unfit, or too large, to be directly included in the Router Advertisements. This information can be used by the networking stack, the applications, or even be partially displayed to the users (e.g., by displaying a localized network service name).

2. Terminology

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

In addition, this document uses the following terminology:

PvD: A Provisioning Domain, a set of network configuration information; for more information, see [RFC7556].

PvD ID: A Fully Qualified Domain Name (FQDN) used to identify a PvD.

Explicit PvD: A PvD uniquely identified with a PvD ID.

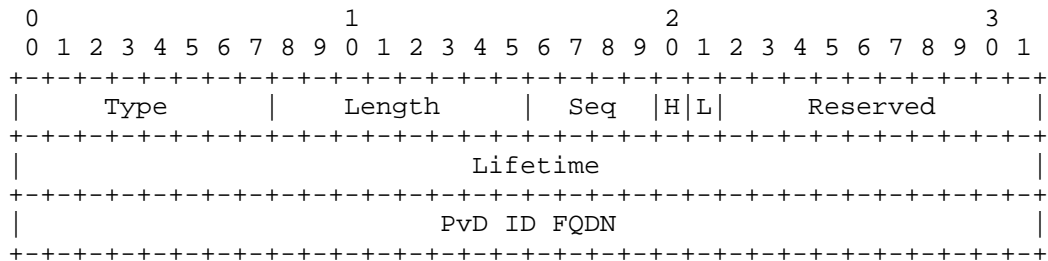
Implicit PvD: A PvD associated with a set of configuration information that, in the absence of a PvD ID, is associated with the advertising router.

3. Provisioning Domain Identification using Router Advertisements

Each provisioning domain is identified by a PvD ID. The PvD ID is a Fully Qualified Domain Name (FQDN) which MUST belong to the network operator in order to avoid ambiguity. The same PvD ID MAY be used in several access networks if the set of configuration information is identical (e.g. in all home networks subscribed to the same service).

3.1. PvD ID Option for Router Advertisements

This document introduces a new Router Advertisement (RA) option called the PvD ID Router Advertisement Option, used to convey the FQDN identifying a given PvD.



PvD ID Router Advertisements Option format

Type : (8 bits) To be defined by IANA.

Length : (8 bits) The length of the option (including the Type and Length fields) in units of 8 octets.

Seq : (4 bits) Sequence number for the PvD Additional Information, as described in Section 5.

H-flag : (1 bit) Whether some PvD Additional Information is made available through HTTP over TLS, as described in Section 5.

L-flag : (1 bit) Whether the router is also providing IPv4 access using DHCPv4 (see Section 3.3.2).

Reserved : (10 bits) Reserved for later use. It MUST be set to zero by the sender and ignored by the receiver.

Lifetime : (32 bits) The length of time in seconds (relative to the time the packet is sent) that the PvD ID option is valid. A value of all one bits (0xffffffff) represents infinity.

PvD ID FQDN : The FQDN used as PvD ID in DNS binary format [RFC1035], padded until the next 8 octets boundary. All the bytes after the first empty DNS label MUST be set to zero by the sender and MUST be ignored by the receiver. The DNS name compression technique described in [RFC1035] MUST NOT be used.

Routers MUST NOT include more than one PvD ID Router Advertisement Option in each RA. In case multiple PvD ID options are found in a given RA, hosts MUST ignore all but the first PvD ID option.

Note: The existence and/or size of the sequence number is subject to discussion. The validity of a PvD Additional Information object is included in the object itself, but this only allows for 'pull based' updates, whereas the RA options usually provide 'push based' updates.

Note: including the lifetime option is congruent with the lifetime option of the other RA option. On the other hand, do we need it ?

3.2. Router Behavior

A router MAY insert at most one PvD ID Option in its RAs. The included PvD ID is associated with all the other options included in the same RA (e.g., Prefix Information [RFC4861], Recursive DNS Server [RFC6106], Routing Information [RFC4191], Captive Portal [RFC7710] options).

In order to provide multiple independent PvDs, a router MUST send multiple RAs using different source link-local addresses (LLA) (as proposed in [I-D.bowbakova-rtgwg-enterprise-pa-multihoming]), each of which MAY include a PvD ID option. In such cases, routers MAY originate the different RAs using the same datalink layer address.

If the router is actually a VRRP instance [RFC5798], then the procedure is identical except that the virtual datalink layer address is used as well as virtual IPv6 addresses.

3.3. Host Behavior

RAs are used to configure IPv6 hosts. When a host receives a RA message including a PvD ID Option with a non-zero lifetime, it MUST associate all the configuration options included in the RA (e.g., Prefix Information [RFC4861], Recursive DNS Server [RFC6106], Routing Information [RFC4191], Captive Portal [RFC7710] options) with the PvD ID).

If the received RA does not include a PvD ID Option, or whenever the included PvD ID Option has a lifetime set to zero, the host **MUST** associate the options included in the RA with an implicit PvD. This implicit PvD is identified by the link-local address of the router sending the RA and the interface on which the RA was received.

This document does not update the way Router Advertisement options are processed. But in addition to the option processing defined in other documents, hosts implementing this specification **MUST** associate each created or updated object (e.g. address, default route, more specific route, DNS server list) with the PvD associated with the received RA as well as the interface and link-local address of the router which last updated the object.

Each configuration object has a PvD validity timer set to the PvD ID option lifetime whenever an RA that updates the object is received. If the received RA does not include a PvD ID option, or whenever the PvD ID option lifetime is zero, the associated objects are immediately associated with an implicit PvD, as mentioned in the previous paragraph. Similarly, whenever an object's PvD timer reaches zero, the object is immediately associated with an implicit PvD identified by the link-local address and interface of the router which last updated the object.

While resolving names, executing the default address selection algorithm [RFC6724] or executing the default router selection algorithm ([RFC2461], [RFC4191] and [RFC8028]), hosts **MAY** consider only the configuration associated with an arbitrary set of PvDs.

For example, a host **MAY** associate a given process with a specific PvD, or a specific set of PvDs, while associating another process with another PvD. A PvD-aware application might also be able to select, on a per-connection basis, which PvDs should be used for a given connection. And particularly constrained devices such as small battery operated devices (e.g. IoT), or devices with limited CPU or memory resources may purposefully use a single PvD while ignoring some received RAs containing different PvD IDs.

The way an application expresses its desire to use a given PvD, or a set of PvDs, or the way this selection is enforced, is out of the scope of this document. Useful insights about these considerations can be found in [I-D.kline-mif-mpvd-api-reqs].

3.3.1. DHCPv6 configuration association

When a host retrieves configuration information from DHCPv6, the configured elements **MUST** be associated with the explicit or implicit PvD of the RA received on the same interface with the O-flag set

[RFC4861]. If multiple RAs with the O-flag set and associated with different PvDs are received on the same interface, the link-local address of the DHCPv6 server MAY be compared with the routers' link-local addresses in order to disambiguate. If the disambiguation is impossible, then the DHCPv6 configuration information MUST be associated with an implicit PvD.

This process requires hosts to keep track of received RAs, associated PvD IDs, and routers link-local addresses.

3.3.2. DHCPv4 configuration association

When a host retrieves configuration from DHCPv4 on an interface, the configured elements MUST be associated with the explicit PvD, received on the same interface, whose PVD ID Options L-flag is set. If multiple different PvDs are found, the datalink layer address used by the DHCPv4 server/relay MAY be compared with the datalink layer address used by on-link advertising routers in order to disambiguate. If no RA associated with a PvD ID option with the L-flag set is found, or if the disambiguation fails, the IPv4 configuration information MUST be associated with an implicit PvD.

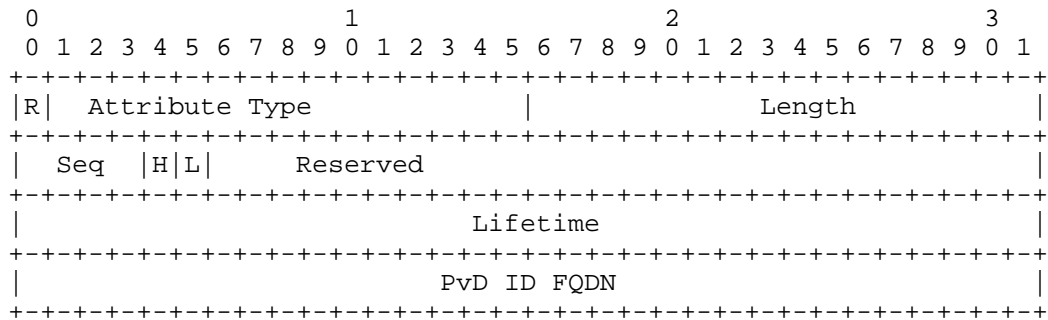
3.3.3. Interconnection Sharing by the Host

The situation when a host becomes also a router by acting as a router or ND proxy on a different interface (such as WiFi) to share the connectivity of another interface (such as cellular), also known as "tethering" is TBD but it is expected that the one or several PvD associated to the shared interface will be also advertised to the clients.

4. Provisioning Domain Identification using IKEv2

RFC 7296 defines Internet Key Exchange version 2 (IKEv2) which includes in sections 2.19 and 3.15 a Configuration Payload (CP) which may be used by an IPsec client to request configuration items (e.g., addresses, recursive DNS servers). IKEv2 also negotiates traffic selectors which represent the 5-tuple(s) to be protected by the Security Association (SA) negotiated by IKEv2. This set of information is another PvD and may include INTERNAL_IP6_ADDRESS, INTERNAL_IP6_LINK, INTERNAL_IP6_SUBNET (to be used to route traffic to this subent), INTERNAL_IP6_DNS, INTERNAL_DNS_DOMAIN.

The PvD ID Configuration option for IKEv2 has the following structure (similar to the RA option):



PvD ID IKEv2 Configuration Payload Attributes format

R-bit Reserved (1 bit) - Defined in IKEv2 RFC [RFC7296].

Attribute Type (15 bits) tbd by IANA PVD_ID.

Length Length (2 octets, unsigned integer) - Length of PvD ID FQDN + 2.

Seq Sequence number (4 bits) for the PvD Additional Information, as described in Section 5.

H-flag (1 bit) Whether some PvD Additional Information is made available through HTTP over TLS, as described in Section 5.

L-flag (1 bit) must be set to 0 if the Configuration Payload contains only IPv6 information, else it must be set to 1.

Reserved (10 bits) Reserved for later use. It MUST be set to zero by the sender and ignored by the receiver.

PvD ID FQDN The FQDN used as PvD ID in DNS binary format [RFC1035], padded until the next 8 octets boundary. All the bytes after the first empty DNS label MUST be set to zero by the sender and MUST be ignored by the receiver. The DNS name compression technique described in [RFC1035] MUST NOT be used.

5. Provisioning Domain Additional Information

Once a new PvD ID is discovered, it may be used to retrieve additional information about the characteristics of the provided connectivity. This set of information is called PvD Additional Information, and is encoded as a JSON object [RFC7159].

The purpose of this additional set of information is to securely provide additional information to hosts about the connectivity that is provided using a given interface and source address pair. It typically includes data that would be considered too large, or not critical enough, to be provided with an RA option. The information contained in this object MAY be used by the operating system, network libraries, applications, or users, in order to decide which set of PvDs should be used for which connection, as described in Section 3.3.

5.1. Retrieving the PvD Additional Information

When the H-flag of the PvD ID Option is set, hosts MAY attempt to retrieve the PvD Additional Information associated with a given PvD by performing an HTTP over TLS [RFC2818] GET query to `https://<PvD-ID>/pvd.json` [RFC5785]. On the other hand, hosts MUST NOT do so whenever the H-flag is not set.

Note: Should the PvD AI retrieval be a MAY or a SHOULD ? Could the object contain critical data, or should it only contain informational data ?

Note that the DNS name resolution of <PvD-ID> as well as the actual query MUST be performed in the PvD context associated to the PvD ID. In other words, the name resolution, source address selection, as well as the next-hop router selection MUST be performed while using exclusively the set of configuration information attached with the PvD, as defined in Section 3.3.

If the HTTP status of the answer is greater than or equal to 400 the host MUST abandon and consider that there is no additional PvD information. If the HTTP status of the answer is between 300 included and 399 included it MUST follow the redirection(s). If the HTTP status of the answer is between 200 included and 299 included the host MAY get a file containing a single JSON object. When a JSON object could not be retrieved, an error message SHOULD be logged and/or displayed in a rate-limited fashion.

After retrieval of the PvD Additional Information, hosts MUST watch the PvD ID Seq field for change. In case a different value than the one in the RA Seq field is observed, or whenever the validity time included in the object is expired, hosts MUST either perform a new query and retrieve a new version of the object, or deprecate the object and stop using it.

Hosts retrieving a new PvD Additional Information object MUST check for the presence and validity of the mandatory fields Section 5.3. A retrieved object including an outdated expiration time or missing a

mandatory element, MUST be ignored. In order to avoid traffic spikes toward the server hosting the PvD Additional Information when an object expires, a host which last retrieved an object at a time A, including a validity time B, SHOULD renew the object at a uniformly random time in the interval $[(B-A)/2, A]$.

In order to prevent PvD spoofing by malicious or misconfigured routers, PvD Additional Information object includes a set of IPv6 prefixes which MUST be checked against all the Prefix Information Options advertised in the Router Advertisement. If any of the prefixes included in the Prefix Information Options is not in the set of prefixes contained in the additional information, the PvD (the one included in the RA and in the additional information) MUST be considered unsafe and MUST NOT be used. While this does not prevent a malicious network provider, it can be used to detect misconfiguration.

The server serving the JSON files SHOULD also check whether the client address is part of the prefixes listed into the additional information and SHOULD return a 403 responsecode if there is no match. The server MAY also use the client address to select the right JSON object to be returned.

Note: In a similar way, a DNS server names list could be included in the PvD AI in order to avoid rogue APs from inserting a different DNS resolver. But this would also prevent CPEs from advertising themselves as default DNS (which is usually done). SPs which really want to use CPEs as DNS, for caching reasons, might find 'creative' ways to go around this rule.

5.2. Providing the PvD Additional Information

Whenever the H-flag is set in the PvD RA Option, a valid PvD Additional Information object MUST be made available to all hosts receiving the RA. In particular, when a captive portal is present, hosts MUST still be allowed to access the object, even before logging into the captive portal.

Routers MAY increment the PVD ID Sequence number (Seq) in order to inform host that a new PvD Additional Information object is available and should be retrieved.

5.3. PvD Additional Information Format

The PvD Additional Information is a JSON object.

The following array presents the mandatory keys which MUST be included in the object:

JSON key	Description	Type	Example
name	Human-readable service name	UTF-8 string	"Awesome Wifi"
expires	Date after which this object is not valid	ISO 8601	"2017-07-23T06:00:00Z"
prefixes	Array of IPv6 prefixes valid for this PVD	Array of strings	["2001:db8:1::/48", "2001:db8:4::/48"]

A retrieved object which does not include a valid string associated with the "name" key at the root of the object, or a valid date associated with the "expiration" key, also at the root of the object, MUST be ignored. In such cases, an error message SHOULD be logged and/or displayed in a rate-limited fashion.

The following table presents some optional keys which MAY be included in the object.

JSON key	Description	Type	Example
localizedName	Localized user-visible service name, language can be selected based on the HTTP Accept-Language header in the request.	UTF-8 string	"Wifi Genial"
noInternet	No Internet, set when the Pvd only provides restricted access to a set of services.	boolean	true
characteristics	Connectivity characteristics	JSON object	See Section 5.3.1
metered	metered, when the access volume is limited.	boolean	false

It is worth noting that the JSON format allows for extensions. Whenever an unknown key is encountered, it MUST be ignored along with its associated elements, unless specified otherwise in future updating documents.

5.3.1. Connectivity Characteristics Information

The following set of keys can be used to signal certain characteristics of the connection towards the PvD.

They should reflect characteristics of the overall access technology which is not limited to the link the host is connected to, but rather a combination of the link technology, CPE upstream connectivity, and further quality of service considerations.

JSON key	Description	Type	Example
maxThroughput	Maximum achievable throughput	object({down(int), up(int)}) in kb/s	{"down": 10000, "up": 5000}
minLatency	Minimum achievable latency	object({down(int), up(int)}) in ms	{"down": 10, "up": 20}
rl	Maximum achievable reliability	object({down(int), up(int)}) in losses every 1000 packets	{"down": 0.1, "up": 1}

5.3.2. Private Extensions

JSON keys starting with "x-" are reserved for private use and can be utilized to provide information that is specific to vendor, user or enterprise. It is RECOMMENDED to use one of the patterns "x-FQDN-KEY" or "x-PEN-KEY" where FQDN is a fully qualified domain name or PEN is a private enterprise number [PEN] under control of the author of the extension to avoid collisions.

5.3.3. Example

Here are two examples based on the keys defined in this section.

```
{
  "name": "Foo Wireless",
  "localizedName": "Foo-France Wifi",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
  "characteristics": {
    "maxThroughput": { "down": 200000, "up": 50000 },
    "minLatency": { "down": 0.1, "up": 1 }
  }
}
```

```
{
  "name": "Bar 4G",
  "localizedName": "Bar US 4G",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
  "metered": true,
  "characteristics": {
    "maxThroughput": { "down":80000, "up": 20000 }
  }
}
```

6. Security Considerations

Although some solutions such as IPSec or SEND [RFC3971] can be used in order to secure the IPv6 Neighbor Discovery Protocol, actual deployments largely rely on link layer or physical layer security mechanisms (e.g. 802.1x [IEEE8021X]).

This specification does not improve the Neighbor Discovery Protocol security model, but extends the purely link-local configuration retrieval mechanisms with HTTP-over-TLS communications.

