Network Working Group Y. Jia
Internet-Draft Z. Chen
Intended status: Standards Track S. Jiang
Expires: 4 May 2021 Huawei
31 October 2020

Flexible IP: An Adaptable IP Address Structure
draft-jia-flex-ip-address-structure-00

Abstract

Along as the popularization and adoption of IP in emerging scenarios, challenges emerge as well due to the ossified address structure. To enable TCP/IP in networks that previously using exclusive protocol, a flexible address structure would be far more preferred for their particular properties [draft-jia-scenarios-flexible-address-structure].

This document describes a flexible address structure -- Flexible IP (FlexIP) acting on limited domains [RFC8799]. FlexIP is expected to proactively adapt scenarios under flexible address structure. Meanwhile, FlexIP still benefit from global reachability based on the IPv6 interoperability.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

As Internet Protocol (IP) gradually turned into the "waist" of the "TCP/IP" protocol stack, it is considered to be the core pillar of the entire Internet [waist]. Although numerous technologies in this "TCP/IP" protocol stack have emerged, evolved, or obsoleted by others, the IPv6 technology [RFC8200] is the only forward in network layer along with the Internet upgrades. IPv6, as the unique successor of IPv4 [RFC0791] defined by IETF, fixes defects for most of its parts. Most notably, the address space is enormously expanded from 32-bit to 128-bit in IPv6 reformation. Despite that IPv6 is expected to serve almost infinite devices in the foreseeable future, several scenarios are found in trouble when IPv6 is in use.

For instance, due to the market and cost requirements, numerous Internet-of-things (IoTs) are devised to be tiny and resource constrained. However, such rigorous requirement induce manufactures to adopt lightweight protocols like bluetooth, other than TCP/IP, for inter communications [iot]. Energy consumption, which is sound in most terminal cases, becomes the greatest challenge when introducing IPv6 to IoTs. Document [draft-jia-scenarios-flexible-address-structure] details the challenges for more cases of IPv6.

To conquer these challenges, several improvements are promoted by the corporation of related standard organizations. Document [RFC4944] depict the transmission of IPv6 over IEEE 802.15.4 network, and such a method enables indirectly support of IPv6 in IoTs, e.g., Thead Protocol [thread]. Besides, document [RFC7668] describe the fusion of IPv6 and Bluetooth Low Energy, and such a method also enable the communications among nodes with Bluetooth and IPv6. Although none of these proposal is superior on the basis of market sharing, all endeavour orientate to the comprehensive adoption of TCP/IP protocol stack in new communication cases.

This document proposes an adaptive IP address format -- Flexible IP (abbr. FlexIP) orienting emerging scenarios [draft-jia-scenarios-flexible-address-structure] within limited domains [RFC8799], and maintain global reachability with IPv6. In general, FlexIP is composed through a hierarchical, self-explanatory address structure. Compared to the patch-like solutions (e.g., [RFC4944], [RFC8724]) that passively tune IP to be compatible with different scenarios, FlexIP proactively makes address structure flexible enough to adapt to various network cases. This variation, opposite to the fixed form of IPv4/IPv6 address, is expected to make Internet Protocol agilely cover futuristic and unknown scenarios.

//TODO: more citations needed.

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Targeted Scenario

As a architectural solution for scenarios detailed in [draft-jia-scenarios-flexible-address-structure], FlexIP is expected to be adopted in limited domains [RFC8799]. According to the definition in [RFC8799], limited domain refers to a single physical network attached to or running in parallel with the Internet, or a defined set of users and nodes distributed over a much wider area, but drawn together by a single virtual network over the Internet. Within the limited domains, requirements, behaviors, and semantics could be noticeable local and network specific.

Figure 1 depicts the orientation of FlexIP in IPv6 and Internet. Generally, Internet with its backbone is principally composed by IPv6, and networks with IPv4 are recognized as legacy and attached at the edge of the Internet with transition mechanisms. The position of networks with FlexIP structure is quite similar as IPv4. For networks within limited domains, FlexIP can be marginally adopted at the edge of the Internet. Such network use FlexIP to fulfill proprietary capabilities, while retain global reachability through IPv6 via transition.

```
               .........
               .........  ->Attaching Point
               ||||||||| /
               ||||||+--+               +-----------------+
               ||<----------------------+ Google Network  |
               ||||||+--+               |      IPv6       |
               |||||||||                +-----------------+
               |||||||||      ->Translator
               |||||||||     /
               ||||||+--+  /-\          +-----------------+
               ||||||<------------------+ Legacy Network  |
               ||||||+--+  \-/          |      IPv4       |
               |||||||||                +-----------------+
               |||||||||
               |||||||||
               ||||||+--+            +--------------------+
               |<--------------------+ Microsoft Network  |
               ||||||+--+            |       IPv6         |
               |||||||||             +--------------------+
               |||||||||
               |||||||||
               ||||||+--+               +-----------------+
               |||<---------------------+ Amazon Network  |
               ||||||+--+               |      IPv6       |
               |||||||||                +-----------------+
               |||||||||
               |||||||||                          ... ...
               |||||||||       ->Translator
               |||||||||      /
               ||||||+--+  /-\  +-------------------------+
               ||||<-------------+ Limited Domain Network |
               ||||||+--+  \-/  |          FlexIP         |
               |||||||||         +------------------------+
               |||||||||              e.g., factory network
               |||||||||
               |||||||||
               |||||||........IPv6 Internet Backbone
               |||||||||
               .........
```

Figure 1: Position for FlexIP in IPv6 Internet

4. Design Considerations

As described in document [draft-jia-scenarios-flexible-address-structure], various address semantics and variable address length are main characteristics for a flexible IP. However, several principles must be considered from the top if such featured address structure become truly practicable. Points below details overall considerations and plannings behind FlexIP.

4.1. Multi-Semantics

For networks with advanced routing, non-topological identifiers could be necessary for reachability [RFC7476]. To enrich routing abilities in complex network, address structure should be flexible enough to accommodate multiple semantics and related identifiers, thus forwarding nodes within the network can dealing with these complex address according based on the routing system built.

Ideally, devices that resided in advanced networks construct special address format for customized routing, while devices that resided in conventional scenario can adopt IPv6 as routine (topology semantic) address.

4.2. Elastic Address Space

The primary orientation of FlexIP is to adapt different network scale, and each network uses different address length that best fit the presumptive scale. The alterable address length refers to a flexible address space, and such resilience makes IP highly adaptable to theoretically infinite space as well as tailored space specifically for low power scenarios.

Ideally, Autonomous System (AS) that composing the Internet is constituted by numerous networks. Each of the network hold the flexible address space that best fit their scenarios.

4.3. Scalability

A constrained space is impossible to accommodate communications among volume devices, while boundless space is unsustainable due to the explosion of global routing table. To makes the flexibility truly values, FlexIP must reach a balance between rigorous requirements of expansive address space and efficient routing performance.

Ideally, the address should be expanded to boundless space, while the structure guarantees the performance of fast forwarding.

4.4. Interoperability

FlexIP network is designed to be an attached network at edges of the Internet, thus interoperability is needed between IPv6 and FlexIP. Based on the interoperability, FlexIP, on the one hand, can benefit from the advanced network abilities and adapt to new scenarios. On the other hand, global reachability from IPv6 Internet makes the network truly values and convenient for realistic use.

Ideally, a translator module is needed wherever a boundary across FlexIP networks and IPv6 networks. The translator depicted in Figure 1 gives a brief architectural instance for interoperability.

5. FlexIP Address structure

In general, FlexIP is composed through a hierarchical, self-explanatory address structure. Such hierarchical, self-explanatory address structure brings FlexIP elastic address space and multi-semantics without sacrificing scalability. Table 1 details the structure of the FlexIP address structure.

Index	Type	Structure (default by topology semantic and 1 segment)
0x01	Restrained Space	topology address - address 1
0x02	Restrained Space	topology address - address 2
...	...	...
0xEF	Restrained Space	topology address - address 239
0xF0	Extendable Space	followed by address with 16-bit length
0xF1	Extendable Space	followed by address with 32-bit length
0xF2	Extendable Space	followed by address with 64-bit length
0xF3	Extendable Space	followed by address with 128-bit length

0xF4	Extendable Space	followed by address with 256-bit length
0xF5	Extendable Space	followed by address with X-bit length
0xF6	Hierarchical Segments	followed by address with 2 segments
0xF7	Hierarchical Segments	followed by address with 3 segments
0xF8	Hierarchical Segments	followed by address with Y segments
0xF9	Multi-Semantics	followed by Non-topological semantic address
0xFA - 0xFF	None	reserved

Table 1: Flexible IP Address Structure

Shapes in Table 1 describes the formal representation of FlexIP and should be used in programing. Text representation of FlexIP is described in Section 6. Particularly, formal representation of FlexIP in this document introduces "/" for readability only. Such "/" MUST be omitted in practical use.

5.1. Restrained Space Format

The first address format is called restrained space format and specific for small address space. Such format includes a 1-byte space customized for constrained resource devices. Structure in such format guarantees that within FlexIP structure, devices can reach the shortest address length under variable length structure and pursuit the maximal routing efficiency.

5.2. Extendable Space Format

The second address format is called extendable space format. By adopting such format, administrator can choose address length that best fit the network.

Specifically, for networks larger than 239 address space, a 16-bit, 32-bit, 64-bit, 128-bit, and 256-bit can be used by the network with Index F0-F4, respectively, and then followed by address itself. Particular, a IPv6 (128-bit) address is regarded as a special indexed by Index F3.

For networks prefer a customized space, a 1-byte LenIndex is emerged between Index and the address. Structure in such format ensures that address space becomes theoretically elastic and boundless. For example, a 56-bit address is presented by F5/07/3B3A297F50C24F under FlexIP structure. Sequence value 07 refers to the 7-byte (56-bit) address length.

5.3. Hierarchical Segments Format

The third address format is called hierarchical segments format. By adopting such format, an FlexIP address is composed by multiple segments. Logically, a segment inside the address could be considered as an individual routing identifier. Thus within different routing areas, routers on the path should forward packets based on the exclusive segment. The partitioning of the address logically splits the large address into several routing segment, and segments are regarded small enough that packets flowing in separate networks could be forwarded efficiently according to the segments. For this reason, structure in such hierarchy format ensures the practicability with boundless space.

Taking an 2-segment address as an example, FlexIP F6/C8/F1/20C12AF2 present a 8-bit segment C8 and a 32-bit segment 220C12AF2, where index F6 indicates the 2 segments behind.

5.4. Multi-Semantics Format

The forth address format is called multi-semantics format. For address adopting such format, networks can forward packets based on the specific semantics.

Under such format, a 1 byte SemIndex is used as the indication of semantic when Index equal to F9. Taking the satellite network [space-routing] as an example, FlexIP F9/01/F2/A32F84C981002E9B can refer to a geographic position embedded address, where 01 refers to the geographic semantic, and F2 refers to a 64-bit address length. Similarly, such pattern can be used for name based routing [ndn], user based routing, or service based routing.

Given that non-topological semantics and addressing are still under open study, specifications for non-topological semantics will be depicted in independent documents when technics become mature.

6. FlexIP Address Text Representation

Literally, text representation of FlexIP should be human friendly compared to the formal representation in Section 5. Considering text representation would be used in extensive written places, FlexIP is such representation should be eminently readable.

This document RECOMMENDED text representation of FlexIP through following structure:
[Length]<SEM>_Value_[Length]<SEM>_Value_...[Length]<SEM>_Value_.
Generally, FlexIP address is concatenated directly by multiple segments. For each of the segment, the text representation is composed by [Length]<SEM>_Value_. Specifically, i.e., for components inside an segment, [Length] refers to the length of current segment with the Byte unit;
Then followed by <SEM>, <SEM> refers to the semantic the segment use. Within a segment, <SEM> is the only field can be omitted if segment points to the default topology semantic -- <TOPO>. Last, _Value_ refers to the address itself. Particularly, _Value_ inherits the same text representation as IPv6 that recommended in [RFC5952].

Table 2 depicts examples for FlexIP representation in text shape. Noted that "/" in the formal representation is for readability only and MUST be removed in practice.

Formal Representation	Text Representation
C8	[1]C8
F1/2A00012F	[4]2A::12F
F5/07/3B3A297F50C24F	[7]3B:3A29:7F50:C24F
F6/C8/F2/2001000000012F	[1]C8[8]2001::12F
F8/04/F0/2F5B/F0/6A3C/F0/9C2B/ F0/735D	[2]2F5B[2]6A3C[2]9C2B[2]735D
F9/01/F2/A32F84C981002E9B	[8]<GEO>A32F84C981002E9B

Table 2: Examples of Flexible IP Address Text Representation

For example, [1]C8[8]2001::12F indicates two segments concatenation: segment C8 with <TOPO> semantic and 1 byte length, and segment 200100000000012F with <TOPO> semantic and 8 byte length. Particular, given that non-topological semantics and addressing are still under open study, <SEM> identification should be officially maintained in IANA.

7. Interoperability

To enable global reachability and inter connectivity between FlexIP network and IPv6 network, an translator is needed wherever packets across the periphery. Figure 2 depicts the core component of the translator, i.e., address mapper, and a sketch for packet traversing from a FlexIP network to a IPv6 network. For any packet leave FlexIP network and enter IPv6 network, the IP addresses of the packet should be converted by rules configured in the mapper, and vice versa.

```
       .........       ->Translator
       |.|.|.|.|      /
       |.|.|.+---+  /-\  +------------------------+
       |.|.<-------------+ Limited Domain Network |
       |.|.|.+---+  \-/  |         FlexIP         |
       |.|.|.|.|     |   +------------------------+
       |.|.|.|.|     |
       |.|.|.|.|     |
       |.|.|.|.|     |
       |.|.|.|.|     |            +-------------+    Rules
       |.|.|.|.|     |          / |             | Distribution
       |.|.|.|.|     +---------|  | +---------+ |      +
       .........                \ | | Mapping | |      |
                                  | |  Rules  <--------+
       Internet                   | +----+----+ |
       Backbone                   |      |      |
                        Packet    | +----v----+ |    Packet
                      +--------+  | | Mapping | |  +--------+
                      |  IPv6  <----+  Engine <----+ FlexIP |
                      +--------+  | +---------+ |  +--------+
                      |  Data  |  |             |  |  Data  |
                      |        |  |   Address   |  |        |
                      +--------+  |    Mapper   |  +--------+
                                  +-------------+
```

Figure 2: Network Address Mapping between FlexIP network and IPv6 network.

Specifically, there are two kind of mapping policy: stateful recording and stateless transforming. Although both two policy is effective for address mapping, stateless transforming is RECOMMENDED for system efficiency and operation complexity. Concrete processes of rules generation, distribution and mapping mechanisms are outside the scope of this specification and should be documented individually.

For FlexIP network with restrained space format Table 1, for instance, a FlexIP C3 should be mapped into IPv6 2001::C3 when affiliated packet flow across the periphery. Conversely, address mapper can simply peel of the prefix 2001:: of when packets flow back to FlexIP network.

8. Security Considerations

As a address format of IP, FlexIP address itself do not involve security issues. While from the viewpoint of the transmission of packets, FlexIP has security properties that are similar to IPv6. These security issues include:

* Eavesdropping, where on-path elements can observe the whole packet (including both contents and metadata) of each datagram.

* Replay, where the attacker records a sequence of packets off of the wire and plays them back to the party that originally received them.

* Packet insertion, where the attacker forges a packet with some chosen set of properties and injects it into the network.

* Packet deletion, where the attacker removes a packet from the wire.

* Packet modification, where the attacker removes a packet from the wire, modifies it, and reinjects it into the network.

* Man-in-the-middle (MITM) attacks, where the attacker subverts the communication stream in order to pose as the sender to receiver and the receiver to the sender.

* Denial-of-service (DoS) attacks, where the attacker sends large amounts of legitimate traffic to a destination to overwhelm it.ss

Specifically, there is not any mechanism for FlexIP to protect against IP spoofing. Defending against these type of attacks is outside the scope of this specification.

9. IANA Considerations

This document does not include an IANA request.

