Internet-Draft   Alexander Bogdanov
draft-bogdanov-ehtp-00.txt   July 2002
Expires: January 2003

End-by-Hop Transmission Protocol

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026 [1].

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

End-by-Hop Transmission Protocol (EHTP) is the connection-oriented transport service for the reliable or unreliable delivery of data packets with possible violation of a sequence. It has the own address space compatible with Unified Memory Space Protocol (UMSP, RFC3018 [5]). EHTP includes the gateway protocol, which supports labels and dynamic resources deallocating. Networks with non-overlapping or incompatible addresses space may be united at EHTP layer with end-to-end interaction and with preservation of a transparency.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [2].

The options names and the headers fields identifiers written by the letters in upper case. At that, the words in the options names are divided by a hyphen, and in the fields identifiers by the underlining symbol.

Table of contents

1 Introduction

EHTP is the connections-oriented transport service for reliable or unreliable delivery of data packets with possible violation of a sequence. It uses the service of unreliable datagram delivery at the lower layer. EHTP is oriented for working with Unified Memory Space Protocol [5] at the upper layer. Nevertheless, EHTP is the universal protocol, with a condition, that the upper layer provides packetization.

EHTP has the own address space, defined over addresses space of the network layer. It is stipulated the protocol of gateway. The endpoint may be connected immediately to global IP network or to work through a gateway. The protocol of EHTP gateway does not include the routing protocol. It is supposed, that the basic work on routing is implemented at a network layer. The definition of EHTP gateways addresses is beyond the scope of this document. The protocol requires that the node have known at least the one gateway address.

EHTP does not provide buffering the received data. All received packets are sending to the upper layer at once. Consequently, the protocol has no the fragmentation function and does not provide the ordered data flows. This decision is based on the supposition, that the allocation of resources, except for minimally necessary for reliable data exchange, should make at application layer. The functional purpose of transmitted data is known at this layer, and it is possible to denial of low-priority traffic service at overloading.

The protocol defines a 4-way handshake establishment of primary connections. All basic functions of connections control are carried out at primary connections layer. Primary connections can be used for the accelerated establishment of 2-way handshaking secondary connection. Primary connection with indefinite port number is stipulated. This connection can be used for sending a connectionless traffic (for the upper layer). The upper layer traffic of any connection can request or not request delivery acknowledgement.

The gateway protocol defines the mixed routing based on addresses and labels. Labels are distributed at a primary connection establishment. The possibility of a connection establishment through the explicit route is stipulated. The protocol gives the mechanism of dynamic resources deallocating on gateways of the explicit route at absence of traffic during established time. Dynamic resource redistribution is executed transparent for the upper layer.

The UDP [3] using at lower layer is specified in this document for EHTP. Allocated IANA port is 1295. The logic, at which UDP is used only at a connection establishment and by sending the big packets, can be used. After connection establishment on the fixed hops, the second layer protocol immediately can be used. The small size of the service information (8 bytes for unreliable data delivery), allows immediately using lower layer service with the small packet size. This document defines UDP using and does not consider other protocols. Nevertheless, any lower layer protocol if it allows identifying EHTP packets, can be used on the fixed hops.

EHTP is the new protocol, and it requires to develop new application programs or to update existing for immediate use of its services. At the same time, it is possible to develop intermediate sublevels above EHTP, which emulate the existing standard protocols services, for example TCP.

Presence of the gateway with state protocol allows to create the multilevel protected systems and to use EHTP for sending the traffic with QoS. Besides, the gateway protocol gives a possibility to unite the switching packets networks with switching channels networks in any combination. This document contains the description of the base protocol and does not consider these questions.

2 Terminology

Node - a device that implements EHTP.

Gateway - an intermediate node that forwards EHTP packets. The gateway always has its own EHTP address.

MTU - maximum packet size in bytes, that can be conveyed between adjacent EHTP nodes without fragmentation on lower layer.

PMTU - minimum MTU of all path hops between a sender node and a receiver node.

Command - an option formed by an endpoint in the EHTP packet, defines operations at EHTP layer.

Network address - a node address on the network layer, for example IPv4.

Transport address - a node address at an EHTP layer. The transport address includes the information about network type and node address in this network. The transport address may be two-level and include the gateway address in a global network and the node address in a local address space of gateway. The "transport address" term does not include a port. In this document text, the term "address" (without type specification) means the transport address.

3 Addressing

3.1 Transport Addresses

Transport addresses are defined over the network addresses. The node transport address has the globally unique value. It includes the information about a network in which the node is located, and the address in this network. One endpoint MUST have only one transport address. The address MUST NOT change, while it is even one open connection.

EHTP packet contains the sender and the receiver transport addresses. Presence of transport addresses allows gateways to realize delivery of packets between different networks.

The transport address includes three fields:

```
 Bits
  0    1    2    3    4    5    6
 +----+----+----+----+----+----+----------------//------------------+
 |    ADDR_LENGTH    |NET_TYPE |             NODE_ADDR              |
 +----+----+----+----+----+----+----------------//------------------+
```

ADDR_LENGTH: 4 bits

The address length. This field contains the number of bytes in the NODE_ADDR field. Two special values %b0000 and %b1111 is defined. Value %b0000 sets the additional length format for the addresses up to 255 bytes length (see section 3.2.4). Value %b1111 set the length of 16 bytes.

NET_TYPE: 2 bits

The network type. This field in a combination with the ADDR_LENGTH field defines a global network, to which the address refers.

NODE_ADDR: 1 - 255 bytes

The node address in the network. This field contains the node address. This field format and a network in which the node is located, is defined by a combination of NET_TYPE and ADDR_LENGTH fields values. There is no the general algorithm connecting these values with the field format of node address. For each combination of NET_TYPE and ADDR_LENGTH fields values the format is defined separately.

Combination of ADDR_LENGTH = 0 and NET_TYPE = 0 values is reserved.

3.2 Transport Address Formats

3.2.1   Immediate IP Addresses

(1) The following transport address fields values are defined for the node in IPv4 global network:

```
ADDR_LENGTH = 4
NET_TYPE = %b00,%b01

   %b00 - value is defined for the node having the interface
          with IPv4 global network and not supporting EHTP.
          Use of this address type is described in [5].
   %b01 - value is defined for the node having the interface
          with IPv4 global network and supporting EHTP.

NODE_ADDR - The field length is equal to 4 bytes.  The field
            contains the global IPv4 address of the node.
```

(2) The following fields values of transport address are defined for the node, which is taking place in IPv6 network:

```
ADDR_LENGTH = 15. This is the special value of the address
                  length.  The actual field length of NODE_ADDR
                  is 16 bytes.

NET_TYPE = 0
NODE_ADDR - This field contains the full IPv6 address.
```

(3) The following transport address fields value is defined for the node having an interfaces in IPv4 and IPv6 networks simultaneously through the compatible address:

```
ADDR_LENGTH = 4
NET_TYPE = %b10
NODE_ADDR - This field length is equal to 4 bytes.  The field
            contains the global IPv4 node address.
```

The nodes of this type optionally represent the gateways. The used network is fixed in first connection establishment command and may be changed only at reconnection.