During the exchange, the server authenticity is verified by the mean of a certificate, validated based on the FQDN found in the Router Advertisement (e.g. using a list of pre-installed CA certificates, or DNSSEC [RFC4035] with DNS Based Authentication of Named Entities [RFC6698]). This authentication creates a secure binding between the information provided by the trusted Router Advertisement, and the HTTP server.

The IPv6 prefixes list included in the PvD Additional Information JSON object is used to validate that the prefixes included in the Router Advertisements are really part of the PvD.

Note: The previous point does not protect against attacker performing NAT66. It would if we mandate the source-address validation on the server side, but not protect against tunnels. In other words, address based security will not protect against rogue APs, but may prevent the use of NAT66.

For privacy reasons, it is desirable that the PvD Additional Information object may only be retrieved by the hosts using the given PvD. Host identity SHOULD be validated based on the client address that is used during the HTTP query.

7. IANA Considerations

IANA is kindly requested to allocate a new IPv6 Neighbor Discovery option number for the PvD ID Router Advertisement option and a new IKEv2 Configuration Payload Attribute Types for PVD_ID.

TBD: JSON keys will need a new register.

8. Acknowledgements

Many thanks to M. Stenberg and S. Barth for their earlier work: [I-D.stenberg-mif-mpvd-dns].

Thanks also to Ray Bellis, Lorenzo Colitti, Thierry Danis, Marcus Keane, Erik Kline, Jen Lenkova, Mark Townsley and James Woodyatt for useful and interesting discussions.

Finally, many thanks to Thierry Danis for his implementation work: [github].

9. References

9.1. Normative references

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2461] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<http://www.rfc-editor.org/info/rfc2818>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <<http://www.rfc-editor.org/info/rfc5785>>.

- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <<http://www.rfc-editor.org/info/rfc7159>>.

9.2. Informative references

- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<http://www.rfc-editor.org/info/rfc4035>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.
- [RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <<http://www.rfc-editor.org/info/rfc5798>>.
- [RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, November 2010.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <<http://www.rfc-editor.org/info/rfc6698>>.
- [RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<http://www.rfc-editor.org/info/rfc7556>>.
- [RFC7710] Kumari, W., Gudmundsson, O., Ebersman, P., and S. Sheng, "Captive-Portal Identification Using DHCP or Router Advertisements (RAs)", RFC 7710, DOI 10.17487/RFC7710, December 2015, <<http://www.rfc-editor.org/info/rfc7710>>.
- [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <<http://www.rfc-editor.org/info/rfc8028>>.

- [I-D.bowbakova-rtgwg-enterprise-pa-multihoming]
Baker, F., Bowers, C., and J. Linkova, "Enterprise Multihoming using Provider-Assigned Addresses without Network Prefix Translation: Requirements and Solution", draft-bowbakova-rtgwg-enterprise-pa-multihoming-01 (work in progress), October 2016.
- [I-D.stenberg-mif-mpvd-dns]
Stenberg, M. and S. Barth, "Multiple Provisioning Domains using Domain Name System", draft-stenberg-mif-mpvd-dns-00 (work in progress), October 2015.
- [I-D.kline-mif-mpvd-api-reqs]
Kline, E., "Multiple Provisioning Domains API Requirements", draft-kline-mif-mpvd-api-reqs-00 (work in progress), November 2015.
- [PEN] IANA, "Private Enterprise Numbers", <<https://www.iana.org/assignments/enterprise-numbers>>.
- [IEEE8021X]
IEEE, "IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control, IEEE Std", .
- [github] Cisco, "IPv6-mPvD github repository", <<https://github.com/IPv6-mPvD>>.

Appendix A. Changelog

Note to RFC Editors: Remove this section before publication.

A.1. Version 00

Initial version of the draft. Edited by Basile Bruneau + Eric Vyncke and based on Basile's work.

A.2. Version 01

Major rewrite intended to focus on the the retained solution based on corridors, online, and WG discussions. Edited by Pierre Pfister. The following list only includes major changes.

PvD ID is an FQDN retrieved using a single RA option. This option contains a sequence number for push-based updates, a new H-flag, and a L-flag in order to link the PvD with the IPv4 DHCP server.

A lifetime is included in the PvD ID option.

Detailed Hosts and Routers specifications.

Additional Information is retrieved using HTTP-over-TLS when the PvD ID Option H-flag is set. Retrieving the object is optional.

The PvD Additional Information object includes a validity date.

DNS-based approach is removed as well as the DNS-based encoding of the PvD Additional Information.

Major cut in the list of proposed JSON keys. This document may be extended later if need be.

Monetary discussion is moved to the appendix.

Clarification about the 'prefixes' contained in the additional information.

Clarification about the processing of DHCPv6.

Appendix B. Connection monetary cost

NOTE: This section is included as a request for comment on the potential use and syntax.

The billing of a connection can be done in a lot of different ways. The user can have a global traffic threshold per month, after which his throughput is limited, or after which he/she pays each megabyte. He/she can also have an unlimited access to some websites, or an unlimited access during the weekends.

An option is to split the bill in elementary billings, which have conditions (a start date, an end date, a destination IP address...). The global billing is an ordered list of elementary billings. To know the cost of a transmission, the host goes through the list, and the first elementary billing whose the conditions are fulfilled gives the cost. If no elementary billing conditions match the request, the host MUST make no assumption about the cost.

B.1. Conditions

Here are the potential conditions for an elementary billing. All conditions MUST be fulfill.

Key	Description	Type	JSON Example
beginDate	Date before which the billing is not valid	ISO 8601	"1977-04-22T06:00:00Z"
endDate	Date after which the billing is not valid	ISO 8601	"1977-04-22T06:00:00Z"
domains	FQDNs whose the billing is limited	array(string)	["deezer.com", "spotify.com"]
prefixes4	IPv4 prefixes whose the billing is limited	array(string)	["78.40.123.182/32", "78.40.123.183/32"]
prefixes6	IPv6 prefixes whose the billing is limited	array(string)	["2a00:1450:4007:80e::200e/64"]

B.2. Price

Here are the different possibilities for the cost of an elementary billing. A missing key means "all/unlimited/unrestricted". If the elementary billing selected has a trafficRemaining of 0 kb, then it means that the user has no access to the network. Actually, if the last elementary billing has a trafficRemaining parameter, it means that when the user will reach the threshold, he/she will not have access to the network anymore.

Key	Description	Type	JSON Example
pricePerGb	The price per Gigabit	float (currency per Gb)	2
currency	The currency used	ISO 4217	"EUR"
throughputMax	The maximum achievable throughput	float (kb/s)	100000
trafficRemaining	The traffic remaining	float (kB)	12000000

B.3. Examples

Example for a user with 20 GB per month for 40 EUR, then reach a threshold, and with unlimited data during weekends and to example.com:

```
[
  {
    "domains": ["example.com"]
  },
  {
    "prefixes4": ["78.40.123.182/32", "78.40.123.183/32"]
  },
  {
    "beginDate": "2016-07-16T00:00:00Z",
    "endDate": "2016-07-17T23:59:59Z",
  },
  {
    "beginDate": "2016-06-20T00:00:00Z",
    "endDate": "2016-07-19T23:59:59Z",
    "trafficRemaining": 12000000
  },
  {
    "throughputMax": 100000
  }
]
```

If the host tries to download data from example.com, the conditions of the first elementary billing are fulfilled, so the host takes this elementary billing, finds no cost indication in it and so deduces that it is totally free. If the host tries to exchange data with foobar.com and the date is 2016-07-14T19:00:00Z, the conditions of the first, second and third elementary billing are not fulfilled.

But the conditions of the fourth are. So the host takes this elementary billing and sees that there is a threshold, 12 GB are remaining.

Another example for a user abroad, who has 3 GB per year abroad, and then pay each MB:

```
[
  {
    "beginDate": "2016-02-10T00:00:00Z",
    "endDate": "2017-02-09T23:59:59Z",
    "trafficRemaining": 3000000
  },
  {
    "pricePerGb": 30,
    "currency": "EUR"
  }
]
```

Authors' Addresses

Basile Bruneau
Ecole Polytechnique
Vannes 56000
France

Email: basile.bruneau@polytechnique.edu

Pierre Pfister
Cisco
11 Rue Camille Desmoulins
Issy-les-Moulineaux 92130
France

Email: ppfister@cisco.com

David Schinazi
Apple

Email: dschinazi@apple.com

Tommy Pauly
Apple

Email: tpaully@apple.com

Eric Vyncke
Cisco
De Kleetlaan, 6
Diegem 1831
Belgium

Email: evyncke@cisco.com

INTAREA
Internet-Draft
Updates: RFC 4884 (if approved)
Intended status: Standards Track
Expires: December 25, 2017

R. Bonica
R. Thomas
Juniper Networks
J. Linkova
Google
C. Lenart
Verizon
M. Boucadair
Orange
June 23, 2017

PROBE: A Utility For Probing Interfaces
draft-ietf-intarea-probe-00

Abstract

This document describes a network diagnostic tool called PROBE. PROBE is similar to PING, in that it can be used to test the status of a probed interface. It differs from PING in that it does not require bidirectional connectivity between the probing and probed interfaces. Alternatively, PROBE requires bidirectional connectivity between the probing interface and a proxy interface. The proxy interface can reside on the same node as the probed interface or it can reside on a node to which the probed interface is directly connected.

Status of This Memo

This Internet-Draft is submitted 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 December 25, 2017.

Copyright Notice

Copyright (c) 2017 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. Introduction	2
1.1. Terminology	4
1.2. Requirements Language	4
2. ICMP Extended Echo Request	4
2.1. Interface Identification Object	6
3. ICMP Extended Echo Reply	7
4. ICMP Message Processing	8
4.1. Code Field Processing	10
5. Use-Cases	11
6. Updates to RFC 4884	11
7. IANA Considerations	11
8. Security Considerations	12
9. References	13
9.1. Normative References	13
9.2. Informative References	14
Appendix A. The PROBE Application	14
Acknowledgments	16
Authors' Addresses	16

1. Introduction

Network operators use PING [RFC2151] to test bidirectional connectivity between two interfaces. For the purposes of this document, we will call these interfaces the probing and probed interfaces. PING sends an ICMP [RFC0792] [RFC4443] Echo message from the probing interface to the probed interface. The probing interface resides on a probing node while the probed interface resides on a probed node.

If the probed interface receives the ICMP Echo message, it returns an ICMP Echo Reply. When the probing interface receives the ICMP Echo

Reply, it has verified bidirectional connectivity between the probing and probed interfaces. Specifically, it has verified that:

- o The probing node can reach the probed interface
- o The probed interface is active
- o The probed node can reach the probing interface
- o The probing interface is active

This document describes a network diagnostic tool called PROBE. PROBE is similar to PING, in that it can be used to test the status of a probed interface. It differs from PING in that it does not require bidirectional connectivity between the probing and probed interfaces. Alternatively, PROBE requires bidirectional connectivity between the probing interface and a proxy interface. The proxy interface can reside on the same node as the probed interface or it can reside on a node to which the probed interface is directly connected. Section 5 of this document describes scenarios in which this characteristic is useful.

Like PING, PROBE executes on a probing node. It sends an ICMP Extended Echo message from a local interface, called the probing interface, to a proxy interface. The proxy interface resides on a probed node.

The ICMP Extended Echo Request contains an ICMP Extension Structure and the ICMP Extension Structure contains an Interface Identification Object. The Interface Identification Object identifies the probed interface. The probed interface can reside on the probed node or it can be directly connected to the probed node.

When the proxy interface receives the ICMP Extended Echo Request, it executes access control procedures. If access is granted, the probed node determines the status of the probed interface and returns an ICMP Extended Echo Reply Message. The ICMP Extended Echo Reply indicates the status of the probed interface.

If the probed interface resides on the probed node, PROBE determines the status of the probed interface as it would determine its MIB-II [RFC2863] `ifOperStatus`. If `ifOperStatus` is equal to `up (1)`, PROBE reports that the probed interface is active. Otherwise, probe reports that the probed interface is inactive.

If the probed interface resides on a node that is directly connected to the probed node, PROBE reports that the interface is up if it appears in the IPv4 Address Resolution Protocol (ARP) table or the

IPv6 Neighbor Cache. Otherwise, it reports that the interface does not exist.

1.1. Terminology

This document uses the following terms:

- o Probing node - The node upon which PROBE executes
- o Probing interface - The interface from which an ICMP Extended Echo originates
- o Proxy interface - The interface to which the ICMP Extended Echo message is sent
- o Probed node - The node upon which the proxy interface resides
- o Probed interface - The interface whose status is being queried

1.2. Requirements Language

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. ICMP Extended Echo Request

The ICMP Extended Echo Request message is defined for both ICMPv4 and ICMPv6. Like any ICMP message, the ICMP Extended Echo Request message is encapsulated in an IP header. The ICMPv4 version of the Extended Echo Request message is encapsulated in an IPv4 header, while the ICMPv6 version is encapsulated in an IPv6 header.

Figure 1 depicts the ICMP Extended Echo Request message.

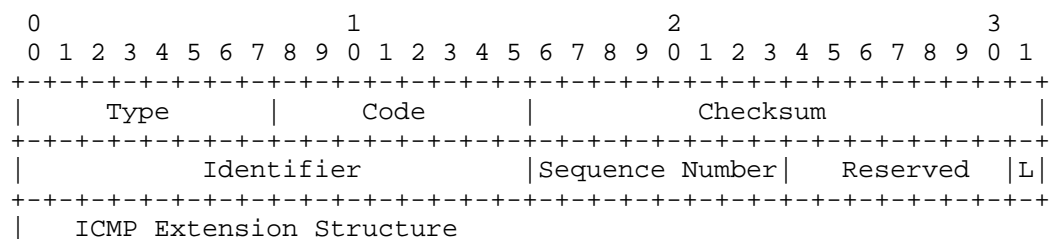


Figure 1: ICMP Extended Echo Request Message

IP Header fields:

- o Source Address: The Source Address identifies the probing interface. It MUST be valid IPv4 or IPv6 unicast address.
- o Destination Address: The Destination Address identifies the proxy interface. It can be a unicast, multicast or anycast address.

ICMP fields:

- o Type: Extended Echo Request. The value for ICMPv4 is TBD by IANA. The value for ICMPv6 is also TBD by IANA.
- o Code: 0
- o Checksum: For ICMPv4, see RFC 792. For ICMPv6, see RFC 4443.
- o Identifier: An identifier to aid in matching Extended Echo Replies to Extended Echo Requests. May be zero.
- o Sequence Number: A sequence number to aid in matching Extended Echo Replies to Extended Echo Requests. May be zero.
- o Reserved: This field MUST be set to zero and ignored upon receipt.
- o L (local) - The L-bit is set if the probed interface resides on the probed node. The L-bit is clear if the probed interface is directly connected to the probed node.
- o ICMP Extension Structure: The ICMP Extension Structure identifies the probed interface.

Section 7 of [RFC4884] defines the ICMP Extension Structure. As per RFC 4884, the Extension Structure contains exactly one Extension Header followed by one or more objects. When applied to the ICMP Extended Echo Request message, the ICMP Extension Structure MUST contain one or two instances of the Interface Identification Object (Section 2.1).

In most cases, a single instance of the Interface Identification Object identifies the probed interface. However, in some cases, a second instance is required for disambiguation.

If the L-bit is set, the Interface Identification Object identifies the probed interface by name, index or address. If the L-bit is clear, the Interface Identification Object identifies the probed interface by address.

If the Interface Identification Object identifies the probed interface by address, that address can be a member of any address family. For example, an ICMPv4 Extended Echo Request message can carry an Interface Identification Object that identifies the probed interface by IPv4, IPv6 or IEEE 802 address. Likewise, an ICMPv6 Extended Echo Request message can carry an Interface Identification Object that identifies the probed interface by IPv4, IPv6 or IEEE 802 address.

2.1. Interface Identification Object

The Interface Identification Object identifies the probed interface by name, index, or address. Like any other ICMP Extension Object, it contains an Object Header and Object Payload. The Object Header contains the following fields:

- o Class-Num: Interface Identification Object. Value is TBD by IANA
- o C-type: Values are: (1) Identifies Interface By Name, (2) Identifies Interface By Index, and (3) Identifies Interface By Address
- o Length: Length of the object, measured in octets, including the object header and object payload.

If the Interface Identification Object identifies the probed interface by name, the object payload contains the human-readable interface name. The interface name SHOULD be the full MIB-II ifName, if less than 255 octets, or the first 255 octets of the ifName, if the ifName is longer. The interface name MAY be some other human-meaningful name of the interface. The interface name MUST be represented in the UTF-8 charset [RFC3629] using the Default Language [RFC2277].

If the Interface Identification Object identifies the probed interface by index, the length is equal to 8 and the payload contains the MIB-II ifIndex [RFC 2863].

If the Interface Identification Object identifies the probed interface by address, the payload is as depicted in Figure 2.

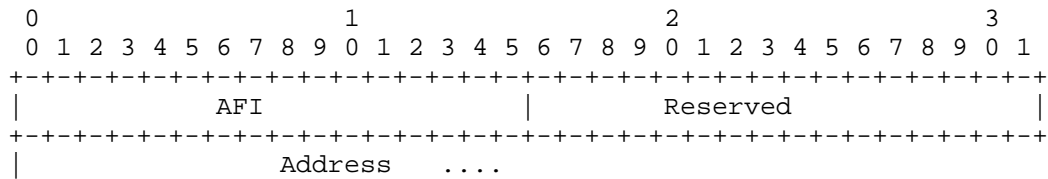


Figure 2: Interface Identification Object - C-type 3 Payload

Payload fields are defined as follows:

- o Address Family Identifier (AFI): This 16-bit field identifies the type of address represented by the Address field. All values found in the IANA registry of Address Family Numbers (available from <<https://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml>>) are valid in this field.
- o Reserved: This field MUST be set to zero and ignored upon receipt.
- o Address: This variable-length field represents an address associated with the probed interface.