10. Informative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

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

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <https://www.rfc-editor.org/info/rfc4944>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/info/rfc5952>.

[RFC7476] Pentikousis, K., Ed., Ohlman, B., Corujo, D., Boggia, G., Tyson, G., Davies, E., Molinaro, A., and S. Eum, "Information-Centric Networking: Baseline Scenarios", RFC 7476, DOI 10.17487/RFC7476, March 2015, <https://www.rfc-editor.org/info/rfc7476>.

[RFC7668] Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., and C. Gomez, "IPv6 over BLUETOOTH(R) Low Energy", RFC 7668, DOI 10.17487/RFC7668, October 2015, <https://www.rfc-editor.org/info/rfc7668>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8724] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zúñiga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <https://www.rfc-editor.org/info/rfc8724>.

[RFC8799] Carpenter, B. and B. Liu, "Limited Domains and Internet Protocols", RFC 8799, DOI 10.17487/RFC8799, July 2020, <https://www.rfc-editor.org/info/rfc8799>.

[draft-jia-scenarios-flexible-address-structure]
Jia, Y., Li, G., and S. Jiang, "Scenarios and Requirements for Flexible Address structure", October 2020, <https://xml2rfc.ietf.org/public/rfc/bibxml-ids/reference.I-D.draft-jia-scenarios-flexible-address-structure.xml>.

[iot] Jara, AJ., Ladid, L., and AF. Gomez-Skarmeta, "The Internet of Everything through IPv6: An Analysis of Challenges, Solutions and Opportunities", Networks Ubiquitous Comput. Dependable Appl. 4(3): 97-118, 2013.

[ndn] Zhang, L., Afanasyev, A., and J. Burke, "Named data networking", ACM SIGCOMM Computer Communication Review 44(3): 66-73, 2014.

[space-routing]
Yang, Z., Li, H., Wu, Q., and J. Wu, "Analyzing and optimizing BGP stability in future space-based internet", International Performance Computing and Communications Conference (IPCCC) , December 2017.

[thread] Thread Group, "Thread Specification", <https://www.threadgroup.org/ThreadSpec>.

[waist] Akhshabi, S. and C. Dovrolis, "The Evolution of Layered Protocol Stacks Leads to an Hourglass-shaped Architecture", Proceedings of the ACM SIGCOMM 2011 Conference 206-217, October 2011, <https://dl.acm.org/doi/abs/10.1145/2018436.2018460>.

Authors' Addresses

Yihao Jia
Huawei Technologies Co., Ltd
156 Beiqing Rd.
Haidian, Beijing
100095
P.R. China

Email: jiayihao@huawei.com

Zhe Chen
Huawei Technologies Co., Ltd
156 Beiqing Rd.
Haidian, Beijing
100095
P.R. China

Email: chenzhe17@huawei.com

Sheng Jiang
Huawei Technologies Co., Ltd
156 Beiqing Rd.
Haidian, Beijing
100095
P.R. China

Email: jiangsheng@huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 4 May 2021

Y. Jia
G. Li
S. Jiang
Huawei

31 October 2020

Scenarios for Flexible Address Structure
draft-jia-scenarios-flexible-address-structure-00

Abstract

Along as the adoption of TCP/IP in increasingly emerging scenarios, challenges emerge as well due to the ossified address structure. To still enable TCP/IP for networks that previously using exclusive protocols, a flexible address structure would be highly preferred for their particular properties. This document describes well-recognized scenarios that typically require a flexible address structure, and states the requirements of such flexible address structure.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

As the unified protocol of the network layer, Internet Protocol (IP) constantly promotes the prosperity of the entire Internet. With the success of TCP/IP protocol stack, IP is gradually replacing exclusive protocols and becomes the core protocol of the entire communication system.

Along as the popularization of TCP/IP, increasingly scenarios long for a flexible address structure. To fulfill the reachability, IP address is designed to hold the topology semantic only [RFC0791]. While within limited domains (ref. [RFC8799]), a multi-semantic address could be increasingly preferred in implementing complex actions and capabilities for particular scenarios. Under such circumstances, a flexible address structure can unleash more possibilities in serving new scenarios.

This document describes well-recognized scenarios that typically require a flexible address structure, and states the requirements of such flexible address structure.

2. Flexibility: Potential Orientation

Since a flexible address is expected be adaptive with different scenarios and routing abilities, potentially orientations of a flexible address structure should include multiple semantics. According to the definition of IP structure [RFC1180], cyberspace topology location serves as the only semantic of IP address. While for emerging scenarios in reality, several semantics are expected for reachability, e.g., contents [CONTENT-NET] or names [ndn]. To accommodate growing requirements of futuristic scenarios, address with multi-semantic embedding should compose the core of the flexibility. With the dynamic semantic embedding, the length of the address should accordingly adaptive.

3. Scenarios

Although new scenarios are ever changing, they principally belong to a fixed scene, i.e., limited domain [RFC8799]. Limited domain, which first defined in [RFC8799], refers to a single physical network attached to or running in parallel with the Internet, or a defined set of users and nodes distributed over a much wider area, but drawn together by a single virtual network over the Internet. Within the limited domains, requirements, behaviors, and semantics could be noticeable local and network specific.

As the dominant status of IP in networking coverage, more networks attempt to adopt TCP/IP in their local networks. Instead of building proprietary network, a TCP/IP stack network facilitates convenient reachability via global Internet and reduces operating expense for exclusive protocols. According to the location that the retrofit of address structure taking effect, targeted scenes perfectly match the demarcation of limited domain, i.e., a edge network attaching to Internet. Thus for networks that seeking for IP convenience but enhanced capabilities, limited domains are scenarios that flexible address structure taking effect.

3.1. Internet of Things (IoTs)

In many IoT scenarios, a simple, low-cost communication network is required, and there are limitations for network devices in computational power, memory, and energy availability. In addition to [IEEE.802.15.4], it can be seen that networks using link layer technologies such as Z-Wave, BLE [BLE], DECT_ULE, MS/TP, NFC, and PLC require end-to-end IPv6 protocols [RFC8200] to run on constrained node networks. The IP protocol allows IoT devices with multiple connection types to connect to each other or to other nodes on the Internet. Generally, a group of IoT network devices form a constrained node network at the edge, and IoT terminals connect to these network devices for data transmission. Devices located on the edge of this network and the Internet can act as gateway devices. To ensure security and reliability, multiple gateways must be deployed. IoT devices on the network can easily select one of gateways for traffic to pass through. This type of network and IoT devices in the network require as little computational power as possible, smaller memory requirements, and better energy availability to reduce the total cost of ownership of the network.

3.2. Satellite Network

In the future, the space-based Internet will provide global Internet connections through satellite and station on the ground interconnection. The low cost of satellite launch and the reduction of the cost of network equipment will promote the development of high-density satellite networks. With the convergence of space-based networks and terrestrial networks, users can experience seamless broadband access. Whatever on cruise ships, flights, and cars, users can switch data communication services over Wi-Fi, cellular, or satellite networks at any time. The network service provider will plan the transmission path of user traffic based on the network coverage, satellite orbit, route, and link load. The advantage of long-distance transmission but shorter delay of space-based networks is fully used to provide high-quality Internet connections for users in areas not covered by terrestrial networks. There is a significant difference between the high dynamics of satellite network and the statics of terrestrial network topology. The traditional satellite network cannot meet the preceding requirements through the networking of the dedicated station on the ground.

3.3. Dynamic Service and Resource

In the future, the network will integrate services and resources from various aspects such as life, production, and learning. Digitalized services and resources are divided into multiple data blocks on the network and multiple copies of data blocks exist, which will become the basic mode. Access to services and resources through URIs has been discussed by many researches, such as NDN [ndn] and ICN [RFC7476]. In practice, the dynamic services and resource management and access mechanisms of integrating ID and address technologies will be more suited to user needs. Providers of services and resources can publish online services and resources through unified identifiers without additional planning of identifiers and locations for data and their replicas. Users can access required services and resources in the nearest and on-demand mode. Further, users can make a request based on the type of service and resource and get a response to the service or a copy of the data.

3.4. Policy-based Traffic Control

Policy-based traffic control is constantly far from flexible. To restrict traffics for specific objects, e.g., devices, users, or group of them, a current solution is subnet partition. Representative technique for subnet partition includes VLAN [RFC5517] and VxLAN [RFC7348]. However, such mechanism usually involve numerous manual configuration, and even small changes in reality could also result in a repartition or manual efforts.

According to the semantic of IP address forwarding, any inconvenience of traffic control stems from the decoupling of address semantic and policy objects. Since address content only present topology location in IPv6, extra out-of-band effort is needed to partition network or recognize traffic from target objects.

For address with objects identifier encoded, policy-based traffic control could be almost automatic. For every node forwarding traffic, object identity could be first extracted from both source and destination address once packets arrive. Then by matching objects and policy-based rules, nodes on the path could trigger particular actions that dynamically assigned by administrators. For examples, action permit, action deny, and action permit but low priority. Particular, such action could also be applied from security restriction.

3.5. Robust Trust and Security

A flexible semantic could be significate in constructing a trust and secure communication. For example, Cryptographically Generated Addresses (CGA) [RFC3972] embeds a truncated public key in the last 57-bit of IPv6 address. Even with a truncated key, authentication and security is greatly enhanced within a IP network via asymmetric cryptography and IPsec [RFC4301]. Similarly, Host Identity Protocol (HIP) [RFC7401] refers such methodology and constructs a enhanced TCP/IP stack.

Within a flexible address structure, any secure-related keys could be intactly included in address structure without any information loss. Under such condition, connection provided by such address could be considered as absolute secure only if the cryptography involved is secure.

4. Requirements

According to the capabilities for scenarios above, this section extracts basic requirements for a flexible address structure. Points below details the basal requirements.

* Multi-Semantics: Since semantics are the core of network routing, multi-semantics compose the main capability of the flexible address.

* Variable Address Length: As for networks with constrained devices, short address becomes necessary for protocol adoption. While on other hand, address space should be adequate enough to accommodate numerous devices.

* IPv6 Interoperability: Without global reachability, a flexible address would be the same as an exclusive protocol. In other words, a flexible address should be valuable only it is inter-operable with IPv6.

5. Security Considerations

This document introduces no new security considerations.

6. IANA Considerations

This document does not include an IANA request.

7. Informative References

[BLE] Bluetooth SIG Working Groups, "Bluetooth Specification", <https://www.bluetooth.com/specifications/>.

[CONTENT-NET] Choi, J., Han, J., and E. Cho, "A survey on content-oriented networking for efficient content delivery", IEEE Communications Magazine 2011, 49(3): 121-127, May 2011.

[IEEE.802.15.4] IEEE 802.15 WPAN Task Group 4, "IEEE 802.15.4 - IEEE Standard for Low-Rate Wireless Networks", May 2020, <https://standards.ieee.org/standard/802_15_4-2020.html>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC1180] Socolofsky, T. and C. Kale, "TCP/IP tutorial", RFC 1180, DOI 10.17487/RFC1180, January 1991, <https://www.rfc-editor.org/info/rfc1180>.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, DOI 10.17487/RFC3972, March 2005, <https://www.rfc-editor.org/info/rfc3972>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC5517] HomChaudhuri, S. and M. Foschiano, "Cisco Systems' Private VLANs: Scalable Security in a Multi-Client Environment", RFC 5517, DOI 10.17487/RFC5517, February 2010, <https://www.rfc-editor.org/info/rfc5517>.

[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, <https://www.rfc-editor.org/info/rfc7348>.

[RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <https://www.rfc-editor.org/info/rfc7401>.

[RFC7476] Pentikousis, K., Ed., Ohlman, B., Corujo, D., Boggia, G., Tyson, G., Davies, E., Molinaro, A., and S. Eum, "Information-Centric Networking: Baseline Scenarios", RFC 7476, DOI 10.17487/RFC7476, March 2015, <https://www.rfc-editor.org/info/rfc7476>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8799] Carpenter, B. and B. Liu, "Limited Domains and Internet Protocols", RFC 8799, DOI 10.17487/RFC8799, July 2020, <https://www.rfc-editor.org/info/rfc8799>.

[ndn] Zhang, L., Afanasyev, A., and J. Burke, "Named data networking", ACM SIGCOMM Computer Communication Review 44(3): 66-73, 2014.

Authors' Addresses

Yihao Jia
Huawei Technologies Co., Ltd
156 Beiqing Rd.
Haidian, Beijing
100095
P.R. China

Email: jiayihao@huawei.com

Guangpeng Li
Huawei Technologies Co., Ltd
156 Beiqing Rd.
Haidian, Beijing
100095
P.R. China

Email: liguangpeng@huawei.com

Sheng Jiang
Huawei Technologies Co., Ltd
156 Beiqing Rd.
Haidian, Beijing
100095
P.R. China

Email: jiangsheng@huawei.com

Internet Area Working Group V. Olteanu
Internet-Draft D. Niculescu
Intended status: Standards Track University Politehnica of Bucharest
Expires: May 6, 2021 November 02, 2020

SOCKS Protocol Version 6
draft-olteanu-intarea-socks-6-11

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

Copyright Notice

Table of Contents

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 [RFC8446] 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:

- 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.

- The connection request also mimics the semantics of TCP Fast Open [RFC7413]. As part of the connection request, the client can supply the potential payload for the initial SYN that is sent out to the server.

- The protocol can be extended via options without breaking backward-compatibility.

- The protocol can leverage the aforementioned options to support 0-RTT authentication schemes.

1.1. Revision log

Typos and minor clarifications are not listed.

draft-11

- Changed intended status to Standards Track

- Renamed Vendor-specific option range to Experimental

- Stack options:

  * Fixed some instances where an unsupported option was indistinguishable from a case where the proxy couldn't or wouldn't honor it (offenders: Happy Eyeballs, IP Fragmentation, UDP Error, Port Parity)

  * MPTCP: changed semantics w.r.t. TCP BIND: the absence of such an option SHOULD no longer lead the proxy to refuse MPTCP

  * Port Parity: relaxed restrictions in case the client supplies a specific port

draft-10

- Removed untrusted sessions

- IP DF

- UDP relay:

  * Support ICMPv6 Too Big

  * Shifted some fields in the error messages

  * RTP support

draft-09

- o Revamped UDP relay

  * Support for ICMP errors: host/net unreachable, TTL exceeded

  * Datagrams can be sent over TCP

  * Timeout for the receipt of the initial datagram

- o TTL stack option (intended use: traceroute)

- o Added the "Privacy Considerations" section

- o SOCKS-provided DNS: the proxy may provide a valid bind address and port

draft-08

- o Removed Address Resolution options

- o Happy Eyeballs options

- o DNS provided by SOCKS

draft-07

- o All fields are now aligned.

- o Eliminated version minors

- o Lots of changes to options

  * 2-byte option kinds

  * Flattened option kinds/types/reply codes; also renamed some options

  * Socket options

    + Proxies MUST always answer them (Clients can probe for support)

    + MPTCP Options: expanded functionality ("please do/don't do MPTCP on my behalf")

    + MPTCP Scheduler options removed

      + Listen Backlog options: code changed to 0x03

   * Revamped Idempotence options

   * Auth data options limited to one per method

- Authentication Reply: all authentication-related information is now in the options

   * Authentication replies no longer have a field indicating the chosen auth. method

   * Method that must proceed (or whereby authentication succeeded) indicated in options

   * Username/password authentication: proxy now sends reply in option

- Removed requirements w.r.t. caching authentication methods by multihomed clients

- UDP: 8-byte association IDs

- Sessions

   * The proxy is now free to terminate ongoing connections along with the session.

   * The session-terminating request is not part of the session that it terminated.

- Address Resolution options

draft-06

- Session options

- Options now have a 2-byte length field.

- Stack options

   * Stack options can no longer contain duplicate information.

   * TFO: Better payload size semantics

   * TOS: Added missing code field.

   * MPTCP Scheduler options:

+ Removed support for round-robin

+ "Default" renamed to "Lowest latency first"

* Listen Backlog options: now tied to sessions, instead of an authenticated user

- Idempotence options

  * Now used in the context of a session (no longer tied to an authenticated user)

  * Idempotence options have a different codepoint: 0x05. (Was 0x04.)