3.2.2 Extended IPv4 addresses

The global IPv4 network is considered in the extended addressing, as a network of peer-to-peer gateways. Each gateway has the own local addresses space. The endpoint transport address includes the gateway address in a global network and the local address. The gateway is completely responsible for sending the traffic between nodes in a global network and nodes in a local address space. The local address space is flat on the part of a global network. The internal structure and the transport protocol inside a local zone may be anyone. Compatibility with EHTP at a gateway level is required only. Nodes from a local zone may be located:

- in a local or virtual gateway network,
- in any addressed point of a global network,
- be connected to gateway by not network communications,
- be virtual devices or application programs of gateway.

The local zone may have several peer gateways, which are named a gateways group. Consecutive address numbers in global IPv4 network are reserved for one group of gateways. The group may consist of four or sixteen gateways. Lower bits of address must contain value %b00-11 for group of 4 gateways and %b0000-1111 for group of 16 gateways. The upper bits of address must have one value for one gateways group. Not less than one gateway must work in-group. Unused addresses from a range must be reserved. Gateways must coordinate the addressing policy inside a zone. Any protocol may be used for this. The zone must have only one group of gateways. The node transport address in local zone does not depend on the gateway address, through which the packets exchange.

The following transport address fields values are defined for extended IPv4 addresses:

```
ADDR_LENGTH = 6, 8, 10, 12

NET_TYPE = %b00,%b01,%b10
```

```
         %b00 - for the local zone having one gateway for communication
                with a global network.
         %b01 - for the local zone having group of 4 gateways for
                communication with a global network.
         %b10 - for the local zone having group of 16 gateways for
                communication with a global network.
```

NODE_ADDR - The length is equal to 6, 8, 10, 12 bytes. General NODE_ADDR format for the node from a local zone is the following:

```
         Bytes
           0   1   2   3
         +---+---+---+---+--------//---------+
         |GATEWAY_ADDRESS|   LOCAL_ADDRESS   |
         +---+---+---+---+--------//---------+
```

GATEWAY_ADDRESS: 4 bytes

This field contains the global IPv4 gateway address. If the zone has group of peer gateways, it is the address of the foreground gateway from group, which should be used for communication with this node. If this gateway is inaccessible, the sender from a global network side must search gateways in increasing order numbers of address, since zero (with zero values of lower address bits).

LOCAL_ADDRESS: 2, 4, 6, 8 bytes

This field contains the node address inside a local zone. The protocol does not impose any restrictions on the format and value of this field.

3.2.3 Telephone number

Value such as network NET_TYPE = %b11 is defined for telephone numbers. ADDR_LENGTH address length value defines number length and may be anyone. Full telephone number written in field NODE_ADDR in the packed decimal format (one decimal number in four bits). If the number length is odd, the value %xF written in last four bits.

3.2.4 Additional Length Format

The additional length format is set by special field of length value in transport address ADDR_LENGTH = %b0000. It is intended for addresses, up to 255 bytes size. The NET_TYPE field values don't influence a general format of transport address of this type. The NODE_ADDR field has the following general format:

```
            Bytes
             0         1
            +--------+------------//--------------+
            | TR_LEN |          RA_ADDR           |
            +--------+------------//--------------+

            TR_LEN: 1 byte

               Indicates the length of the RA_ADDR in bytes.

            RA_ADDR: 1-255 byte

               The node address.
```

3.2.4.1 Character Address

This document defines the transport address containing the node address as character ASCII string:

```
      ADDR_LENGTH = 0

      NET_TYPE = %b01

      NODE_ADDR :
         TR_LEN  - The address length
         RA_ADDR - Domain name or character representation of the
                   transport address or URL.
```

3.3 Character Presentation of the Transport Address

The combination of ADDR_LENGTH and NET_TYPE values is named the global network number (or network number) and is represented together by decimal numbers. Initial zero is omitted. The final zero is omitted, if the penultimate number is more than 3, i. e. it may not be considered as a network type. The global network address is written behind a network number through slash "/" by the rules of this network. It may be the gateway address or the endpoint address. For a gateway the local zone node address is written through slash "/", by the rules of addresses writing for this zone. Transport addresses in global IPv4 or IPv6 network may be written without number of a network. For example:

```
      41/198.47.50.3 (or 198.47.50.3) - The endpoint in the IPv4 global
                                        network

      8/198.47.50.3/123.100.27.1 - Endpoint in the local IPv4 network,
                                   working through a single gateway of
                                   global IPv4 network.
```

This document defines only two rules of the addresses writing in a local zone. It may be the local IPv4 address, or representation of any address as decimal number.

The node address may be written down as the domain name or any URL. In that case, the global network number is not presented. It is supposed, that this name has globally unique value.

The number of global telephone network is presented in one value 3 irrespective of length. The network number may be absent, if the telephone number may not be defined as the network number. For example, the number length exceeds three, or the last number is more than five, or the number value is more than 153. Blanks and hyphen digits are ignored. For example: 123 456-7890. Prefixes of an output in a global telephone system are not included in number. The telephone system subscriber address may be presented by alphanumeric line with slash "/" for separating hierarchical components. For example: 3/Moscow/123-45-67. The protocol of transformation of such addresses in a binary kind lies beyond the scope of this document.

The transport address in character representation is entered in EHTP packet in a character address format (see section 3.2.4.1). It is NOT RECOMMENDED to use the character address, if the node may transform it to a binary format.

4 Format of EHTP Packet

4.1 General Format of Packet

Packet EHTP has the following structure:

```
+--------+---------+-------+---------+------------+-------+--------+
|Gateways|Addresses| Basic |Endpoint |    DATA    |PADDING|CHECKSUM|
|Options |Header   | Header|Options  |            |       |        |
+--------+---------+-------+---------+------------+-------+--------+
```

Gateways Options

The gateway options are optional headers with a variable length. Its may be formed by gateways or by an endpoint. On a route of transmission, gateways options may be added, be deleted from a packet, or be modified. In EHTP packet, gateways options MUST be located before the basic header and the addresses header.

Addresses Header

The addresses header is the optional header with variable length. It contains the transport addresses of the sender and of the receiver. The addresses header is formed by the endpoint and it

MUST be located in a packet directly before of the basic header. The addresses header should not be deleted from a packet on gateways and its fields values should not be changed.

Basic Header

The basic header is the obligatory header with the fixed length. It is formed by the endpoint and it contains information, minimally necessary for delivery and processing of packet by the receiver. The fields values of the basic header should not be changed by gateways.

Endpoint Options

The endpoint options are optional headers with variable length. They are formed by an endpoint. They MUST NOT be added, deleted from a packet, or modified on a route of sending. The endpoint options MUST be located after the basic header in EHTP packet.

DATA

It is the optional field, containing the upper layer data. The field length is 0-65535 bytes.

PADDING

These are bytes, which are padded in the end of data field (if necessary) to make a multiple of four bytes. Values may be anyone. At using of the checksum, it is recommended set to 0.