3. ICMP Extended Echo Reply

The ICMP Extended Echo Reply message is defined for both ICMPv4 and ICMPv6. Like any ICMP message, the ICMP Extended Echo Reply message is encapsulated in an IP header. The ICMPv4 version of the Extended Echo Reply message is encapsulated in an IPv4 header, while the ICMPv6 version is encapsulated in an IPv6 header.

Figure 3 depicts the ICMP Extended Echo Reply message.

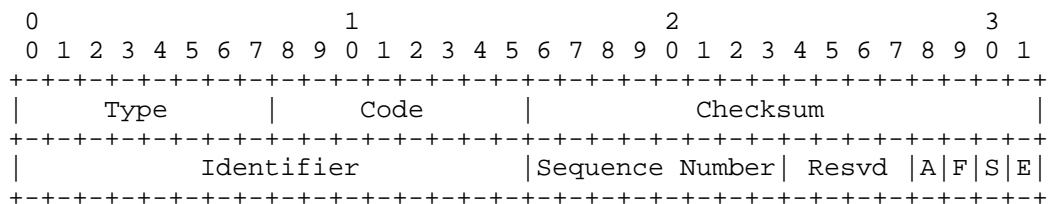


Figure 3: ICMP Extended Echo Reply Message

IP Header fields:

- o Source address: Copied from the Destination Address field of the invoking Extended Echo Request message
- o Destination address: Copied from the Source Address field of the invoking Extended Echo Request message

ICMP fields:

- o Type: Extended Echo Reply. The value for ICMPv4 is TBD by IANA. The value for ICMPv6 is also TBD by IANA
- o Code: (0) No Error, (1) Malformed Query, (2) No Such Interface, (3) Multiple Interfaces Satisfy Query
- o Checksum: For ICMPv4, see RFC 792. For ICMPv6, see RFC 4443
- o Identifier: Copied from the Identifier field of the invoking Extended Echo Request packet
- o Sequence Number: Copied from the Sequence Number field of the invoking Extended Echo Request packet
- o Resvd - This field MUST be set to zero and ignored upon receipt
- o A (Active) - The A-bit is set if Code is equal to zero and the probed interface is active. Otherwise, the A-bit is clear.
- o F (IPv4) - The F-bit is set if the A-bit is also set and IPv4 is running on the probed interface. Otherwise, the F-bit is clear.
- o S (IPv6) - The S-bit is set if the A-bit is also set and IPv6 is running on the probed interface. Otherwise, the S-bit is clear.
- o E (Ethernet) - The E-bit is set if the A-bit is also set and IPv4 is running on the probed interface. Otherwise, the E-bit is clear.

4. ICMP Message Processing

When a node receives an ICMP Extended Echo Request message and any of the following conditions apply, the node MUST silently discard the incoming message:

- o The node does not recognize ICMP Extended Echo Request messages

- o The node has not explicitly enabled ICMP Extended Echo functionality
- o The incoming ICMP Extend Echo Request carries a source address that is not explicitly authorized for the incoming ICMP Extended Echo Request L-bit setting
- o The incoming ICMP Extend Echo Request carries a source address that is not explicitly authorized for the incoming ICMP Extended Echo Request type (i.e., by ifName, by IfIndex, by Address)
- o The Source Address of the incoming messages is not a unicast address

Otherwise, when a node receives an ICMPv4 Extended Echo Request, it MUST format an ICMP Extended Echo Reply as follows:

- o Don't Fragment flag (DF) is 1
- o More Fragments flag is 0
- o Fragment Offset is 0
- o TTL is 255
- o Protocol is ICMP

When a node receives an ICMPv6 Extended Echo Request, it MUST format an ICMPv6 Extended Echo Reply as follows:

- o Hop Limit is 255
- o Next Header is ICMPv6

In either case, the responding node MUST:

- o Copy the source address from the Extended Echo Request message to the destination address of the Extended Echo Reply
- o Copy the destination address from the Extended Echo Request message to the source address of the Extended Echo Reply
- o Set the DiffServ codepoint to CS0 [RFC4594]
- o Set the ICMP Type to Extended Echo Reply
- o Copy the Identifier from the Extended Echo Request message to the Extended Echo Reply

- o Copy the Sequence Number from the Extended Echo Request message to the Extended Echo Reply
- o Set the Code field as described Section 4.1
- o If the Code Field is equal to No Error (0) and the L-bit is clear, set the A-Bit.
- o If the Code Field is equal to No Error (0) and the L-bit is set and the probed interface is active, set the A-bit.
- o If the A-bit is set, set the F-bit, S-bit and E-bit as appropriate. Otherwise, clear the F, S and E bits.
- o Set the checksum appropriately
- o Forward the ICMP Extended Echo Reply to its destination

4.1. Code Field Processing

The Code field MUST be set to Malformed Query (1) if any of the following conditions apply:

- o The ICMP Extended Echo Request does not include an ICMP Extension Structure
- o The ICMP Extension Structure does not include an Interface Identification Object
- o The ICMP Extension Structure contains more than two Interface Identification Objects
- o The L-bit is clear and the Interface Identification Object identifies the probed interface by ifName or ifIndex
- o The query is otherwise malformed

The Code field MUST be set to No Such Interface (2) if any of the following conditions apply:

- o The L-bit is set and the ICMP Extension Structure does not identify any local interfaces
- o The L-bit is clear and the address or addresses found in the Interface Identification object appear in neither the IPv4 Address Resolution Protocol (ARP) Table nor the IPv6 Neighbor Cache

The Code field MUST be set to Multiple Interfaces Satisfy Query (3) if any of the following conditions apply:

- o The L-bit is set and the ICMP Extension Structure identifies more than one local interfaces
- o The L-bit is clear and the address or addresses found in the Interface Identification object map to multiple IPv4 ARP or IPv6 Neighbor Cache entries

Otherwise, the Code field MUST be set to No Error (0)

5. Use-Cases

In the scenarios listed below, network operators can use PROBE to determine the status of a probed interface, but cannot use PING for the same purpose. In all scenarios, assume bidirectional connectivity between the probing and proxy interfaces. However, bidirectional connectivity between the probing and probed interfaces is lacking.

- o The probed interface is unnumbered
- o The probing and probed interfaces are not directly connected to one another. The probed interface has an IPv6 link-local address, but does not have a more globally scoped address
- o The probing interface runs IPv4 only while the probed interface runs IPv6 only
- o The probing interface runs IPv6 only while the probed interface runs IPv4 only
- o For lack of a route, the probing node cannot reach the probed interface.

6. Updates to RFC 4884

Section 4.6 of RFC 4884 provides a list of extensible ICMP messages (i.e., messages that can carry the ICMP Extension Structure). This document adds the ICMP Extended Echo message and the ICMP Extended Echo Reply message to that list.

7. IANA Considerations

This document requests the following actions from IANA:

- o Add an entry to the "ICMP Type Number" registry, representing the Extended Echo Request. This entry has one code (0).
- o Add an entry to the "Internet Control Message Protocol version 6 (ICMPv6) Parameters" registry, representing the Extended Echo Request. This entry has one code (0).
- o Add an entry to the "ICMP Type Number" registry, representing the Extended Echo Reply. This entry has the following codes: (0) No Error, (1) Malformed Query, (2) No Such Interface, (3) Multiple Interfaces Satisfy Query. Protocol Flag Bit mappings are as follows: Bit 0 (IPv4), Bit 1 (IPv6), Bit 2 (Ethernet), Bits 3-15 (Reserved).
- o Add an entry to the "Internet Control Message Protocol version 6 (ICMPv6) Parameters" registry, representing the Extended Echo Reply. This entry has the following codes: (0) No Error, (1) Malformed Query, (2) No Such Interface, (3) Multiple Interfaces Satisfy Query. Protocol Flag Bit mappings are as follows: Bit 0 (IPv4), Bit 1 (IPv6), Bit 2 (Ethernet), Bits 3-15 (Reserved).
- o Add an entry to the "ICMP Extension Object Classes and Class Subtypes" registry, representing the Interface Identification Object. It has C-types Reserved (0), Identifies Interface By Name (1), Identifies Interface By Index (2), Identifies Interface By Address (3)

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

The following are legitimate uses of PROBE:

- o to determine the operational status of an interface
- o to determine which protocols (e.g., IPv4, IPv6) are active on an interface

However, malicious parties can use PROBE to obtain additional information. For example, a malicious party can use PROBE to discover interface names. Having discovered an interface name, the malicious party may be able to infer additional information. Additional information may include:

- o interface bandwidth

- o the type of device that supports the interface (e.g., vendor identity)
- o the operating system version that the above-mentioned device executes

Understanding this risk, network operators establish policies that restrict access to ICMP Extended Echo functionality. In order to enforce these policies, nodes that support ICMP Extended Echo functionality MUST support the following configuration options:

- o Enable/disable ICMP Extended Echo functionality. By default, ICMP Extend Echo functionality is disabled.
- o Define enabled L-bit settings. By default, L-bit set is enabled and L-bit clear is disabled.
- o Define enabled query types (i.e., by ifName, by ifIndex, by Address). By default, all query types are disabled.
- o For each enabled query type, define the prefixes from which ICMP Extended Echo Request messages are permitted
- o For each interface, determine whether ICMP Echo Request messages are accepted

When a node receives an ICMP Extended Echo Request message that it is not configured to support, it MUST silently discard the message. See Section 4 for details.

In order to protect local resources, implementations SHOULD rate-limit incoming ICMP Extended Echo Request messages.

9. References

9.1. Normative References

- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <<http://www.rfc-editor.org/info/rfc792>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

- [RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, DOI 10.17487/RFC2277, January 1998, <<http://www.rfc-editor.org/info/rfc2277>>.
- [RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000, <<http://www.rfc-editor.org/info/rfc2863>>.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <<http://www.rfc-editor.org/info/rfc3629>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <<http://www.rfc-editor.org/info/rfc4443>>.
- [RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, DOI 10.17487/RFC4884, April 2007, <<http://www.rfc-editor.org/info/rfc4884>>.

9.2. Informative References

- [RFC2151] Kessler, G. and S. Shepard, "A Primer On Internet and TCP/IP Tools and Utilities", FYI 30, RFC 2151, DOI 10.17487/RFC2151, June 1997, <<http://www.rfc-editor.org/info/rfc2151>>.
- [RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, DOI 10.17487/RFC4594, August 2006, <<http://www.rfc-editor.org/info/rfc4594>>.

Appendix A. The PROBE Application

The PROBE application accepts input parameters, sets a counter and enters a loop to be exited when the counter is equal to zero. On each iteration of the loop, PROBE emits an ICMP Extended Echo Request, decrements the counter, sets a timer, waits for the timer to expire. If an expected ICMP Extended Echo Reply arrives while PROBE is waiting for the timer to expire, PROBE relays information returned by that message to its user. However, on each iteration of the loop, PROBE waits for the timer to expire, regardless of whether an Extended Echo Reply message arrives.

PROBE accepts the following parameters:

- o Count
- o Wait
- o Probing Interface Address
- o Hop Count
- o Proxy Interface Address
- o Local
- o Probed Interface Identifier

Count is a positive integer whose default value is 3. Count determines the number of times that PROBE iterates through the above-mentioned loop.

Wait is a positive integer whose minimum and default values are 1. Wait determines the duration of the above-mentioned timer, measured in seconds.

Probing Interface Address specifies the source address of ICMP Extended Echo Request. The Probing Interface Address MUST be a unicast address and MUST identify an interface that is local to the probing node.

The Proxy Interface Address identifies the interface to which the ICMP Extended Echo Request message is sent. It can be an IPv4 or IPv6 address. If it is an IPv4 address, PROBE emits an ICMPv4 message. If it is an IPv6 address, PROBE emits an ICMPv6 message.

Local is a boolean value. It is TRUE if the proxy and probed interfaces both reside on the probed node. It is FALSE if the proxy interface resides on the probed node and the probed interface is directly connected to the probed node.

The probed interface is the interface whose status is being queried. It is identified by one of the following:

- o an interface name
- o an address from any address family (e.g., IPv4, IPv6, IEEE 802, 48-bit MAC, 64-bit MAC)
- o an ifIndex

If the probed interface identifier is an address, it does not need to be of the same address family as the proxy interface address. For example, PROBE accepts an IPv4 destination interface address and an IPv6 probed interface identifier

Acknowledgments

Thanks to Jeff Haas, Carlos Pignataro, Jonathan Looney and Joe Touch for their thoughtful review of this document.

Authors' Addresses

Ron Bonica
Juniper Networks
2251 Corporate Park Drive
Herndon, Virginia 20171
USA

Email: rbonica@juniper.net

Reji Thomas
Juniper Networks
Elnath-Exora Business Park Survey
Bangalore, Karnataka 560103
India

Email: rejithomas@juniper.net

Jen Linkova
Google
1600 Amphitheatre Parkway
Mountain View, California 94043
USA

Email: furry@google.com

Chris Lenart
Verizon
22001 Loudoun County Parkway
Ashburn, Virginia 20147
USA

Email: chris.lenart@verizon.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Transport Area Working Group
Internet-Draft
Updates: 6040, 2661, 2784, 3931, 4380
(if approved)
Intended status: Standards Track
Expires: January 4, 2018

B. Briscoe
Simula Research Laboratory
July 3, 2017

Propagating Explicit Congestion Notification Across IP Tunnel Headers
Separated by a Shim
draft-ietf-tsvwg-rfc6040update-shim-04

Abstract

RFC 6040 on "Tunnelling of Explicit Congestion Notification" made the rules for propagation of ECN consistent for all forms of IP in IP tunnel. This specification updates RFC 6040 to clarify that its scope includes tunnels where two IP headers are separated by at least one shim header that is not sufficient on its own for wide area packet forwarding. It surveys widely deployed IP tunnelling protocols separated by such shim header(s) and updates the specifications of those that do not mention ECN propagation (L2TPv2, L2TPv3, GRE and Teredo). This specification also updates RFC 6040 with configuration requirements needed to make any legacy tunnel ingress safe.

Status of This Memo

This Internet-Draft is submitted 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 January 4, 2018.

Copyright Notice

Copyright (c) 2017 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. Introduction	2
2. Terminology	3
3. Scope of RFC 6040	3
3.1. Feasibility of ECN Propagation between Tunnel Headers . .	4
3.2. Desirability of ECN Propagation between Tunnel Headers .	5
4. Making a non-ECN Tunnel Ingress Safe by Configuration	5
5. IP-in-IP Tunnels with Tightly Coupled Shim Headers	6
5.1. Specific Updates to Protocols under IETF Change Control .	8
5.1.1. L2TP (v2 and v3) ECN Extension	8
5.1.2. GRE	11
5.1.3. Teredo	12
6. IANA Considerations	12
7. Security Considerations	13
8. Comments Solicited	13
9. Acknowledgements	13
10. References	13
10.1. Normative References	13
10.2. Informative References	14
Author's Address	16

1. Introduction

RFC 6040 on "Tunnelling of Explicit Congestion Notification" [RFC6040] made the rules for propagation of Explicit Congestion Notification (ECN [RFC3168]) consistent for all forms of IP in IP tunnel.

A common pattern for many tunnelling protocols is to encapsulate an inner IP header (v4 or v6) with shim header(s) then an outer IP header (v4 or v6). Some of these shim headers are designed as generic encapsulations, so they do not necessarily directly encapsulate an inner IP header. Instead they can encapsulate headers such as link-layer (L2) protocols that in turn often encapsulate IP.

To clear up confusion, this specification clarifies that the scope of RFC 6040 includes any IP-in-IP tunnel, including those with shim

header(s) and other encapsulations between the IP headers. Where necessary, it updates the specifications of the relevant encapsulation protocols with the specific text necessary to comply with RFC 6040.

This specification also updates RFC 6040 to state how operators ought to configure a legacy tunnel ingress to avoid unsafe system configurations.

2. Terminology

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] when, and only when, they appear in all capitals, as shown here.

This specification uses the terminology defined in RFC 6040 [RFC6040].

3. Scope of RFC 6040

In section 1.1 of RFC 6040, its scope is defined as:

"...ECN field processing at encapsulation and decapsulation for any IP-in-IP tunnelling, whether IPsec or non-IPsec tunnels. It applies irrespective of whether IPv4 or IPv6 is used for either the inner or outer headers. ..."

This was intended to include cases where shim header(s) sit between the IP headers. Many tunnelling implementers have interpreted the scope of RFC 6040 as it was intended, but it is ambiguous. Therefore, this specification updates RFC 6040 by adding the following scoping text after the sentences quoted above:

It applies in cases where an outer IP header encapsulates an inner IP header either directly or indirectly by encapsulating other headers that in turn encapsulate (or might encapsulate) an inner IP header.

There is another problem with the scope of RFC 6040. Like many IETF specifications, RFC 6040 is written as a specification that implementations can choose to claim compliance with. This means it does not cover two important cases:

1. those cases where it is infeasible for an implementation to access an inner IP header when adding or removing an outer IP header;

2. those implementations that choose not to propagate ECN between IP headers.

However, the ECN field is a non-optional part of the IP header (v4 and v6). So any implementation that creates an outer IP header has to give the ECN field some value. There is only one safe value a tunnel ingress can use if it does not know whether the egress supports propagation of the ECN field; it has to zero the ECN field in any outer IP header.