  * Clarified that implementations that support Idempotence Options must support all Idempotence Option Types.

  * Shifted Idempotence Option Types by 1. (Makes implementation easier.)

- Shrunk vendor-specific option range to 32 (down from 64).

- Removed reference to dropping initial data. (It could no longer be done as of -05.)

- Initial data size capped at 16KB.

- Application data is never encrypted by SOCKS 6. (It can still be encrypted by the TLS layer under SOCKS.)

- Messages now carry the total length of the options, rather than the number of options. Limited options length to 16KB.

- Security Considerations

  * Updated the section to reflect the smaller maximum message size.

  * Added a subsection on resource exhaustion.

draft-05

- Limited the "slow" authentication negotiations to one (and Authentication Replies to 2)

- Revamped the handling of the first bytes in the application data stream

   * False starts are now recommended. (Added the "False Start" section.)

   * Initial data is only available to clients willing to do "slow" authentication. Moved the "Initial data size" field from Requests to Authentication Method options.

   * Initial data size capped at 2^13. Initial data can no longer be dropped by the proxy.

   * The TFO option can hint at the desired SYN payload size.

- Request: clarified the meaning of the Address and Port fields.

- Better reverse TCP proxy support: optional listen backlog for TCP BIND

- TFO options can no longer be placed inside Operation Replies.

- IP TOS stack option

- Suggested a range for vendor-specific options.

- Revamped UDP functionality

   * Now using fixed UDP ports

   * DTLS support

- Stack options: renamed Proxy-Server leg to Proxy-Remote leg

draft-04

- Moved Token Expenditure Replies to the Authentication Reply.

- Shifted the Initial Data Size field in the Request, in order to make it easier to parse.

draft-03

- Shifted some fields in the Operation Reply to make it easier to parse.

- Added connection attempt timeout response code to Operation Replies.

- Proxies send an additional Authentication Reply after the authentication phase. (Useful for token window advertisements.)

- o Renamed the section "Connection Requests" to "Requests"

- o Clarified the fact that proxies don't need to support any command in particular.

- o Added the section "TCP Fast Open on the Client-Proxy Leg"

- o Options:

  * Added constants for option kinds

  * Salt options removed, along with the relevant section from Security Considerations. (TLS 1.3 Makes AEAD mandatory.)

  * Limited Authentication Data options to one per method.

  * Relaxed proxy requirements with regard to handling multiple Authentication Data options. (When the client violates the above bullet point.)

  * Removed interdependence between Authentication Method and Authentication Data options.

  * Clients SHOULD omit advertising the "No authentication required" option. (Was MAY.)

  * Idempotence options:

    + Token Window Advertisements are now part of successful Authentication Replies (so that the proxy-server RTT has no impact on their timeliness).

    + Proxies can't advertise token windows of size 0.

    + Tweaked token expenditure response codes.

    + Support no longer mandatory on the proxy side.

  * Revamped Socket options

    + Renamed Socket options to Stack options.

    + Banned contradictory socket options.

    + Added socket level for generic IP. Removed the "socket" socket level.

    + Stack options no longer use option codes from setsockopt().

      + Changed MPTCP Scheduler constants.

draft-02

- Made support for Idempotence options mandatory for proxies.
- Clarified what happens when proxies can not or will not issue tokens.
- Limited token windows to 2^31 - 1.
- Fixed definition of "less than" for tokens.
- NOOP commands now trigger Operation Replies.
- Renamed Authentication options to Authentication Data options.
- Authentication Data options are no longer mandatory.
- Authentication methods are now advertised via options.
- Shifted some Request fields.
- Option range for vendor-specific options.
- Socket options.
- Password authentication.
- Salt options.

draft-01

- Added this section.
- Support for idempotent commands.
- Removed version numbers from operation replies.
- Request port number for SOCKS over TLS. Deprecate encryption/ encapsulation within SOCKS.
- Added Version Mismatch Replies.
- Renamed the AUTH command to NOOP.
- Shifted some fields to make requests and operation replies easier to parse.

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

```
    CLIENT                                                        PROXY

            +------------------------+
            | Authentication methods | Request
    --------> Command code           +------------------------------>
            | Address                |
            | Port                   |
            | Options                |
            +------------------------+

            +------------------------+
    --------> Initial data           +------------------------------>
            +------------------------+

                                        +-----------------------+
                   Authentication Reply | Type                  |
     <----------------------------------+ Options               <-----
                                        +-----------------------+

     <-------------------(Authentication protocol)------------------>


                                        +-----------------------+
                   Authentication Reply | Type = Success        |
     <----------------------------------+ Options               <-----
                                        +-----------------------+

                          +-----------------------+
        Operation Reply   | Reply code            |
     <--------------------+ Bind address          <------------------
                          | Bind port             |
                          | Options               |
                          +-----------------------+
```

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 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. Requests

The client starts by sending a request to the proxy.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------------------------------+
|  Version = 6  | Command Code  |        Options Length         |
+---------------+---------------+---------------+---------------+
|             Port              |  Padding = 0  | Address Type  |
+-------------------------------+---------------+---------------+
|                                                             ...
...                Address (variable length)                 ...
...                                                             |
+---------------------------------------------------------------+
|                                                             ...
...                Options (variable length)                 ...
...                                                             |
+---------------------------------------------------------------+
```

Figure 2: SOCKS 6 Request

o Version: 6

o Command Code:

* 0x00 NOOP: does nothing.

* 0x01 CONNECT: requests the establishment of a TCP connection. TFO MUST NOT be used unless explicitly requested.

* 0x02 BIND: requests the establishment of a TCP port binding.

* 0x03 UDP ASSOCIATE: requests a UDP port association.

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, but padded by NUL characters, if needed.

* IPv6: a 16-byte IPv6 address

o Port: the port in network byte order.

o Padding: set to 0

o Options Length: the total size of the SOCKS options that appear in the Options field. MUST NOT exceed 16KB.

o Options: see Section 8.

The Address and Port fields have different meanings based on the Command Code:

o NOOP: The fields have no meaning. The Address Type field MUST be either 0x01 (IPv4) or 0x04 (IPv6). The Address and Port fields MUST be 0.

- o CONNECT: The fields signify the address and port to which the client wishes to connect.

- o BIND, UDP ASSOCIATE: The fields indicate the desired bind address and port. If the client does not require a certain address, it can set the Address Type field to 0x01 (IPv4) or 0x04 (IPv6), and the Address field to 0. Likewise, if the client does not require a certain port, it can set the Port field to 0.

Clients can advertise their supported authentication methods by including an Authentication Method Advertisement option (see Section 8.2).

5. Version Mismatch Replies

Upon receipt of a request starting with a version number other than 6, the proxy sends the following response:

```
 0 1 2 3 4 5 6 7
+---------------+
|  Version = 6  |
+---------------+
```

Figure 3: SOCKS 6 Version Mismatch Reply

- o Version: 6

A client MUST close the connection after receiving such a reply.

6. Authentication Replies

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

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------------------------------+
|  Version = 6  |      Type     |        Options Length         |
+---------------+---------------+-------------------------------+
|                                                             ...
...                  Options (variable length)               ...
...                                                             |
+---------------------------------------------------------------+
```

Figure 4: SOCKS 6 Authentication Reply

- Version: 6

- Type:

  * 0x00: authentication successful.

  * 0x01: authentication failed.

- Options Length: the total size of the SOCKS options that appear in the Options field. MUST NOT exceed 16KB.

- Options: see Section 8.

If the server signals that the authentication has failed and does not signal that any authentication negotiation can continue (via an Authentication Method Selection option), 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 Data option. Application data is not subject to any encryption negotiated during this phase. Descriptions of such negotiations are beyond the scope of this memo.

When the negotiation is complete (either successfully or unsuccessfully), the proxy sends a second Authentication Reply. The second Authentication Reply MUST NOT allow for further negotiations.

7. Operation Replies

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

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------------------------------+
|  Version = 6  |  Reply Code   |        Options Length         |
+---------------+---------------+---------------+---------------+
|           Bind Port           |  Padding = 0  | Address Type  |
+-------------------------------+---------------+---------------+
|                                                             ...
...              Bind Address (variable length)              ...
...                                                             |
+---------------------------------------------------------------+
|                                                             ...
...                 Options (variable length)                ...
...                                                             |
+---------------------------------------------------------------+
```

Figure 5: SOCKS 6 Operation Reply

- Version: 6
- Reply Code:
  * 0x00: Success
  * 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
  * 0x09: Connection attempt timed out
- Bind Port: the proxy bound port in network byte order.
- Padding: set to 0
- Address Type:

   * 0x01: IPv4

   * 0x03: Domain Name

   * 0x04: IPv6

- 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, but padded by NUL characters, if needed.

   * IPv6: a 16-byte IPv6 address

- Options Length: the total size of the SOCKS options that appear in the Options field. MUST NOT exceed 16KB.

- Options: see Section 8.

Proxy implementations MAY support any subset of the client commands listed in Section 4.

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

If the client issued an NOOP command, the client MUST close the connection after receiving the Operation Reply.

7.1. Handling CONNECT

In case the client has issued a CONNECT request, data can now pass.

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

Proxies offering UDP functionality may be configured with a UDP port used for relaying UDP datagrams to and from the client, and/or a port used for relaying datagrams over DTLS.

Following a successful Operation Reply, the client and the proxy begin exchanging messages with the following header:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------------------------------+
|  Version = 6  | Message Type  |         Message Length        |
+---------------+---------------+-------------------------------+
|                        Association ID                         |
|                           (8 bytes)                           |
+---------------------------------------------------------------+
```

Figure 6: UDP Association Header

o Message Type:

   * 0x01: Association Initialization

   * 0x02: Association Confirmation

   * 0x03: Datagram

   * 0x04: Error

o Message Length: the total length of the message

o Association ID: the identifier of the UDP association

First, the proxy picks an Association ID sends a an Association Initialization message:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------------------------------+
|  Version = 6  |     0x01      |       Message Length = 12     |
+---------------+---------------+-------------------------------+
|                        Association ID                         |
|                           (8 bytes)                           |
+---------------------------------------------------------------+
```

Figure 7: UDP Association Initialization

Proxy implementations SHOULD generate Association IDs randomly or pseudo-randomly.

Clients may start sending datagrams to the proxy either:

o over the TCP connection,

o in plaintext, using the proxy's configured UDP port(s), or

o over an established DTLS session.

A client's datagrams are prefixed by a Datagram Header, indicating the remote host's address and port:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------------------------------+
|  Version = 6  |     0x03      |        Message Length         |
+---------------+---------------+-------------------------------+
|                        Association ID                         |
|                           (8 bytes)                           |
+---------------+---------------+-------------------------------+
| Address Type  | Padding = 0   |             Port              |
+---------------+---------------+-------------------------------+
|                                                            ...
...                 Address (variable length)                 ...
...                                                             |
+---------------------------------------------------------------+
```

Figure 8: Datagram Header

o Version: 0x06

o Association ID: the identifier of the UDP association

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.

Datagrams sent over UDP MAY be padded with arbitrary data (i. e., the Message Length MAY be smaller than the actual UDP/DTLS payload). Client and proxy implementations MUST ignore the padding. If the Message Length is larger than the size of the UDP or DTLS payload, the message MUST be silently ignored.

Following the receipt of the first datagram from the client, the proxy makes a one-way mapping between the Association ID and:

o the TCP connection, if it was received over TCP, or

o the 5-tuple of the UDP conversation, if the datagram was received over plain UDP, or

o the DTLS connection, if the datagram was received over DTLS. The DTLS connection is identified either by its 5-tuple, or some other mechanism, like [I-D.ietf-tls-dtls-connection-id].

The proxy SHOULD close the TCP connection if the initial datagram is not received after a timeout.

Further datagrams carrying the same Association ID, but not matching the established mapping, are silently dropped.

The proxy then sends an UDP Association Confirmation message over the TCP connection with the client:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------------------------------+
|  Version = 6  |      0x02     |       Message Length = 12     |
+---------------+---------------+-------------------------------+
|                        Association ID                         |
|                           (8 bytes)                           |
+---------------------------------------------------------------+
```

Figure 9: UDP Association Confirmation

Following the confirmation message, UDP packets bound for the proxy's bind address and port are relayed to the client, also prefixed by a Datagram Header.

The UDP association remains active for as long as the TCP connection between the client and the proxy is kept open.

7.3.1. Proxying UDP servers

Under some circumstances (e.g. when hosting a server), the SOCKS client expects the remote host to send UDP datagrams first. As such, the SOCKS client must trigger a UDP Association Confirmation without having the proxy relay any datagrams on its behalf.

To that end, it sends an empty datagram prefixed by a Datagram Header with an IP address and port consisting of zeroes. If it is using UDP, the client SHOULD resend the empty datagram if an UDP Association Confirmation is not received after a timeout.

7.3.2. Proxying multicast traffic

The use of multicast addresses is permitted for UDP traffic only.

7.3.3. Reporting ICMP Errors

If a client has opted in (see Section 8.1.8), the proxy MAY relay information contained in some ICMP Error packets. The message format is as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------------------------------+
|  Version = 6  |     0x04      |        Message Length         |
+---------------+---------------+-------------------------------+
|                        Association ID                         |
|                           (8 bytes)                           |
+---------------+---------------+-------------------------------+
| Address Type  |  Padding = 0  |             Port              |
+---------------+---------------+-------------------------------+
|                                                            ...
...                 Address (variable length)                 ...
...                                                             |
+---------------+---------------+-------------------------------+
| Reporter ATYP |  Error Code   |          Padding = 0          |
+---------------+---------------+-------------------------------+
|                                                            ...
...           Reporter Address (variable length)              ...
...                                                             |
+---------------------------------------------------------------+
```

Figure 10: Datagram Error Message

- Address: The destination address of the IP header contained in the ICMP payload o Address Type: Either 0x01 (IPv4) or 0x04 (IPv6)

o Port: The destination port of the UDP header contained in the ICMP payload

o Reporter Address: The IP address of the host that issued the ICMP error

o Reporter Address Type (ATYP): Either 0x01 (IPv4) or 0x04 (IPv6)

o Error code:

   * 0x01: Network unreachable

   * 0x02: Host unreachable

   * 0x03: TTL expired

   * 0x04: Datagram too big (IPv6 only)

It is possible for ICMP Error packets to be spurious, and not be related to any UDP packet that was sent out. The proxy is not required to check the validity of ICMP Error packets before reporting them to the client.

Clients MUST NOT send Datagram Error messages to the proxy. Proxies MUST NOT send Error messages unless the clients have opted in.

