Internet Area Working Group
Internet-Draft
Intended status: Experimental
Expires: September 6, 2018

V. Olteanu
D. Niculescu
University Politehnica of Bucharest

March 05, 2018

SOCKS Protocol Version 6
draft-olteanu-intarea-socks-6-02

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 2
  1.1. Revision log 3
2. Requirements language 5
3. Mode of operation 5
4. Connection Requests 6
5. Version Mismatch Replies 7
6. Authentication Replies 8
7. Operation Replies 9
  7.1. Handling CONNECT 10
  7.2. Handling BIND 11
  7.3. Handling UDP ASSOCIATE 11
8. SOCKS Options 11
  8.1. Socket options 11
    8.1.1. TFO options 12
    8.1.2. Multipath TCP options 13
    8.1.3. MPTCP Scheduler options 14
  8.2. Authentication Method options 14
  8.3. Authentication Data options 15
  8.4. Idempotence options 16
    8.4.1. Requesting a fresh token window 17
    8.4.2. Spending a token 18
    8.4.3. Handling Token Window Advertisements 19
  8.5. Salt options 19
9. Username/Password Authentication 20
10. Security Considerations 20
  10.1. Large requests 20
  10.2. Replay attacks 21
  10.3. Identical request profiling 21
11. IANA Considerations 21
12. Acknowledgements 22
13. References 22
  13.1. Normative References 22
  13.2. Informative References 22
Authors’ Addresses 22

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 satelite.

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

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

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

- 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-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                  |
     <----------------------------------+ Method                <-----
                                        | Options               |
                                        +-----------------------+

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

                          +-----------------------+
        Operation reply   | Reply code            |
     <--------------------+ Bind address          <------------------
                          | Bind port             |
                          | Options               |
                          | Initial data offset   |
                          +-----------------------+
```

Figure 1: The SOCKS version 6 protocol message exchange

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

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

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

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

4. Connection Requests

The client starts by sending a request to the proxy.

Version		Command	Port	Address	Address	Number of	Options	Initial Data	Initial Data
Major	Minor	Code		Type		Options		Size	
1	1	1	2	1	Variable	1	Variable	2	Variable

Figure 2: SOCKS 6 Request

- Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.

- Command Code:

  * 0x00 NOOP: authenticate the client and do nothing.

  * 0x01 CONNECT: requests the establishment of a TCP connection.

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

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

- 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.

- Number of Options: the number of SOCKS options that appear in the Options field.

- Options: see Section 8.

- Initial Data Size: A two-byte number in network byte order. In case of NOOP, BIND or UDP ASSOCIATE, this field MUST be set to 0. In case of CONNECT, this is the number of bytes of initial data that are supplied in the following field.

- Initial Data: The first octets of the data stream.

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

The server MAY truncate the initial data to an arbitrary size and disregard the rest. This is will be communicated later to the client, should the authentication process be successful (see Section 7). As such, server implementations do not have to buffer the initial data while waiting for the (potentially malicious) client to authenticate.

5. Version Mismatch Replies

Upon receipt of a request starting with a version number other than 6.0, the proxy sends the following response:

```
   +---------------+
   |    Version    |
   | Major | Minor |
   +-------+-------+
   |   1   |   1   |
   +-------+-------+
```

Figure 3: SOCKS 6 Version Mismatch Reply

- Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.

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:

```
   +---------------+------+--------+-----------+----------+
   |    Version    | Type | Method | Number of | Options  |
   | Major | Minor |      |        |  Options  |          |
   +-------+-------+------+--------+-----------+----------+
   |   1   |   1   |  1   |   1    |     1     | Variable |
   +-------+-------+------+--------+-----------+----------+
```

Figure 4: SOCKS 6 Authentication Reply

- Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.

- Type:

  * 0x00: authentication successful.

  * 0x01: further authentication needed.

- Method: The chosen authentication method.

- Number of Options: the number of SOCKS options that appear in the Options field.

- Options: see Section 8.

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

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

The client and proxy begin a method-specific negotiation. During such negotiations, the proxy MAY supply information that allows the client to authenticate a future request using an Authentication Data option. The client and proxy SHOULD NOT negotiate the encryption of the application data. Descriptions of such negotiations are beyond the scope of this memo.

7. Operation Replies

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