However, an RFC has no jurisdiction over implementations that choose not to comply with it or cannot comply with it, including all those implementations that pre-dated the RFC. Therefore it would have been unreasonable to add such a requirement to RFC 6040. Nonetheless, to ensure safe propagation of the ECN field over tunnels, it is reasonable to add requirements on operators, to ensure they configure their tunnels safely (where possible). Before stating these configuration requirements in Section 4, the factors that determine whether propagating ECN is feasible or desirable will be briefly introduced.

3.1. Feasibility of ECN Propagation between Tunnel Headers

In many cases shim header(s) and an outer IP header are always added to (or removed from) an inner IP packet as part of the same procedure. We call this a tightly coupled shim header. Processing the shim and outer together is often necessary because the shim(s) are not sufficient for packet forwarding in their own right; not unless complemented by an outer header. In these cases it will often be feasible for an implementation to propagate the ECN field between the IP headers.

In some cases a tunnel adds an outer IP header and a tightly coupled shim header to an inner header that is not an IP header, but that in turn encapsulates an IP header (or might encapsulate an IP header). For instance an inner Ethernet (or other link layer) header might encapsulate an inner IP header as its payload. We call this a tightly coupled shim over an encapsulating header.

Digging to arbitrary depths to find an inner IP header within an encapsulation is strictly a layering violation so it cannot be a required behaviour. Nonetheless, some tunnel endpoints already look within a L2 header for an IP header, for instance to map the Diffserv codepoint between an encapsulated IP header and an outer IP header [RFC2983]. In such cases at least, it should be feasible to also (independently) propagate the ECN field between the same IP headers. Thus, access to the ECN field within an encapsulating header can be a useful and benign optimization. The guidelines in section 6 of

[I-D.ietf-tsvwg-ecn-encap-guidelines] give the conditions for this layering violation to be benign.

3.2. Desirability of ECN Propagation between Tunnel Headers

Developers and network operators are encouraged to implement and deploy tunnel endpoints compliant with RFC 6040 (as updated by the present specification) in order to provide the benefits of wider ECN deployment [RFC8087]. Nonetheless, propagation of ECN between IP headers, whether separated by shim headers or not, has to be optional to implement and to use, because:

- o Legacy implementations of tunnels without any ECN support already exist
- o A network might be designed so that there is usually no bottleneck within the tunnel
- o If the tunnel endpoints would have to search within an L2 header to find an encapsulated IP header, it might not be worth the potential performance hit

4. Making a non-ECN Tunnel Ingress Safe by Configuration

Even when ECN propagation is not implemented or is not being used, it ought to be possible to render a tunnel ingress safe by configuration. The main safety concern is to disable the ECN capability in the outer IP header if the egress of the tunnel does not implement ECN logic to propagate any ECN markings into the packet forwarded beyond the tunnel. Otherwise the non-ECN egress could discard any ECN marking introduced within the tunnel, which would break all the ECN-based control loops that regulate the traffic load over the tunnel.

Therefore this specification updates RFC 6040 by inserting the following text just before the last paragraph of section 4.3:

When the implementation of a tunnel ingress does not support [RFC6040] or one of its compatible predecessors ([RFC4301] or the full functionality mode of [RFC3168]) and when the outer tunnel header is IP (v4 or v6), if possible, the operator MUST configure the ingress to zero the outer ECN field in any of the following cases:

- * if it is known that the tunnel egress does not support propagation of the ECN field (RFC 6040, RFC 4301 or the full functionality mode of RFC 3168)

- * or if the behaviour of the egress is not known or an egress with unknown behaviour might be dynamically paired with the ingress.
- * or if an IP header might be encapsulated within a non-IP header that the tunnel ingress is encapsulating, but the ingress does not inspect within the encapsulation.

In order that the network operator can comply with the above safety rules, even if a tunnel ingress does not support RFC 6040, RFC 4301 or the full functionality mode of RFC 3168, the implementation of the tunnel ingress:

- o MUST make propagation of the ECN field between inner and outer IP headers independent of any configuration of Diffserv codepoint propagation;
- o SHOULD zero the outer ECN field in its default configuration.

There might be concern that the above "MUST" makes compliant equipment non-compliant at a stroke. However, any equipment that is still treating the ToS octet (IPv4) or the Traffic Class octet (IPv6) as a single 8-bit field is already non-compliant, and has been since 1998 when the upper 6 bits were separated off for the Diffserv codepoint (DSCP) [RFC2474]. For instance, copying the ECN field as a side-effect of copying the DSCP is a seriously unsafe bug that risks breaking the feedback loops that regulate load on a tunnel.

Permanently zeroing the outer ECN field is safe, but it is not sufficient to claim compliance with RFC 6040 because it does not meet the aim of introducing ECN support to tunnels (see Section 4.3 of [RFC6040]).

5. IP-in-IP Tunnels with Tightly Coupled Shim Headers

There follows a list of specifications of encapsulations with tightly coupled shim header(s), in rough chronological order. The list is confined to standards track or widely deployed protocols. The list is not necessarily exhaustive so, for the avoidance of doubt, the scope of RFC 6040 is defined in Section 3 and is not limited to this list.

- o PPTP (Point-to-Point Tunneling Protocol) [RFC2637];
- o L2TP (Layer 2 Tunnelling Protocol), specifically L2TPv2 [RFC2661] and L2TPv3 [RFC3931], which not only includes all the L2-specific specializations of L2TP, but also derivatives such as the Keyed IPv6 Tunnel [RFC8159];

- o GRE (Generic Routing Encapsulation) [RFC2784] and NVGRE (Network Virtualization using GRE) [RFC7637];
- o GTP (GPRS Tunnelling Protocol), specifically GTPv1 [GTPv1], GTP v1 User Plane [GTPv1-U], GTP v2 Control Plane [GTPv2-C];
- o Teredo [RFC4380];
- o CAPWAP (Control And Provisioning of Wireless Access Points) [RFC5415];
- o LISP (Locator/Identifier Separation Protocol) [RFC6830];
- o VXLAN (Virtual eXtensible Local Area Network) [RFC7348] and VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe];
- o SFC (Service Function Chaining) [RFC7665];
- o Geneve [I-D.ietf-nvo3-geneve];
- o GUE (Generic UDP Encapsulation) [I-D.ietf-intarea-gue].

Some of the listed protocols enable encapsulation of a variety of network layer protocols as inner and/or outer. This specification applies in the cases where there is an inner and outer IP header as described in Section 3. Otherwise [I-D.ietf-tsvwg-ecn-encap-guidelines] gives guidance on how to design propagation of ECN into other protocols that might encapsulate IP.

Where protocols in the above list need to be updated to specify ECN propagation and they are under IETF change control, update text is given in the following subsections. For those not under IETF control, it is RECOMMENDED that implementations of encapsulation and decapsulation comply with RFC 6040. It is also RECOMMENDED that their specifications are updated to add a requirement to comply with RFC 6040 (as updated by the present document).

PPTP is not under the change control of the IETF, but it has been documented in an informational RFC [RFC2637]. However, there is no need for the present specification to update PPTP because L2TP has been developed as a standardized replacement.

NVGRE is not under the change control of the IETF, but it has been documented in an informational RFC [RFC7637]. NVGRE is a specific use-case of GRE (it re-purposes the key field from the initial specification of GRE [RFC1701] as a Virtual Subnet ID). Therefore the text that updates GRE in Section 5.1.2 below is also intended to update NVGRE.

Although the definition of the various GTP shim headers is under the control of the 3GPP, it is hard to determine whether the 3GPP or the IETF controls standardization of the process of adding both a GTP and an IP header to an inner IP header. Nonetheless, the present specification is provided so that the 3GPP can refer to it from any of its own specifications of GTP and IP header processing.

The specification of CAPWAP already specifies RFC 3168 ECN propagation and ECN capability negotiation. Without modification the CAPWAP specification already interworks with the backward compatible updates to RFC 3168 in RFC 6040.

LISP made the ECN propagation procedures in RFC 3168 mandatory from the start. RFC 3168 has since been updated by RFC 6040, but the changes are backwards compatible so there is still no need for LISP tunnel endpoints to negotiate their ECN capabilities.

VXLAN is not under the change control of the IETF but it has been documented in an informational RFC. In contrast, VXLAN-GPE (Generic Protocol Extension) is being documented under IETF change control. It is RECOMMENDED that VXLAN and VXLAN-GPE implementations comply with RFC 6040 when the VXLAN header is inserted between (or removed from between) IP headers. The authors of any future update to these specifications are encouraged to add a requirement to comply with RFC 6040 as updated by the present specification.

Although the Service Function Chaining (SFC) architecture [RFC7665] depends on a shim-based encapsulation to identify the service function path (SFP), it does not specify the processing of ECN when handling transport encapsulation.

The specifications of Geneve and GUE already refer to RFC 6040 for ECN encapsulation.

5.1. Specific Updates to Protocols under IETF Change Control

5.1.1. L2TP (v2 and v3) ECN Extension

The L2TP terminology used here is defined in [RFC2661] and [RFC3931].

L2TPv3 [RFC3931] is used as a shim header between any packet-switched network (PSN) header (e.g. IPv4, IPv6, MPLS) and many types of layer 2 (L2) header. The L2TPv3 shim header encapsulates an L2-specific sub-layer then an L2 header that is likely to contain an inner IP header (v4 or v6). Then this whole stack of headers can be encapsulated optionally within an outer UDP header then an outer PSN header that is typically IP (v4 or v6).

L2TPv2 is used as a shim header between any PSN header and a PPP header, which is in turn likely to encapsulate an IP header.

Even though these shims are rather fat (particularly in the case of L2TPv3), they still fit the definition of a tightly coupled shim header over an encapsulating header (Section 3.1), because all the headers encapsulating the L2 header are added (or removed) together. L2TPv2 and L2TPv3 are therefore within the scope of RFC 6040, as updated by Section 3 above.

L2TP maintainers are RECOMMENDED to implement the ECN extension to L2TPv2 and L2TPv3 defined in Section 5.1.1.2 below, in order to provide the benefits of ECN [RFC8087], whenever a node within an L2TP tunnel becomes the bottleneck for an end-to-end traffic flow.

5.1.1.1. Safe Configuration of a 'Non-ECN' Ingress LCCE

The following text is appended to both Section 5.3 of [RFC2661] and Section 4.5 of [RFC3931] as an update to the base L2TPv2 and L2TPv3 specifications:

An LCCE that does not support the ECN Extension in Section 5.1.1.2 of RFCXXXX MUST follow the configuration requirements in Section 4 of RFCXXXX for when the outer PSN header is IP (v4 or v6).
{RFCXXXX refers to the present document so it will need to be inserted by the RFC Editor}

In particular, for an LCCE implementation that does not support the ECN Extension, this means that configuration of how it propagates the ECN field between inner and outer IP headers MUST be independent of any configuration of the Diffserv extension of L2TP [RFC3308].

5.1.1.2. ECN Extension for L2TP (v2 or v3)

When the outer PSN header and the payload inside the L2 header are both IP (v4 or v6), to comply with RFC 6040, an LCCE will follow the rules for propagation of the ECN field at ingress and egress in Section 4 of RFC 6040 [RFC6040].

Before encapsulating any data packets, RFC 6040 requires an ingress LCCE to check that the egress LCCE supports ECN propagation. If the egress supports ECN, the ingress LCCE can use the normal mode of encapsulation. Otherwise, the ingress LCCE has to use compatibility mode [RFC6040]. An LCCE can determine the remote LCCE's support for ECN either statically (by configuration) or by dynamic discovery during setup of each control connection between the LCCEs, using the Capability AVP defined in Section 5.1.1.2.1 below.

Where the outer PSN header is some protocol other than IP that supports ECN, the appropriate ECN propagation specification will need to be followed, e.g. "Explicit Congestion Marking in MPLS" [RFC5129]. Where no specification exists for ECN propagation by a particular PSN, [I-D.ietf-tsvwg-ecn-encap-guidelines] gives general guidance on how to design ECN propagation into a protocol that encapsulates IP.

5.1.1.2.1. LCCE Capability AVP for ECN Capability Negotiation

The LCCE Capability Attribute Value Pair (AVP) defined here has Attribute Type ZZ. The Attribute Value field for this AVP is a bit-mask with the following 16-bit format:

```

      0                               1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
  +-----+-----+-----+-----+
  |X X X X X X X X X X X X X X E|
  +-----+-----+-----+-----+

```

This AVP MAY be present in the following message types: SCCRQ and SCCRP (Start-Control-Connection-Request and Start-Control-Connection-Reply). This AVP MAY be hidden (the H-bit set to 0 or 1) and is optional (M-bit not set). The length (before hiding) of this AVP MUST be 8 octets. The Vendor ID is the IETF Vendor ID of 0.

Bit 15 of the Value field of the LCCE Capability AVP is defined as the ECN Capability flag (E). When the ECN Capability flag is set to 1, it indicates that the sender supports ECN propagation. When the ECN Capability flag is cleared to zero, or when no LCCE Capability AVP is present, it indicates that the sender does not support ECN propagation. All the other bits are reserved. They MUST be cleared to zero when sent and ignored when received or forwarded.

An LCCE initiating a control connection will send a Start-Control-Connection-Request (SCCRQ) containing an LCCE Capability AVP with the ECN Capability flag set to 1. If the tunnel terminator supports ECN, it will return a Start-Control-Connection-Reply (SCCRP) that also includes an LCCE Capability AVP with the ECN Capability flag set to 1. Then, for any sessions created by that control connection, both ends of the tunnel can use the normal mode of RFC 6040 to propagate the ECN field when encapsulating data packets.

If, on the other hand, the tunnel terminator does not support ECN it will ignore the ECN flag in the LCCE Capability AVP and send an SCCRP to the tunnel initiator without a Capability AVP (or with a Capability AVP but with the ECN Capability flag cleared to zero). The tunnel initiator interprets the absence of the ECN Capability

flag in the SCCRP as an indication that the tunnel terminator is incapable of supporting ECN. When encapsulating data packets for any sessions created by that control connection, the tunnel initiator will then use the compatibility mode of RFC 6040 to clear the ECN field of the outer IP header to 0b00.

If the tunnel terminator does not support this ECN extension, the network operator is still expected to configure it to comply with the safety provisions set out in Section 5.1.1.1 above, when it acts as an ingress LCCE.

5.1.2. GRE

The GRE terminology used here is defined in [RFC2784]. GRE is often used as a tightly coupled shim header between IP headers. Sometimes the GRE shim header encapsulates an L2 header, which might in turn encapsulate an IP header. Therefore GRE is within the scope of RFC 6040 as updated by Section 3 above.

GRE tunnel endpoint maintainers are RECOMMENDED to support [RFC6040] as updated by the present specification, in order to provide the benefits of ECN [RFC8087] whenever a node within a GRE tunnel becomes the bottleneck for an end-to-end IP traffic flow tunnelled over GRE using IP as the delivery protocol (outer header).

GRE tunnels do not support dynamic configuration based on capability negotiation, so the ECN capability has to be manually configured. For a GRE ingress implementation that supports ECN propagation, manual configuration requirements are specified in Section 4.3 of RFC 6040. Otherwise they are specified in Section 5.1.2.1 below.

Where the delivery protocol is some protocol other than IP that supports ECN, the appropriate ECN propagation specification will need to be followed, e.g Explicit Congestion Marking in MPLS [RFC5129]. Where no specification exists for ECN propagation by a particular PSN, [I-D.ietf-tsvwg-ecn-encap-guidelines] gives more general guidance on how to propagate ECN to and from protocols that encapsulate IP.

5.1.2.1. Safe Configuration of a 'Non-ECN' GRE Ingress

The following text is appended to Section 3 of [RFC2784] as an update to the base GRE specification:

A GRE tunnel ingress that does not support RFC 6040 or one of its compatible predecessors (RFC 4301 or the full functionality mode of RFC 3168) MUST follow the configuration requirements in Section 4 of RFCXXXX for when the outer delivery protocol is IP

(v4 or v6). {RFCXXXX refers to the present document so it will need to be inserted by the RFC Editor}

5.1.3. Teredo

Teredo [RFC4380] provides a way to tunnel IPv6 over an IPv4 network, with a UDP-based shim header between the two.

For Teredo tunnel endpoints to provide the benefits of ECN, the Teredo specification would have to be updated to include negotiation of the ECN capability between Teredo tunnel endpoints. Otherwise it would be unsafe for a Teredo tunnel ingress to copy the ECN field to the IPv6 outer.

It is believed that current implementations do not support propagation of ECN, but that they do safely zero the ECN field in the outer IPv6 header. However the specification does not mention anything about this. To make existing Teredo deployments safe it will not be feasible to require them to be configured correctly, because Teredo tunnel endpoints are generally deployed on hosts. Therefore, the only feasible safety precaution available here is to update the specification of Teredo implementations until ECN support is added. The following text updates the Teredo specification [RFC4380], as a new section 5.1.3:

"5.1.3 Safe 'Non-ECN' Teredo Encapsulation

A Teredo tunnel ingress implementation that does not support ECN propagation as defined in RFC 6040 or one of its compatible predecessors (RFC 4301 or the full functionality mode of RFC 3168) MUST zero the ECN field in the outer IPv6 header."

6. IANA Considerations

IANA is requested to assign the following L2TP Control Message Attribute Value Pair:

Attribute Type	Description	Reference
ZZ	ECN Capability	RFCXXXX

[TO BE REMOVED: This registration should take place at the following location: <https://www.iana.org/assignments/l2tp-parameters/l2tp-parameters.xhtml>]

7. Security Considerations

The Security Considerations in [RFC6040] and [I-D.ietf-tsvwg-ecn-encap-guidelines] apply equally to the scope defined for the present specification.

8. Comments Solicited

Comments and questions are encouraged and very welcome. They can be addressed to the IETF Transport Area working group mailing list <tsvwg@ietf.org>, and/or to the authors.