8. SOCKS Options

SOCKS options have the following format:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|             Kind              |            Length             |
+-------------------------------+-------------------------------+
|                                                             ...
...                Option Data (variable length)              ...
...                                                             |
+---------------------------------------------------------------+
```

Figure 11: SOCKS 6 Option

o Kind: Allocated by IANA. (See Section 15.)

o Length: The total length of the option. MUST be a multiple of 4.

o Option Data: The contents are specific to each option kind.

Unless otherwise noted, client and proxy implementations MAY omit supporting any of the options described in this document. Upon encountering an unsupported option, a SOCKS endpoint MUST silently ignore it.

8.1. Stack options

Stack options can be used by clients to alter the behavior of the protocols on top of which SOCKS is running, as well the protocols used by the proxy to communicate with the remote host (i.e. IP, TCP, UDP). A Stack option can affect either the proxy's protocol on the client-proxy leg or on the proxy-remote leg. Clients can only place Stack options inside SOCKS Requests.

Proxies MAY choose not to honor any Stack options sent by the client.

Proxies include Stack options in their Operation Replies to signal their behavior, and MUST do so for every supported Stack option sent by the client. Said options MAY also be unsolicited, i. e. the proxy MAY send them to signal behavior that was not explicitly requested by the client.

If a particular Stack option is unsupported, the proxy MUST silently ignore it.

In case of UDP ASSOCIATE, the stack options refer to the UDP traffic relayed by the proxy.

Stack options that are part of the same message MUST NOT contradict one another or contain duplicate information.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 1            |            Length             |
+---+-----------+---------------+-------------------------------+
|Leg|   Level   |     Code      |                             ...
+---+-----------+---------------+                             ...
...                                                           ...
...              Option Data (variable length)                ...
...                                                             |
+---------------------------------------------------------------+
```

Figure 12: Stack Option

o Leg:

* 1: Client-Proxy Leg

* 2: Proxy-Remote Leg

* 3: Both Legs

o Level:

* 1: IP: options that apply to either IPv4 or IPv6

* 2: IPv4

* 3: IPv6

* 4: TCP

* 5: UDP

o Code: Option code

o Option Data: Option-specific data

8.1.1. IP TOS options

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 1            |          Length = 8           |
+---+-----------+---------------+---------------+---------------+
|Leg| Level = 1 |   Code = 1    |      TOS      |  Padding = 0  |
+---+-----------+---------------+---------------+---------------+
```

Figure 13: IP TOS Option

o TOS: The IP TOS code

The client can use IP TOS options to request that the proxy use a certain value for the IP TOS field. Likewise, the proxy can use IP TOS options to advertise the TOS values being used.

8.1.2. Happy Eyeballs options

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|            Kind = 1           |          Length = 8           |
+---+-----------+---------------+-------------------------------+
| 2 | Level = 1 |   Code = 2    | Availability  | Padding = 0   |
+---+-----------+---------------+-------------------------------+
```

Figure 14: Happy Eyeballs Option

- Availability:

  * 0x01: Happy Eyeballs is not desired (client) or was not performed (proxy)

  * 0x02: Happy Eyeballs is desired (client) or was attempted (proxy)

This memo provides enough features for clients to implement a mechanism analogous to Happy Eyeballs [RFC8305] over SOCKS. However, when the delay between the client and the proxy, or the proxy's vantage point, is high, doing so can become impractical or inefficient.

In such cases, the client can instruct the proxy to employ the Happy Eyeballs technique on its behalf when connecting to a remote host.

The client MUST supply a Domain Name as part of its Request. Otherwise, the proxy MUST silently ignore the option.

TODO: Figure out which knobs to include.

8.1.3. TTL options

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|            Kind = 1           |          Length = 8           |
+---+-----------+---------------+---------------+---------------+
|Leg| Level = 1 |   Code = 3    |      TTL      | Padding = 0   |
+---+-----------+---------------+---------------+---------------+
```

Figure 15: IP TTL Option

- TTL: The IP TTL or Hop Limit

8.1.4. No Fragmentation options

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 1            |          Length = 8           |
+---+-----------+---------------+-------------------------------+
|Leg| Level = 1 |   Code = 4    | Availability  |  Padding = 0  |
+---+-----------+---------------+-------------------------------+
```

Figure 16: No Fragmentation Option

o Availability:

   * 0x01: IP fragmentation is allowed (client) or the lack thereof is not enforced (proxy)

   * 0x02: IP fragmentation is not desired (client) or avoidance of fragmentation is enforced (proxy)

A No Fragmentation option can be used to instruct the proxy to avoid IP fragmentation. In the case of IPv4, this also entails setting the DF bit on outgoing packets.

8.1.5. TFO options

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 1            |          Length = 8           |
+---+-----------+---------------+-------------------------------+
| 2 | Level = 4 |   Code = 1    |         Payload Size          |
+---+-----------+---------------+-------------------------------+
```

Figure 17: TFO Option

o Payload Size: The desired payload size of the TFO SYN. Ignored in case of a BIND command.

If a SOCKS Request contains a TFO option, the proxy SHOULD attempt to use TFO in case of a CONNECT command, or accept TFO in case of a BIND command. Otherwise, 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 a CONNECT command, the client can indicate the desired payload size of the SYN. If the field is 0, the proxy can use an arbitrary payload size. If the field is non-zero, the proxy MUST NOT use a payload size larger than the one indicated. The proxy MAY use a smaller payload size than the one indicated.

8.1.6. Multipath options

In case of a CONNECT or BIND command, the client can inform the proxy whether MPTCP is desired on the proxy-remote leg by sending a Multipath option.

Conversely, the proxy can use a Multipath option to convey the following information:

- o whether or not the connection uses MPTCP or not, when replying to a CONNECT command, or in the second Operation reply to a BIND command, or

- o whether an MPTCP connection will be accepted, when first replying to a BIND command.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 1            |          Length = 8           |
+---+-----------+---------------+---------------+---------------+
| 2 | Level = 4 |   Code = 2    | Availability  |  Padding = 0  |
+---+-----------+---------------+---------------+---------------+
```

Figure 18: Multipath Option

- o Availability:

  - * 0x01: MPTCP is not desired (client) or available (proxy)

  - * 0x02: MPTCP is desired (client) or available (proxy)

In the absence of such an option, the proxy SHOULD NOT enable MPTCP for CONNECT commands.

8.1.7. Listen Backlog options

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 1            |          Length = 8           |
+---+-----------+---------------+-------------------------------+
| 2 | Level = 4 |   Code = 3    |            Backlog            |
+---+-----------+---------------+-------------------------------+
```

Figure 19: Listen Backlog Option

- o Backlog: The length of the listen backlog.

The default behavior of the BIND does not allow a client to simultaneously handle multiple connections to the same bind address. A client can alter BIND's behavior by adding a TCP Listen Backlog Option to a BIND Request, provided that the Request is part of a Session.

In response, the proxy sends a TCP Listen Backlog Option as part of the Operation Reply, with the Backlog field signaling the actual backlog used. The proxy SHOULD NOT use a backlog longer than requested.

Following the successful negotiation of a backlog, the proxy listens for incoming connections for as long as the initial connection stays open. The initial connection is not used to relay data between the client and a remote host.

To accept connections, the client issues further BIND Requests using the bind address and port supplied by the proxy in the initial Operation Reply. Said BIND requests must belong to the same Session as the original Request.

If no backlog is issued, the proxy signals a backlog length of 0, and BIND's behavior remains unaffected.

8.1.8. UDP Error options

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 1            |          Length = 8           |
+---+-----------+---------------+-------------------------------+
| 2 | Level = 5 |   Code = 1    | Availability  |  Padding = 0  |
+---+-----------+---------------+-------------------------------+
```

Figure 20: UDP Error Option

o Availability:

   * 0x01: Error reporting is not dersired (client) or will not be performed (proxy)

   * 0x02: Error reporting is dersired (client) or will be performed (proxy)

Clients can use this option to turn on error reporting for a particular UDP association. See Section 7.3.3.

8.1.9. Port Parity options

The RTP specification [RFC3550] recommends running the protocol on consecutive UDP ports, where the even port is the lower of the two.

SOCKS clients can specify the desired port parity when issuing a UDP ASSOCIATE command, and request that the port's counterpart be reserved.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 1            |          Length = 8           |
+---+-----------+---------------+---------------+---------------+
| 2 | Level = 5 |   Code = 2    |    Parity     |    Reserve    |
+---+-----------+---------------+---------------+---------------+
```

Figure 21: Port Parity Option

o Parity:

   * 0x00: No particular parity

   * 0x01: Even

   * 0x02: Odd

- Reserve: whether or not to reserve the port's counterpart

   * 0x00: Don't reserve

   * 0x01: Reserve

If the UDP ASSOCIATE request does not have the Port field set to 0 (indicating that an arbitrary port can be chosen), the proxy MUST ignore the suggested parity.

A port's counterpart is determined as follows:

- for even ports, it is the next higher port and

- for odd ports, it is the next lower port.

If the proxy can not or will not comply with the requested parity, it also does not reserve the allocated port's counterpart.

Port reservations are in place until either:

- the original association ends, or

- an association involving the reserved port is made.

An association involving a reserved port can only be made if a client explicitly requests said port. Further, if the original association is part of a session (see Section 8.4), the reserved port can only be claimed from within the same session.

8.2. Authentication Method options

A client that is willing to go through the authentication phase MUST include an Authentication Method Advertisement option in its Request. In case of a CONNECT Request, the option is also used to specify the amount of initial data supplied before any method-specific authentication negotiations take place.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 2            |            Length             |
+-------------------------------+-------------------------------+
|      Initial Data Length      |                             ...
+-------------------------------+                             ...
...                                                           ...
...                 Methods (variable length)                 ...
...                                                           ...
...             +-----------------------------------------------+
...             |   Padding = 0 (variable length, 0-3 bytes)    |
+---------------+-----------------------------------------------+
```

Figure 22: Authentication Method Advertisement Option

o Initial Data Size: A two-byte number in network byte order. In case of CONNECT, this is the number of bytes of initial data that are supplied by the client immediately following the Request. This number MUST NOT be larger than 2^14.

o Methods: One byte per advertised method. Method numbers are assigned by IANA.

o Padding: A minimally-sized sequence of zeroes, such that the option length is a multiple of 4. Note that 0 coincides with the value for "No Authentication Required".

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

The proxy indicates which authentication method must proceed by sending an Authentication Method Selection option in the corresponding Authentication Reply:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 3            |          Length = 8           |
+---------------+---------------+-------------------------------+
|    Method     |                  Padding = 0                  |
+---------------+-----------------------------------------------+
```

Figure 23: Authentication Method Selection Option

o Method: The selected method.

If the proxy selects "No Acceptable Methods", the client MUST close the connection.

If authentication is successful via some other means, or not required at all, the proxy silently ignores the Authentication Method Advertisement option.

8.3. Authentication Data options

Authentication Data options carry method-specific authentication data. They can be part of SOCKS Requests and Authentication Replies.

Authentication Data options have the following format:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 4            |            Length             |
+---------------+---------------+-------------------------------+
|    Method     |                                             ...
+---------------+                                             ...
...                                                           ...
...          Authentication Data (variable length)            ...
...                                                             |
+---------------------------------------------------------------+
```

Figure 24: Authentication Data Option

o Method: The number of the authentication method. These numbers are assigned by IANA.

o Authentication Data: The contents are specific to each method.

Clients MUST only place one Authentication Data option per authentication method.

8.4. Session options

Clients and proxies can establish SOCKS sessions, which span one or more Requests. All session-related negotiations are done via Session Options, which are placed in Requests and Authentication Replies by the client and, respectively, by the proxy.

Client and proxy implementations MUST either support all Session Option Types, or none.

8.4.1. Session initiation

A client can initiate a session by sending a Session Request Option:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 5            |          Length = 4           |
+-------------------------------+-------------------------------+
```

Figure 25: Session Request Option

The proxy then replies with a Session ID Option in the successful Operation Reply:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 6            |            Length             |
+-------------------------------+-------------------------------+
|                                                             ...
...                 Session ID (variable length)              ...
...                                                             |
+---------------------------------------------------------------+
```

Figure 26: Session ID Option

- o Session ID: An opaque sequence of bytes specific to the session. The size MUST be a multiple of 4. MUST NOT be empty.

The Session ID serves to identify the session and is opaque to the client.

The credentials, or lack thereof, used to initiate the session are tied to the session.

The SOCKS Request that initiated the session is considered part of the session. A client MUST NOT attempt to initiate a session from within a different session.

If the proxy can not or will not honor the Session Request, it does so silently.

8.4.2. Further SOCKS Requests

Any further SOCKS Requests that are part of the session MUST include a Session ID Option (as seen in Figure 26). The proxy MUST silently ignore any authentication attempt in the Request, and MUST NOT require any authentication.

The proxy then replies by placing a Session OK option in the successful Authentication Reply:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 8            |          Length = 4           |
+-------------------------------+-------------------------------+
```

Figure 27: Session OK Option

If the Session ID is invalid, the first Authentication Reply MUST signal that authentication failed and can not continue (by setting the Type field to 0x01). Further, it SHALL contain a Session Invalid option:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 9            |          Length = 4           |
+-------------------------------+-------------------------------+
```

Figure 28: Session Invalid Option

8.4.3. Tearing down the session

Proxies can, at their discretion, tear down a session and free all associated state. Proxy implementations SHOULD feature a timeout mechanism that destroys sessions after a period of inactivity. When a session is terminated, the proxy MAY close all connections associated with said session.

Clients can signal that a session is no longer needed, and can be torn down, by sending a Session Teardown option in addition to the Session ID option:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 10           |          Length = 4           |
+-------------------------------+-------------------------------+
```

Figure 29: Session Teardown Option

After sending such an option, the client MUST assume that the session is no longer valid. The proxy MUST treat the session-terminating request as if it were not part of any session.

8.5. Idempotence options

To protect against duplicate SOCKS Requests, clients can request, and then spend, idempotence tokens. A token can only be spent on a single SOCKS request.

Tokens are 4-byte unsigned integers in a modular 4-byte space. Therefore, if x and y are tokens, x is less than y if 0 < (y - x) < 2^31 in unsigned 32-bit arithmetic.

Proxies grant contiguous ranges of tokens called token windows. Token windows are defined by their base (the first token in the range) and size.

All token-related operations are done via Idempotence options.

Idempotence options are only valid in the context of a SOCKS Session. If a SOCKS Request is not part of a Session (either by supplying a valid Session ID or successfully initiating one via a Session Request), the proxy MUST silently ignore any Idempotence options.

Token windows are tracked by the proxy on a per-session basis. There can be at most one token window for every session and its tokens can only be spent from within said session.

Client and proxy implementations MUST either support all Idempotence Option Types, or none.

8.5.1. Requesting a token window

A client can obtain a window of tokens by sending an Idempotence Request option as part of a SOCKS Request:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 11           |           Length = 8          |
+-------------------------------+-------------------------------+
|                          Window Size                          |
+---------------------------------------------------------------+
```

Figure 30: Token Request

o Window Size: The requested window size.

Once a token window is issued, the proxy MUST include an Idempotence Window option in all subsequent successful Authentication Replies:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 12           |           Length = 12         |
+-------------------------------+-------------------------------+
|                          Window Base                          |
+---------------------------------------------------------------+
|                          Window Size                          |
+---------------------------------------------------------------+
```

Figure 31: Idempotence Window

o Window Base: The first token in the window.

o Window Size: The window size. This value MAY differ from the requested window size. Window sizes MUST be less than 2^31. Window sizes MUST NOT be 0.

If no token window is issued, the proxy MUST silently ignore the Token Request. If there is already a token window associated with the session, the proxy MUST NOT issue a new window.

8.5.2. Spending a token

The client can attempt to spend a token by including a Idempotence Expenditure option in its SOCKS request:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 13           |          Length = 4           |
+-------------------------------+-------------------------------+
|                             Token                             |
+---------------------------------------------------------------+
```

Figure 32: Idempotence Expenditure

- o Kind: 13 (Idempotence Expenditure option)
- o Length: 8
- o Token: The token being spent.

Clients SHOULD prioritize spending the smaller tokens.

The proxy responds by sending either an Idempotence Accepted or Rejected option as part of the Authentication Reply:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 14           |          Length = 4           |
+-------------------------------+-------------------------------+
```

Figure 33: Idempotence Accepted

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 15           |          Length = 4           |
+-------------------------------+-------------------------------+
```

Figure 34: Idempotence Rejected

If eligible, the token is spent before attempting to honor the Request. If the token is not eligible for spending, the Authentication Reply MUST indicate failure.

8.5.3. Shifting windows

Windows can be shifted (i. e. have their base increased, while retaining their size) unilaterally by the proxy.

Proxy implementations SHOULD shift the window: * as soon as the lowest-order token in the window is spent and * when a sufficiently high-order token is spent.

Proxy implementations SHOULD NOT shift the window's base beyond the highest unspent token.

8.5.4. Out-of-order Window Advertisements

Even though the proxy increases the window's base monotonically, there is no mechanism whereby a SOCKS client can receive the Token Window Advertisements in order. As such, clients SHOULD disregard Token Window Advertisements with a Window Base less than the previously known value.

9. Username/Password Authentication

Username/Password authentication is carried out as in [RFC1929].