CHECKSUM

The checksum or the data authentication of packet.

All packet headers MUST have a length, which is a multiple of 4 bytes.

One EHTP packet MUST be located in a separate packet of the lower layer.

All headers and options are identified by the first 4 or 5 bits. If the node does not know these bits definition, it MUST silently discard a packet.

4.2 Basic Header

Three formats of base header of 12, 8 and 4 bytes length are defined.

The basic header with a length of 12 bytes MUST be used in packets with commands of a connection establishment. It has the following format:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 0 0 1 0|E|P|R|                SEQUENCE_NUM                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           SERVICE_ID          |          DATA_LENGTH          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         CONNECTION_ID                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

E: 1 bit

Flag of the following option. If it is set, the endpoint option is located after the basic header. If it is not set, the upper layer data are located behind the basic header.

P: 1 bit

Push flag. If it is set, the packet must not be deferred in sending queues. The return packet, containing response of this packet, also must have PSH=1. Push flag does not change a sequence of sending packets, sent on separate connection.

R: 1 bit

Reserved. This bit MUST be set to zero by sending. If this bit is set to 1 on reception, the packet MUST be silently discard without no further action. This bit MUST NOT is analyzed by gateways.

SEQUENCE_NUMBER: 24 bits

A packet sequence number. Numbering is conducted for one connection. It is necessary for calculations to use arithmetic, given in [11]. Value 0 is reserved and it must be excluded at consecutive numeration.

SERVICE_ID: 16 bits

The upper layer service identifier (destination port). This document defines value 2110 for UMSP protocol [5].

DATA_LENGTH: 16 bits

      Indicates the length of DATA field in bytes. A range of
      allowable values is 0 - 65535.

   CONNECTION_ID: 32 bits

      The connection identifier, assigned by a receiver endpoint.

Basic headers with smaller length have the same purpose of fields.

Basic headers of 8 bytes length are using after a connection establishment, if the receiver endpoint can unequivocally identify connection by two lower bytes of CONNECTION_ID field. The header has the following format:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 0 0 1 1|E|P|R|                SEQUENCE_NUM                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          CONNECTION_ID        |          DATA_LENGTH          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Basic headers in length of 4 bytes are used after a connection establishment at an execution of all following conditions:

- o The receiver endpoint can unequivocally identify connection by two lower bytes of CONNECTION_ID field.
- o The data length does not exceed 255 bytes.
- o Data of the upper layer does not require delivery acknowledgement.

The header has the following format:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 0 1 0 0|E|P|R|  DATA_LENGTH  |        CONNECTION_ID          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.3 The Addresses Header

The addresses header contains transport addresses of the sender and the receiver. Fields values of the transport address are given in section 3. The addresses header has the following format:

```
  0                   1
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |1 0 0 0|  SAL  |STP|  RAL  |RTP|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                               |
 ~          SND_NODE_ADDR        ~
 |                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                               |
 ~          RCV_NODE_ADDR        ~
 |                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                               |
 ~          ZERO_PADDING         ~
 |                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

SAL: 4 bits

The ADDR_LENGTH field of the sender transport address (see section 3.1).

STP: 2 bits

The NET_TYPE field of the sender transport address.

RAL: 4 bits

The ADDR_LENGTH field of the receiver transport address.

RTP: 2 bits

The NET_TYPE field of the receiver transport address.

SND_NODE_ADDR: 1-256 bytes

The NODE_ADDR field of the sender transport address.

RCV_NODE_ADDR: 1-256 bytes

The NODE_ADDR field of the receiver transport address.

ZERO_PADDING: 0-3 bytes

Zero addition bytes. They are intended for alignment of the end of addresses header on four bytes border.

If SAL = 0 and STP = 0, field SND_NODE_ADDR is absent.

4.4 Options

Options of a gateway and of an endpoint have a equal format, and differ a position about basic header:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 0 1 0 1|E|D|G|  HEAD_LENGTH  |    OPCODE     | HEADER_DATA1  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ~                          HEADER_DATA2                         ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

E: 1 bit

The purpose of this bit differs for gateways options and the endpoint options:

- For the gateway option, it is the flag of preservation in a stack on the following route hop. If it is not set, the option must be deleted from a packet after processing on the next gateway of explicit route. If flag is set, the option must not be deleted from a packet on gateways.
- For the endpoint option, it is a flag of the following option. If it is not set, the data are located after this option. If the flag is set, the following endpoint option is located after this option.

D: 1 bit

A flag of obligatory processing in the endpoint. It defines the order of an option processing, if the endpoint does not know purpose of this option or may not process it because of any reason. If D = 1, the data transmitted in a packet should be ignored. If D = 0, the packet is received irrespective of a possibility of this option processing. The endpoint must analyze all options of a packet.

G: 1 bit

The flag of obligatory processing on gateways. It defines the order of option processing, if a gateway does not know the purpose of this option or may not process it because of any reason. If G flag is set, the packet must be silently discarded. The gateway must process only its options. The

gateway must not analyze the options of other gateways. If G flag is not set, the packet is forwarded on a route, irrespective of a possibility of this option processing.

HEAD_LENGTH: 8 bits

Indicates the number of 4-byte words in HEADER_DATA2 field (HEADER_DATA1 field always is present at header).

OPCODE: 8 bits

This field value defines operation, which is set by an option.

HEADER_DATA1 + HEADER_DATA2: 1 - 1021 bytes

The format of these fields is defined for each OPCODE value separately.

4.5 Data

DATA field contains the upper layer data. If DATA_LENGTH = 0, the DATA field is absent.

4.6 Checksum

The packet contains the checksum of 4 bytes length by default. This sum is calculated with using of CRC-32 algorithm. Gateways options MUST NOT are included in calculation of the checksum. All other headers, including addresses header, basic header and endpoint options MUST be included in calculation. The upper layer data may be included in calculation not completely. The quantity is defined at a connection establishment (see section 5.1.1.2).

If the Security Parameters Index (SPI) value is defined at a connection establishment, the packet contains authentication data instead of the checksum. The length of authentication data MUST NOT exceed 1024 bytes and MUST be multiple to 4 bytes. This document does not impose any other restrictions on a format of authentication data.

It is supposed, that integrity of data flow is one of the basic upper layer requirements. It is impossible to create the steady distributed systems based on a network, which are not carrying out this requirement. Therefore, the checksum must be used, only if it is impossible to agree on packets authentication parameters.

Irrespective of SPI, gateway options MUST NOT be included in the checksum calculation or authentication data, as they may be modified, added and deleted from a packet on gateways. Thereof, these options

are not protected from distortion and fake. It should be taken into account in the logic of packets processing.

After a connection establishment under some conditions, the addresses header and full basic header may be absent in packets. As the protocol does not guarantee correct packets routing, the full packet including addresses header and 12-byte basic header MUST be used for calculation of the checksum in endpoints. The endpoint of the receiver MUST store these values in state variables. For calculation of the authentication data, the addresses header MUST not be included in calculation, if it is not really sent in a network. The 12-byte basic header MUST be included in calculation of authentication data, irrespective of the real format, transmitted through a network.

5 The Endpoints Protocol

5.1 Connections control

5.1.1 Primary Connection Establishment

The purpose of primary connection establishment is:

- o Transmission of the upper layer data
- o Reservation of resource for the accelerated establishment of secondary connections,
- o Fixing of a route through gateways (see section 6).
- o The establishment of initial connection, if there are switching channels networks in a route.

The endpoints execute during procedure of establishment:

- o exchange of connection identifiers (each side appropriates its identifier to connection),
- o coordinate of initial sequence numbers,
- o set a maximum quantity of secondary connections, which may be in this primary connection,
- o coordinate PMTU.

At the description of a connection establishment procedure in this document, the node that initiates connections is called "initiator". The node that answers a connection establishment is called "responder".

The connection establishment consists of 4-way handshake:

(1) The initiator sends INIT command.
(2) The responder confirms the possibility of connection establishment by sending of INIT-ACK command.

(3) The initiator sends the CONNECT command. Data of the upper layer may be transmitted together with this command.
(4) Responder confirms a connection establishment by sending CONNECT-ACK command to initiator. This command may be transmitted together with the upper layer data.

All connections identifiers, assigned by the defined node, must have the unique values for this node at each moment of time. Identifiers, assigned by different nodes, may coincide. The combination of values <the connection identifier> + <the node transport address> + <time (implicit)> is the global unique key of the connection. Under some conditions, the transport address and/or the full connection identifier can be absent in a packet. For the control of wrong routing, the endpoints (the sender and the receiver) must include globally unique key in calculation of a packet checksum.

Values of connections identifiers must be allocated in view of necessity to reject the packets, sent on already closed or emergency broken off connections. Such packets may be in a network after closing connection for some time. The identifier from the closed connection may be used repeatedly in the following cases:

- o The time passed from the moment of closing connection, exceeding double maximal lifetime of packets in a network.
- o The difference between SEQUENCE_NUMBER for the first sent packet in the new connection and the last value in the closed connection guarantees, that the consecutive increase of sequence numbers in new connection will not achieve the last value of the closed connection during double maximal lifetime of packets in a network. The node does not choose initial sequence number; therefore, this condition may be not executed always.

The connections identifiers, established right after the emergency reload of the node, also must take into account these restrictions.

The node is completely responsible for allocated identifiers and may change or add the given rules. For increase of security from attacks, it is recommended, that connections identifiers would have values, which are difficult for predicting.

To use the short-cut format of base header, an endpoint must assigned connections identifiers with two lower bytes, unique for the node. In this case, at a connection establishment and at calculation of the checksum, full 32-bit identifier value is used. Packets after a connection establishment can include only two low bytes of the identifier. Right after connection closing, the value of low bytes can be used in new connection, if higher bytes have the other value.

5.1.1.1 INIT, INIT-ACK

INIT command is used on a first step of primary connection establishment. INIT-ACK command is the positive response to INIT command. These commands MUST be formed as a gateway options. The packet containing these commands, MUST NOT includes the upper layer data. Fields of base header must have the following values:

```
   SEQUENCE_NUM -  In INIT command, this field is set to zero on
                   transmit and ignored on receipt.  For INIT-ACK
                   command, the packet sender allocates this field
                   value.  It MUST be copied by the receiver in
                   SEQUENCE_NUM field of base header of a packet with
                   CONNECT command.  Values 0 and %xFFFFFF are
                   reserved.
   SERVICE_ID -    This field contains the service identifier of
                   established connection.
   DATA_LENGTH =   0
   CONNECTION_ID - Values 0 and %xFFFFFFFF are reserved.  In INIT
                   command, this field is not used by the protocol
                   and may contain any value.  In INIT-ACK command,
                   value of this field is copied from
                   TMP_CONNECTION_ID field of INIT command.
```

The options of INIT and INIT-ACK commands have an identical format and differ only in OPCODE value:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 0 1 0 1|E|D|G|  HEAD_LENGTH  |    OPCODE     |RES_VER_FLAGS|F|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |RSV|         MIN_HL_PMTU       |RSV|            PMTU           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       TMP_CONNECTION_ID                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              SPI                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The fields of general format have the following values:

```
      E = 1
      D = 1
      G = 1
      HEAD_LENGTH = 2/3 (depends on SPI presence)
      OPCODE =
               1 for INIT
               2 for INIT-ACK
```

RES_VER_FLAGS: 7 bits

The reserved versions flags. Values of these bits are set to zero on transmit. If even one of them is set to 1 at receiving, the packet must be silently rejected. Each new version of the protocol will use one bit of this field. The endpoint has a possibility to provide a few protocol versions. INIT command may contain a several not zero bits of the version. INIT-ACK command MUST contain only one not zero bit of the version.

F: 1 bit

Flag of the protocol first version. This bit MUST be set to 1. It sets the protocol version, which is defined in this document.

RSV: 2 + 2 bits

Reserved. These fields should be set to zero on transmit and ignored on receipt.

MIN_HL_PMTU: 14 bits

Indicates the minimal size of upper layer PMTU in 32-bit words. This value sets the upper layer at request of connection establishment. Connection PMTU MUST NOT be less than this value. MIN_HL_PMTU field indicates the payload size.

PMTU: 14 bits

Indicates the real PMTU size in 32-bit words. This value defines the full EHTP packet size, including all necessary headers and an authentication data. PMTU value may be reduced by gateways up to size from MIN_HL_PMTU field in view of EHTP headers length. If the gateway may not provide the minimally necessary value, the following variants are possible:

- o MTU, which may be provided, is written in packet, and the packet is sent forward. The INIT receiver forms INIT-ACK command with changed PMTU. The initiator of connection establishment may silently discard INIT-ACK, if PMTU does not arrange it.
- o Boundary nodes of a hop, which may not provide PMTU, fix themselves in connection route (see section 6). All subsequent connection packets are fragmented and assembled on these boundary nodes by lower level

protocol. A fragmentation MUST NOT is for end-to-end interactions at EHTP layer.

PMTU connection MAY be less, than a size of packets with connection establishment commands (for example, if connection requires the small packets size for maintenance of desired QoS).

TMP_CONNECTION_ID: 32 bits

Indicates the temporary identifier, which assigned the sender of this packet. It MUST be copied by the receiver in base header CONNECTION_ID field of a response command.

SPI: 32 bits

This field contains the security parameters index (SPI), which defines the authentication of this packet at EHTP layer. The endpoints must agree upon this field values beforehand. The protocol does not impose any restrictions on SPI values and does not give any rules of its using. SPI field is present at an option, if HEAD_LENGTH = 3. Otherwise, it is absent. Value SPI from INIT and INIT-ACK commands is temporary and may be changed in the following packets of a connection establishment.

The receiver of INIT command must not save a state of connection and allocate resources before correct CONNECT command receiving.

The gateway, immediately connected with the endpoint of INIT receiver, may form INIT-ACK command. For example, if the endpoint is connected on the switched channel.

5.1.1.2 CONNECT, CONNECT-ACK

CONNECT command is sent by the initiator of connection establishment after INIT-ACK command reception. CONNECT-ACK command is sent, as the positive response to CONNECT command on last step of connection establishment. CONNECT and CONNECT-ACK commands MUST be formed as a gateway option. The packet containing these commands MAY include the upper layer data relating to this connection. CONNECT and CONNECT-ACK commands have an identical format and differ only by OPCODE value. Fields of base header must have the following values:

SEQUENCE_NUM -	For CONNECT command this field contains a sequence number from SEQUENCE_NUM field of INIT-ACK command. For command CONNECT-ACK this field contains a sequence number from FIRST_SEQUENCE_NUM field of CONNECT command.