9. Acknowledgements

Thanks to Ing-jyh (Inton) Tsang for initial discussions on the need for ECN propagation in L2TP and its applicability. Thanks also to Carlos Pignataro, Tom Herbert, Ignacio Goyret, Alia Atlas, Praveen Balasubramanian, Joe Touch and Mohamed Boucadair for helpful advice and comments.

10. References

10.1. Normative References

- [I-D.ietf-tsvwg-ecn-encap-guidelines]
Briscoe, B., Kaippallimalil, J., and P. Thaler,
"Guidelines for Adding Congestion Notification to
Protocols that Encapsulate IP", draft-ietf-tsvwg-ecn-
encap-guidelines-08 (work in progress), March 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black,
"Definition of the Differentiated Services Field (DS
Field) in the IPv4 and IPv6 Headers", RFC 2474,
DOI 10.17487/RFC2474, December 1998,
<<http://www.rfc-editor.org/info/rfc2474>>.
- [RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn,
G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"",
RFC 2661, DOI 10.17487/RFC2661, August 1999,
<<http://www.rfc-editor.org/info/rfc2661>>.

- [RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <<http://www.rfc-editor.org/info/rfc2784>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<http://www.rfc-editor.org/info/rfc3168>>.
- [RFC3931] Lau, J., Ed., Townsley, M., Ed., and I. Goyret, Ed., "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, DOI 10.17487/RFC3931, March 2005, <<http://www.rfc-editor.org/info/rfc3931>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<http://www.rfc-editor.org/info/rfc4301>>.
- [RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <<http://www.rfc-editor.org/info/rfc4380>>.
- [RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <<http://www.rfc-editor.org/info/rfc5129>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<http://www.rfc-editor.org/info/rfc6040>>.

10.2. Informative References

- [GTPv1] 3GPP, "GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface", Technical Specification TS 29.060.
- [GTPv1-U] 3GPP, "General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", Technical Specification TS 29.281.
- [GTPv2-C] 3GPP, "Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C)", Technical Specification TS 29.274.

- [I-D.ietf-intarea-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-intarea-gue-04 (work in progress), May 2017.
- [I-D.ietf-nvo3-geneve]
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-04 (work in progress), March 2017.
- [I-D.ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-04 (work in progress), April 2017.
- [RFC1701] Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 1701, DOI 10.17487/RFC1701, October 1994, <<http://www.rfc-editor.org/info/rfc1701>>.
- [RFC2637] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and G. Zorn, "Point-to-Point Tunneling Protocol (PPTP)", RFC 2637, DOI 10.17487/RFC2637, July 1999, <<http://www.rfc-editor.org/info/rfc2637>>.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <<http://www.rfc-editor.org/info/rfc2983>>.
- [RFC3308] Calhoun, P., Luo, W., McPherson, D., and K. Peirce, "Layer Two Tunneling Protocol (L2TP) Differentiated Services Extension", RFC 3308, DOI 10.17487/RFC3308, November 2002, <<http://www.rfc-editor.org/info/rfc3308>>.
- [RFC5415] Calhoun, P., Ed., Montemurro, M., Ed., and D. Stanley, Ed., "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification", RFC 5415, DOI 10.17487/RFC5415, March 2009, <<http://www.rfc-editor.org/info/rfc5415>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<http://www.rfc-editor.org/info/rfc6830>>.

- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<http://www.rfc-editor.org/info/rfc7348>>.
- [RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <<http://www.rfc-editor.org/info/rfc7637>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<http://www.rfc-editor.org/info/rfc7665>>.
- [RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <<http://www.rfc-editor.org/info/rfc8087>>.
- [RFC8159] Konstantynowicz, M., Ed., Heron, G., Ed., Schatzmayr, R., and W. Henderickx, "Keyed IPv6 Tunnel", RFC 8159, DOI 10.17487/RFC8159, May 2017, <<http://www.rfc-editor.org/info/rfc8159>>.

Author's Address

Bob Briscoe
Simula Research Laboratory
UK

EMail: ietf@bobbriscoe.net
URI: <http://bobbriscoe.net/>

Network Working Group
Internet Draft
Intended status: Experimental
Expires: December 2017

G. Lencse
NAIST
Sz. Szilagyi
F. Fejes
University of Debrecen
M. Georgescu
RCS&RDS
June 30, 2017

MPT Network Layer Multipath Library
draft-lencse-tsvwg-mpt-00.txt

Abstract

Although several contemporary IT devices have multiple network interfaces, communication sessions are restricted to use only one of them at a time due to the design of the TCP/IP protocol stack: the communication endpoint is identified by an IP address and a TCP or UDP port number. The simultaneous use of these multiple interfaces for a communication session would improve user experience through higher throughput and improved resilience to network failures.

MPT is a network layer multipath solution, which provides a tunnel over multiple paths using the GRE-in-UDP specification, thus being different from both MPTCP and Huawei's GRE Tunnel Bonding Protocol.

MPT can also be used as a router, routing the packets among several networks between the tunnel endpoints, thus establishing a multipath site-to-site connection.

The version of tunnel IP and the version of path IP are independent from each other, therefore MPT can also be used for IPv6 transition purposes.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and 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 December 30, 2017.

Copyright Notice

Copyright (c) 2017 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. Introduction	3
1.1. Design Assumptions	3
1.2. MPT in the Networking Stack	3
1.3. Terminology	4
1.4. MPT Concept	5
2. Conventions Used in this Document	6
3. Operation Overview	6
4. MPT Control	8
4.1. Configuration Information	8
4.1.1. General Information for the MPT Server	8
4.1.2. Connection Specifications	9
4.2. MPT Configuration Commands	13
5. Possible Mappings of the Tunnel Traffic to Paths	14
5.1. Per Packet Based Mapping	17
5.2. Flow Based Mapping	18
5.3. Combined Mapping	18
6. Packet Reordering	18

7. Why MPT is Considered Experimental?	19
7.1. Published Results	20
7.1.1. MPT Concept and First Implementation	20
7.1.2. Estimation of the Channel Aggregation Capabilities	20
7.1.3. Demonstrating the Resilience of an MPT Connection	20
7.2. Open questions	21
7.2.1. Parameters	21
7.2.2. Development of Further Mapping Algorithms.....	21
7.2.3. Performance Issues	21
8. Security Considerations	21
9. IANA Considerations	22
10. Conclusions	22
11. References	22
11.1. Normative References	22
11.2. Informative References	22
12. Acknowledgments	23
Appendix A. Sample C code for packet reordering	24

1. Introduction

MPT is a multipath extension of the GRE-in-UDP encapsulation [RFC8086].

1.1. Design Assumptions

MPT is intended to be used as a preconfigured tunnel and the application of MPT does not require any modifications to the applications using the TCP/IP socket interface API.

1.2. MPT in the Networking Stack

The layer architecture of MPT is shown in Fig. 1. MPT extends the GRE-in-UDP [RFC8086] architecture by allowing multiple physical paths. To that end it can be compared to MPTCP [RFC6824], but unlike MPTCP, MPT uses UDP in the underlying layer, builds on GRE-in-UDP, and provides a tunnel IP layer, over which both UDP and TCP can be used. The aim of Huawei's GRE tunnel bonding protocol [Lay2016] is also similar to that of MPT: it targets to bonded access to wired and wireless network in customer premises. However, it uses GRE (not GRE-in-UDP) which is less supported in ISP networks than UDP, and it seems to limit the number of physical interfaces to two. For the comparison of MPT with other multipath solutions, please refer to [Alm2017].

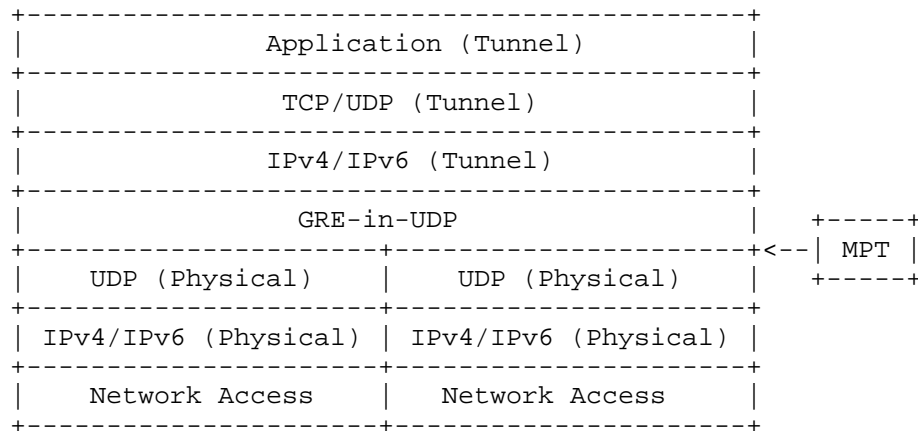


Figure 1: MPT Layer Architecture

1.3. Terminology

This document uses a number of terms that are either MPT specific or have defined meaning in the context of MPT as follows:

MPT server: An MPT server is a software that implements network layer multipath communication by providing an UDP tunnel (named "connection" in the MPT terminology) over several underlying "paths".

MPT client: An MPT client is a software tool, which is used to control the local MPT server (e.g. start/stop connections, add paths to connections, etc.).

Connection: An MPT connection (also referred as communication session) is an UDP tunnel between two MPT servers, which can be used to carry user data. A connection can be established over one or more paths. A connection is initiated on the basis of a "connection specification".

Path: A path is used to refer to the pair of the network cards of the end nodes (identified by the pair of IP addresses of the cards). Using a specified path, the packet transmission runs between the given pair of network cards.

Connection specification: A connection specification is stored in a configuration file and it is used by an MPT server to establish an MPT connection with another MPT server. It contains all the configuration information for the connection (e.g. endpoint IP

versions and addresses, number of paths and configuration information for all paths). The precise definition of the connection specification can be found in Section 4.1.2.

Data port: Data port means the GRE-in-UDP port defined in [RFC8086] as 4754. It is used for transmitting data packets.

Local command port: An MPT server accepts commands from the MPT client at the local command port.

Remote command port: An MPT server MAY accept commands from other MPT servers at the remote command port.

Data plane: The parts and functions of MPT, which are responsible for handling user data packets.

Control plane: All parts and functions of MPT except the data plane. E.g.: handling connections and paths, all the communication through local or remote command ports, etc.

1.4. MPT Concept

When an MPT server is started, it reads its configuration files, and depending on its contents, it MAY wait for and accept connection(s) initiated by other MPT server(s) and/or it MAY initiate one or more MPT connection(s) with other MPT server(s). In the simplest case, the MPT server uses the connection specifications described in its configuration files for initiating connections. In addition to that, new connections MAY be established, connections MAY be closed, the parameters of the connections MAY be changed later (e.g. some paths may be switched on and off) dynamically by issuing the appropriate commands using an MPT client.

MPT connections between MPT servers implement tunnels. The traffic comes from the tunnel interface is distributed over the active paths of the MPT connection by the MPT server. There are three possible mappings (see Section 5 for details and illustrative examples):

- o per packet based mapping, where a mapping decision is made for every single packet
- o flow based mapping, where the flows, identified by the usual five tuple, are always mapped to a given path.
- o combined mapping, where the flows, identified by the usual five tuple, are always mapped to a given connection and a mapping decision is made for every single packet of each connections.

The peer MPT server receives and de-encapsulates the traffic from the different paths and restores the tunnel traffic using the optional GRE sequence numbers for packet reordering if necessary.

2. Conventions Used in this Document

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].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying significance described in RFC 2119.

3. Operation Overview

In this Section, we describe the operation of the data plane, whereas the operation of the control plane can be found in Section 4.

The data packet transmission and receive mechanism of MPT is summarized in Fig. 2. Now, we shall follow the route and processing of data packets.

When a packet is read from the tunnel interface, the MPT software looks up the appropriate connection specification, which determines the mapping of the packets to the paths. The connection specification determines the behavior of the multipath communication, especially the distribution of the packets among the paths (see Section 5 for possible mapping methods). The path is selected and the user data packet encapsulated into a GRE-in-UDP data unit, which may optionally contain GRE Sequence Numbers for reordering. The simplest GRE header contains 4 octets: 16 bits of zeros and 16 bits of protocol type identification value (i.e. 0x86DD in the case of using IPv6 on the tunnel interface, or 0x0800 in the case of IPv4). Then the GRE-in-UDP data unit is encapsulated into the UDP/IP data unit of the selected path, where the destination UDP port number is the 4754 GRE-in-UDP port and the IP addresses (either both IPv4 or both IPv6) are determined by the path definition. Finally, the encapsulated packet is transmitted through the physical interface. The encapsulation of the different protocol data units is shown in Fig. 3.

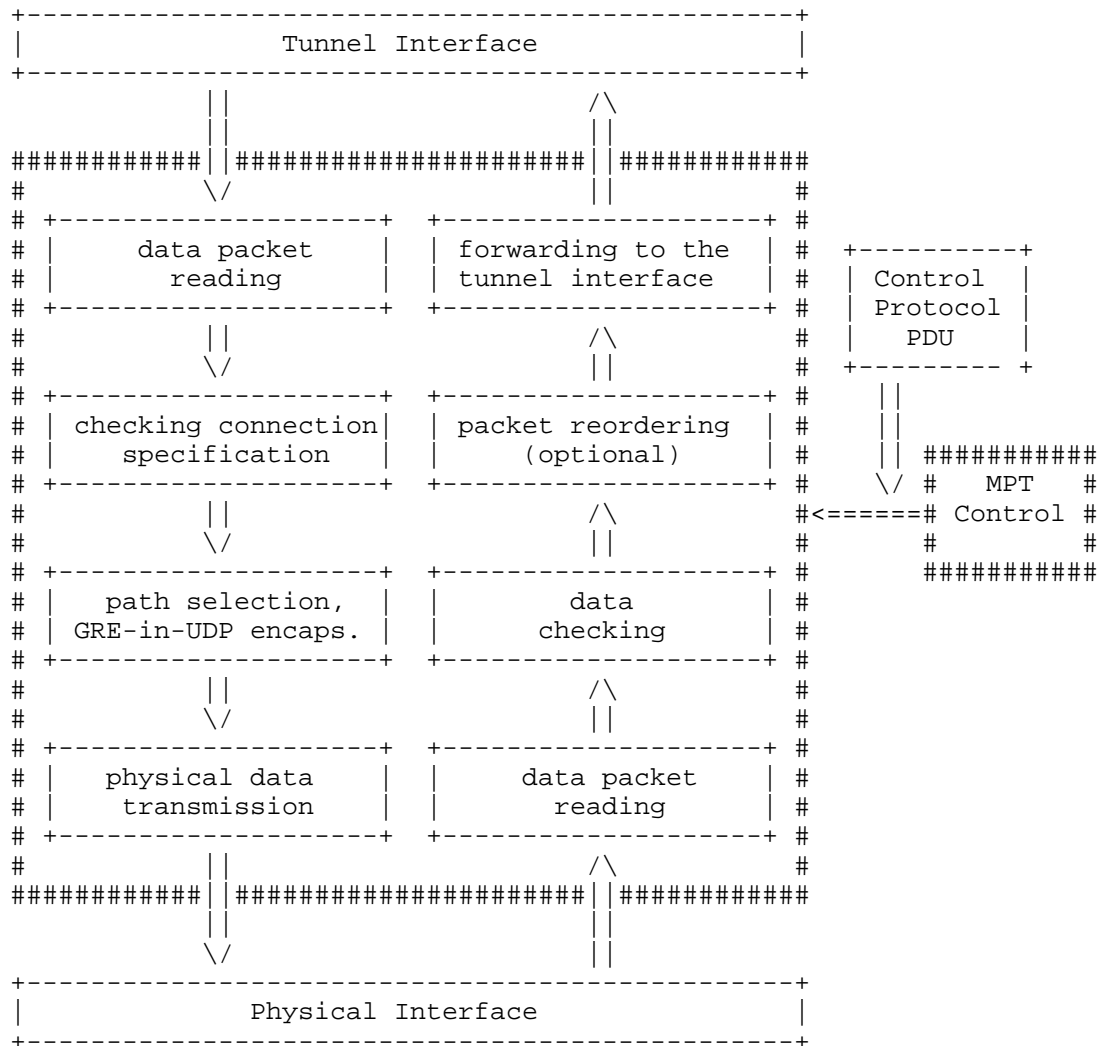


Figure 2: Conceptual architecture of MPT working mechanism

When a packet is read from the physical interface, its destination UDP port number is the 4754 GRE-in-UDP port. MPT reads the packet, identifies the connection the packet belongs to (by the source and destination IP addresses of the tunnel IP header) and runs checking mechanisms (e.g. connection validity check, GRE sequence number check or GRE Key value check, if present). If all the checking mechanisms finish successfully and no reordering is necessary, then the packet is promptly transmitted to the Transport and Application

Layers through the tunnel interface. If reordering is on and GRE sequence number indicates that one or more data unit(s) are missing, then the packet is placed into a buffer array for reordering purposes. (Reordering is discussed in Section 6.)

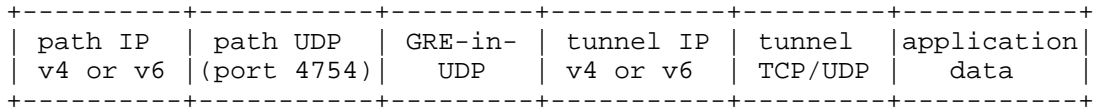


Figure 3: PDU encapsulation of the MPT data communication

4. MPT Control

A connection can be established between two MPT servers in two ways:

1. When the MPT server is started, it establishes the connection on the basis of a connection specification from the configuration files. In this case, the connection specification contains all the necessary parameters. MPT client commands still can be used to modify the parameters, switch off and on paths, etc. as described in Section 4.2.
2. The connection is established by using MPT client commands. In this case the command line arguments of the MPT commands and configuration files contain the necessary parameters.

