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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 6, 2021.

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

- 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
- TTL stack option (intended use: traceroute)
- Added the "Privacy Considerations" section
- SOCKS-provided DNS: the proxy may provide a valid bind address and port

draft-08

- Removed Address Resolution options
- Happy Eyeballs options
- DNS provided by SOCKS

draft-07

- All fields are now aligned.
- Eliminated version minors
- 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.)

- Renamed the section "Connection Requests" to "Requests"
- Clarified the fact that proxies don't need to support any command in particular.
- Added the section "TCP Fast Open on the Client-Proxy Leg"
- 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

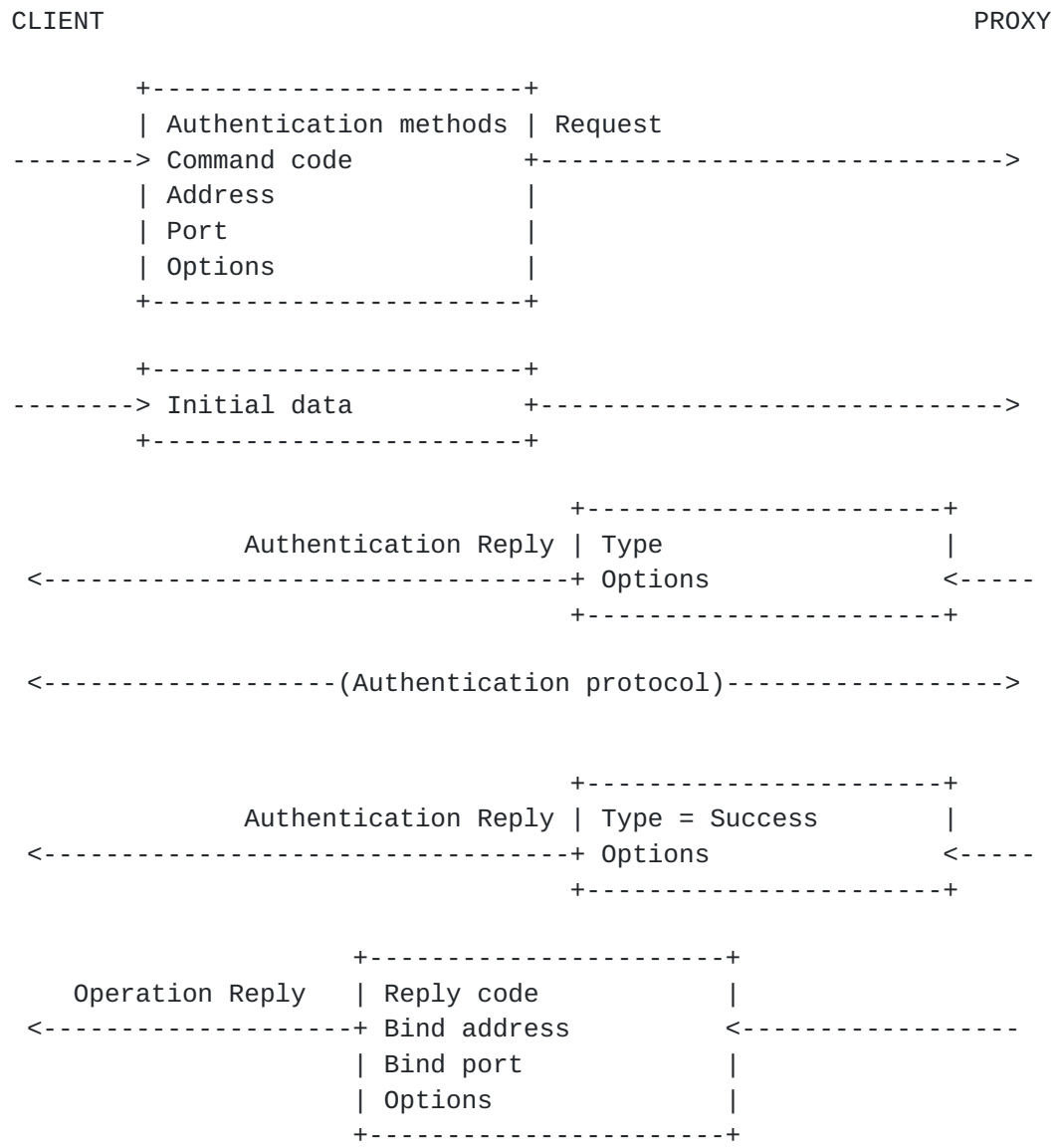

Figure 1: The SOCKS version 6 protocol message exchange

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.