```
      SERVICE_ID -    This field contains the service identifier of
                      established connection.  It must coincide with
                      INIT-ACK command.
      DATA_LENGTH =   0-65535
      CONNECTION_ID - For CONNECT command, this field value is copied
                      from TMP_CONNECTION_ID field of INIT-ACK command.
                      For CONNECT-ACK command, value of this field is
                      copied from PERM_CONNECTION_ID field of CONNECT
                      command.
```

The option of CONNECT and CONNECT-ACK commands has the following format:

```
      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
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |1 0 1 0 1|E|D|G|  HEAD_LENGTH  |    OPCODE     |    SC_MAX     |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |T|V|         MIN_HL_PMTU       |RSV|            PMTU           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | INACTION_TIME |              FIRST_SEQUENCE_NUM               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                      PERM_CONNECTION_ID                       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |RSV|  CHK_LEN  |                 RESERVED                      |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                              SPI                              |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The fields of general format have the following values:

```
         E = 1
         D = 1
         G = 1
         HEAD_LENGTH = 4/5 (depends on SPI presence)
         OPCODE =
                  3 for CONNECT
                  4 for CONNECT-ACK
```

SC_MAX: 8 bits

Indicates the maximal number of secondary connections, which may be connected with this initial. The zero value forbids establishing the secondary connections, connected with this primary connection. Value 255 specifies that the maximum quantity of secondary connections is not fixed at connection establishment and may be anyone. When the limit will be exhausted during work, S-DISCONNECT command with ERROR_CODE = 8

must be used for the connections restriction. Final value of SC_MAX field is defined in CONNECT-ACK command.

T: 1 bit

The short connection identifier flag. If it is set, the packet sender can unequivocally identify connection on lowest two bytes of the PERM_CONNECTION_ID field. After a connection establishment to address of this packet sender, the packets with short base header may be sent.

V: 1 bit

If this bit is set, secondary connections with other SERVICE_ID value may be connected with this primary connection. If this bit is not set, secondary connections MUST be only with same SERVICE_ID value. Final value of the V bit in connection is set in CONNECT-ACK command.

RSV + RESERVED: 2 + 2 + 24 bits

These fields should be set to zero on transmit and ignored on receipt.

MIN_HL_PMTU: 14 bits

Indicates the upper layer PMTU minimal size in 32-bit words (see above section).

PMTU: 14 bits

Indicates the realizable PMTU size in 32-bit words (see above section).

INACTION_TIME: 8 bits

The reserved inactive time. This field contains time interval in seconds, at which expiration gateways will deallocate connection resources at traffic absence (see section 6.8).

FIRST_SEQUENCE_NUM: 24 bits

This field contains the initial sequence number of the receiver. The consecutive numeration of connection packets, which will be sent by this command receiver, must be begun from this number. It is RECOMMENDED, that this field value will be difficult for predicting. For the receiver of CONNECT command, this number MUST be copied in a packet with CONNECT-ACK

command. For the receiver of CONNECT-ACK command, this number MUST be copied in the first connection data packet.

PERM_CONNECTION_ID: 32 bits

This field contains the permanent identifier, assigned to connection by this packet sender.

CHK_LEN: 5 bits

Specifies number of the first 4-byte words from DATA field, which will be included in calculation of the checksum or authentication data. If this field is set to zero, DATA field MUST NOT be included in calculation. If value is set to %b11111, calculation includes full DATA field. Value CHK_LEN is applied to all packets of new connection, transmitted by the sender of this command. Headers EHTP are included in calculation of the checksum irrespective of this field value. If CHK_LEN value to not multiply required value, conditional zero addition is used at calculation of the checksum.

SPI: 32 bits

This field contains the security parameters index (SPI), which defines a packets authentication of this connection at EHTP layer downstream of this packet. Not all subsequent packets of connection contain SPI field. Therefore, this value must be stored, as a state variable of this connection by both endpoints. SPI value from CONNECT-ACK and CONNECT commands may differ. If HEAD_LENGTH = 4, SPI field is absent.

V, SC_MAX and INACTION_TIME fields of the CONNECT command MAY be modified by the gateways fixed in a route. The receiver of the CONNECT command MUST NOT increase these fields value.

The connection establishment initiator MUST NOT sends data packets before reception of CONNECT-ACK command. All data packets received on connection before reception of CONNECT-ACK command MUST be discard silently. After reception of CONNECT-ACK command connection is considered established.

After sending CONNECT-ACK command, the sender node considers connection established. The first data packet received by CONNECT-ACK sender, MUST have initial sequence number (from FIRST_SEQUENCE_NUM field of option). Before reception of this packet, all other packets of connection MUST be discard silently.

5.1.1.3 Identifiers Exchanging Scheme

The scheme of identifiers exchange in procedure of primary connection establishment is represent in the following figure:

```
      Initiator                                       Responder
      ---------                                       ----------

   INIT [ SEQUENCE_NUM = 0,
          CONNECTION_ID = any value
          TEMP_CONNECTION_ID = TempID1 ] ---------->


                <---------- INIT-ACK [ SEQUENCE_NUM = TempNum1,
                                       CONNECTION_ID = TempID1,
                                       TEMP_CONNECTION_ID = TempID2 ]

   CONNECT [ SEQUENCE_NUM = TempNum1,
             CONNECTION_ID = TempID2,
             FIRST_SEQUENCE_NUM = SeqNum1,
             PERM_CONNECTION_ID = PermConnID1 ] ---------->


         <----------  CONNECT-ACK [ SEQUENCE_NUM = SeqNum1,
                                    CONNECTION_ID = PermConnID1,
                                    FIRST_SEQUENCE_NUM = SeqNum2,
                                    PERM_CONNECTION_ID = PermConnID2 ]

   DATA [ SEQUENCE_NUM = SeqNum2,
          CONNECTION_ID = PermConnID2 ] ---------->

                <----------  DATA [ SEQUENCE_NUM = SeqNum1 + 1,
                                    CONNECTION_ID = PermConnID1 ]
```

5.1.2 Secondary Connection Establishment

The purpose of a secondary connection establishment is:

- o An allocation of a separate data flow within the framework of the same service of the upper layer,
- o The accelerated establishment of new connection with the same or new service of the upper layer.

For the upper layer the service of primary and secondary connections not differ.

All secondary connections have the following general procedures with primary connection:

- o One sequence of sequence numbers in SEQUENCE_NUM field and general congestions control is conducted.