4.1. Configuration Information

The MPT configuration files contain various pieces of information. They can be divided into two groups:

1. general information for the MPT server
2. connections specification(s)

4.1.1. General Information for the MPT Server

The MPT configuration file is made up of sections. The "general" section **MUST** be present and it contains general information for the operation of the MPT server, whereas there **MAY** several sections follow, each of which describes a different tunnel.

The general section **MUST** contain the following elements:

- o tunnel number: the number of tunnels to create (they are to be described in separate sections)

- o accept remote: it is a key (yes/no) whether this MPT server should accept commands from other MPT servers to build up connections, which are not defined in the local configuration files
- o local command port: the port number on which the local MPT client software can give commands to the MPT server
- o command timeout: the timeout value for the MPT client

For each tunnel, a separate section is to be used to describe the following parameters:

- o name: The name is used by the operating system to access to the interface.
- o MTU: The maximum transmission unit of the tunnel interface. For the Ethernet environment, the value should be set between 1436 and 1468 (depending on the additional header sizes, used by the actual system). It can be calculated as: $1500 - \text{Path_IP_header_size} - \text{UDP_header_size} - \text{GRE_header_size}$.
- o ipv4_addr: IPv4 address and mask
- o ipv6_addr: IPv6 address and mask

Note that both ipv4_addr and ipv6_addr MAY be present. At least one of them MUST be present.

It is important that the same tunnel may be used by several connections. A connection can be uniquely identified by the IP addresses of the two endpoints, which have to be of the same type (IPv4 or IPv6).

4.1.2. Connection Specifications

A connection specification is made up of sections. The "connection" section contains parameters that are to be specified only ones for each connection. The "paths" section contains one or more path definitions. The optional "networks" section contains network definitions for routing purposes.

The general section (called "connection") MUST contain the following elements:

- o name: The unique name of the connection. If we use multiple connections, the name must uniquely identify the connection.

- o permissions: There MAY be SEND and RECEIVE permissions, which allow sending and receiving connection updates. The term SEND means that the local MPT environment is allowed to start configuration change to the peer. The term RECEIVE means that the peer is allowed to start a configuration change, and the local MPT environment will accept it. (The actual execution of the requested change depends on further conditions, e.g. successful authentication.)
- o IP version: its possible values are 4 or 6.
- o local IP address: must be of the IP version specified above and must be the same as defined for the tunnel.
- o remote IP address: the IP address of the remote peer, must be of the IP version specified above
- o local data port number: used for data communication, SHOULD be set to the 4754 GRE-in-UDP port number
- o remote data port number: used for data communication, SHOULD be set to the 4754 GRE-in-UDP port number
- o remote command port number: The UDP port number of the peer, which is used to accept control commands. If the local MPT client starts an MPT command (e.g. turning off a path usage), the MPT server will communicate this action to the peer by using the remote command port number as the destination port number.
- o path count: The key is an integer P, denoting the number of paths defined for this connection. The minimum value is 1, the maximum value is implementation dependent, e.g. 20. This configuration file MUST have P sections (usually named [path_n]), where $0 < n < P$, describing all paths of the connection.
- o network count: The MPT environment can be used to establish a tunnel between networks (i.e. not only the tunnel peers can use the tunnel for communication). The minimum value is 0, the maximum value is implementation dependent, e.g. 20. This key is an integer L, denoting the number of networks on which the actual connection is able to route. This configuration file MUST have L sections (usually named [net_n]), where $0 < n < L$, describing all networks that belong to the connection.
- o status: The key indicates the initial status of the connection. The value 0 means OK.

- o authentication type: The MPT system uses control communication between the tunnel endpoints. The control communication can be requested to use authentication. The value 0 means no authentication.

The following elements are OPTIONAL:

- o reorder window: The reorder window value specifies the length (or size) of the buffer-array for the optional packet reordering on the basis of the GRE sequence numbers, see Section 6 for more information. The value of 0 (which is the default value when omitting the key) means, that no sorting will be performed at the receiver (the packets are transferred to the tunnel interface immediately when they arrive).
- o maximum buffer delay: This key is used only if we require ordered packet transmission (i.e. reorder window > 0). If ordered packet transmission is required, maximum buffer delay specifies the maximum time (in milliseconds) while the packet may be stored in the buffer-array.
- o authentication key: The authentication key contains the key value of the control communication authentication. Some algorithms do not need authentication keys. In this case the specification of the authentication key is not necessary, or will be ignored.

A path definition section MUST contain the following elements:

- o interface name: The value is the name of the physical interface used by the given path for packet forwarding (e.g. eth0, wlan0).
- o IP version: Specifies the version of IP used by the path. The value can be 4 or 6.
- o public IP address: Specifies the public IP address of the interface used for the tunnel communication. If the host is placed into the Global Address Realm, the public IP address is the IP address of the interface, otherwise (i.e. when the host is behind a NAT-Box) it is the public address assigned by the NAT-Box to the tunnel communication session. If the path uses IPv4 and NAT, then the special address value of 0.0.0.0 can be used to force the MPT server program to determine the public IP address automatically.
- o remote IP address: Indicates the public IP address of the remote endpoint.

- o gateway IP address: The IP address of the gateway, used to reach the peer (i.e. remote IP address) using the given path. If the operating system uses the Network Manager (nmcli) software for network configuration, then the value of 0.0.0.0 can be used to find the gateway of the named interface automatically.
- o weight out: This is the "weight of the path" in the system expressing the estimated transmission capacity of the path. The MPT server program distributes the outgoing packets between the available paths according to their weights, if per packet based mapping is used. The value must be between 1 and 10,000.
- o status: This key means the initial state of the path after starting the MPT server. The value "up" means that the path is usable (working), and the state of the path is OK. If required, may be set initially as "down".

A path definition section MAY contain the following elements:

- o private IP address: The IP address of the physical interface. Can be omitted, if the public IP address is assigned directly to the interface. When using IPv4 and NAT, the special value of 0.0.0.0 can be used to force the MPT server application to read and use the first IPv4 address assigned to the interface.
- o keepalive time: The MPT system monitors the availability of each path by sending keepalive messages regularly. The key specifies the frequency (i.e. the time between the keepalive messages in seconds) that the MPT server uses for sending keepalives. The value of zero (which is the default value) means switching off the keepalive mechanism.
- o dead time: If the keepalive mechanism is active, and the host does not receive any keepalive message on the given path from the peer for dead time seconds, then the path is considered as "dead" and will not be used for data transmission. (The default value is 3*keepalive time.)
- o weight in: This field is used at the "mpt path up" command (see Section 4.2) to set the outgoing weight of the corresponding path at the peer. The default value is 1.
- o command default: This key can be used to specify one path as the default path for control command communication. In the case of receiving the control command of "create connection", the system will use this path for the control communication.

The optional "networks" section contains network definitions for routing purposes. Each network definition begins with its name in the [net_n] format and contains the following parameters:

- o IP version: Specifies the version of IP used in the network definitions. The value can be 4 or 6.
- o source address: specifies the source network and its prefix length in the CIDR notation.
- o destination address: specifies the destination network and its prefix length in the CIDR notation.

The network configuration can also be used to provide multipath Internet connection by specifying 0.0.0.0/0 as destination address and prefix length. (The source is our tunnel address in this case.)

4.2. MPT Configuration Commands

The same control interface is used for the local administration of the MPT server (by the MPT client accessing the MPT server at the local command port through the loopback interface) and for the communication of the local MPT server with the remote MPT server (accessing it at its remote command port).

Now, some client commands will follow. Although some of the syntax of our MPT implementation will be used, the focus is not on their syntax, which may be implementation dependent, but rather on their functionalities. The execution of these commands may also involve communication between the local MPT server and a/the remote MPT server.

```
mpt address {add|del} IPADDRESS/PREFIX dev INTERFACE
```

An IPv4 or IPv6 address can be added to or deleted from a (local) interface.

```
mpt interface INTERFACE {up|down}
```

The specified interface is turned up or down plus all the paths, that are based on the given local physical interface are also turned on or off by starting the "mpt path {up|down}" command (see below) for each considered path.

```
mpt path PATH {up|down}
```

This command can be used to turn on or off a specified path. If the path status is changed to down, then it is not used by the connection, (i.e. no data is sent through that path by the MPT software).

```
mpt connection CONNECTION {create|delete}
```

This command can be used to establish or tear down a connection between the local and a remote MPT server. (The parameters are taken from local configuration files.) If the remote server is configured so, then it accepts the parameters of the connection from the local server.

```
mpt save [FILENAME]
```

The current configuration can be changed during runtime by remote peers. (This can be enabled with the accept remote key and with the permissions key.) This command is used to write these connection changes to the configuration files, so the new settings will remain after server startup or after mpt reload.

```
mpt reload [FILENAME]
```

Warm restart: the MPT server build up its connections according to its configuration files. (Our implementation only establishes, but it does not tears down connections.)

5. Possible Mappings of the Tunnel Traffic to Paths

The data packets coming from the tunnel interface must be forwarded through one of the active paths of the connection. Three possible mapping solutions are proposed:

- o Per packet based mapping means that the tunnel traffic is distributed among the paths on the basis of the parameters of the paths only, and regardless of what network flow a given packet belongs to.
- o Flow based mapping means that packets which belong to a given network flow, identified by the usual five tuple of source IP address, destination IP address, source port number, destination port number, and protocol number (TCP or UDP), or three tuple of source IP address, destination IP address, and protocol number (TCP, UDP or ICMP), are always mapped to the same path.
- o Combined mapping means the combinations of the two above in the way that packets which belong to a given network flow, identified by the way described above, are always mapped to the same connection. And the packets that belong to a connection are distributed among the paths of that connection by per packet decisions on the basis of the parameters of the paths of the connection.

We illustrate the three mapping solutions by examples.

Definitions for the examples:

Computers A and B are interconnected by 3 different paths:

path_1: 100Base-TX Ethernet

path_2: 802.11g WiFi

path_3: LTE

Connection_1 has 3 paths with the following weight out values:

path_1: 5

path_2: 2

path_3: 3

Example 1 (Per packet based mapping)

All the traffic between the two computers is distributed among the three paths of Connection_1 proportionally to their weight out

values. A decision is made about every single packet as described in Section 5.1, regardless of the fact what application it belongs to.

Advantage: The transmission capacity of all the paths can be utilized.

Disadvantage: There is no possibility to use different mappings for different applications.

Example 2 (Per flow based mapping)

Based on the destination port number or port range, the traffic of different applications are mapped to paths as follows:

HTTP, VoD: path_1

FTP, Bit-Torrent: path_2

VoIP: path3

Advantage: Application can be differentiated: e.g. the delay critical VoIP can use LTE, whereas the free WiFi is satisfactory for the non-mission critical Bit-Torrent.

Disadvantage: The mapping of the traffic is too rigid, all the traffic of applications of a given type is mapped to a single path, therefore, the applications (and thus their users) do not experience the benefits of multipath transmission.

Example 3 (Combined mapping)

We define further two connections:

Connection_2

path_1: 5

path_2: 2

Connection_2

path_1: 5

path_3: 3

Based on the destination port number or port range, the traffic of different applications are mapped to paths as follows:

HTTP: connection_1

FTP, Bit-Torrent: connecton_2

VoIP, VoD: connection_3

Advantage: The applications may benefit from the multipath transmission, whereas each types of applications use those paths, which are beneficial and affordable for them.

Disadvantage: The price of the above resilience is the time and computational complexity of the execution of both algorithms.

Conclusion: The appropriate choice of the mapping algorithm depends on the expectations of the user.

5.1. Per Packet Based Mapping

The aim of the "per packet based" mapping is to distribute the tunnel traffic to the paths proportionally to their transmission capacity. This mapping facilitates the aggregation of the transmission capacities of the paths.

In MPT, the transmission capacity of the paths is represented by their WEIGHT_OUT parameter.

The following algorithm calculates the sending vector, which contains the indices of the paths in the order they are to be used for transmission.

ALGORITHM calculate_sending_vector

INPUT: $W[i]$ ($1 \leq i \leq N$), the vector of the weights of the paths.

(Note: We have N paths with indices $(1, \dots, N)$)

OUTPUT: $O[j]$ ($1 \leq j \leq M$), the sending vector containing the indices of the paths; where M is the length of the sending cycle.

lcm := Least Common Multiple for ($W[1], \dots, W[N]$)

```
M := 0

s[i] := 0, for all i (1 <= i <= N)

(Note: s[i] will store the sum of the increments for path i, where
the increment is lcm/W[i])
```

```
WHILE TRUE DO

    z := min(s[1]+lcm/W[1], . . . , s[N]+lcm/W[N])

    k := The smallest index i, for which z == s[i]+lcm/W[i]

    M := M+1

    s[k] := z

    O[M] := k

    IF s[i] == z for all i (1 <= i <= N) THEN RETURN

DONE

END
```

A sample C code can be found in the Appendix.

5.2. Flow Based Mapping

The aim of the flow based mapping is to be able to distinguish the packets belong to different network flows and map them to the path that was set for them. (E.g. WiFi is used for Torrent traffic and LTE is used for VoIP calls.)

TBD

5.3. Combined Mapping

TBD

6. Packet Reordering

As the delay of the different paths can be different, packet reordering may appear in a packet sequence transmission. The MPT environment offers an optional feature to ensure the right ordered

packet transmission for the tunnel communication. If this feature is enabled, the receiver uses a buffer-array to store the incoming (unordered) packets. Then the packets are sorted according to the GRE sequence numbers, so ensuring the ordered transmission to the receiver's tunnel interface.

There are two parameters aimed to control the reordering. The reorder window parameter specifies the length of the buffer array used for reordering. The maximum buffer delay parameter specifies the maximum time (in milliseconds) while the packet is stored in the buffer-array. If the packet is delayed in the buffer-array for the specified time, it will be transmitted to the tunnel interface, even in the case, when some packets are missing before the considered packet. The missing packets are considered as lost packets (i.e. we will not wait more for a lost packet). The arrived packets are transferred to the tunnel interface according to their GRE sequence number, so the ordered delivery will be kept also in the case of packet loss.

How to set the values of these parameters?

As for maximum buffer delay, if its value is too small, then MPT may incorrectly consider a sequence number as lost, and if it arrives later, MPT has to drop it to keep on the order-right delivery. If its value is too large, then the packet loss will be determined too late, and thus the communication performance may decrease. Our experience shows that a feasible choice could be: a few times the RTT (Round-Trip Time) of the slowest path.

As for reorder window, it MUST be large enough to store packets arriving at maximum line rate from all the active paths of the given connection during a maximum buffer delay interval.

The appropriate choice of these parameters is still subject of research.

7. Why MPT is Considered Experimental?

We view MPT as a research area rather than a solution which is ready for deployment. We have an MPT implementation, which is workable, but it contains only the "per packet based" mapping of the tunnel traffic to the paths. One of our aims of writing this Internet Draft is to enable others to write MPT implementations. It is our hope that the experience gained with preparing other implementations as well as the results of their testing and performance analysis will lead to a better MPT specification, which may then serve as a

standard track specification of an improved MPT, which will be ready for deployment.

In this section, we summarize the most important results as well as the open questions of the MPT related research.

7.1. Published Results

7.1.1. MPT Concept and First Implementation

The conceptual architecture of MPT, comparison with other multipath solutions, some details of the first implementation and some test results are available in [Alm2017].

The user manual of the first MPT implementation and the precompiled MPT libraries for Linux (both i386 and amd64) and Raspbian are available from [Mpt2017].

7.1.2. Estimation of the Channel Aggregation Capabilities

The channel aggregation capabilities of an early MPT implementation, which did not use GRE-in-UDP, were analyzed up to twelve 100Mbps links in [Len2015].

Some of the above tests were repeated with the current GRE-in-UDP based MPT implementation, and the path aggregation capabilities of MPT were compared to that of MPTCP in [Kov2016].

7.1.3. Demonstrating the Resilience of an MPT Connection

The resilience property of the early MPT implementation, which did not use GRE-in-UDP, was demonstrated in [Alm2014] and in [Alm2015].

The fast connection recovery of the GRE-in-UDP based MPT implementation was demonstrated in [Fej2016].

Playout buffer length triggered path switching algorithm was developed for the GRE-in-UDP based MPT, and its effectiveness was demonstrated by the elimination of the stalling events on YouTube video playback [Fej2017].

7.2. Open questions

7.2.1. Parameters

The optimal (or good enough) choice of the reorder window size and maximum buffer delay parameters are important questions, which should be solved before MPT can be deployed.

7.2.2. Development of Further Mapping Algorithms

The current MPT implementation [Mpt2017] includes only the per packet base mapping. For a precise specification of the further two mapping algorithms, we would like to use our experiences with them. There are some open questions e.g. how to handle the traffic that is neither TCP nor UDP?

7.2.3. Performance Issues

The current MPT implementation [Mpt2017] works in user space. Thus, it is not surprising, that multipath transmission of the same amount of traffic by MPT results in higher CPU load than its multipath transmission by MPTCP [Kov2017]. How much CPU power could a kernel space MPT implementation save?

It was also pointed out by [Kov2017], that MPT is not able to utilize the computing power of more than two CPU cores. It is so, because MPT uses only two threads (one for each direction). This is not a serious issue, when MPT is used on personal computers. However, when MPT is used to connect several networks, it is an important question, how MPT could utilize the computing power of the modern CPUs with several cores.

8. Security Considerations

Threats that apply to GRE-in-UDP tunneling, apply here, too. For the security considerations of GRE-in-UDP, please refer to Section 11 of [RFC8086].

If an MPT server is configured so, its peer is allowed to build up connections. It may lead to resource exhaustion and thus successful DoS (Denial of Service) attacks.