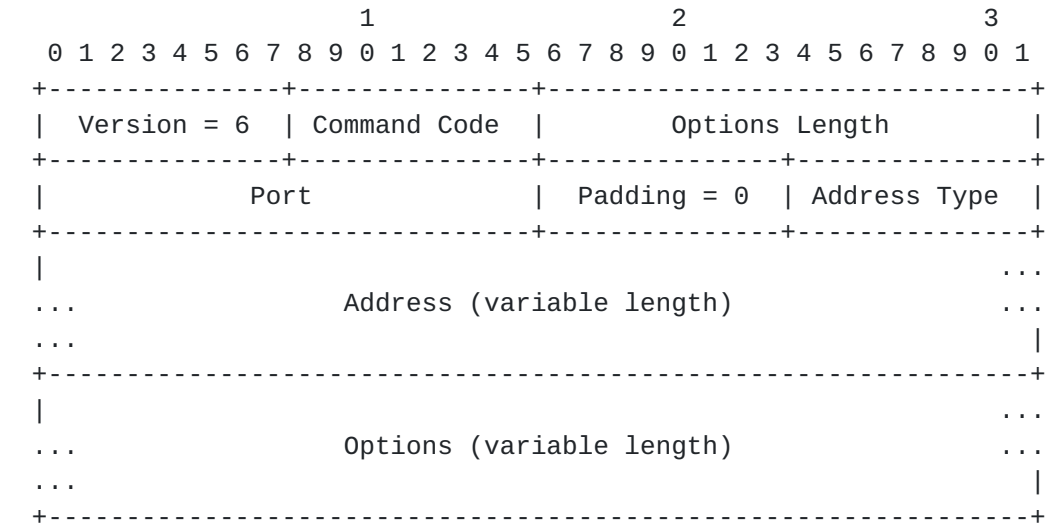

Figure 2: SOCKS 6 Request

- Version: 6
- 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.
- Address Type:
  - 0x01: IPv4
  - 0x03: Domain Name
  - 0x04: IPv6
- 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
- Port: the port in network byte order.
- Padding: set to 0
- Options Length: the total size of the SOCKS options that appear in the Options field. MUST NOT exceed 16KB.
- Options: see Section 8.

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

- 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.

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

- 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:

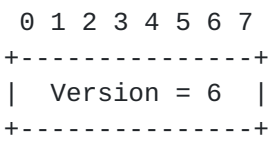

Figure 3: SOCKS 6 Version Mismatch Reply

- 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:

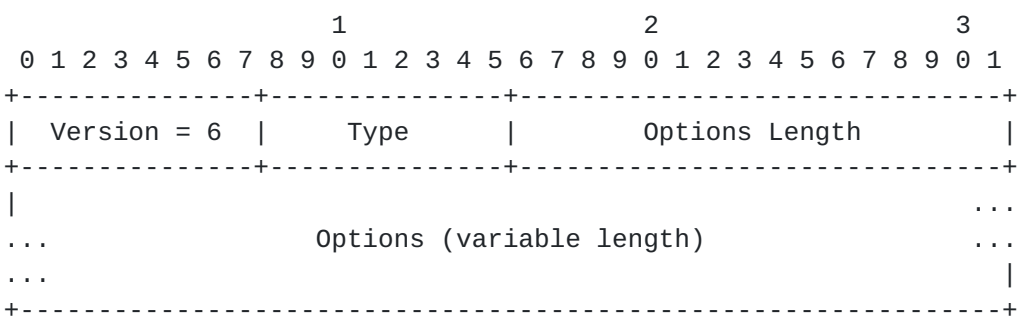

Figure 4: SOCKS 6 Authentication Reply

- 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:

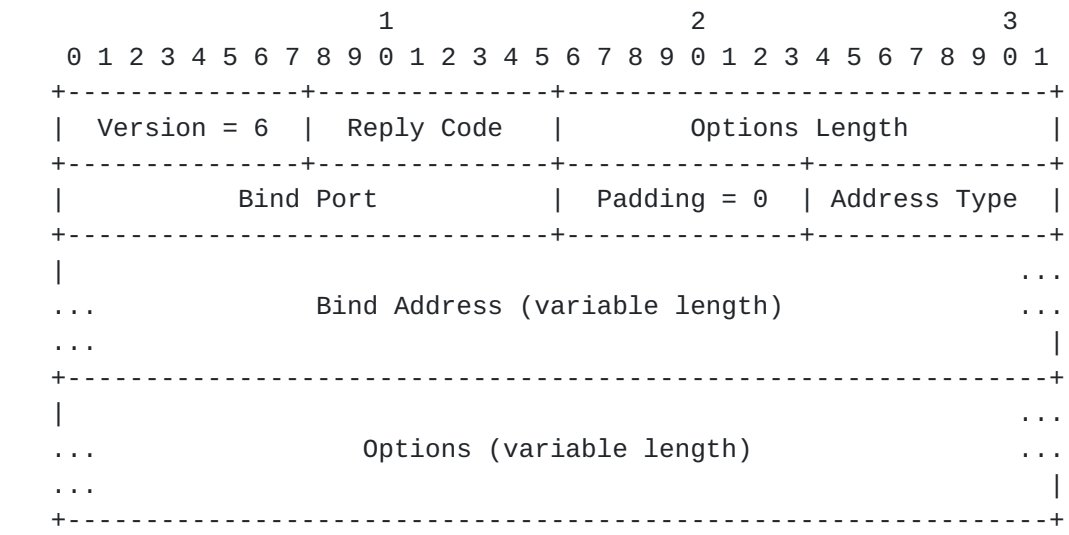

Figure 5: SOCKS 6 Operation Reply

- 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:

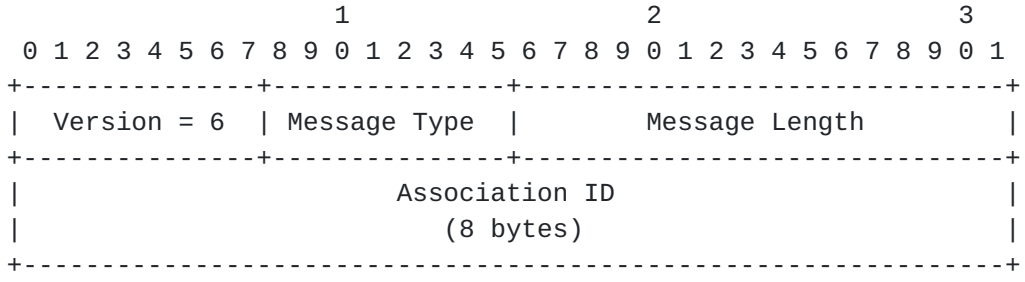

Figure 6: UDP Association Header

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:

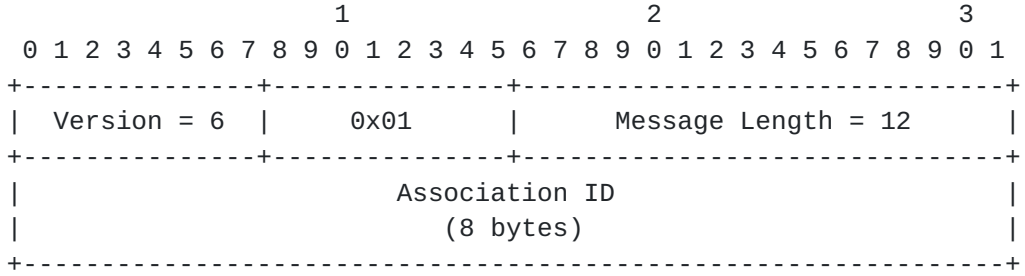

Figure 7: UDP Association Initialization

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

Clients may start sending datagrams to the proxy either:

- over the TCP connection,
- in plaintext, using the proxy's configured UDP port(s), or
- over an established DTLS session.

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

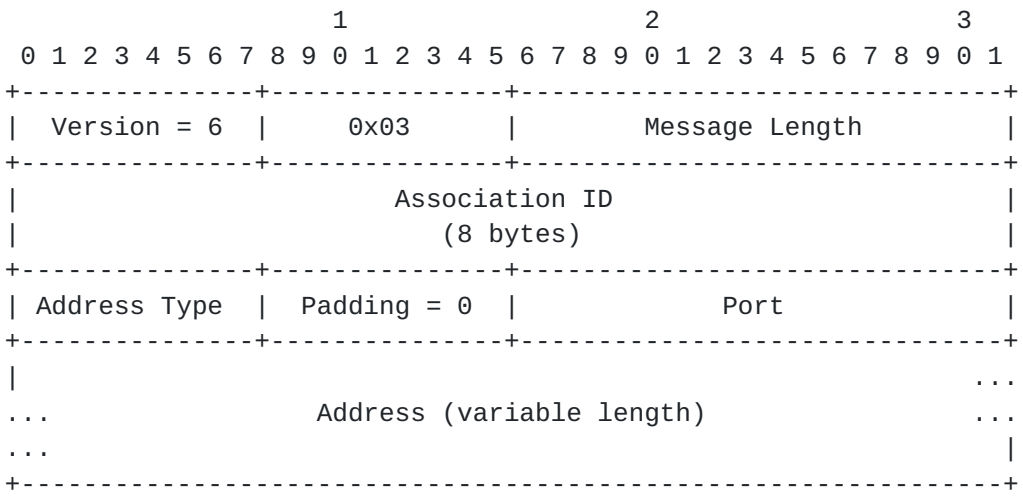

Figure 8: Datagram Header

- Version: 0x06
- Association ID: the identifier of the UDP association
- Address Type:
  - 0x01: IPv4
  - 0x03: Domain Name
  - 0x04: IPv6
- 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

- 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:

- the TCP connection, if it was received over TCP, or
- the 5-tuple of the UDP conversation, if the datagram was received over plain UDP, or
- 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:

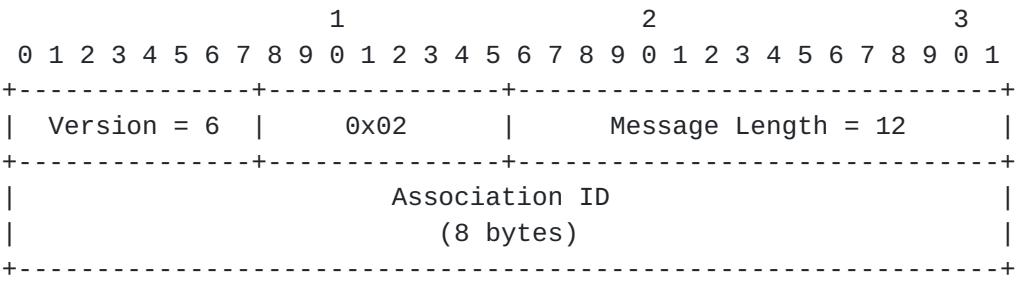

Figure 9: UDP Association Confirmation

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:

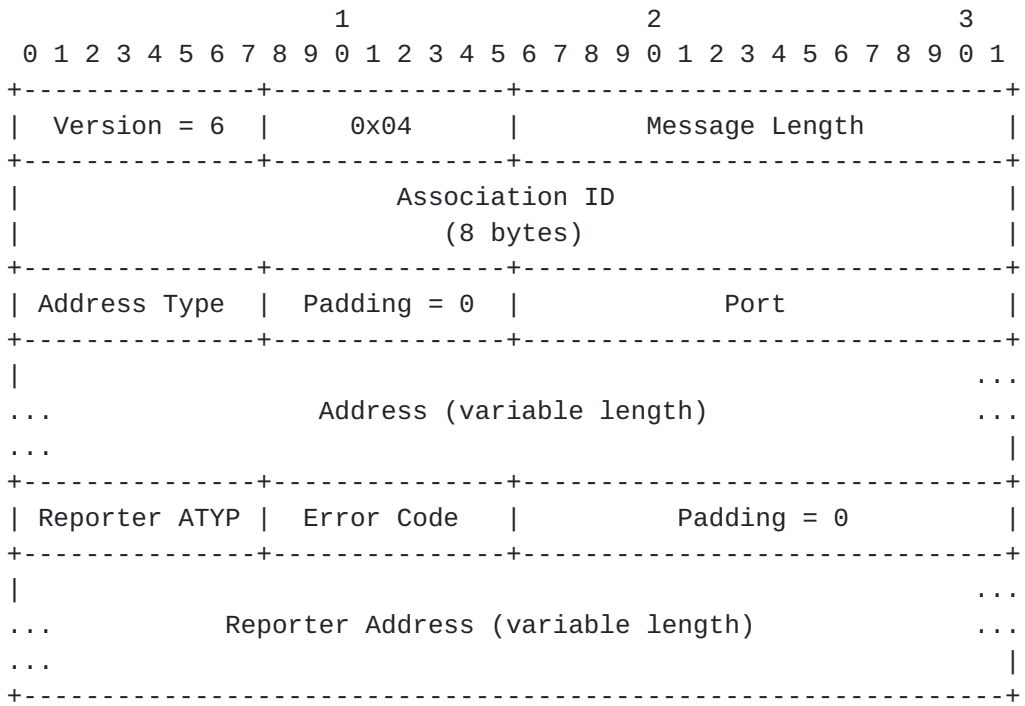

Figure 10: Datagram Error Message

- Address: The destination address of the IP header contained in the ICMP payload

- Address Type: Either 0x01 (IPv4) or 0x04 (IPv6)
- Port: The destination port of the UDP header contained in the ICMP payload
- Reporter Address: The IP address of the host that issued the ICMP error
- Reporter Address Type (ATYP): Either 0x01 (IPv4) or 0x04 (IPv6)
- 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:

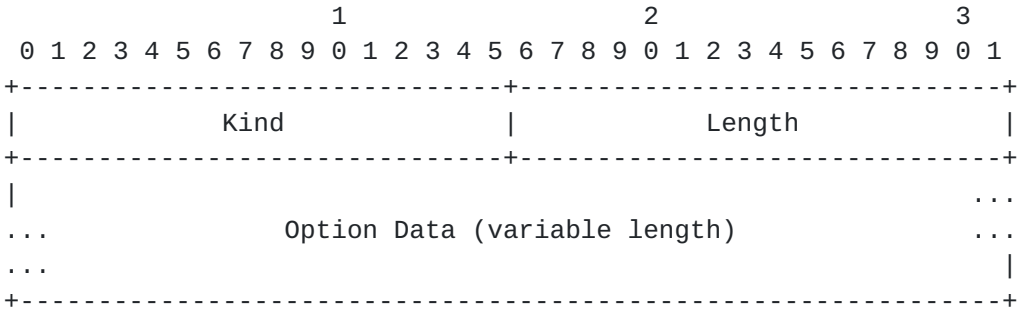

Figure 11: SOCKS 6 Option

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

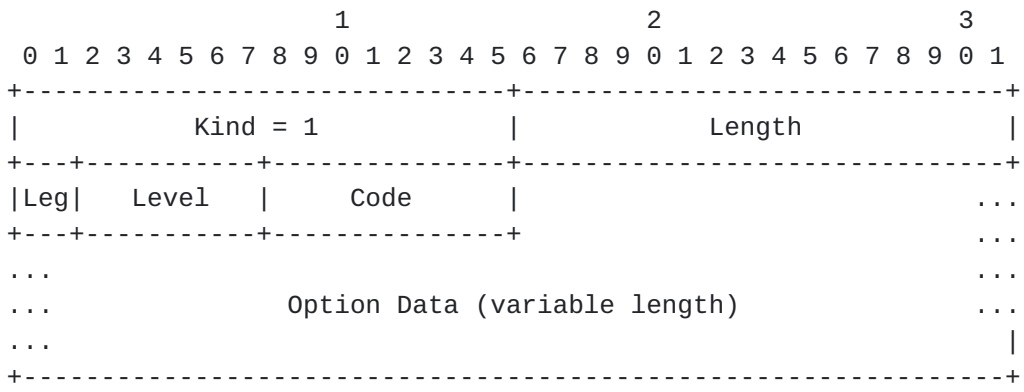

Figure 12: Stack Option

- Leg:
  - 1: Client-Proxy Leg
  - 2: Proxy-Remote Leg
  - 3: Both Legs
- Level:
  - 1: IP: options that apply to either IPv4 or IPv6
  - 2: IPv4
  - 3: IPv6
  - 4: TCP
  - 5: UDP
- Code: Option code
- Option Data: Option-specific data