```
+-------+---------+------+----------+--------------+
| Reply | Address | Bind |   Bind   | Initial Data |
| Code  |  Type   | Port | Address  |    Offset    |
+-------+---------+------+----------+--------------+
|   1   |    1    |  2   | Variable |      2       |
+-------+---------+------+----------+--------------+
+-----------+----------+
| Number of | Options  |
|  Options  |          |
+-----------+----------+
|     1     | Variable |
+-----------+----------+
```

Figure 5: SOCKS 6 Operation Reply

- Reply Code:

  * 0x00: Succes

  * 0x01: General SOCKS server failure

  * 0x02: Connection not allowed by ruleset

  * 0x03: Network unreachable

  * 0x04: Host unreachable

  * 0x05: Connection refused

  - 0x06: TTL expired
  - 0x07: Command not supported
  - 0x08: Address type not supported

- 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.
  - IPv6: a 16-byte IPv6 address

- Bind Port: the proxy bound port in network byte order.

- Number of Options: the number of SOCKS options that appear in the Options field.

- Options: see Section 8.

- Initial Data Offset: A two-byte number in network byte order. In case of BIND or UDP ASSOCIATE, this field MUST be set to 0. In case of CONNECT, it represents the offset in the plain data stream from which the client is expected to continue sending data.

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

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. The client MUST resume the data stream at the offset indicated by the Initial Data Offset field.

7.2. Handling BIND

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

7.3. Handling UDP ASSOCIATE

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

8. SOCKS Options

SOCKS options have the following format:

Kind	Length	Option Data
1	1	Variable

Figure 6: SOCKS 6 Option

- o Kind: MUST be allocated by IANA. (See Section 11.)

- o Length: The length of the option.

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

8.1. Socket options

Socket options are be used by clients to alter the behavior of the sockets created by the proxy. A socket option can affect either the proxy’s socket on the client-proxy leg or on the proxy-server leg. Clients can only place Socket options inside SOCKS Requests.

Proxies MAY include Socket options in their Operation Replies to signal their sockets’ behavior. Said options MAY be unsolicited, i. e. the proxy MAY send them to signal behaviour that was not explicitly requested by the client.

Kind	Length	Leg	Level	Code	Data
1	1	2 bits	6 bits	1	Variable

Figure 7: Socket Option

- Kind: MUST be allocated by IANA. (See Section 11.)
- Length: The length of the option.
- Leg:
  - 0x1: Client-Proxy Leg
  - 0x2: Proxy-Server Leg
  - 0x3: Both Legs
- Level:
  - 0x01: Socket
  - 0x02: IPv4
  - 0x03: IPv6
  - 0x04: TCP
  - 0x05: UDP
- Code: Option code
- Data: Option-specific data

8.1.1. TFO options

Kind	Length	Leg	Level	Code
1	1	2 bits	6 bits	1

Figure 8: TFO Option

- Kind: MUST be allocated by IANA. (See Section 11.)

- Length: MUST be 4.

- Leg: MUST be 0x2 (Proxy-Server Leg).

- Level: 0x04 (TCP).

- Code: 0x17

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 proxy MAY include a TFO option in the Operation reply if TFO was attempted, the operation succeded and the remote server supports TFO. In case of a BIND command, the proxy MAY include a TFO option in the first Operation reply to signal that it will accept an incoming TFO connection.

8.1.2. Multipath TCP options

In case of a CONNECT command, the proxy can inform the client that the connection to the server is an MPTCP connection.

```
+---------------+--------+--------+------+
| Kind | Length |  Leg   | Level  | Code |
+------+--------+--------+--------+------+
|  1   |   1    | 2 bits | 6 bits |  1   |
+------+--------+--------+--------+------+
```

Figure 9: Multipath TCP Option

- Kind: MUST be allocated by IANA. (See Section 11.)

- Length: 4.

- Leg: MUST be 0x2 (Proxy-Server Leg).

- Level: 0x04 (TCP).

- Code: 0x2a

8.1.3. MPTCP Scheduler options

In case of a CONNECT or BIND command, a client can use an MPTCP Scheduler option to indicate its preferred scheduler for the connection.

A proxy can use an MPTCP Scheduler option to inform the client about what scheduler is in use.

Kind	Length	Leg	Level	Code	Scheduler
1	1	2 bits	6 bits	1	1

Figure 10: MPTCP Scheduler Option

- Kind: MUST be allocated by IANA. (See Section 11.)
- Length: MUST be 5.
- Leg: Either 0x01, 0x02, or 0x03 (Client-Proxy, Proxy-Client or Both legs).
- Level: 0x04 (TCP).
- Code: 0x2b
- Scheduler:
  * 0x00: Default
  * 0x01: Round-Robin
  * 0x02: Redundant

8.2. Authentication Method options

Authentication Method options are used by clients to advertise supported authentication methods. They can be part of SOCKS Requests.

Kind	Length	Methods
1	1	Variable

Figure 11: Authentication Method Option

- o Kind: MUST be allocated by IANA. (See Section 11.)

- o Length: The length of the option.

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

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

Kind	Length	Method	Authentication Data
1	1	1	Variable

Figure 12: Authentication Data Option

- o Kind: MUST be allocated by IANA. (See Section 11.)

- o Length: The length of the 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 MAY omit advertising authentication methods for which they have included at least an Authentication Data option.

8.4. Idempotence options

To protect against duplicate SOCKS Requests, authenticated 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. Windows can be shifted (i. e. have their base increased, while retaining their size) unilaterally by the proxy.

Requesting and spending tokens is done via Idempotence options:

Kind	Length	Type	Option Data
1	1	1	Variable

Figure 13: Idempotence Option

- Kind: MUST be allocated by IANA. (See Section 11.)
- Length: The length of the option.
- Type:
  - 0x00: Token Request
  - 0x01: Token Window Advertisement
  - 0x02: Token Expenditure
  - 0x03: Token Expenditure Reply
- Option Data: The contents are specific to each type.

All proxy implementations MUST support Idempotetence options, even if they do not issue token windows.

8.4.1. Requesting a fresh token window

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