Authentication between MPT servers is optional, which may lead to security issues.

9. IANA Considerations

Port numbers may be reserved for local command port and remote command port.

10. Conclusions

Hereby we publish the specifications of the MPT network layer multipath library in the hope, that it can be made better by the review and comments of the WG members and, after answering several open questions, one day MPT can mature to be a production tool. We seek for interested volunteers for a different implementation and we would be happy to take part in research cooperation. We welcome all kinds of feedback from anyone to make MPT better.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC8086] Young, L. (Editor), Crabbe, E., Xu, X., and T. Herbert, "GRE-in-UDP Encapsulation", RFC 8086, DOI: 10.17487/RFC8086, March 2017.

11.2. Informative References

- [Alm2014] Almasi, B., "A solution for changing the communication interfaces between WiFi and 3G without packet loss., in Proc. 37th Int. Conf. on Telecommunications and Signal Processing (TSP 2014), Berlin, Germany, Jul. 1-3, 2014, pp. 73.77
- [Alm2015] Almasi, B., Kosa, M., Fejes, F., Katona, R., and L. Pusok, "MPT: a solution for eliminating the effect of network breakdowns in case of HD video stream transmission", in: Proc. 6th IEEE Conf. on Cognitive Infocommunications (CogInfoCom 2015), Gyor, Hungary, 2015, pp. 121.126, doi: 10.1109/CogInfoCom.2015.7390576 .
- [Alm2017] Almasi, B., Lencse, G., and Sz. Szilagyi, "Investigating the Multipath Extension of the GRE in UDP Technology", Computer Communications (Elsevier), vol. 103, no. 1, (2017.) pp. 29-38, DOI: 10.1016/j.comcom.2017.02.002

- [Fej2016] Fejes, F., Katona, R., and L. Pusok, "Multipath strategies and solutions in multihomed mobile environments", in: Proc. 7th IEEE Conf. on Cognitive Infocommunications (CogInfoCom 2016), Wroclaw, Poland, 2016, pp. 79.84, doi: 10.1109/CogInfoCom.2016.7804529
- [Fej2017] Fejes, F., Racz, S., and G. Szabo, "Application agnostic QoE triggered multipath switching for Android devices", In: Proc. 2017 IEEE International Conference on Communications (IEEE ICC 2017), Paris, France, 21-25 May 21-25, 2017. pp. 1585-1591.
- [Kov2016] Kovacs, A., "Comparing the aggregation capability of the MPT communications library and multipath TCP", in: Proc. 7th IEEE Conf. on Cognitive Infocommunications (CogInfoCom 2016), Wroclaw, Poland, 2016, pp. 157.162, doi: 10.1109/CogInfoCom.2016.7804542
- [Kov2017] Kovacs, A., "Evaluation of the Aggregation Capability of the MPT Communications Library and Multipath TCP", unpublished
- [Len2015] Lencse, G. and A. Kovacs, "Advanced Measurements of the Aggregation Capability of the MPT Multipath Communication Library", International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems, vol. 4. no. 2. (2015.) pp 41-48. DOI: 10.11601/ijates.v4i2.112
- [Ley2016] Leymann, N., Heidemann, C., Zhang, M., Sarikaya, B, and M. Cullen, "Huawei's GRE Tunnel Bonding Protocol", Internet Draft, draft-zhang-gre-tunnel-bonding-05.txt
- [Mpt2017] MPT - Multipath Communication Library, <http://irh.inf.unideb.hu/user/szilagyi/mpt/>
- [RFC6824] Ford, A., Raiciu, C, Handley, M., and O Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI: 10.17487/RFC6824, January, 2013.

12. Acknowledgments

The MPT Network Layer Multipath Library was invented by Bela Almasi, the organizer and original leader of the MPT development team.

This document was prepared using 2-Word-v2.0.template.dot.

Appendix A. Sample C code for packet reordering

```
void calculate_pathselection(connection_type *con) {
    long long lcm;
    long gcd, min_inc, cinc;
    int i,j, min_idx;
    path_type *p;

    con->pathselectionlength = 0;
    gcd = con->mpath[0].weight_out;
    lcm = gcd;
    for (i = 0; i < M; i++)
        O[i] = NULL;

    for (i = 0; i < con->path_count; i++) {
        gcd = CALCULATE_GCD(gcd, con->mpath[i].weight_out);
        lcm = (lcm * con->mpath[i].weight_out) / gcd;
        con->mpath[i].selection_increment = 0;
    }

    for (j = 0; j < M; j++) {
        min_idx = 0;
        min_inc = lcm + 1;
        for (i = 0; i < con->path_count; i++) {
            p = &con->mpath[i];
            cinc = p->selection_increment + (lcm / p->weight_out);
            if ((p->weight_out) && (cinc < min_inc)) {
                min_idx = i;
                min_inc = cinc;
            }
        }
        O[j] = &con->mpath[min_idx];
        con->mpath[min_idx].selection_increment = min_inc;

        for (i = 0; i < con->path_count; i++) // check if ready
            if (con->mpath[i].selection_increment != min_inc)
                goto NEXT_SELECTION;
        break;
    }

NEXT_SELECTION:
    continue;
}

con->path_index = 0;
con->pathselectionlength = j + 1;
}
```

Copyright (c) 2017 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

Authors. Addresses

Gabor Lencse
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara
630-0192, JAPAN

Phone: +81 743 72 5215
Email: gabor-l@is.naist.jp

Szabolcs Szilagyi
University of Debrecen
Egyetem ter 1.
H-4032 Debrecen
Hungary

Phone: +36 52 512 900 / 75015
Email: szilagyi.szabolcs@inf.unideb.hu

Ferenc Fejes
University of Debrecen
Egyetem ter 1.
H-4032 Debrecen
Hungary

Phone: +36 70 545 48 07
Email: fejes@openmailbox.org

Marius Georgescu
RCS&RDS
Strada Dr. Nicolae D. Staicovici 71-75
Bucharest 030167
Romania

Phone: +40 31 005 0979
Email: marius.georgescu@rcs-rds.ro

Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 29, 2017

T. Mizrahi
Marvell
J. Fabini
Vienna University of Technology
A. Morton
AT&T Labs
June 27, 2017

Guidelines for Defining Packet Timestamps
draft-mizrahi-intarea-packet-timestamps-00

Abstract

This document specifies guidelines for defining binary packet timestamp formats in networking protocols at various layers. It also presents three recommended timestamp formats. The target audience of this memo includes network protocol designers. It is expected that a new network protocol that requires a packet timestamp will, in most cases, use one of the recommended timestamp formats. If none of the recommended formats fits the protocol requirements, the new protocol specification should specify the format of the packet timestamp according to the guidelines in this document.

Status of This Memo

This Internet-Draft is submitted 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 December 29, 2017.

Copyright Notice

Copyright (c) 2017 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. Introduction	2
2. Terminology	3
2.1. Requirements Language	3
2.2. Abbreviations	3
3. Packet Timestamp Format Specification	3
4. Recommended Timestamp Formats	4
4.1. NTP Timestamp Formats	4
4.1.1. NTP 64-bit Timestamp Format	4
4.1.2. NTP 32-bit Timestamp Format	6
4.2. The PTP Concatenated Timestamp Format	7
5. Packet Timestamp Control Field	8
6. IANA Considerations	9
7. Security Considerations	9
8. References	10
8.1. Normative References	10
8.2. Informative References	10
Authors' Addresses	11

1. Introduction

Timestamps are widely used in network protocols for various purposes, including delay measurement, clock synchronization, and logging or reporting the time of an event.

Timestamps are represented in the RFC series in one of two forms: text-based timestamps, and packet timestamps. Text-based timestamps [RFC3339] are represented as user-friendly strings, and are widely used in the RFC series, for example in information objects and data models, e.g., [RFC5646], [RFC6991], and [RFC7493]. Packet timestamps, on the other hand, are represented by a compact binary field that has a fixed size, and are not intended to have a human-friendly format. Packet timestamps are also very common in the RFC series, and are used for example for measuring delay and for synchronizing clocks, e.g., [RFC5905], [RFC4656], and [RFC1323].

This memo presents guidelines for defining a packet timestamp format in network protocols. Three recommended timestamp formats are presented. It is expected that a new network protocol that requires

a packet timestamp will, in most cases, use one of the recommended timestamp formats. If none of the recommended formats fits the protocol requirements, the new protocol specification should specify the format of the packet timestamp according to the guidelines in this document.

2. Terminology

2.1. Requirements Language

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].

2.2. Abbreviations

NTP Network Time Protocol [RFC5905]

PTP Precision Time Protocol [IEEE1588]

3. Packet Timestamp Format Specification

This section defines a template for specifying packet timestamp formats. A timestamp format specification **MUST** include the following aspects:

Timestamp field format:

The format of the timestamp field consists of:

+ Size: The number of bits (or octets) used to represent the packet timestamp field.

+ Units: The units used to represent the timestamp.

If the timestamp is comprised of more than one field, the format of each field is specified.

Epoch:

The origin of the timescale used for the timestamp; the moment in time used as a reference for the timestamp value.

Wraparound:

The wraparound period of the timestamp. Any further wraparound-related considerations should be described here.

Synchronization aspects:

Any assumptions or requirements related to synchronization should be specified, for example, whether it is assumed that nodes populating the timestamps should be synchronized, and whether the timestamp is measured with respect to a central reference clock such as a stratum 1 NTP server.

4. Recommended Timestamp Formats

This memo recommends to use one of the three timestamp formats specified below. In cases where the three timestamp formats below do not satisfy the protocol requirements, the timestamp specification should clearly state the reasons for defining a new format.

Clearly, different network protocols (and the use cases they serve) may have different requirements and constraints, and consequently may use different timestamp formats. The choice of the specific timestamp format for a given protocol may depend on a various factors. A few examples of factors that may affect the choice of the timestamp format:

- o Timestamp size: while some network protocols may allow a large timestamp fields, in other cases there may be constraints with respect to the timestamp size, affecting the choice of the timestamp format.
- o Resolution: the time resolution is another factor that may directly affect the selected timestamp format. Similarly, the wraparound periodicity of the timestamp may also affect the selected format.
- o Common format for multiple protocols: if there are two or more network protocols that use timestamps and are often used together in typical systems, using a common timestamp format should be preferred if possible.

4.1. NTP Timestamp Formats

4.1.1. NTP 64-bit Timestamp Format

The Network Time Protocol (NTP) 64-bit timestamp format is defined in [RFC5905]. This timestamp format is used in several network protocols, including [RFC6374], [RFC4656], and [RFC5357]. Since this timestamp format is used in NTP, this timestamp format should be preferred in network protocols that are typically deployed in concert with NTP.

The format is presented in this section according to the template defined in Section 3.

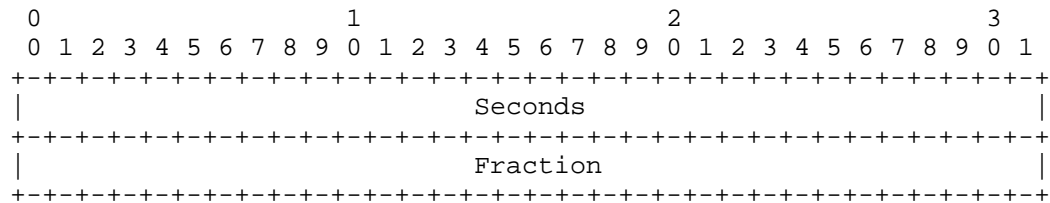


Figure 1: NTP [RFC5905] 64-bit Timestamp Format

Timestamp field format:

Seconds: specifies the integer portion of the number of seconds since the epoch.

+ Size: 32 bits.

+ Units: seconds.

Fraction: specifies the fractional portion of the number of seconds since the epoch.

+ Size: 32 bits.

+ Units: the unit is 2^{-32} seconds, which is roughly equal to 233 picoseconds.

Epoch:

The epoch is 1 January 1900 at 00:00 UTC.

Wraparound:

This time format wraps around every 2^{32} seconds, which is roughly 136 years. The next wraparound will occur in the year 2036.

Synchronization aspects:

The timestamp format itself does not place a requirement on the degree of synchronization between nodes; such requirements emerge from the protocol and use cases served. Note that if the nodes that use this timestamp format use NTP-based synchronization, the timestamp may be derived from the NTP-synchronized clock, allowing

the timestamp to be measured with respect to the clock of an NTP server.

4.1.2. NTP 32-bit Timestamp Format

The Network Time Protocol (NTP) 32-bit timestamp format is defined in [RFC5905]. This timestamp format is used in [I-D.morton-ippm-mbm-registry]. This timestamp format should be preferred in network protocols that are typically deployed in concert with NTP. The 32-bit format can be used either when space constraints do not allow the use of the 64-bit format, or when the 32-bit format satisfies the resolution and wraparound requirements.

The format is presented in this section according to the template defined in Section 3.

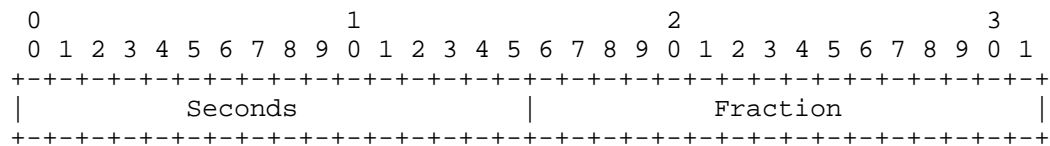


Figure 2: NTP [RFC5905] 32-bit Timestamp Format

Timestamp field format:

Seconds: specifies the integer portion of the number of seconds since the epoch.

+ Size: 16 bits.

+ Units: seconds.

Fraction: specifies the fractional portion of the number of seconds since the epoch.

+ Size: 16 bits.

+ Units: the unit is 2^{-16} seconds, which is roughly equal to 15.3 microseconds.

Epoch:

The epoch is 1 January 1900 at 00:00 UTC.

Wraparound:

This time format wraps around every 2^{16} seconds, which is roughly 18 hours.

Synchronization aspects:

The timestamp format itself does not place a requirement on the degree of synchronization between nodes; such requirements emerge from the protocol and use cases served. Note that if the nodes that use this timestamp format use NTP-based synchronization, the timestamp may be derived from the NTP-synchronized clock, allowing the timestamp to be measured with respect to the clock of an NTP server.

4.2. The PTP Concatenated Timestamp Format

The Precision Time Protocol (PTP) [IEEE1588] uses an 80-bit timestamp format. The concatenated timestamp format is a 64-bit field, which is the 64 least significant bits of the 80-bit PTP timestamp. Since this timestamp format is similar to the one used in PTP, this timestamp format should be preferred in network protocols that are typically deployed in PTP-capable devices.

The PTP concatenated timestamp format is used in several protocols, such as [RFC6374], [RFC7456], [RFC8186] and [ITU-T-Y.1731].

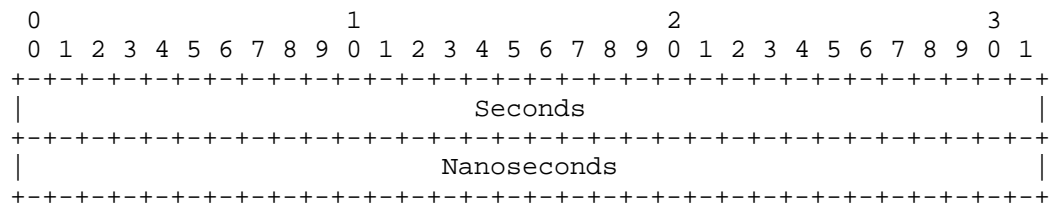


Figure 3: PTP [IEEE1588] Concatenated Timestamp Format

Timestamp field format:

Seconds: specifies the integer portion of the number of seconds since the epoch.

+ Size: 32 bits.

+ Units: seconds.

Nanoseconds: specifies the fractional portion of the number of seconds since the epoch.

+ Size: 32 bits.

+ Units: nanoseconds. The value of this field is in the range 0 to $(10^9)-1$.

Epoch:

The PTP [IEEE1588] epoch is 1 January 1970 00:00:00 TAI, which is 31 December 1969 23:59:51.999918 UTC.

Wraparound:

This time format wraps around every 2^{32} seconds, which is roughly 136 years. The next wraparound will occur in the year 2106.

Synchronization aspects:

The timestamp format itself does not place a requirement on the degree of synchronization between nodes; such requirements emerge from the protocol and use cases served. Note that if the nodes that use this timestamp format use PTP-based synchronization, the timestamp may be derived from the PTP-synchronized clock, allowing the timestamp to be measured with respect to the clock of an PTP Grandmaster clock.

5. Packet Timestamp Control Field

In some cases it is desirable to have a control field that includes information about the timestamp format. This section defines a recommended format of a timestamp-related control field that is intended for network protocols that require such timestamp-related control information.

The recommended control field includes the following sub-fields:

- o Timestamp format.
- o Precision - the resolution or granularity of the system clock.
- o Epoch.
- o Era - the number of times the time has wrapped around since the epoch.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

A network protocol that uses a packet timestamp MUST specify the security considerations that result from using the timestamp. This section provides an overview of some of the common security considerations of using timestamps.

Any metadata that is attached to control or data packets, and specifically packet timestamps, can facilitate network reconnaissance; by passively eavesdropping to timestamped packets an attacker can gather information about the network performance, and about the level of synchronization between nodes.

Timestamps can be spoofed or modified by on-path attackers, thus attacking the application that uses the timestamps. For example, if timestamps are used in a delay measurement protocol, an attacker can modify en route timestamps in a way that manipulates the measurement results. Integrity protection mechanisms, such as Hashed Message Authentication Codes (HMAC), can mitigate such attacks. The specification of an integrity protection mechanism is outside the scope of this document, as typically integrity protection will be defined on a per-network-protocol basis, and not specifically for the timestamp field.