8.1.1. IP TOS options

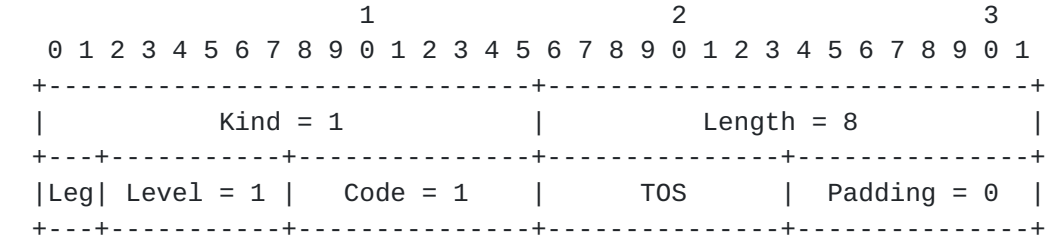

Figure 13: IP TOS Option

- 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

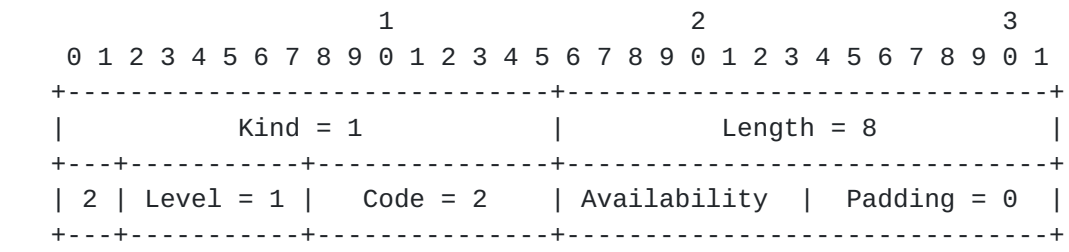

Figure 14: Happy Eyeballs Option

o 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

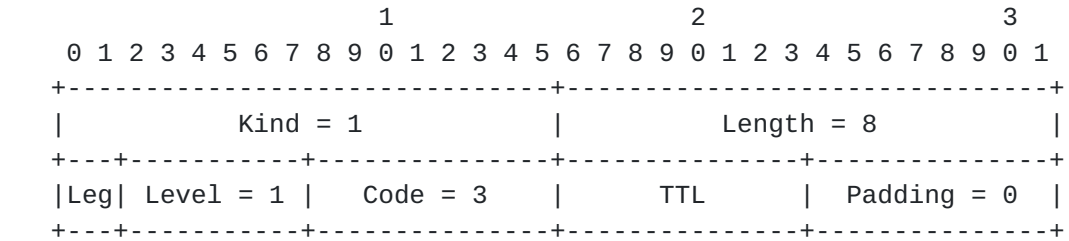

Figure 15: IP TTL Option

o TTL: The IP TTL or Hop Limit

8.1.4. No Fragmentation options

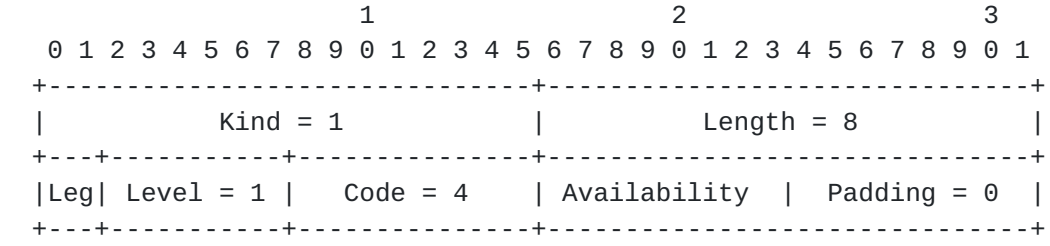

Figure 16: No Fragmentation Option

- 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

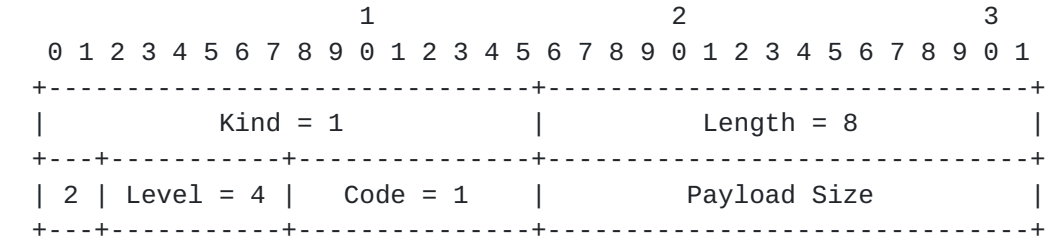

Figure 17: TFO Option

- 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:

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

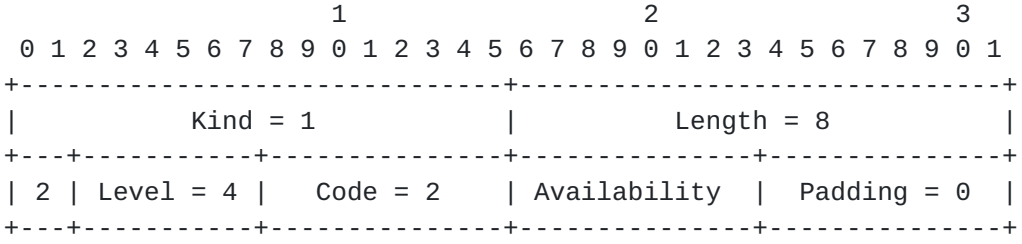

Figure 18: Multipath Option

- 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

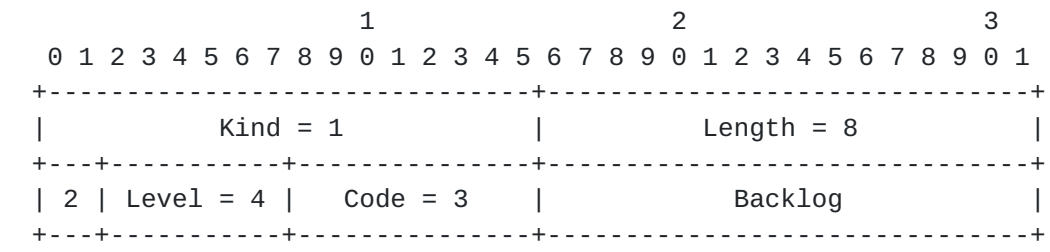

Figure 19: Listen Backlog Option

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

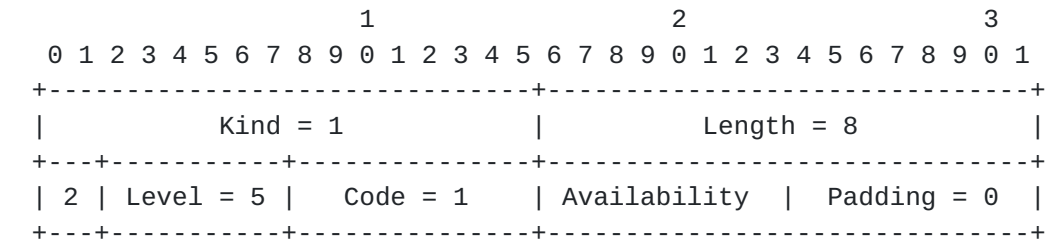

Figure 20: UDP Error Option

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.