- o Have general SPI and PMTU.
- o Use one explicit route (see section 6).
- o Closing of primary connection results in closing all secondary connections related to it.

Procedure of a secondary connection establishment consists of two steps. The initiator of secondary connection sends BIND command on primary connection. In reply to it, BIND-ACK command is sent. Both commands may contain a upper layer data of new connection. The initiator of secondary connection may be the initiator or the responder of primary connection.

The packet with BIND command MUST have the ordered sequence number. If the SEQUENCE_NUM value is not ordered, the packet with the command is silently discarded.

Packets of secondary connection have the new CONNECTION_ID values and do not contain the information on primary connection. This information should be saved in endpoints variable states.

BIND and BIND-ACK commands have an identical format. Difference is the OPCODE and CONNECTION_ID values of base header. Fields of base header must have the following values:

```
   SEQUENCE_NUM  - The current sequence number of primary connection.
   SERVICE_ID    - This field contains the service identifier of
                   establishing secondary connection.
   DATA_LENGTH   = 0-65535
   CONNECTION_ID - For BIND command, it is the identifier of primary
                   connection assigned by the receiver of this
                   packet.  For BIND-ACK command, it is the
                   identifier from CONNECTION_ID field of packet with
                   BIND command.
```

BIND and BIND-ACK are formed as endpoints options and have the following format:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 0 1 0 1|E|D|0|  HEAD_LENGTH  |    OPCODE     |T|RSV| CHK_LEN |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       SEC_CONNECTION_ID                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The fields of general format have the following values:

```
   D = 1
   HEAD_LENGTH = 1
```

OPCODE =
5 for BIND
6 for BIND-ACK

T: 1 bit

The short connection identifier flag. If it is set, all packets of new connection, sent to address of sender of this command, must include the base header of 8 or 4 bytes length. If it is not set, the base header MUST have the length of 12 bytes.

RSV: 2 bits

Reserved. This bits should be set to 0 on transmit and ignored on receipt.

CHK_LEN: 5 bits

Specifies number of the first 4-byte words from DATA field, which will be included in calculation of the checksum or authentication data. If this field is set to zero, DATA field MUST NOT be included in calculation. If value is set to %b11111, calculation includes full DATA field. Value CHK_LEN is applied to all packets of new connection, transmitted by the sender of this command.

SEC_CONNECTION_ID: 32 bits

This field contains the identifier of secondary connection, assigned by the sender of this packet.

The initiator sends BIND command. After that, it must wait for BIND-ACK command up to sending of the following data packet of the new connection. The disordered data packets of new connection may be received before BIND-ACK reception. Cumulative acknowledgement number does not confirm reception of BIND command.

After sending of BIND-ACK positive response, the sender considers that connection is established. The DISCARDED-PACKET option is sent as the negative acknowledgement to BIND (see section 6.4).

Data packets of secondary connection have the CONNECTION_ID fields copied from the SEC_CONNECTION_ID field of the received command. BIND and BIND-ACK command does not stop data transmission on other connections.

5.1.3   0 - connection

The primary connection with zero SERVICE_ID value (0-connection) may be established between two endpoints. The purpose of 0-connection establishment may be:

- o restriction quantity of connections, which may be simultaneously open between two nodes,
- o grant of connectionless service for the upper layer.

It MUST NOT be established more than one 0-connection, which is not using an authentication or having one value of an security parameter index SPI between two nodes. Reservation parameters may be considered as the additional index, allowing to establish repeated 0-connections.

0-connection may be established by the following fashions:

(1) The initiator of primary connection set the SERVICE_ID to zero in INIT command. All other commands of the primary connection establishment MUST have zero value of this field.

(2) The responder of primary connection establishment makes a decision on 0-connection independently. In this case it set SERVICE_ID = 0 in reciprocal INIT-ACK command. The initiator MUST accept this value and copy it in CONNECT command.

(3) 0-connection may request a gateway. For this purpose it must set SERVICE_ID = 0 in INIT command.

Secondary connections are used for upper layer service irrespective from the 0-connection establishment initiator. BIND command may be attached to a packet containing CONNECT command of 0-connection.

Receiving of the second request about an establishment 0 - connection may be examined as an emergency reload of the source node. In this case, the INIT command receiver MUST execute the following:

(1) To send the INIT-ACK command not saving a state of new connection.

(2) The node must start activity check on existing connection after reception of reciprocal CONNECT command (see section 6.8). The CONNECT-ACK command must not be sent before end of checking. If existing connection is active, the packet with CONNECT command MUST be silently discard.

Receiving of incorrect INIT-ACK command, CONNECT or CONNECT-ACK with SERVICE_ID = 0 is examined as a error. Such packet MUST be silently discarded.

0-connection may be used for connectionless data transmission (for upper layer service). For such packets the following conditions MUST be satisfied:

- CONNECTION_ID has the 0-connections value, assigned by the packet receiver.
- SERVICE_ID has nonzero value.
- The Packet contains an upper layer data.
- The Packet does not contain BIND command.

All packets of 0-connection MUST contain the base header in length of 12 bytes.

5.1.4 Receiving Window Restriction

RCV-WINDOW option is used to inform an opposite endpoint in connection the maximal receiving window size for upper layer data. It is formed as an endpoint option and has the following format:

```
     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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1 0 1 0 1|E|D|0|  HEAD_LENGTH  |    OPCODE     |  WINDOW_SIZE1 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          WINDOW_SIZE2                         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The fields of general format have the following values:

D = 1
HEAD_LENGTH = 0/1
OPCODE = 7

WINDOW_SIZE1: 1 byte

If HEAD_LENGTH = 0, this field indicates the window size in 4-KByte blocks. If HEAD_LENGTH = 1, this field is set to 0.

WINDOW_SIZE2: 4 bytes

If HEAD_LENGTH = 1, this field indicates the window size in bytes. The value MUST be multiple to four. If HEAD_LENGTH = 0, this field is absent.

The RCV-WINDOW option sets a maximum data quantity of the upper layer, which are sent and are not confirmed with cumulative

acknowledgement. The option may be transferred in any connection packet, starting with CONNECT or BIND command.

The RCV-WINDOW option may be used for primary and secondary connections. This option concerns only to connection on which it was transmitted. The window restriction of primary connection does not influence on secondary connections. The RCV-WINDOW option MUST NOT be used for 0-connection.

At EHTP layer, the received data is sent to the upper layer without buffering. Therefore, window restriction is service, which is given on demand of the upper layer. Thus, RCV-WINDOW option limits a window at EHTP layer. If the upper layer has its functions of a window restriction or the ordered data flows is not require to it, this option is not used.

5.1.5 Security Parameters Index Changing

SPI must be changed, if the sequence numbers counter has passed a full cycle or if SPI lifetime has expired. The CHG-SPI option is used for SPI change. It has the following format:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0 1 0 1|E|D|0|  HEAD_LENGTH  |    OPCODE     |    RESERVED   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            NEW_SPI                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The fields of general format have the following values:

D = 1
HEAD_LENGTH = 1
OPCODE = 8

RESERVED : 8 bits

Reserved. These bits should be set to 0 on transmit and ignored on receipt.

NEW_SPI : 32 bits

Indicates the new value of SPI connection that will be used by the sender of this packet.

CHG-SPI option MUST be formed as an endpoint option. It joins any packet of primary or secondary connection and changes SPI for primary connection and everything secondary connections connected to it. The

packet with CHG-SPI option uses old SPI. All packets with following numbers MUST use new SPI. The option changes SPI only in one direction.

5.1.6 Connection Termination

All commands of connections termination: DISCONNECT, S-DISCONNECT, DISCONNECT-ACK and S-DISCONNECT-ACK are formed as an endpoints options, have an identical format and differ only in OPCODE value:

```
     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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1 0 1 0 1|E|D|0|  HEAD_LENGTH  |    OPCODE     |   ERROR_CODE  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The fields of general format have the following values:

```
      D = 1
      HEAD_LENGTH = 0
      OPCODE =
               9  for DISCONNECT
               10 for S-DISCONNECT
               11 for DISCONNECT-ACK
               12 for S-DISCONNECT-ACK
```

ERROR_CODE: 8 bits

Termination error code. Zero value defines normal end. Nonzero value defines abend. Error codes are described in section 5.1.7.

End of primary connection automatically finishes all secondary connections connected to it. End of secondary connection does not influence on other connections. For primary connection termination, DISCONNECT and DISCONNECT-ACK is used. For secondary connection termination, S-DISCONNECT and S-DISCONNECT-ACK are used. Packets with DISCONNECT-ACK and S-DISCONNECT-ACK commands MUST NOT include the payload.

The initiator of primary connection termination sends DISCONNECT command. After itÆs sending, the endpoint closes primary connection and all secondary connections connected to it. Incoming packets must be silently discarded. The addressee of the DISCONNECT command immediately sends DISCONNECT-ACK command and closes connection.

The initiator of secondary connection termination sends S-DISCONNECT command. After itÆs sending, the endpoint stops the outgoing traffic of secondary connection. The DISCARDED-PACKET option is sent instead

of the packets acknowledgment of this connection, in order not to stop cumulative sequence number counter. The addressee of DISCONNECT command immediately sends DISCONNECT-ACK command and closes connection.

Connection termination may be initiated, since the answer to CONNECT command. Connection termination always MUST be from two steps.

If the endpoint has received DISCONNECT-ACK and did not send DISCONNECT on this connection, it is recommended to start reconnection procedure (see section 6.2.3.3). If the endpoint has received S-DISCONNECT-ACK and it did not send S-DISCONNECT on this secondary connection, it is recommended to send S-DISCONNECT-ACK command and to close connection.

5.1.7 Termination Error Codes

The following error codes are defined for termination commands:

1 -	The connection establishment without payload is not supposed. The endpoint may demand to establish connection only if the upper level has data for transfer. In this case, the data must be transferred on the third step of the connection establishment.
2 -	The connection establishment is impossible at the present time because of temporal network overload or the channel employment. The repeated establishment MUST begin with INIT command.
3 -	Connection is impossible because of a network steady overload.
4 -	The endpoint is temporarily inaccessible.
5 -	The expectation time-out of the CONNECT-ACK command is exceeded.
6 -	Inadmissible value of reserved inactive time.
7 -	Connection through the not fixed route is not allowed.
8 -	the limit of secondary connections is exhausted.
9 -	Erroneous request on new 0-connection establishment. Such connection is already established.
10 -	the unknown destination address.
11 -	Connection is completed under the initiative of the upper level.
12 -	Connection is completed because of endpoint inactivity.
15 -	The non-supported protocol version.
16 -	Inadmissible limit of secondary connections.
255 -	Unexpected error.

5.2 User Data Transfer

The confirmed packet is identified on a combination of the

CONNECTION_ID, SERVICE_ID and SEQUENCE_NUM fields values. If the upper level demands delivery acknowledgement, SEQUENCE_NUM field MUST have nonzero value. If delivery acknowledgement is not required, SEQUENCE_NUM field MUST be set to zero.

The destination endpoint does not write received data packets in the buffer and transfers their to upper level at once. The full size of a packet must not exceed PMTU connection. The upper level must implement packing functions. The upper level may set minimal PMTU, which is required for its work, at a connection establishment.

The SEQUENCE_NUM field value in a combination with P flag of the basic header operates acknowledgement formation. The following logic is defined:

```
   SEQUENCE_NUM = 0 & P = 0 - The acknowledgement MUST NOT be formed
                              on this packet.

   SEQUENCE_NUM # 0 & P = 0 - Acknowledgement on this packet MUST be
                              sent, probably, with a small delay for
                              traffic optimization.

   P = 1 - Acknowledgement MUST be sent.  It is NOT RECOMMENDED to
           detain itÆs sending.  If SEQUENCE_NUM of the received
           packet is set to 0, acknowledgement MUST be sent on last
           ordered packet.  If the upper level data are acknowledged,
           the packet with acknowledgement must have P = 1.
```

For time-outs and repeated transfers number calculation, it is possible to use the recommendations, given in [20]. If the upper level does not demand delivery acknowledgement (SEQUENCE_NUM = 0), it is not transferred repeatedly.

If the received packet has incorrect CONNECTION_ID value, it MUST be silently discarded. All disordered packets with displacement from the ordered number, exceeding 65535, MUST be silently discarded.

It is supposed, that acknowledgement occupies essential traffic volume. Therefore, the protocol defines three variants of acknowledgement for an opportunity of optimization.

5.2.1 The Packet of Cumulative Data Acknowledgement

The sending of cumulative acknowledgement packet may be optimum if there is no the upper level data traffic in the necessary direction on connection. It has the special format of base header:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 0 1 1 0|E|P|R|               SEQUENCE_NUM                    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           SERVICE_ID          |          DATA_LENGTH          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         CONNECTION_ID                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

E, P, R, DATA_LENGTH, SERVICE_ID, CONNECTION_ID

Assignment of these fields is described in section 4.2.

SEQUENCE_NUM: 24 bits

Indicates the cumulative acknowledged sequence number.

The packet of cumulative acknowledgement MUST NOT contains the upper level data (DATA_LENGTH = 0).

5.2.2 CUM-ACK

CUM-ACK option is used for cumulative acknowledgement. The packet including this option may contain the payload. CUM-ACK is formed in a packet as an endpoint option and has the following format (differs from the general option format):

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 1 0 0 0|E|1|0|             CUM_SEQUENCE_NUM                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

E: 1 bit

Flag of the following option (see section 4.4).

CUM_SEQUENCE_NUM: 24 bits

Indicates the cumulative acknowledged sequence number.