Another potential threat that can have a similar impact is delay attacks. An attacker can maliciously delay some or all of the en route messages, with the same harmful implications as described in the previous paragraph. Mitigating delay attacks is a significant challenge; in contrast to spoofing and modification attacks, the delay attack cannot be prevented by cryptographic integrity protection mechanisms. In some cases delay attacks can be mitigated by sending the timestamped information through multiple paths, allowing to detect and to be resilient to an attacker that has access to one of the paths.

In many cases timestamping relies on an underlying synchronization mechanism. Thus, any attack that compromises the synchronization mechanism can also compromise protocols that use timestamping. Attacks on time protocols are discussed in detail in [RFC7384].

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

8.2. Informative References

- [I-D.morton-ippm-mbm-registry] Morton, A. and M. Mathis, "Initial Performance Metric Registry Entries Part 2: MBM", draft-morton-ippm-mbm-registry-01 (work in progress), March 2017.
- [IEEE1588] IEEE, "IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Version 2", 2008.
- [ITU-T-Y.1731] ITU-T, "OAM functions and mechanisms for Ethernet based Networks", 2013.
- [RFC1323] Jacobson, V., Braden, R., and D. Borman, "TCP Extensions for High Performance", RFC 1323, DOI 10.17487/RFC1323, May 1992, <<http://www.rfc-editor.org/info/rfc1323>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<http://www.rfc-editor.org/info/rfc3339>>.
- [RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006, <<http://www.rfc-editor.org/info/rfc4656>>.
- [RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <<http://www.rfc-editor.org/info/rfc5357>>.
- [RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <<http://www.rfc-editor.org/info/rfc5646>>.

- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<http://www.rfc-editor.org/info/rfc5905>>.
- [RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, DOI 10.17487/RFC6374, September 2011, <<http://www.rfc-editor.org/info/rfc6374>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<http://www.rfc-editor.org/info/rfc6991>>.
- [RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <<http://www.rfc-editor.org/info/rfc7384>>.
- [RFC7456] Mizrahi, T., Senevirathne, T., Salam, S., Kumar, D., and D. Eastlake 3rd, "Loss and Delay Measurement in Transparent Interconnection of Lots of Links (TRILL)", RFC 7456, DOI 10.17487/RFC7456, March 2015, <<http://www.rfc-editor.org/info/rfc7456>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <<http://www.rfc-editor.org/info/rfc7493>>.
- [RFC8186] Mirsky, G. and I. Meilik, "Support of the IEEE 1588 Timestamp Format in a Two-Way Active Measurement Protocol (TWAMP)", RFC 8186, DOI 10.17487/RFC8186, June 2017, <<http://www.rfc-editor.org/info/rfc8186>>.

Authors' Addresses

Tal Mizrahi
Marvell
6 Hamada st.
Yokneam
Israel

Email: talmi@marvell.com

Joachim Fabini
Vienna University of Technology
Gusshausstrasse 25/E389
Vienna 1040
Austria

Phone: +43 1 58801 38813
Fax: +43 1 58801 38898
Email: Joachim.Fabini@tuwien.ac.at
URI: <http://www.tc.tuwien.ac.at/about-us/staff/joachim-fabini/>

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Phone: +1 732 420 1571
Fax: +1 732 368 1192
Email: acmorton@att.com
URI: <http://home.comcast.net/~acmacm/>

Internet Area Working Group
Internet-Draft
Intended status: Experimental
Expires: December 30, 2017

V. Olteanu
D. Niculescu
University Politehnica of Bucharest
June 28, 2017

SOCKS Protocol Version 6
draft-olteanu-intarea-socks-6-00

Abstract

The SOCKS protocol is used primarily to proxy TCP connections to arbitrary destinations via the use of a proxy server. Under the latest version of the protocol (version 5), it takes 2 RTTs (or 3, if authentication is used) before data can flow between the client and the server.

This memo proposes SOCKS version 6, which reduces the number of RTTs used, takes full advantage of TCP Fast Open, and adds support for 0-RTT authentication.

Status of This Memo

This Internet-Draft is submitted 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 December 30, 2017.

Copyright Notice

Copyright (c) 2017 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. Introduction	2
2. Requirements language	3
3. Mode of operation	3
4. Connection Requests	5
5. SOCKS Options	7
5.1. Authentication options	7
6. Authentication Replies	8
7. Operation Replies	9
7.1. Handling CONNECT	10
7.2. Handling BIND	10
7.3. Handling UDP ASSOCIATE	11
8. Security Considerations	11
9. IANA Considerations	11
10. Acknowledgements	11
11. References	11
11.1. Normative References	11
11.2. Informative References	12
Authors' Addresses	12

1. Introduction

Versions 4 and 5 [RFC1928] of the SOCKS protocol were developed two decades ago and are in widespread use for circuit level gateways or as circumvention tools, and enjoy wide support and usage from various software, such as web browsers, SSH clients, and proxifiers. However, their design needs an update in order to take advantage of the new features of transport protocols, such as TCP Fast Open [RFC7413], or to better assist newer transport protocols, such as MPTCP [RFC6824].

One of the main issues faced by SOCKS version 5 is that, when taking into account the TCP handshake, method negotiation, authentication, connection request and grant, it may take up to 5 RTTs for a data exchange to take place at the application layer. This is especially costly in networks with a large delay at the access layer, such as 3G, 4G, or satellite.

The desire to reduce the number of RTTs manifests itself in the design of newer security protocols. TLS version 1.3 [I-D.ietf-tls-tls13] defines a zero round trip (0-RTT) handshake mode for connections if the client and server had previously communicated.

TCP Fast Open [RFC7413] is a TCP option that allows TCP to send data in the SYN and receive a response in the first ACK, and aims at obtaining a data response in one RTT. The SOCKS protocol needs to concern itself with at least two TFO deployment scenarios: First, when TFO is available end-to-end (at the client, at the proxy, and at the server); second, when TFO is active between the client and the proxy, but not at the server.

This document describes the SOCKS protocol version 6. The key improvements over SOCKS version 5 are:

- o The client sends as much information upfront as possible, and does not wait for the authentication process to conclude before requesting the creation of a socket.
- o The connection request also mimics the semantics of TCP Fast Open [RFC7413]. As part of the connection request, the client can supply the payload for the initial SYN that is sent out to the server.
- o The protocol can be extended via options without breaking backward-compatibility.
- o The protocol can leverage the aforementioned options to support 0-RTT authentication schemes.

2. Requirements language

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 [RFC2119].

3. Mode of operation

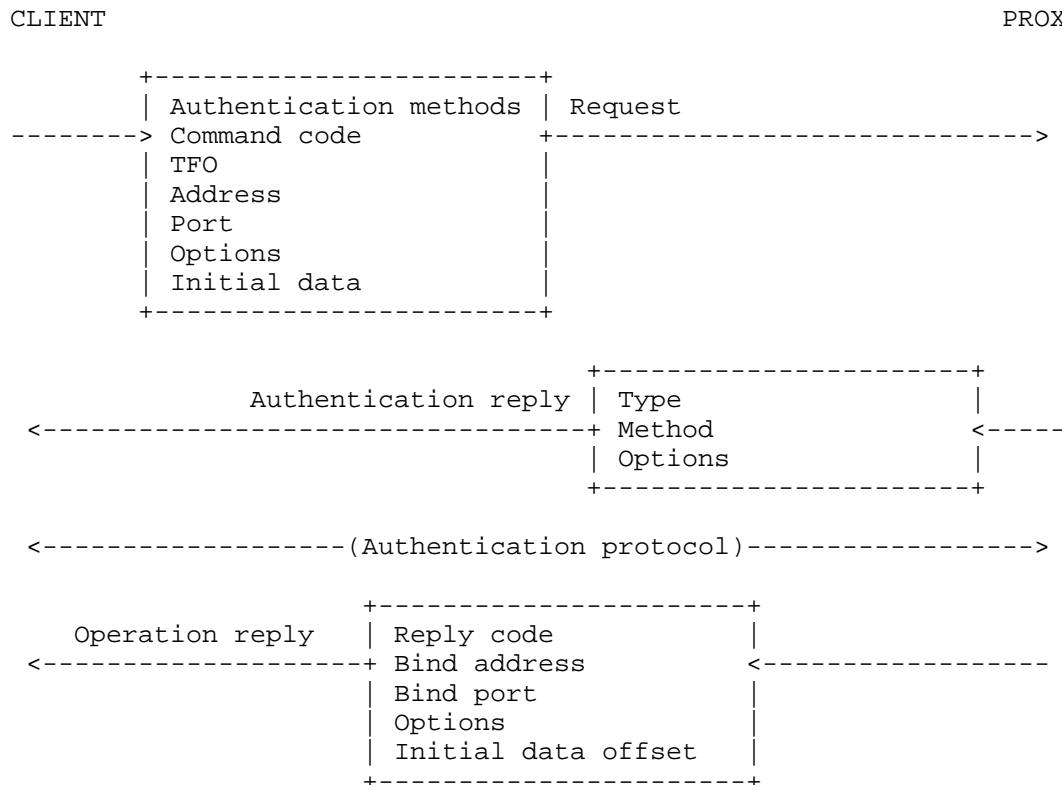


Figure 1: The SOCKS version 6 protocol message exchange

When a TCP-based client wishes to establish a connection to a server, it must open a TCP connection to the appropriate SOCKS port on the SOCKS proxy. The client then enters a negotiation phase, by sending the request in figure Figure 1, that contains, in addition to fields present in SOCKS 5 [RFC1928], fields that facilitate low RTT usage and faster authentication negotiation.

Next, the server sends an authentication reply. If the request did not contain the necessary authentication information, the proxy indicates an authentication method that must proceed. This may trigger a longer authentication sequence that could include tokens for ulterior faster authentications. The part labeled "Authentication protocol" is specific to the authentication method employed and is not expected to be employed for every connection between a client and its proxy server. The authentication protocol typically takes up 1 RTT or more.

If the authentication is successful, an operation reply is generated by the proxy. It indicates whether the proxy was successful in creating the requested socket or not.

In the fast case, when authentication is properly set up, the proxy attempts to create the socket immediately after the receipt of the request, thus achieving an operational connection in one RTT (provided TFO functionality is available at the client, proxy, and server).

4. Connection Requests

The client starts by sending a request to the proxy.

Version		Number of Methods		Methods
Major	Minor			
1	1	1	Variable	

Command Code	TFO	Address Type	Address	Port
1	1	1	Variable	2

Number of Options	Options	Initial Data Size	Initial Data
1	Variable	2	Variable

Figure 2: SOCKS 6 Request

- o Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.
- o Number of Methods: The number of supported authentication methods that the client wishes to advertise.
- o Methods: One byte per advertised method. Method numbers are assigned by IANA.
- o Command Code:
 - * 0x00 AUTH: authenticate the client and do nothing.

- * 0x01 CONNECT: requests the establishment of a TCP connection.
- * 0x02 BIND: requests the establishment of a TCP port binding.
- * 0x03 UDP ASSOCIATE: requests a UDP port association.
- o TFO:
 - * 0x00 indicates that the proxy MUST NOT attempt to use TFO in case of a CONNECT command, or accept TFO in case of a BIND command. In case of an AUTH or UDP ASSOCIATE command, this field MUST be set to 0x00.
 - * 0x01 indicates that the proxy SHOULD attempt to use TFO in case of a CONNECT command, or accept TFO in case of a BIND command.
- o Address Type:
 - * 0x01: IPv4
 - * 0x03: Domain Name
 - * 0x04: IPv6
- o Address: this field's format depends on the address type:
 - * IPv4: a 4-byte IPv4 address
 - * Domain Name: one byte that contains the length of the FQDN, followed by the FQDN itself. The string is not NUL-terminated.
 - * IPv6: a 16-byte IPv6 address
- o Port: the port in network byte order.
- o Number of Options: the number of SOCKS options that appear in the Options field.
- o Options: see section Section 5.
- o Initial Data Size: A two-byte number in network byte order. In case of AUTH, BIND or UDP ASSOCIATE, this field MUST be set to 0. In case of CONNECT, this is the number of bytes of initial data that are supplied in the following field.
- o Initial Data: The first octets of the data stream.

Clients MUST support the "No authentication required" method. Clients MAY omit advertising the "No authentication required" option.

Clients SHOULD NOT issue AUTH commands unless they advertise authentication methods with support for 0-RTT authentication.

The server MAY truncate the initial data to an arbitrary size and disregard the rest.

5. SOCKS Options

SOCKS options have the following format:

Kind	Length	Option Data
1	1	Variable

Figure 3: SOCKS 6 Option

- o Kind: MUST be allocated by IANA. (See section Section 9.)
- o Length: The length of the option data.
- o Option Data: the contents are specific to each option kind.

5.1. Authentication options

Authentication options have the following format:

Kind	Length	Method	Authentication Data
1	1	1	Variable

Figure 4: Authentication Option

- o Kind: MUST be allocated by IANA. (See section Section 9.)
- o Length: the length of the option data.
- o Method: the number of the authentication method. These numbers are assigned by IANA.

- o Authentication Data: the contents are specific to each method.

All proxy implementations MUST support authentication method options. Clients MAY omit advertising authentication methods for which they have included at least an authentication option.

6. Authentication Replies

Upon receipt of a request, the proxy sends an Authentication Reply:

Version		Type	Method	Number of	Options
Major	Minor			Options	
1	1	1	1	1	Variable

Figure 5: SOCKS 6 Authentication Reply

- o Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.
- o Type:
 - * 0x00: authentication successful.
 - * 0x01: further authentication needed.
- o Method: The chosen authentication method.
- o Number of Options: the number of SOCKS options that appear in the Options field.
- o Options: see section Section 5.

Multihomed clients SHOULD cache the chosen method on a per-interface basis and SHOULD NOT include authentication options related to any other methods in further requests originating from the same interface.

If the server signals that further authentication is needed and selects "No Acceptable Methods", the client MUST close the connection.

The client and proxy begin a method-specific negotiation. During such negotiations, the proxy MAY supply information that allows the client to authenticate a future request using an authentication

option. Descriptions of such negotiations are beyond the scope of this memo.

If the cliend issued an AUTH command, the client MUST close the connection after the negotiation is complete.

7. Operation Replies

After the authentication negotiations are complete, the server sends an Operation Reply:

Version		Reply	Address	Bind	Bind
Major	Minor	Code	Type	Address	Port
1	1	1	1	Variable	2

Number of	Options	Initial Data
Options		Offset
1	Variable	2

Figure 6: SOCKS 6 Operation Reply

- o Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.
- o Reply Code:
 - * 0x00: Succes
 - * 0x01: General SOCKS server failure
 - * 0x02: Connection not allowed by ruleset
 - * 0x03: Network unreachable
 - * 0x04: Host unreachable
 - * 0x05: Connection refused
 - * 0x06: TTL expired
 - * 0x07: Command not supported

- * 0x08: Address type not supported
- o Address Type:
 - * 0x01: IPv4
 - * 0x03: Domain Name
 - * 0x04: IPv6
- o Bind Address: the proxy bound address in the following format:
 - * IPv4: a 4-byte IPv4 address
 - * Domain Name: one byte that contains the length of the FQDN, followed by the FQDN itself. The string is not NUL-terminated.
 - * IPv6: a 16-byte IPv6 address
- o Bind Port: the proxy bound port in network byte order.
- o Number of Options: the number of SOCKS options that appear in the Options field.
- o Options: see section Section 5
- o Initial Data Offset: A two-byte number in network byte order. In case of BIND or UDP ASSOCIATE, this field MUST be set to 0. In case of CONNECT, it represents the offset in the plain data stream from which the client is expected to continue sending data.

If the proxy returns a reply code other than "Success", the client MUST close the connection.

7.1. Handling CONNECT

In case the client has issued a CONNECT request, data can now pass. The client MUST resume the data stream at the offset indicated by the Initial Data Offset field.

7.2. Handling BIND

In case the client has issued a BIND request, it must wait for a second Operation reply from the proxy, which signifies that a host has connected to the bound port. The Bind Address and Bind Port fields contain the address and port of the connecting host. Afterwards, application data may pass.

7.3. Handling UDP ASSOCIATE

The relay of UDP packets is handled exactly as in SOCKS 5 [RFC1928].

8. Security Considerations

Given the format of the request message, a malicious client could craft a request that is in excess of 100 KB and proxies could be prone to DDoS attacks.

To mitigate such attacks, proxy implementations SHOULD be able to incrementally parse the requests. Proxies MAY close the connection to the client if:

- o the request is not fully received after a certain timeout, or
- o the number of options exceeds an imposed hard cap, or
- o the total size of the options exceeds an imposed hard cap, or
- o the size of the initial data exceeds a hard cap.

Further, the server MAY choose not to buffer any initial data beyond what would fit in a TFO SYN's payload.

9. IANA Considerations

This document requests that IANA allocate option codes for SOCKS 6 options. Further, this document requests an option code for authentication options.

10. Acknowledgements

The protocol described in this draft builds upon and is a direct continuation of SOCKS 5 [RFC1928].

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

11.2. Informative References

- [I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-20 (work in progress), April 2017.
- [RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, DOI 10.17487/RFC1928, March 1996, <<http://www.rfc-editor.org/info/rfc1928>>.
- [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <<http://www.rfc-editor.org/info/rfc6824>>.
- [RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <<http://www.rfc-editor.org/info/rfc7413>>.

Authors' Addresses

Vladimir Olteanu
University Politehnica of Bucharest

Email: vladimir.olteanu@cs.pub.ro

Dragos Niculescu
University Politehnica of Bucharest

Email: dragos.niculescu@cs.pub.ro