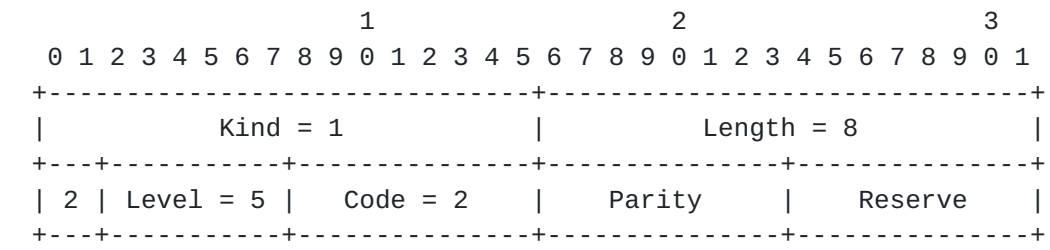

Figure 21: Port Parity Option

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.

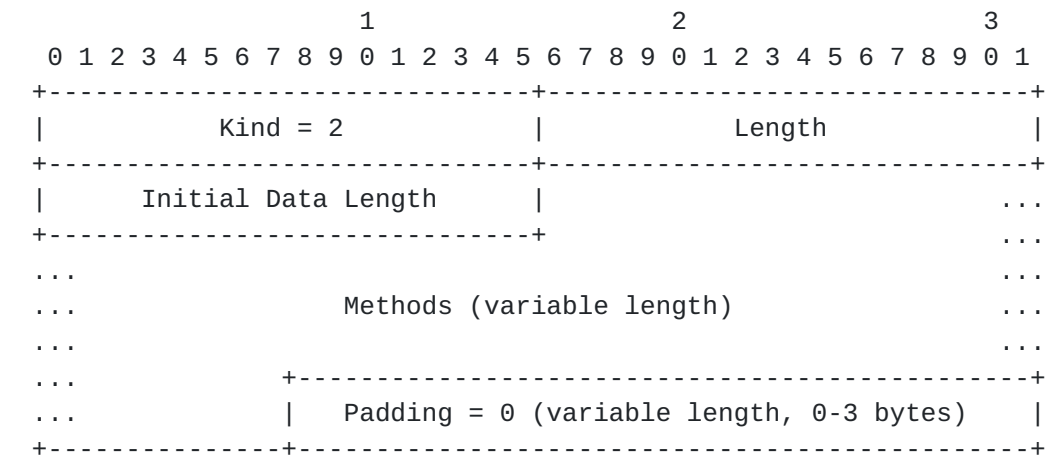

Figure 22: Authentication Method Advertisement Option

- 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.

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

- 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:

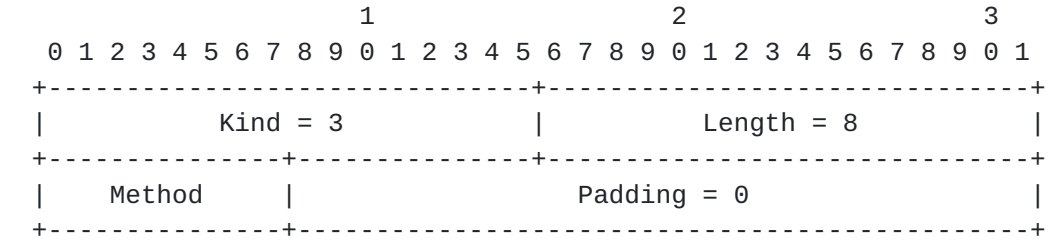

Figure 23: Authentication Method Selection Option

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:

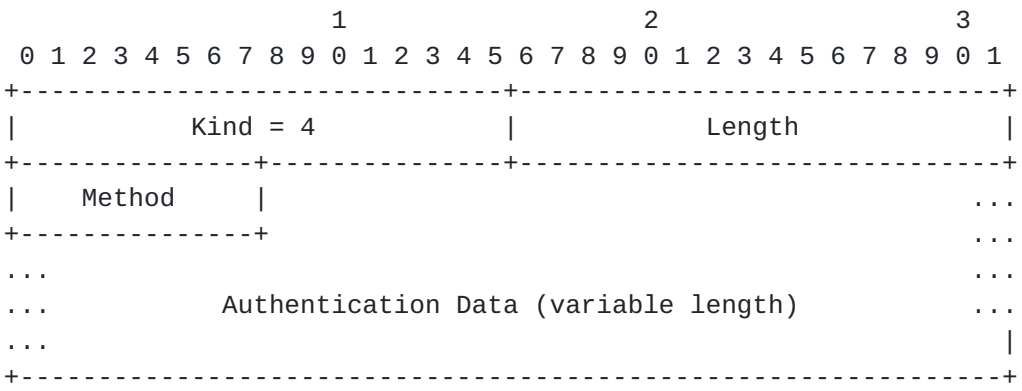

Figure 24: Authentication Data Option

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:

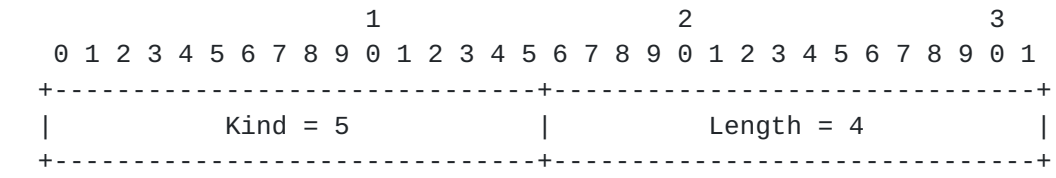

Figure 25: Session Request Option

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

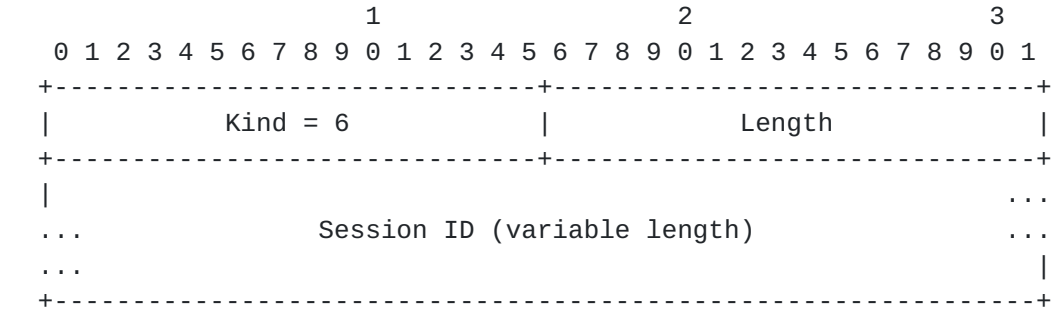

Figure 26: Session ID Option

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:

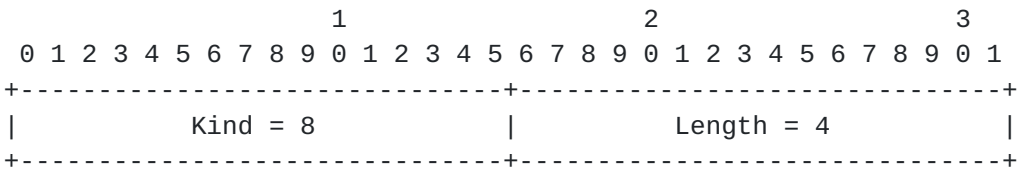

Figure 27: Session OK Option

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:

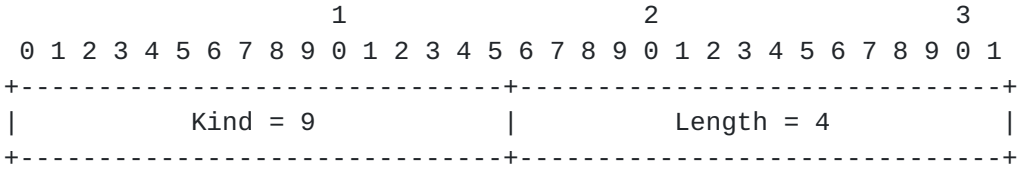

Figure 28: Session Invalid Option

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:

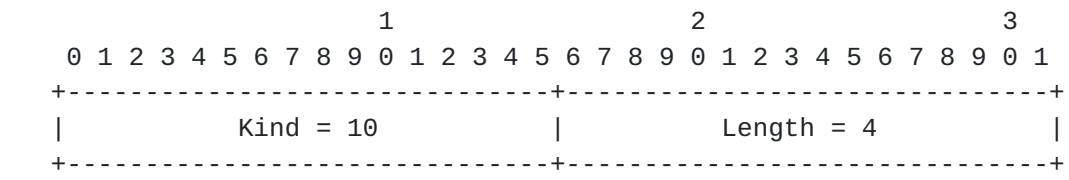

Figure 29: Session Teardown Option

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:

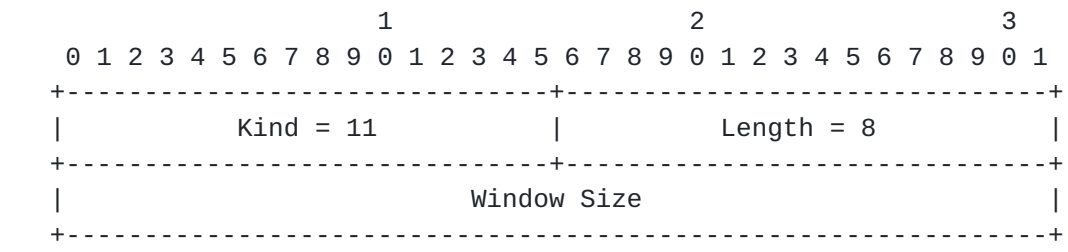

Figure 30: Token Request

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:

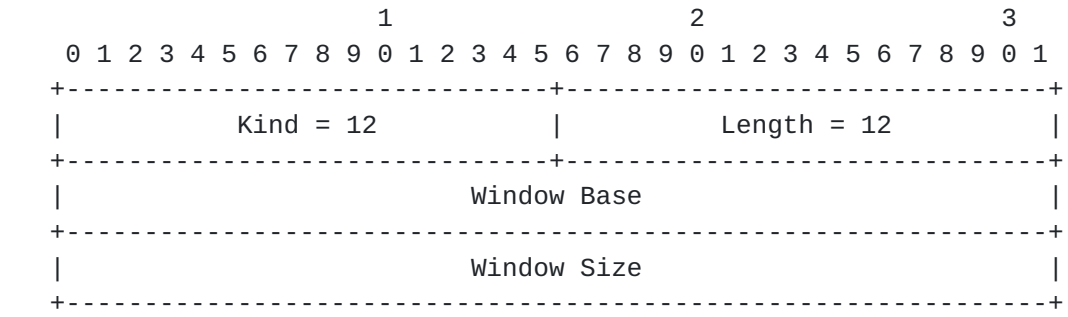

Figure 31: Idempotence Window

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:

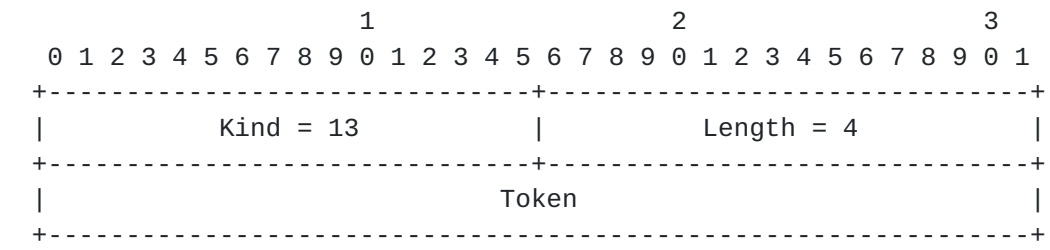

Figure 32: Idempotence Expenditure

- Kind: 13 (Idempotence Expenditure option)
- Length: 8
- 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:

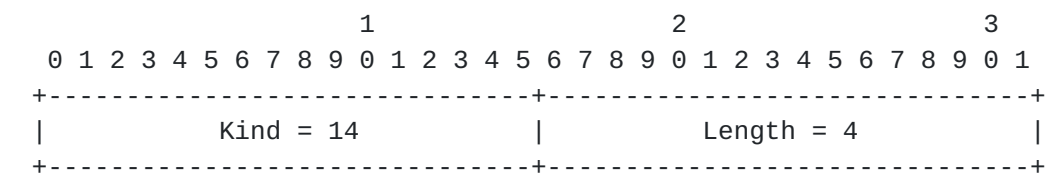

Figure 33: Idempotence Accepted

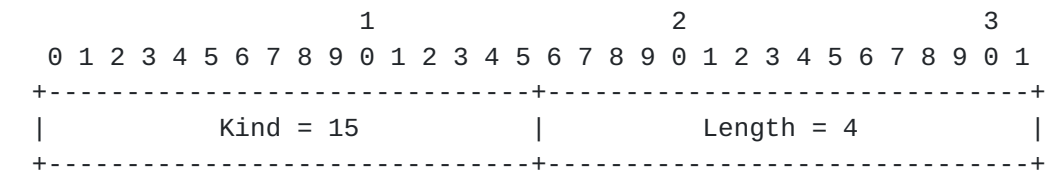

Figure 34: Idempotence Rejected

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.

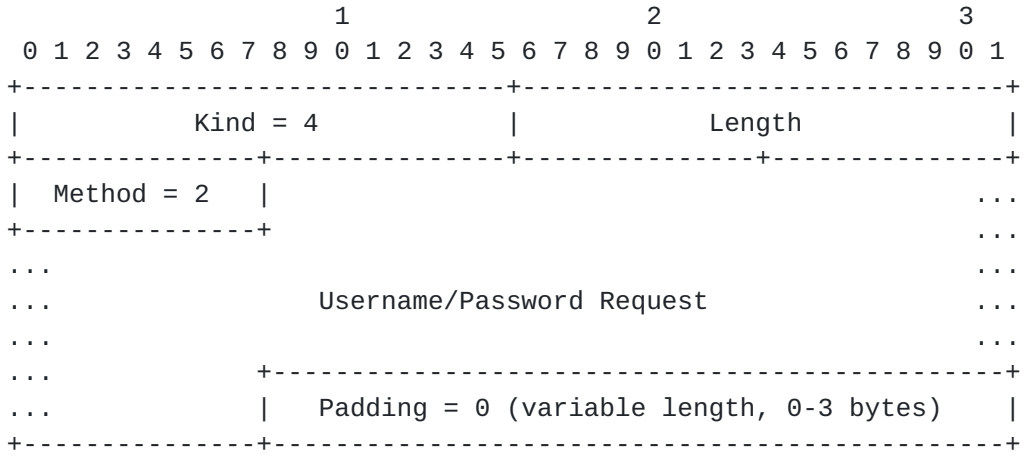

Figure 35: Password authentication via a SOCKS Option

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:

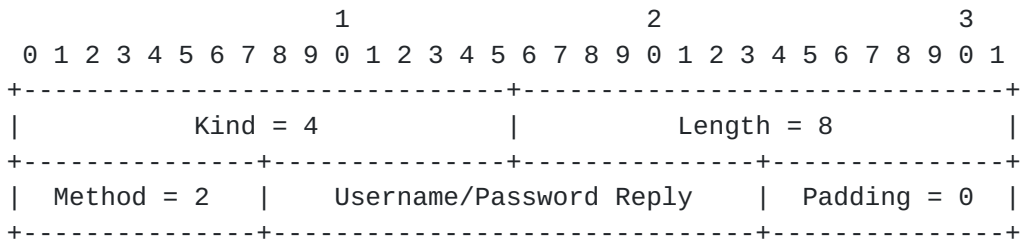

Figure 36: Reply to password authentication via a SOCKS Option

- 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:

- The protocol running on top of SOCKS tolerates the risks of TFO, or
- 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
- The client uses Idempotence Options, making replays very unlikely, or
- 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:

- the request is not fully received after a certain timeout, or
- 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:

- 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.
- 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:

- 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