Clients can also attempt to authenticate by placing the Username/ Password request in an Authentication Data Option.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 4            |            Length             |
+---------------+---------------+---------------+---------------+
|  Method = 2   |                                             ...
+---------------+                                             ...
...                                                           ...
...                 Username/Password Request                 ...
...                                                           ...
...             +-----------------------------------------------+
...             |   Padding = 0 (variable length, 0-3 bytes)    |
+---------------+-----------------------------------------------+
```

Figure 35: Password authentication via a SOCKS Option

o Username/Password Request: The Username/Password Request, as described in [RFC1929].

Proxies reply by including a Authentication Data Option in the next Authentication Reply which contains the Username/Password reply:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|           Kind = 4            |          Length = 8           |
+---------------+---------------+---------------+---------------+
|  Method = 2   |    Username/Password Reply    |  Padding = 0  |
+---------------+-------------------------------+---------------+
```

Figure 36: Reply to password authentication via a SOCKS Option

o Username/Password Reply: The Username/Password Reply, as described in [RFC1929].

10. TCP Fast Open on the Client-Proxy Leg

TFO breaks TCP semantics, causing replays of the data in the SYN's payload under certain rare circumstances [RFC7413]. A replayed SOCKS Request could itself result in a replayed connection on behalf of the client.

As such, client implementations SHOULD NOT use TFO on the client-proxy leg unless:

o The protocol running on top of SOCKS tolerates the risks of TFO, or

o The SYN's payload does not contain any application data (so that no data is replayed to the server, even though duplicate connections are still possible), or

o The client uses Idempotence Options, making replays very unlikely, or

o SOCKS is running on top of TLS and Early Data is not used.

11. False Starts

In case of CONNECT Requests, the client MAY start sending application data as soon as possible, as long as doing so does not incur the risk of breaking the SOCKS protocol.

Clients must work around the authentication phase by doing any of the following:

- If the Request does not contain an Authentication Method Advertisement option, the authentication phase is guaranteed not to happen. In this case, application data MAY be sent immediately after the Request.

- Application data MAY be sent immediately after receiving an Authentication Reply indicating success.

- When performing a method-specific authentication sequence, application data MAY be sent immediately after the last client message.

12. DNS provided by SOCKS

Clients may require information typically obtained from DNS servers, albeit from the proxy's vantage point.

While the CONNECT command can work with domain names, some clients' workflows require that addresses be resolved as a separate step prior to connecting. Moreover, the SOCKS Datagram Header, as described in Section 7.3, can be reduced in size by providing the resolved destination IP address, rather than the FQDN.

Emerging techniques may also make use of DNS to deliver server-specific information to clients. For example, Encrypted SNI [I-D.ietf-tls-esni] relies on DNS to publish encryption keys.

Proxy implementations MAY provide a default plaintext DNS service. A client looking to make use of it issues a CONNECT Request to IP address 0.0.0.0 or 0:0:0:0:0:0:0:0 on port 53. Following successful authentication, the Operation Reply MAY indicate an unspecified bind address (0.0.0.0 or ::) and port (0). The client and proxy then behave as per [RFC7766].

The service itself can be provided directly by the proxy daemon, or by proxying the client's request to a pre-configured DNS server.

If the proxy does not implement such functionality, it MAY return an error code signaling "Connection refused".

13. Security Considerations

13.1. Large requests

Given the format of the request message, a malicious client could craft a request that is in excess of 16 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 or their size exceeds an imposed hard cap.

13.2. Replay attacks

In TLS 1.3, early data (which is likely to contain a full SOCKS request) is prone to replay attacks.

While Token Expenditure options can be used to mitigate replay attacks, anything prior to the initial Token Request is still vulnerable. As such, client implementations SHOULD NOT make use of TLS early data unless the Request attempts to spend a token.

13.3. Resource exhaustion

Malicious clients can issue a large number of Session Requests, forcing the proxy to keep large amounts of state.

To mitigate this, the proxy MAY implement policies restricting the number of concurrent sessions on a per-IP or per-user basis, or barring unauthenticated clients from establishing sessions.

14. Privacy Considerations

The timing of Operation Replies can reveal some information about a proxy's recent usage:

- o The DNS resolver used by the proxy may cache the answer to recent queries. As such, subsequent connection attempts to the same hostname are likely to be slightly faster, even if requested by different clients.
- o Likewise, the proxy's OS typically caches TFO cookies. Repeated TFO connection attempts tend to be sped up, regardless of the client.

15. IANA Considerations

This document requests that IANA allocate 2-byte option kinds for SOCKS 6 options. Further, this document requests the following option kinds:

- o Unassigned: 0

- Stack: 1
- Authentication Method Advertisement: 2
- Authentication Method Selection: 3
- Authentication Data: 4
- Session Request: 5
- Session ID: 6
- Session OK: 8
- Session Invalid: 9
- Session Teardown: 10
- Idempotence Request: 11
- Idempotence Window: 12
- Idempotence Expenditure: 13
- Idempotence Accepted: 14
- Idempotence Rejected: 15
- Resolution Request: 16
- IPv4 Resolution: 17
- IPv6 Resolution: 18
- Experimental: 64512-0xFFFF

This document also requests that IANA allocate a TCP and UDP port for SOCKS over TLS and DTLS, respectively.

16. Acknowledgments

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

17. References

17.1. Normative References

[RFC1929] Leech, M., "Username/Password Authentication for SOCKS V5", RFC 1929, DOI 10.17487/RFC1929, March 1996, <https://www.rfc-editor.org/info/rfc1929>.

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

[RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016, <https://www.rfc-editor.org/info/rfc7766>.

[RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <https://www.rfc-editor.org/info/rfc8305>.

17.2. Informative References

[I-D.ietf-tls-dtls-connection-id] Rescorla, E., Tschofenig, H., and T. Fossati, "Connection Identifiers for DTLS 1.2", draft-ietf-tls-dtls-connection-id-07 (work in progress), October 2019.

[I-D.ietf-tls-esni] Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "TLS Encrypted Client Hello", draft-ietf-tls-esni-08 (work in progress), October 2020.

[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, <https://www.rfc-editor.org/info/rfc1928>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[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, <https://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, <https://www.rfc-editor.org/info/rfc7413>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Authors' Addresses

Vladimir Olteanu
University Politehnica of Bucharest
313 Splaiul Independentei, Sector 6
Bucharest
Romania

Email: vladimir.olteanu@cs.pub.ro

Dragos Niculescu
University Politehnica of Bucharest
313 Splaiul Independentei, Sector 6
Bucharest
Romania

Email: dragos.niculescu@cs.pub.ro

Network Working Group   L. Colitti
Internet-Draft   Google
Intended status: Informational   T. Pauly
Expires: 19 May 2021   Apple Inc.
                                                        15 November 2020

Per-Application Networking Considerations
draft-per-app-networking-considerations-00

Abstract

This document describes considerations for and implications of using application identifiers as a method of differentiating traffic on networks. Specifically, it discusses privacy considerations, possible mitigations, and considerations for user experience and API design.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/tfpauly/per-app-networking-considerations.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

There are a number of use cases where network operators, or applications, might desire for application traffic to be treated differently by the network. Some examples are:

* Network-specific services. Applications might want to access local resources on a network that does not otherwise provide Internet access (for example, an entertainment system on an airplane).

* Per-application private networks. Certain applications, such as enterprise applications, might want to connect directly to the enterprise network in a secure fashion without using a device-wide VPN.

* Mobile network services. In mobile networks, applications like voice over LTE, IMS and RCS often use a different virtual network than general Internet traffic.

* Applications with specific performance requirements. Certain applications would benefit from particular scheduling or QoS policies - for example applications requiring low latency such as voice might be scheduled and queued differently from latency-insensitive traffic.

* Local breakout. In a mobile networks, applications might want to access resources through a different network interface (e.g., one that uses IPv6 addresses that are local to a specific area, and do not have a wide mobility).

* Zero-rating traffic. As allowed by regulators, certain classes of traffic (e.g., messaging or streaming video) might be exempt from metering on networks that are otherwise metered.

In existing networks, this is sometimes implemented by the network using deep packet inspection (e.g., flow tracking coupled with inspection of the SNI handshake). This is complex, implicates public policy concerns, and generally conflicts with the recommendations in [RFC7258]. The move towards encrypted protocols such as [RFC8484] and [I-D.ietf-tls-esni] will make this more difficult for some operators. Thus, if an application is to receive different treatment, the host or the application itself should be involved in requesting specific network treatment. This document explores the implications.

In this document, the term "application" refers to an application as understood by the user of the device.

1.1. Conventions and Definitions

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Requesting differential treatment

There are already mechanisms for applications to request and obtain particular treatment by the network, or to communicate application identity to the network in order to obtain particular treatment. These include:

* Diffserv

* APN6

* Network tokens

* Slicing in 3GPP 5G networks

* Explicit application selection of a given Provisioning Domain (PvD) [RFC8801]

3. Open Internet implications

In certain regulatory regions, networks that provide general Internet access may not be permitted to discriminate between traffic sent to or from different lawful applications or websites, or such discrimination may be prohibited if commercially based. In a situation where the network operator has influence on the implementation of the user host (e.g., mobile networks where the handset is sold by the carrier), the device may be able to implement network policies directly, and thus may be impacted by neutrality considerations.

Neutrality concerns can be addressed by providing user control over assignment of particular applications to the particular network resources available to that user. Further, network neutrality implications may be reduced or avoided in some jurisdictions if the differential treatment occurs between different classes of traffic with different network requirements (e.g., bandwidth-intensive traffic vs. low-latency traffic) as opposed to between different applications with similar network requirements, and thus, by ensuring that the mechanism used to communicate requests to the network only specifies traffic classes and not individual applications.

4. Privacy implications

IETF guidance to avoid pervasive monitoring [RFC7258] is for network protocols to expose as little information as possible. Some of the proposed technologies for application signalling rely on the application exposing its identity to the network so that the network can then implement appropriate policies. This may provide the network with much more information than is needed to implement the desired behaviour. Information about which users are using specific applications, or visiting certain destinations, and when, can be highly privacy-sensitive.

Note that application identity can be exposed to the network even in the absence of explicit signalling. For example, if the host were to implement a network-set policy that requires that traffic from application X be sent on a different network path than all other traffic, the identity of application X would be exposed to the network as soon as it sends traffic.

Privacy concerns may also be reduced or avoided if the mechanism to request a different class of service only specifies the class of service (e.g., "low latency" or "streaming video") instead of the application originating the traffic.

In a situation where the network operator has influence over the implementation of the user host, the operator can still impose policies on what requests are possible - for example, the operator might choose to limit access to specialized services such as carrier messaging only to carrier applications. It is possible for such policies to preserve privacy if the policies specify general categories of traffic as opposed to specifying applications.

5. Mitigating implications via traffic categories

Many of these implications can be mitigated if the mechanism does not request different treatment of a service for a particular application, but instead specifies a general category of traffic, especially one that is defined based on traffic properties rather than commercial agreements.

Categories of traffic need to be sufficiently broad to not identify individual applications, and should be general enough that details about a user cannot be inferred merely by use of the category.

Consider the example a network that wants to provide differentiated service for a role-playing game application that can take advantage of a low-latency path. Several levels of categories could be defined. The following list shows some examples, in order of decreasing specificity:

1. Role-playing game

2. Game

3. Real-time/low-latency

The first category would not be an appropriate choice due to the privacy implications of identifying what kind of game a user plays. The second category is preferable, but the third is best since it defines a way to manage the network traffic without identifying anything about the content of the application.

Some use cases for traffic differentiation might need other kinds of categories. For example, operators might wish to zero-rate applications using categories based on payment tiers and rate-limiting.

6. User experience considerations

Privacy and neutrality concerns can be mitigated if the host's user is informed that particular applications are seeking or designated for particular treatment and consents to it. In order for consent to be meaningful, the user should be presented with a message that they understand. It may be difficult to balance the goal of providing complete and accurate information with the goal of ensuring that the user understands the implications.

7. API considerations

It is desirable to provide an API layer that is not tied to specific network technologies (e.g., URSP, VPN, etc.). Having applications select a specific Provisioning Domain (PvD) could provide a useful layer of abstraction, as described in [I-D.ietf-taps-interface].

Any API should not involve revealing an application or user identity to the network via metadata without network authentication. Instead, the API should allow a given setting to be conditional on the identity of the network. For example, an application should express "use the zero-rated service for my app when on a particular carrier network", instead of blindly saying "this is my application identifier".

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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative References

[I-D.ietf-taps-interface]
Trammell, B., Welzl, M., Enghardt, T., Fairhurst, G., Kuehlewind, M., Perkins, C., Tiesel, P., Wood, C., and T. Pauly, "An Abstract Application Layer Interface to Transport Services", Work in Progress, Internet-Draft, draft-ietf-taps-interface-10, 2 November 2020, <http://www.ietf.org/internet-drafts/draft-ietf-taps-interface-10.txt>.

[I-D.ietf-tls-esni]
Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-08, 16 October 2020, <http://www.ietf.org/internet-drafts/draft-ietf-tls-esni-08.txt>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/info/rfc7258>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/info/rfc8484>.

[RFC8801] Pfister, P., Vyncke, É., Pauly, T., Schinazi, D., and W. Shao, "Discovering Provisioning Domain Names and Data", RFC 8801, DOI 10.17487/RFC8801, July 2020, <https://www.rfc-editor.org/info/rfc8801>.

Acknowledgments

Thanks to Adi Masputra and Elliot Briggs for their inputs to this discussion.

Authors' Addresses

Lorenzo Colitti
Google
Shibuya 3-21-3,
Japan

Email: lorenzo@google.com

Tommy Pauly
Apple Inc.
One Apple Park Way
Cupertino, California 95014,
United States of America

Email: tpauly@apple.com

Internet Area Working Group M. Piraux
Internet-Draft O. Bonaventure
Intended status: Experimental UCLouvain
Expires: 6 May 2021 A. Masputra
Apple Inc.
2 November 2020

Tunneling Internet protocols inside QUIC
draft-piraux-intarea-quic-tunnel-00

Abstract

This document specifies methods for tunneling packets of Internet protocols inside a QUIC connection.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Mobile devices such as laptops, smartphones or tablets have different requirements than the traditional fixed devices. These mobile devices often change their network attachment. They are often attached to trusted networks, but sometimes they need to be connected to untrusted networks where their communications can be eavesdropped, filtered or modified. In these situations, the classical approach is to rely on VPN protocols such as DTLS or IPSec. These VPN protocols provide the encryption and authentication functions to protect those mobile clients from malicious behaviors in untrusted networks.

However, some networks have deployed filters that block these VPN protocols. When faced with such filters, users can either switch off their connection or find alternatives, e.g. by using TLS to access some services over TCP port 443. The planned deployment of QUIC [I-D.ietf-quic-transport] [I-D.ietf-quic-tls] opens a new opportunity for such users. Since QUIC will be used to access web sites, it should be less affected by filters than VPN solutions such as IPSec or DTLS. Furthermore, the flexibility of QUIC makes it possible to easily extend the protocol to support VPN services.

This document explores how QUIC could be used to enable devices to communicate securely in untrusted networks. The QUIC protocol opens up a new way to find a clean solution to this problem. First, QUIC includes the same encryption and authentication techniques as deployed VPN protocols. Second, QUIC is intended to be widely used to support web-based services, making it unlikely to be filtered in many networks, in contrast with VPN protocols. Third, the QUIC migration mechanism enables handovers between several network interfaces.

This document is organized as follows. Section 3 describes the reference environment. Then, we propose a first mode of operation, explained in Section 4, that use the recently proposed datagram extension ([I-D.pauly-quic-datagram]) for QUIC to transport plain packets over a QUIC connection. Section 5 specifies how a connection is established in this document proposal. Section 7 details the format of the messages introduced by this document.

2. Conventions and Definitions

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Reference environment

The reference scenario is a client that uses a QUIC tunnel to send all its packets to a concentrator. The concentrator processes the packets received from the client over the QUIC connection and forwards them to their final destination. It also receives the packets destined to the client and tunnels them through the QUIC connection.

```
                                                        +--------------+
 +--------+              +---------------+              | Final        |
 | Client |              | Concentrator  |<===\ ... \===>| destination  |
 +--------+              +---------------+              | server       |
       ^    +----------+    ^                           +--------------+
       |    | Access   |    |             Legend:
       .----| network  |----.               --- QUIC connection
            +----------+                    === Tunneled flow
```

Figure 1: A client attached to a concentrator

In a nutshell, the solution proposed in this document works as follows. The client opens a QUIC connection to a concentrator. The concentrator authenticates the client through means that are outside the scope of this document such as client certificates, usernames/passwords, OAuth, ... If the authentication succeeds, the client can send packets via the concentrator by tunneling them through the concentrator.