```
+---------------+------+-------------+
| Kind | Length | Type | Window Size |
+------+--------+------+-------------+
|  1   |   1    |  1   |      4      |
+------+--------+------+-------------+
```

Figure 14: Token Request

- Kind: MUST be allocated by IANA. (See Section 11.)
- Length: 7
- Type: 0x00 (Token Request)
- Window Size: The requested window size.

If a token window is issued, the proxy then includes a Token Window Advertisement option in the corresponding Operation Reply:

```
+---------------+------+-------------+-------------+
| Kind | Length | Type | Window Base | Window Size |
+------+--------+------+-------------+-------------+
|  1   |   1    |  1   |      4      |      4      |
+------+--------+------+-------------+-------------+
```

Figure 15: Token Window Advertisement

- Kind: MUST be allocated by IANA. (See Section 11.)
- Length: 11
- Type: 0x01 (Token Grant)
- Window Base: The first token in the window.
- Window Size: The window size. This value SHOULD be lower or equal to the requested window size. Window sizes MUST be less than 2^31.

If no token window is issued, the proxy MUST silently ignore the Token Request.

8.4.2. Spending a token

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

Kind	Length	Type	Token
1	1	1	4

Figure 16: Token Expenditure

- Kind: MUST be allocated by IANA. (See Section 11.)
- Length: 7
- Type: 0x02 (Token Expenditure)
- Token: The token being spent.

Clients SHOULD prioritize spending the smaller tokens.

The server responds by sending a Token Expenditure Reply option as part of the Operation Reply:

Kind	Length	Type	Response Code
1	1	1	1

Figure 17: Token Expenditure Response

- Kind: MUST be allocated by IANA. (See Section 11.)
- Length: 4
- Type: 0x03 (Token Expenditure Response)
- Response Code:
  * 0x00: Success: The token was spent successfully.
  * 0x01: No Window: The proxy does not have a token window associated with the client.

   - 0x02: Out of Window: The token is not within the window.

   - 0x03: Duplicate: The token has already been spent.

If eligible, the token is spent as soon as the client authenticates. If the token is not eligible for spending, the proxy MUST NOT attempt to honor the client’s SOCKS Request; further, it MUST indicate a General SOCKS server failure in the Operation Reply.

Proxy implementations SHOULD also send a Token Window Advertisement if:

- the token is out of window, or

- by the proxy’s internal logic, successfully spending the token caused the window to shift.

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

Proxy implementations MAY include a Token Window Advertisement in any Operation Reply.

8.4.3. Handling Token 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 unsollicited Token Window Advertisements with a Window Base less than the previously known value.

8.5. Salt options

Clients can use Salt options so that otherwise identical requests are unique. (See Section 10.3.)

```
+---------------+------+
| Kind | Length | Salt |
+------+--------+------+
|  1   |   1    |  4   |
+------+--------+------+
```

Figure 18: Salt Option

- Kind: MUST be allocated by IANA. (See Section 11.)

- Length: 6

- Salt: An arbitrary value.

Proxies MUST silently ignore Salt options.

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, provided that it is no longer than 252 bytes.

```
+---------------+--------+---------------------------+
| Kind | Length | Method | Username/Password request |
+------+--------+--------+---------------------------+
|  1   |   1    |   1    |          Variable         |
+------+--------+--------+---------------------------+
```

Figure 19: Password authentication via a SOCKS Option

- Kind: MUST be allocated by IANA. (See Section 11.)

- Length: The length of the option.

- Method: 0x02 (Username/Password).

- Username/Password request: The Username/Password request, as described in [RFC1929].

10. Security Considerations

10.1. Large requests

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

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

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

- the number of options exceeds an imposed hard cap, or

- the total size of the options exceeds an imposed hard cap, or

- the size of the initial data excedes a hard cap.

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

10.2. 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, the initial Token Request is still vulnerable. As such, client implementations SHOULD NOT make use of TLS early data when sending a Token Request.

10.3. Identical request profiling

If sent via TLS early data, identical SOCKS requests can also be identical on the wire. An attacker with the capability to capture a client’s SOCKS traffic can attempt to profile it by identifying identical requests.

A client can use Salt options to make all of its requests unique.

11. IANA Considerations

This document requests that IANA allocate 1-byte option codes for SOCKS 6 options. Further, this document requests option codes for:

- Socket options

- Authentication Method options

- Authentication Data options

- Idempotence options

- Salt options

- Vendor-specific options

This document also requests that IANA allocate a port for SOCKS over TLS.

12. Acknowledgements

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

13. References

13.1. 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>.

13.2. Informative References

[I-D.ietf-tls-tls13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-26 (work in progress), March 2018.

[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>.

[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>.

Authors’ Addresses

Vladimir Olteanu
University Politehnica of Bucharest

Email: vladimir.olteanu@cs.pub.ro

Dragos Niculescu
University Politehnica of Bucharest

Email: dragos.niculescu@cs.pub.ro