The concentrator captures the packets destined to the client and tunnels them over the QUIC connection. This solution is intended to provide a similar service as the one provided by IPSec tunnels or DTLS. This document leaves address assignment mechanisms out of scope, deployments can rely on out-of-band configurations for that purpose.

4. The tunnel mode

The "tunnel mode" of operation leverages the recently proposed QUIC datagram extension [I-D.pauly-quic-datagram]. In a nutshell, to send a packet to a remote host, the client simply encapsulates the entire packet inside a QUIC DATAGRAM frame sent over the QUIC connection established with the concentrator.

The frame transmission is subject to congestion control, but the frame that contains the packet is not retransmitted in case of loss as specified in [I-D.pauly-quic-datagram].

This mode adds a minimal byte overhead for packet encapsulation in QUIC. It does not define ways of indicating the protocol of the conveyed packets, which can be useful in deployments for which out-of-band signaling may be used.

5. Connection establishment

During QUIC connection establishment, the "tunnel mode" of operation support is indicated by setting the ALPN token "qt" in the TLS handshake. Draft-version implementations MAY specify a particular draft version by suffixing the token, e.g. "qt-00" refers to the first version of this document.

After the QUIC connection is established, the client can start using the "tunnel mode". The client may use PCP [RFC6887] to request the concentrator to accept inbound connections on their behalf. After the negotiation of such port mappings, the concentrator can start sending packets containing inbound connections in QUIC DATAGRAM frame.

6. Reporting access networks availability

When the access network is unstable or its performance is degrading, for instance due to signal loss, being able to report its availability to the concentrator can help reduce the amount of packets sent over unstable or unavailable paths. It can also resume quickly the sending of packets over a previously unavailable access network.

To do so, we define in Section 7 a message called Access Report TLV. The message can be sent by the client to the concentrator. It identifies the type of access network reported and its associated status. This message is sent over the QUIC connection in a separate unidirectional stream.

7. Messages format

In the following sections, we specify the format of each message introduced in this document. The messages are encoded as TLVs, i.e. (Type, Length, Value) tuples, as illustrated in Figure 2. All TLV fields are encoded in network-byte order.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type (8)   |  Length (8)   |          [Value (*)]        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: QUIC tunnel TLV Format

The Type field is encoded as a byte and identifies the type of the TLV. The Length field is encoded as a byte and indicate the length in bytes of the Value field. A value of zero indicates that no Value field is present. The Value field is a type-specific value whose length is determined by the Length field.

7.1. QUIC tunnel control TLVs

This document specifies the following QUIC tunnel control TLVs:

Type	Size	Sender	Mode	Name
0x00	4 bytes	Client	all	Access Report TLV

Figure 3: QUIC tunnel control TLVs

The Access Report TLV is sent by the client to periodically report on access networks availability. Each Access Report TLV MUST be sent on a separate unidirectional stream. The stream FIN bit MUST be set following the end of the TLV.

7.1.1. Access Report TLV

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type = 0x00  | Length = 0x02 | AI (4)| R (4) |   Signal (8)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Access Report TLV

The Access Report TLV contains the following:

* AI (Access ID) - a four-bit-long field that identifies the access network, e.g., 3GPP (Radio Access Technologies specified by 3GPP) or Non-3GPP (accesses that are not specified by 3GPP) [TS23501]. The value is one of those listed below (all other values are invalid and the TLV that contains it MUST be discarded):

Access ID	Description
1	3GPP Network
2	Non-3GPP Network

* R (Reserved) - a four-bit-long field that MUST be zeroed on transmission and ignored on receipt.

* Signal - a one-octet-long field that identifies the report signal, e.g., available or unavailable. The value is supplied to the QUIC tunnel through some mechanism that is outside the scope of this document. The value is one of those listed in Section 9.3.

The client that includes the Access Report TLV sets the value of the Access ID field according to the type of access network it reports on. Also, the client sets the value of the Signal field to reflect the operational state of the access network. The mechanism to determine the state of the access network is outside the scope of this specification.

The client MUST be able to cancel the sending of an Access Report TLV that is pending delivery, i.e. by resetting its corresponding unidirectional stream. This can be used when the information contained in the TLV is no longer relevant, e.g. the access network availability has changed. The time of canceling is based on local policies and network environment.

Reporting the unavailability of an access network to the concentrator can serve as an indication to stop sending packets over this network while maintaining the QUIC tunnel connection. Upon reporting of the availability of this network, the concentrator can quickly resume sending packets over this network.

8. Security Considerations

8.1. Privacy

The Concentrator has access to all the packets it processes. It MUST be protected as a core IP router, e.g. as specified in [RFC1812].

8.2. Ingress Filtering

Ingress filtering policies MUST be enforced at the network boundaries, i.e. as specified in [RFC2827].

9. IANA Considerations

9.1. Registration of QUIC tunnel Identification String

This document creates one new registration for the identification of the QUIC tunnel protocol in the "Application Layer Protocol Negotiation (ALPN) Protocol IDs" registry established in [RFC7301].

The "qt" string identifies the QUIC tunnel protocol datagram mode.

Protocol: QUIC Tunnel

Identification Sequence: 0x71 0x74 ("qt")

Specification: This document

9.2. QUIC tunnel control TLVs

IANA is requested to create a new "QUIC tunnel control Parameters" registry.

The following subsections detail new registries within "QUIC tunnel control Parameters" registry.

9.2.1. QUIC tunnel control TLVs Types

IANA is request to create the "QUIC tunnel control TLVs Types" sub-registry. New values are assigned via IETF Review (Section 4.8 of [RFC8126]).

The initial values to be assigned at the creation of the registry are as follows:

Code	Name	Reference
0	Access Report TLV	[This-Doc]

9.3. QUIC tunnel Access Report Signal Codes

This document establishes a registry for QUIC tunnel Access Report Signal codes. The "QUIC tunnel Access Report Signal Code" registry manages a 62-bit space. New values are assigned via IETF Review (Section 4.8 of [RFC8126]).

The initial values to be assigned at the creation of the registry are as follows:

Code	Name	Reference
1	Access Available	[This-Doc]
2	Access Unavailable	[This-Doc]

10. References

10.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[TS23501] 3GPP (3rd Generation Partnership Project), "Technical Specification Group Services and System Aspects; System Architecture for the 5G System; Stage 2 (Release 16)", 3GPP TS23501, 2019.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

10.2. Informative References

[I-D.pauly-quic-datagram]
Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", Work in Progress, Internet-Draft, draft-pauly-quic-datagram-05, 4 November 2019, <http://www.ietf.org/internet-drafts/draft-pauly-quic-datagram-05.txt>.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-32, 20 October 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-transport-32.txt>.

[I-D.ietf-quic-tls]
Thomson, M. and S. Turner, "Using TLS to Secure QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-tls-32, 20 October 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-tls-32.txt>.

[RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers", RFC 1812, DOI 10.17487/RFC1812, June 1995, <https://www.rfc-editor.org/info/rfc1812>.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <https://www.rfc-editor.org/info/rfc2827>.

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <https://www.rfc-editor.org/info/rfc6887>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

Appendix A. Change Log

A.1. Since draft-piraux-quic-tunnel-03

* Make the lightweight mode the default "tunnel mode"

* Rename the datagram mode as tunnel session mode in draft-piraux-intarea-quic-tunnel-session

A.2. Since draft-piraux-quic-tunnel-02

* Add the lightweight mode

A.3. Since draft-piraux-quic-tunnel-01

* Add the Access Report TLV for reporting access networks availability

* Add a section with examples of use of the Packet Tag

A.4. Since draft-piraux-quic-tunnel-00

* Separate the document in two and put the stream mode in another document

* Remove TCP Extended TLV

* Add a mechanism for joining QUIC connections in a QUIC tunneling session

* Add a format for encoding any network-layer protocol packets and Ethernet frames in QUIC DATAGRAM frames

Acknowledgments

Thanks to Quentin De Coninck and Francois Michel for their comments and the proofreading of the first version of draft-piraux-quic-tunnel. Thanks to Gregory Vander Schueren for his comments on the first version of draft-piraux-quic-tunnel. Thanks to Florin Baboescu for his comments on the first version of this document.

Authors' Addresses

Maxime Piraux
UCLouvain

Email: maxime.piraux@uclouvain.be

Olivier Bonaventure
UCLouvain

Email: olivier.bonaventure@uclouvain.be

Adi Masputra
Apple Inc.

Email: adi@apple.com

Internet Area Working Group
Internet-Draft
Intended status: Experimental
Expires: 6 May 2021

M. Piraux
O. Bonaventure
UCLouvain
A. Masputra
Apple Inc.

2 November 2020

Session mode for multiple QUIC Tunnels
draft-piraux-intarea-quic-tunnel-session-00

Abstract

This document specifies methods for grouping QUIC tunnel connections in a single session enabling the exchange of packets of Internet protocols over several QUIC connections.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Mobile devices such as laptops, smartphones or tablets have different requirements than the traditional fixed devices. These mobile devices often change their network attachment. They are often attached to trusted networks, but sometimes they need to be connected to untrusted networks where their communications can be eavesdropped, filtered or modified. In these situations, the classical approach is to rely on VPN protocols such as DTLS or IPSec. These VPN protocols provide the encryption and authentication functions to protect those mobile clients from malicious behaviors in untrusted networks.

Today's mobile devices are often multihomed and many expect to be able to perform seamless handovers from one access network to another without breaking the established VPN sessions. In some situations it can also be beneficial to combine two or more access networks together to increase the available host bandwidth. A protocol such as Multipath TCP [RFC6824] supports those handovers and allows aggregating the bandwidth of different access links. It could be combined with single-path VPN protocols to support both seamless handovers and bandwidth aggregation above VPN tunnels. Unfortunately, Multipath TCP is not yet deployed on most Internet servers and thus few applications would benefit from such a use case.

This document explores how QUIC could be used to enable multi-homed mobile devices to communicate securely in untrusted networks.

This document is organized as follows. Section 3 describes the reference environment. Then, we propose a new mode of operation, explained in Section 4, that use the recently proposed datagram extension ([I-D.pauly-quic-datagram]) for QUIC to transport plain IP packets over a QUIC connection. Section 5 specifies how a connection is established in this document proposal. Section 6 details the format of the messages introduced by this document.

2. Conventions and Definitions

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Reference environment

The reference environment is illustrated in Figure 1, in which a client-initiated flow is tunneled through the concentrator.

```
              +---------+
         .----| Access  |----.
         |    | network |    |
         |    |    A    |    |
         v    +----------    v                           +-------------+
+--------+                +---------------+              | Final       |
| Client |                | Concentrator  |<===\ ... \===>| destination |
+--------+                +---------------+              | server      |
         ^    +---------+    ^                           +-------------+
         |    | Access  |    |
         |    | network |    |            Legend:
         .----|    B    |----.              --- QUIC connection
              +---------+                   === Tunneled flow
```

Figure 1: Example environment

Such a multihomed client would like to benefit from the different access networks available to reach the concentrator. These access networks can be used for load-sharing, failover or other purposes. One possibility to efficiently use these two access networks is to rely on the proposed Multipath extensions to QUIC [I-D.deconinck-quic-multipath]. Another approach is to create one QUIC connection using the single-path QUIC protocol [I-D.ietf-quic-transport] over each access network and glue these different connections together in a single session on the concentrator. Given the migration capabilities of QUIC, this approach could support failover with a single active QUIC connection at a time.

4. The tunnel session mode

The "tunnel session" mode enables the client and the concentrator to exchange packets of several network protocols through the QUIC tunnel connection at the same time. It also leverages the QUIC datagram extension [I-D.pauly-quic-datagram].

This document specifies the following format for encoding packets in QUIC DATAGRAM frame. It allows encoding packets from several protocols by identifying the corresponding protocol of the packet in each QUIC DATAGRAM frame. Figure 2 describes this encoding.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Protocol Type (16)      |        Packet Tag (16)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Packet (*)                        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Encoding packets in QUIC DATAGRAM frame

This encoding defines three fields.

* Protocol Type: The Protocol Type field contains the protocol type of the payload packet. The values for the different protocols are defined as "ETHER TYPES" in [IANA-ETHER-TYPES]. A QUIC tunnel that receives a Protocol Type representing an unsupported protocol MAY drop the associated Packet. QUIC tunnel endpoints willing to exchange Ethernet frames can use the value 0x6558 for [Transparent-Ethernet-Bridging].

* Packet Tag: An opaque 16-bit value. The QUIC tunnel application is free to decide its semantic value. For instance, a QUIC tunnel endpoint MAY encode the sending order of packets in the Packet Tag, e.g. as a timestamp or a sequence number, to allow reordering on the receiver.

* Packet: The packet conveyed inside the QUIC tunnel connection.

```
              ,->+----------+
              |  |    IP    |
QUIC packet   |  +----------+
containing    |  |   UDP    |
a DATAGRAM    |  +----------+
frame         |  |   QUIC   |
              |  |..........|
              |  | DATAGRAM |
              |  |  P. Type |
              |  |  P. Tag  |
              |  |+--------+|<-.
              |  ||   IP   ||  |
              |  |+--------+|  | Tunneled
              |  ||  UDP   ||  | UDP packet
              |  |+--------+|  |
              |  |   ....   |<-.
              `->+----------+
```

Figure 3: QUIC packet sent by the client when tunneling a UDP packet

Figure 3 illustrates how a UDP packet is tunneled using the tunnel session mode. The main advantage of the tunnel session mode is that it supports IP and any protocol above the network layer. Any IP packet can be transported using the datagram extension over a QUIC connection. However, this advantage comes with a large per-packet overhead since each packet contains both a network and a transport header. All these headers must be transmitted in addition with the IP/UDP/QUIC headers of the QUIC connection. For TCP connections for instance, the per-packet overhead can be large.

4.1. Joining a tunneling session

If the client is multihomed, it can use Multipath QUIC [I-D.deconinck-quic-multipath] to efficiently use its different access networks. This version of the document does not elaborate in details on this possibility. If the concentrator does not support Multipath QUIC, then the client creates several QUIC connections and joins them at the application layer. This works as illustrated in figure Figure 4. Each message is exchanged over a dedicated unidirectional QUIC stream. Their format is detailed in Section 6. When the client opens the first QUIC connection with the concentrator, (1) it can request a QUIC tunnel session identifier. (2) The concentrator replies with a variable-length opaque value that identifies the QUIC tunneling session. When opening a QUIC connection over another access network, (3) the client can send this identifier to join the QUIC tunneling session. The concentrator matches the session identifier with the existing session with the client. It can then use both sessions to reach the client and receive tunneled packets from the client.

```
        1-Req. Sess. ID->
       .------------------------------.
       |               <-Sess. ID.-2 |
       v                              v
+--------+                      +---------------+
| Client |                      | Concentrator |
+--------+                      +---------------+
       ^                              ^
       | 3-Join. Sess.->              |      Legend:
       .------------------------------.        --- QUIC connection
```

Figure 4: Creating sessions over different access networks

Joining a tunneling session allows grouping several QUIC connections to the concentrator. Each endpoint can then coordinate the use of the Packet Tag across the tunneling session as presented in Section 4.1.1.

Both QUIC tunnel endpoints open their first unidirectional stream (i.e. stream 2 and 3), hereafter named the QUIC tunnel control stream, to exchange these messages. A QUIC tunnel endpoint MUST NOT close its control stream and SHOULD provide enough flow control credit to its peer.

The messages format used for this purpose are described in Section 6. The client initiates the procedure and MAY either start a new session or join an existing session. This negotiation MUST NOT take place more than once per QUIC connection.

4.1.1. Coordinate use of the Packet Tag

When using the tunnel session mode, each packet is associated with a 16-bit value called Packet Tag. This document leaves defining the meaning of this value to implementations. This section provides some examples on how it can be used to implement packet reordering across several QUIC tunnel connections grouped in a tunneling session.

A first simple example of use is to encode the timestamp at which the datagram was sent. Using a millisecond precision and encoding the 16 lower bits of the timestamp makes the value wrapping around in a bit more than 65 seconds.

Another example of use is to maintain a value counting the datagrams sent over all QUIC tunnel connections of the tunneling session. The 16-bit value allows distinguishing at most 32768 packets in flight.

The QUIC tunnel receiver can then distinguish, buffer and reorder packets based on this value. Mechanisms for managing the datagram buffer and negotiating the use of the Packet Tag are out of scope of this document.

5. Connection establishment

During connection establishment, the tunnel session mode support is indicated by setting the ALPN token "qt-session" in the TLS handshake. Draft-version implementations MAY specify a particular draft version by suffixing the token, e.g. "qt-session-00" refers to the first version of this document.

6. Messages format

In the following sections, we specify the format of each message introduced in this document. They are encoded as TLVs, following the format defined in Section 7 of [I-D.piraux-intarea-quic-tunnel].

6.1. QUIC tunnel control TLVs

This document specifies additional QUIC tunnel control TLVs:

Type	Size	Sender	Mode	Name
0x01	Variable	Client	tunnel session	New Session TLV
0x02	Variable	Concentrator	tunnel session	Session ID TLV
0x03	Variable	Client	tunnel session	Join Session TLV

Figure 5: QUIC tunnel control TLVs

The New Session TLV is used by the client to initiate a new tunneling session. The Session ID TLV is used by the concentrator to communicate to the client the Session ID identifying this tunneling session. The Join Session TLV is used to join a given tunneling session, identified by a Session ID. All QUIC these tunnel control TLVs MUST NOT be sent on other streams than the QUIC tunnel control streams.

When the tunnel session mode is in use, the Access Report TLV defined in Section 7.1.1 of [I-D.piraux-intarea-quic-tunnel] MUST be sent on other streams than the QUIC tunnel control stream.

6.1.1. New Session TLV

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type (8)   |  Length (8)   |  [QoS Flow Indication (*)]  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: New Session ID TLV

The New Session TLV contains an optional value. It initiates a new tunneling session at the concentrator, and can contain an opaque value giving an indication on the type of traffic conveyed over this session. The concentrator can use this indication for QoS purposes for instance.

The concentrator MUST send a Session ID TLV in response, with the Session ID corresponding to the tunneling session created. After sending a New Session TLV, the client MUST close the QUIC tunnel control stream.

The concentrator MUST NOT send New Session TLVs.

6.1.2. Session ID TLV

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type (8)   |  Length (8)   |        Session ID (*)       ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Session ID TLV

The Session ID TLV contains an opaque value that identifies the current tunneling session. It can be used by the client in subsequent QUIC connections to join them to this tunneling session. The concentrator MUST send a Session ID TLV in response of a New Session TLV, with the Session ID corresponding to the tunneling session created.

The client MUST NOT send a Session ID TLV. The concentrator MUST close the QUIC tunnel control stream after sending a Session ID TLV.

6.1.3. Join Session TLV

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type (8)   |  Length (8)   |        Session ID (*)       ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: Join Session TLV

The Join Session TLV contains an opaque value that identifies a tunneling session to join. The client can send a Join Session TLV to join the QUIC connection to a particular tunneling session. The tunneling session is identified by the Session ID. After sending a Join Session TLV, the client MUST close the QUIC tunnel control stream.

The concentrator MUST NOT send Join Session TLVs. After receiving a Join Session TLV, the concentrator MUST use the Session ID to join this QUIC connection to the tunneling session. Joining the tunneling session implies merging the state of this QUIC tunnel connection to the session. A successful joining of connection is indicated by the closure of the QUIC tunnel control stream of the concentrator.

In cases of failure when joining a tunneling session, the concentrator MUST send a RESET_STREAM with an application error code discerning the cause of the failure. The possible codes are listed below:

* UNKNOWN_ERROR (0x0): An unknown error occurred when joining the tunneling session. QUIC tunnel endpoints SHOULD use more specific error codes when applicable.

* UNKNOWN_SESSION_ID (0x1): The Session ID used in the Join Session TLV is not a valid ID. It was not issued in a Session ID TLV or refers to an expired tunneling session.

* CONFLICTING_STATE (0x2): The current state of the QUIC tunnel connection could not be merged with the tunneling session.

7. Security Considerations

The security considerations of [I-D.piraux-intarea-quic-tunnel] are also applicable to this document.

8. IANA Considerations

8.1. Registration of QUIC tunnel Identification String

This document creates a new registration for the identification of the QUIC tunnel protocol in the "Application Layer Protocol Negotiation (ALPN) Protocol IDs" registry established in [RFC7301].

The "qt-session" string identifies the QUIC tunnel protocol tunnel session mode.

Protocol: QUIC Tunnel session mode

Identification Sequence: 0x71 0x74 0x2d 0x73 0x65 0x73 0x73 0x69 0x6f 0x6e ("qt-session")

Specification: This document

8.2. QUIC tunnel control TLVs

The following subsections detail new registries within "QUIC tunnel control Parameters" registry.

8.2.1. QUIC tunnel control TLVs Types

This document creates three new registrations to identify the QUIC tunnel control TLVs defined in this document in the "QUIC tunnel control TLVs Types" sub-registry defined in [I-D.piraux-intarea-quic-tunnel].

The values to be added in the registry are as follows:

Code	Name	Reference
1	New Session TLV	[This-Doc]
2	Session ID TLV	[This-Doc]
3	Join Session TLV	[This-Doc]

8.3. QUIC tunnel control Error Codes

This document establishes a registry for QUIC tunnel control stream error codes. The "QUIC tunnel control Error Code" registry manages a 62-bit space. New values are assigned via IETF Review (Section 4.8 of [RFC8126]).

The initial values to be assigned at the creation of the registry are as follows:

Code	Name	Reference
0	UNKNOWN_ERROR	[This-Doc]
1	UNKNOWN_SESSION_ID	[This-Doc]
2	CONFLICTING_STATE	[This-Doc]

9. References

9.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[I-D.piraux-intarea-quic-tunnel] Piraux, M., Bonaventure, O., and A. Masputra, "Tunneling Internet protocols inside QUIC", Work in Progress, Internet-Draft, draft-piraux-intarea-quic-tunnel-00, 2 November 2020, <http://www.ietf.org/internet-drafts/draft-piraux-intarea-quic-tunnel-00.txt>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[I-D.pauly-quic-datagram]
Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", Work in Progress, Internet-Draft, draft-pauly-quic-datagram-05, 4 November 2019, <http://www.ietf.org/internet-drafts/draft-pauly-quic-datagram-05.txt>.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-32, 20 October 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-transport-32.txt>.

[I-D.deconinck-quic-multipath]
Coninck, Q. and O. Bonaventure, "Multipath Extensions for QUIC (MP-QUIC)", Work in Progress, Internet-Draft, draft-deconinck-quic-multipath-05, 20 August 2020, <http://www.ietf.org/internet-drafts/draft-deconinck-quic-multipath-05.txt>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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, <https://www.rfc-editor.org/info/rfc6824>.

[IANA-ETHER-TYPES]
"IANA ETHER TYPES", https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.txt , 2019.

[Transparent-Ethernet-Bridging]
Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 1701, DOI 10.17487/RFC1701, October 1994, <https://www.rfc-editor.org/info/rfc1701>.

Appendix A. Change Log

Acknowledgments

Authors' Addresses

Maxime Piraux
UCLouvain

Email: maxime.piraux@uclouvain.be

Olivier Bonaventure
UCLouvain

Email: olivier.bonaventure@uclouvain.be

Adi Masputra
Apple Inc.

Email: adi@apple.com

Internet Area Working Group   M. Piraux
Internet-Draft   O. Bonaventure
Intended status: Experimental   UCLouvain
Expires: 6 May 2021   2 November 2020

Tunneling TCP inside QUIC
draft-piraux-intarea-quic-tunnel-tcp-00

Abstract

This document specifies a new operating mode for a QUIC tunnel to efficiently convey TCP bytestreams.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The recently proposed QUIC tunnel protocol [I-D.piraux-intarea-quic-tunnel] supports the exchange of IP packets and Ethernet frames over a QUIC connection. Its two existing operating modes transports plain packets inside QUIC frames. Their main advantage is that they support any network-layer protocol. However, this advantage comes with a large per-packet overhead since each packet contains both a network and a transport header. All these headers must be transmitted in addition to the IP/UDP/QUIC headers of the QUIC connection. For TCP connections for instance, the per-packet overhead can be large.

In this document, we propose a new operating mode for the QUIC tunnel protocol, called the stream mode. It takes advantage of the QUIC streams to efficiently transport TCP bytestreams over a QUIC connection. Section 3 describes this new mode. Section 5 specifies the format of the messages introduced by this document. Section 6 contains example flows.

2. Conventions and Definitions

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. The stream mode

Since QUIC supports multiple streams, another possibility to carry the data exchanged over TCP connections between the client and the concentrator is to transport the bytestream of each TCP connection as one of the bidirectional streams of the QUIC connection. For this, we base our approach on the 0-RTT Convert protocol [I-D.ietf-tcpm-converters] that was proposed to ease the deployment of TCP extensions. In a nutshell, it is an application proxy that converts TCP connections, allowing the use of new TCP extensions through an intermediate relay.

A similar approach is used in the stream mode. When a client opens a stream, it sends at the beginning of the bytestream one or more TLV messages indicating the IP address and port number of the remote destination of the bytestream. Their format is detailed in section Section 5.1. Upon reception of such a TLV message, the concentrator opens a TCP connection towards the specified destination and connects the incoming bytestream of the QUIC connection to the bytestream of the new TCP connection (and similarly in the opposite direction).

Figure 1 summarizes how the new TCP connection is mapped to the QUIC stream. Actions and events of a TCP connection are translated to actions and events of the associated QUIC stream, so that a state transition on one is translated to the other.

TCP	QUIC Stream
SYN received	Open Stream, send TLVs
FIN received	Send Stream FIN
RST received	Send STOP_SENDING
	Send RESET_STREAM
Data received	Send Stream data

QUIC Stream	TCP
Stream opened, TLVs received	Send SYN
Stream FIN received	Send FIN
STOP_SENDING received	Send RST
RESET_STREAM received	Send RST
Stream data received	Send data

Figure 1: TCP connection to QUIC stream mapping

When sending STOP_SENDING or RESET_STREAM frames in response to the receipt of a TCP RST, QUIC tunnel peers MUST use the application protocol error code 0x00 (TCP_CONNECTION_RESET).

The QUIC stream-level flow control can be tuned to match the receive window size of the corresponding TCP connection, so that no excessive data needs to be buffered.

4. Connection establishment

The support of the stream mode is negotiated during the connection establishment by including the quic_tunnel_stream_mode transport parameter (value TBD). The parameter value has no meaning and SHOULD be null.

During the connection establishment, the concentrator can control the number of TCP bytestreams that can be opened initially by setting the initial_max_streams_bidi QUIC transport parameter as defined in [I-D.ietf-quic-transport].

5. Messages format

In the following sections, we specify the format of each message introduced in this document. They are encoded using the TLV format described in [I-D.piraux-intarea-quic-tunnel].

5.1. QUIC tunnel stream TLVs

When using the stream mode, one or more messages are used to trigger and confirm the establishment of a connection towards the final destination for a given stream. Those messages are exchanged on this QUIC stream before the TCP connection bytestream. This section describes the format of these messages.

This document specifies the following QUIC tunnel stream TLVs:

Type	Size	Name
0x00	20 bytes	TCP Connect TLV
0x01	2 bytes	TCP Connect OK TLV
0x02	Variable	Error TLV
0xff	2 bytes	End TLV

Figure 2: QUIC tunnel stream TLVs

The TCP Connect TLV is used to request the establishment a TCP connection by the concentrator towards the final destination. The TCP Connect OK TLV confirms the establishment of this TCP connection. The Error TLV is used to indicate any error that occurred during the establishment of a TCP connection. Finally, the End TLV marks the end of the series of TLVs and the start of the bytestream on a given QUIC stream. These TLVs are detailed in the following sections.

Future versions of this document may define new TLVs. The End TLV allows a QUIC tunnel peer to send several TLVs before the start of the bytestream.

```
      Offset 0         Offset 20   Offset 22
         |                  |         |
Client   v                  v         v
         +------------------+---------+----------------
Stream 0 | TCP Connect TLV  | End TLV | TCP bytestream ...
         +------------------+---------+----------------
```

Figure 3: Example of use of QUIC tunnel stream TLVs

Figure 3 illustrates an example of use of QUIC tunnel streams TLVs. In this example, the client opens Stream 0 and sends two TLVs. The first one requests the concentrator to establish a new TCP connection. The second TLV marks the end of the series of TLV and the start of the TCP bytestream.

5.1.1. TCP Connect TLV

The TCP Connect TLV indicates the final destination of the TCP connection associated to a given QUIC stream. The fields Remote Peer Port and Remote Peer IP Address contain the destination port number and IP address of the final destination.

The Remote Peer IP Address MUST be encoded as an IPv6 address. IPv4 addresses MUST be encoded using the IPv4-Mapped IPv6 Address format defined in [RFC4291]. Further, the Remote Peer IP address field MUST NOT include multicast, broadcast, and host loopback addresses [RFC6890].

A QUIC tunnel peer MUST NOT send more than one TCP Connect TLV per QUIC stream. A QUIC tunnel peer MUST NOT send a TCP Connect TLV on non-self initiated streams.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type (8)   |  Length (8)   |     Remote Peer Port (16)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                  Remote Peer IP Address (128)                 |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: TCP Connect TLV

5.1.2. TCP Connect OK TLV

The TCP Connect OK TLV does not contain a value. Its presence confirms the successful establishment of the requested TCP connection to the final destination. A QUIC peer MUST NOT send a TCP Connect OK TLV on self-initiated streams.

5.1.3. Error TLV

The Error TLV indicates out-of-band errors that occurred during the establishment of the TCP connection to the final destination. These errors can be ICMP Destination Unreachable messages for instance. In this case the ICMP packet received by the concentrator is copied inside the Error Payload field.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type (8)   |   Length (8)  |         Error Code (16)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      [Error Payload (*)]                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Error TLV

The following bytestream-level error codes are defined in this document:

Code	Name
0x0	Protocol Violation
0x1	ICMP Packet Received
0x2	Malformed TLV
0x3	Network Failure

Figure 6: Bytestream-level Error Codes

* Protocol Violation (0x0): A general error code for all non-conforming behaviors encountered. A QUIC tunnel peer SHOULD use a more specific error code when possible.

* ICMP Packet Received (0x1): This code indicates that the concentrator received an ICMP packet while trying to create the associated TCP connection. The Error Payload contains the packet.

* Malformed TLV (0x2): This code indicates that a received TLV was not successfully parsed or formed. A peer receiving a TCP Connect TLV with an invalid IP address MUST send an Error TLV with this error code.

* Network Failure (0x3): This codes indicates that a network failure prevented the establishment of the connection.

After sending one or more Error TLVs, the sender MUST send an End TLV and terminate the stream, i.e. set the FIN bit after the End TLV.

5.1.4. End TLV

The End TLV does not contain a value. Its existence signals the end of the series of TLVs. The next byte in the QUIC stream after this TLV is part of of the tunneled bytestream.

6. Example flows

This section illustrates the different messages described previously and how they are used in a QUIC tunnel connection. For QUIC STREAM frames, we use the following syntax: STREAM[ID, Stream Data [, FIN]]. The first element is the Stream ID, the second is the Stream Data contained in the frame and the last one is optional and indicates that the FIN bit is set.

```
Client                        Concentrator          Final Destination
 | STREAM[0, "TCP Connect, End"] ||                                |
 |------------------------------>||              SYN               |
 |                               ||===============================>|
 |                               ||            SYN+ACK             |
 |STREAM[0,"TCP Connect OK, End"]||<===============================|
 |<------------------------------||                                |
 | STREAM[0, "bytestream data"]  ||                                |
 |------------------------------>||      bytestream data, ACK      |
 |                               ||===============================>|
 |                               ||      bytestream data, ACK      |
 |  STREAM[0, "bytestream data"] ||<===============================|
 |<------------------------------||              FIN               |
 |      STREAM[0, "", FIN]       ||<===============================|
 |<------------------------------||              ACK               |
 |      STREAM[0, "", FIN]       ||===============================>|
 |------------------------------>||              FIN               |
 |                               ||===============================>|
 |                               ||              ACK               |
 |                               ||<===============================|

Legend:
   --- QUIC connection
   === TCP connection
```

Figure 7: Example flow for the stream mode

On Figure 7, the client is initiating a TCP connection in stream mode to the Final Destination. A request and a response are exchanged, then the connection is torn down gracefully. A remote-initiated connection accepted by the concentrator on behalf of the client would have the order and the direction of all messages reversed.

7. Security Considerations

7.1. Denial of Service

There is a risk of an amplification attack when the Concentrator sends a TCP SYN in response of a TCP Connect TLV. When a TCP SYN is larger than the client request, the Concentrator amplifies the client traffic. To mitigate such attacks, the Concentrator SHOULD rate limit the number of pending TCP Connect from a given client.

8. IANA Considerations

8.1. QUIC tunnel stream TLVs

IANA is requested to create a new "QUIC tunnel stream Parameters" registry.

The following subsections detail new registries within "QUIC tunnel stream Parameters" registry.

8.1.1. QUIC tunnel stream TLVs Types

IANA is request to create the "QUIC tunnel stream TLVs Types" sub-registry. New values are assigned via IETF Review (Section 4.8 of [RFC8126]).

The initial values to be assigned at the creation of the registry are as follows:

Code	Name	Reference
0	TCP Connect TLV	[This-Doc]
1	TCP Connect OK TLV	[This-Doc]
2	Error TLV	[This-Doc]
255	End TLV	[This-Doc]

8.1.2. QUIC tunnel streams TLVs Error Types

IANA is request to create the "QUIC tunnel stream TLVs Error Types" sub-registry. New values are assigned via IETF Review (Section 4.8 of [RFC8126]).

The initial values to be assigned at the creation of the registry are as follows:

Code	Name	Reference
0	Protocol Violation	[This-Doc]
1	ICMP packet received	[This-Doc]
2	Malformed TLV	[This-Doc]
3	Network Failure	[This-Doc]

8.2. QUIC Transport Parameter Registry

This document defines a new transport parameter for the negotiation of the stream mode. The following entry in Table 1 should be added to the "QUIC Transport Parameters" registry under the "QUIC Protocol" heading.

Value	Parameter Name	Specification
TBD	quic_tunnel_stream_mode	Section 4

Table 1: Addition to QUIC Transport Parameters Entries

9. References

9.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <https://www.rfc-editor.org/info/rfc6890>.

[I-D.piraux-intarea-quic-tunnel]
Piraux, M., Bonaventure, O., and A. Masputra, "Tunneling Internet protocols inside QUIC", Work in Progress, Internet-Draft, draft-piraux-intarea-quic-tunnel-00, 2 November 2020, <http://www.ietf.org/internet-drafts/draft-piraux-intarea-quic-tunnel-00.txt>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[I-D.ietf-tcpm-converters]
Bonaventure, O., Boucadair, M., Gundavelli, S., Seo, S., and B. Hesmans, "0-RTT TCP Convert Protocol", Work in Progress, Internet-Draft, draft-ietf-tcpm-converters-19, 22 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-tcpm-converters-19.txt>.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-32, 20 October 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-transport-32.txt>.

[I-D.piraux-quic-tunnel]
Piraux, M., Bonaventure, O., and A. Masputra, "Tunneling Internet protocols inside QUIC", Work in Progress, Internet-Draft, draft-piraux-quic-tunnel-03, 12 August 2020, <http://www.ietf.org/internet-drafts/draft-piraux-quic-tunnel-03.txt>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

Appendix A. Change Log

A.1. Since draft-piraux-quic-tunnel-tcp-01

* Nits

A.2. Since draft-piraux-quic-tunnel-tcp-00

* Add the quic_tunnel_stream_mode transport parameter for negotiation

Acknowledgments

This documents draws heavily on the initial version of [I-D.piraux-quic-tunnel]. Their contributors are thanked again here. This work was partially supported by the MQUIC project.

Authors' Addresses

Maxime Piraux
UCLouvain

Email: maxime.piraux@uclouvain.be

Olivier Bonaventure
UCLouvain

Email: olivier.bonaventure@uclouvain.be

LPWAN   P. Thubert, Ed.
Internet-Draft   Cisco Systems
Updates: 5172 (if approved)   22 September 2020
Intended status: Standards Track
Expires: 26 March 2021

SCHC over PPP
draft-thubert-intarea-schc-over-ppp-02

Abstract

This document extends RFC 5172 to signal the use of SCHC as the compression method between a pair of nodes over PPP. Combined with RFC 2516, this enables the use of SCHC over Ethernet and Wi-Fi.

Table of Contents

1. Introduction

The Point-to-Point Protocol (PPP) [RFC5172] provides a standard method of encapsulating network-layer protocol information over serial (point-to-point and bus) links. "A Method for Transmitting PPP Over Ethernet (PPPoE)" [RFC2516] transports PPP over Ethernet between a pair of nodes. It is compatible with a translating bridge to Wi-Fi, and therefore enables PPP over Wi-Fi as well.

PPP also proposes an extensible Link Control Protocol and a family of Network Control Protocols (NCPs) for establishing and configuring different network-layer protocols. "IP Version 6 over PPP" [RFC5072] specifies the IPv6 Control Protocol (IPV6CP), which is an NCP for a PPP link, and allows for the negotiation of desirable parameters for an IPv6 interface over PPP. "Negotiation for IPv6 Datagram Compression Using IPv6 Control Protocol" [RFC5172] defines the IPv6 datagram compression option that can be negotiated by a node on the link through the IPV6CP.

The "Static Context Header Compression (SCHC) and fragmentation for LPWAN, application to UDP/IPv6" [SCHC] is a new technology that can provide an extreme compression performance but requires a same state to be provisionned on both ends before it can be operated. To enable SCHC over PPP and therefore Ethernet and Wi-Fi, this specification extends [RFC5172] to signal SCHC as an additional compression method for use over PPP.

An example use case for SCHC over PPP over Ethernet (SCHCoPPPoE) is to apply SCHC to periodic flows and maintain them at a protocol-independant size and rate. The constant size may be too small for a particular flow or protocol. The SCHC fragmentation can then be used to transport a protocol data unit (PDU) as N compressed SCHC fragments, in which case the effective PDU rate is the TSN frame rate divided by N.

This can be useful to streamline the frames and simplifies the scheduling of Deterministic Networking [DetNet] and Operational Technology (OT) control flows over IEEE Std 802.1 Time-Sensitive Networking (TSN) [IEEE802.1TSNTG] or one of the [RAW Technologies].

2. BCP 14

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 BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Extending RFC 5172

With this specification, a PPP session defines a vitual link where a SCHC context is established with a particular set of Rules, which is indicated at the set up of the PPP session as follows:

[RFC5172] defines an IPV6CP option called the IPv6-Compression-Protocol Configuration option with a type of 2. The option contains an IPv6-Compression-Protocol field value that indicates a compression protocol and an optional data field as shown in Figure 1:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |   IPv6-Compression-Protocol   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Data ...
+-+-+-+-+
```

Figure 1: The IPv6-Compression-Protocol Configuration Option

This specification indicates a new IPv6-Compression-Protocol field value for [SCHC] (see Section 5), and enables to transport a Uniform Resource Identifier (URI) [RFC3986] of the set of rules in the optional data. The default format for the set of rules is YANG using the "Data Model for SCHC" [SCHC_DATA_MODEL] encoded in JSON as specified in [RFC7951]. The size of the URL is computed based on the

Length of the option as Length-4. If the encoding is asymetrical, the initiator of the session is considered downstream, playing the role of the device in an LPWAN network.

4. Profiling SCHC for high speed links

Appendix D of [SCHC] specifies the profile information that technology specifications such as this must provide. The following section address this requirement.

4.1. Mapping the SCHC Architecture

This specification leverages SCHC between an end point that is an IP Host and possibly a serial DTE (Data Terminal Equipment), and another that is an IP Node (either another IP Host or a Router) and possibly a serial DCE (Data Control Equipment), or a more modern physical or emulated endpoint, e.g., Ethernet devices that echange IP packets over PPPoE.

Both endpoints MUST support the function of SCHC Compressor/ Decompressor (C/D) as shown in Figure 2.

```
+----------+  Wi-Fi /   +----------+                ....
|    IP    |  Ethernet  |    IP    |            ..          )
|   Host   +-----/------+  Router  +----------(   Internet   )
| SCHC C/D |  Serial    | SCHC C/D |            (          )
+----------+            +----------+               ...
            <-- SCHC -->
              over PPP
```

Figure 2: Typical Deployment

The SCHC Fragmenter/Reassembler (F/R) is generally not needed, because the maximum transmission unit (MTU) is expected to be large enough and SCHC only reduces the frame size vs. native IP. But it may be used to obtain a small protocol-independant frame size for the compressed packets, possibly way smaller than MTU.

A context may be generated for a particular upper layer application, such as a control loop using an industrial automation protocol, to protect the particular flow with a DetNet service. The context can be asymetric, e.g., when connecting a primary and a secondary endpoints, a client and a server, or a programmable logic controller with a sensor or an actuator.

4.2. SCHC Parameters

Compared to typical LPWANs, most serial links and emulations such as PPPoE are very fast and most of the constraints can be alleviated. For this reason, the SCHC profile for PPP is defined as follows:

RuleID numbering scheme: The RuleID for a compression rule is expressed as 2 bytes. The first (leftmost) 2 bits of that RuleId MUST be set to 0 This leaves 14 bits to index the rule. A SCHC compressed packet is always in the form:

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+----------...--------+~~~~~~~~~~
|0 0         RuleID             | Compression Residue | Payload
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+----------...--------+~~~~~~~~~~
|------- Compressed Header (byte aligned) ------------|
```

Figure 3: SCHC Compressed Packet

This specification only supports the No-ACK Mode of SCHC fragmentation as specified in section 8.4.1 of [SCHC]. The SCHC Fragment Header is 2 bytes long.

The RuleID for a fragmenation rule is expressed as 4 bits. The bits MUST all set to 1 for a fragmentation rule in No-ACK Mode. The DTag field is 11 bits long (T=11) and the FCN field is one bit (N=1), which is set to 1 on the last fragment as illustrated in Appendix B of [SCHC] and to 0 otherwise. There is no W field (M=0).

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+--------...-------+~~~~~~~~~~~
|1 1 1 1|        DTag         |F| Fragment Payload |  padding
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+--------...-------+~~~~~~~~~~~
        |--------- T ---------|N|
|---- SCHC Fragment Header  ----|
```

Figure 4: SCHC Fragment

The No-ACK mode has been designed under the assumption that data unit out-of-sequence delivery does not occur between the entity performing fragmentation and the entity performing reassembly and a DetNet PREOF function might be needed to reorder the fragments.

Maximum packet size: MAX_PACKET_SIZE is aligned to the PPP Link MTU.

Padding: The Compression Residue MUST be aligned to the L2 word. For Ethernet, the L2 word is one byte, so padding is needed up to the next byte boundary. If a compression rule produces a residue that is not byte aligned, then it is implicitly terminated with a statement that indicates padding till the next byte boundary. The padding bit is 0.

4.2.1. Resulting Packet Format

In the case of PPPoE, the sequence of compression and encapsulation is as follows:

```
   A packet (e.g., an IPv6 packet)
             |                                            ^ (padding bits
             v                                            |   dropped)
    +------------------+                      +--------------------+
    | SCHC Compression |                      | SCHC Decompression |
    +------------------+                      +--------------------+
             |                                            ^
             +--      No        -+                        |
             |   fragmentation   |   +------------------->+
             v                   |   |                    |
    +--------------------+       |   |         +-------------------+
    | SCHC Fragmentation |       |   |         |  SCHC Reassembly  |
    +--------------------+       |   |         +-------------------+
             |                   |   |                    ^
             +<------------------+   |         No         |
             |                       +-- fragmentation   -+
             v                                            |
    +--------------------+                    +--------------------+
    | PPP Session encaps |                    | PPP Session decaps |
    +--------------------+                    +--------------------+
             |                                            ^
             |                                            |
             v                                            |
     +------------------+                      +------------------+
     | PPPoE(oE) encaps |                      | PPPoE(oE) encaps |
     +------------------+                      +------------------+
             |                                            ^
             |                                            |
             +--------------------------------------------+
           Sender                                    Receiver
```

Figure 5: Stack Operation (no fragment)

In the case of PPPoE, a frame that transports an IPv6 packet compressed with SCHC with no fragmentation shows as follows:

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +     Source MAC Address        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                               |                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   Destination MAC Address     +
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Ethernet Frame Type(0x8864)   | Ver=1 | Type=1|   Code=0      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Session ID                |            Length             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | PPP Protocol (IPv6) = 0x0057  |0|0|       SCHC RuleID         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
...                        Compression                          ...
 |                           Residue                     +-+-+-+-+
 |                                                       | Pad   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 |                                                               |
 |                         Uncompressed                          |
...                          Original                           ...
 |                           Payload                             |
 |                                                               |
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: SCHC over PPP over Ethernet Format

4.3. Security Considerations

This draft enables to use the SCHC compression and fragmentation over PPP and therefore Ethernet and Wi-Fi with PPPoE. It inherits the possible threats against SCHC listed in the "Security considerations" section of [SCHC].

5. IANA Considerations

This document requests the allocation of a new value in the registry "IPv6-Compression-Protocol Types" for "SCHC". A suggested value is proposed in Table 1:

Value	Description	Reference
4	Static Context Header Compression (SCHC)	This document

Table 1: IP Header Compression Configuration Option Suboption Types

6. Acknowledgments

7. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2516] Mamakos, L., Lidl, K., Evarts, J., Carrel, D., Simone, D., and R. Wheeler, "A Method for Transmitting PPP Over Ethernet (PPPoE)", RFC 2516, DOI 10.17487/RFC2516, February 1999, <https://www.rfc-editor.org/info/rfc2516>.

[RFC5072] Varada, S., Ed., Haskins, D., and E. Allen, "IP Version 6 over PPP", RFC 5072, DOI 10.17487/RFC5072, September 2007, <https://www.rfc-editor.org/info/rfc5072>.

[RFC5172] Varada, S., Ed., "Negotiation for IPv6 Datagram Compression Using IPv6 Control Protocol", RFC 5172, DOI 10.17487/RFC5172, March 2008, <https://www.rfc-editor.org/info/rfc5172>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[SCHC] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zúñiga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <https://www.rfc-editor.org/info/rfc8724>.

8. Informative References

[SCHC_DATA_MODEL]
Minaburo, A. and L. Toutain, "Data Model for Static Context Header Compression (SCHC)", Work in Progress, Internet-Draft, draft-ietf-lpwan-schc-yang-data-model-03, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-lpwan-schc-yang-data-model-03>.

[RAW Technologies]
Thubert, P., Cavalcanti, D., Vilajosana, X., Schmitt, C., and J. Farkas, "Reliable and Available Wireless Technologies", Work in Progress, Internet-Draft, draft-thubert-raw-technologies-05, 18 May 2020, <https://tools.ietf.org/html/draft-thubert-raw-technologies-05>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[DetNet] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", RFC 8655, DOI 10.17487/RFC8655, October 2019, <https://www.rfc-editor.org/info/rfc8655>.

[IEEE802.1TSNTG]
IEEE, "Time-Sensitive Networking (TSN) Task Group", <https://1.ieee802.org/tsn/>.

Author's Address

Pascal Thubert (editor)
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
06254 Mougins - Sophia Antipolis
France

Phone: +33 497 23 26 34
Email: pthubert@cisco.com