5.2.3 GAP-ACK

GAP-ACK is an endpoint option. It contains cumulative acknowledgement and selective acknowledgement blocks. The packet

including this option may contain the payload. GAP-ACK option has the following format:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 0 1 0 1|E|D|0|  HEAD_LENGTH  |    OPCODE     |  CUM_SEQ_NUM  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  CUM_SEQ_NUM (continuation)   |           GAP_PACKET          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       GAP_BLOCK_1_START       |         GAP_BLOCK_1_END       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ~                                                               ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       GAP_BLOCK_n_START       |         GAP_BLOCK_n_END       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The fields of general format have the following values:

D = 1
HEAD_LENGTH = 1 - 255
OPCODE = 13

CUM_SEQ_NUM: 24 bits

Indicates the cumulative acknowledged sequence number (all packets with this and smaller sequence numbers were received).

GAP_PACKET: 16 bits

The acknowledgement of one disordered packet. Value is positive displacement from CUM_SEQ_NUM of this option. Zero value means absence of single acknowledgement.

GAP_BLOCK_n_START + GAP_BLOCK_n_END: 16 + 16 bits

These fields contain the beginning and the end (inclusive) of the selective acknowledgement block. These values are positive displacement from cumulative acknowledgement CUM_SEQ_NUM value of this option. The option may contain several selective acknowledgement blocks.

5.3 Congestion Control

EHTP uses the congestion control determined for TCP [6,15] and SCTP [7] for endpoint with the one address. EHTP primary connection and all secondary connections connected to it use common sequence

numbering packets and the common congestion control. In too time, EHTP allows establishing a rwnd receiving window separately for primary connection and for everyone secondary. Besides, the receiving window can be not established. In this case, it is considered infinite large. Therefore there are following essential features:

- o At operations with ssthresh and cwnd values, primary connection and everything connected to it secondary connections are considered as one stream. Thus, cumulative acknowledgement value is taken into account only, and selective acknowledgements are ignored.
- o Restriction rwnd is taken into account separately for primary connection for each secondary connection.

If the data packet does not demand acknowledgement, it has a zero sequence number. Besides, a cumulative acknowledgement packets (see section 5.2.1) which also have no the sequence number, may be transferred on connection. Thus, there may be traffic in connection, which is taken into account by transfer and has no acknowledgement. The packet sender without acknowledgement conditionally counts sequence number of this packet on unit more, than last sent packet with acknowledgement. The packet without acknowledgement request is considered delivered to the receiver (in functions of congestion control) in the following cases:

- o If cumulative acknowledgement of the following packet (with number equal to conditional number of a packet without acknowledgement) is received.
- o In case, if the traffic without request of acknowledgement is transferred only in connection, the data sender may take advantage of P pushing flag. If P = 1, acknowledgement is sent irrespective of SEQUENCE_NUM value. Packets without acknowledgement request must not be sent no more, than two per round trip time (RTT) if P flag is not used and there are no packets with acknowledgement request. Use of P flag for implicit congestions control is not defined in this document and left for experiments.

The algorithm of management of congestions is used by endpoints. In too time gateway with state may supervise a transfer policy of separate nodes and lower a priority for the data, not sticking to rules.

EHTP gives a sluice an optional means obviously to inform about the rejected packets (see section 6.4) except for implicit congestion control.

6 The Gateway Protocol

This document supposes that the quantity of global networks is limited at a network layer, and the typical endpoint has the global network address or it is connected with a global network through one EHTP gateway. In such configuration, the basic work on routing packets is conducted at a network layer. This document does not include the routing protocol. It is supposed, that EHTP can work at static routing in such configuration. Besides, the routing protocol is supposed, as a separate problem.

The protocol includes three routing variants:

(1) Implicit routing based on transport addresses.
(2) The route is explicitly set by sequence of gateways transport addresses, which are included in a packet.
(3) Explicit routing based on labels [17,19].

The explicit route of packets select at a primary connection establishment. All three routing ways may be applied to one connection simultaneously. The packet may contain a route as sequence of the following address objects:

(1) The gateway address without label. This object contains only the gateway transport address.
(2) The gateway address with label. In addition to the gateway transport address, this object includes the label allocated by this gateway.
(3) Label.

Address objects are formed in gateways options before basic header. The sequence of address objects in a packet defines sequence in a route. The first object defines the first hop.

The protocol gives a possibility to deallocate resources of gateways dynamically at absence of the connection traffic. Therefore, two labels types are defined:

- The stateless label. This label does not control activity of connection. It has unlimited or uncertain period of validity.
- The label with state. This label deallocates resources of a gateway after expiration of the reserved inactive time (see section 6.8).

In the text of this document, the "label" term means a label of any type. The label may be allocated by the lower layer or EHTP layer.

The protocol defines the following gateways types:

(1) The obligatory gateway. Connection MUST be established through this gateway. If the sequence of obligatory gateways is defined, it MUST NOT be broken (without consideration gateways of other types).

(2) The fixed gateway. Direct and return traffic of connection through such gateway MUST coincide.

(3) The explicit gateway. The gateway is fixed in a route of connection, but direct and return traffic through these gateways may differ.

(4) The free gateway. This gateway do not contain in an explicit route of connection. The traffic through it may be anyone.

Gateways of first three types contain in an explicit route of a packet.

It may be any quantity of EHTP gateways in a route. The route of connection may traverse several MPLS areas; areas with routing based on addresses and switched channels in any combination.

EHTP does not require from gateways to store a connections state. Therefore, independent formation of packets is not defined by EHTP gateway. The gateway may only attach options to endpoints packets or modify them. If it is necessary to transfer the error message for packet, the gateway must delete DATA field from this packet and set to 0 the checksum.

6.1 Gateway Options

6.1.1 GATEWAY-HEADER

GATEWAY-HEADER indicates the gateway of packet explicit route. It has the special format, which differs from the general option format:

```
     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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1 0 1 1 1|E|D|G|RSV|  GL   |GTP|W|B|F|S|I|N|  RESV |  O_COUNT  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    ~                          GTW_NODE_ADDR                        ~
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    ~                          ZERO_PADDING                         ~
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

E, D, G

These bits have assignment that is defined in section 4.4 for the general option format.

RSV: 2 bits

Reserved. This bit must be set to zero by the sender. If these bits are not set to 0 on receipt, the packet MUST be silently discard without no further action.

GL: 4 bits

The ADDR_LENGTH field of the gateway transport address (see section 3.1).

GTP: 2 bits

The NET_TYPE field of the gateway transport address.

W: 1 bit

The processing flag. If it is not set, this header indicates not traversed gateway. If it is set, the heading indicates on traversed gateway. This flag is set by gateway at forwarding.

B: 1 bit

The flag of obligatory gateway. If it is set, the connection MUST be established through this gateway.

F: 1 bit

The traffic fixing flag. If it is set, all direct and return connection traffic MUST pass through this gateway.

S: 1 bit

The label with state flag. If it is set, the gateway has allocated a label with state, which will be deallocated by gateway after the expiration of reserved inactive time (see section 6.8). If it is not set, the gateway will not supervise connection activity. The S flag is used at a connection establishment only.

I: 1 bit

The immediate hop flag. It is used only at a connection establishment. Value is set after processing header by the

gateway. If it is set, the label allocated by the gateway defines direct hop up to the previous gateway in explicit route. If it is not set, it can be a free gateway between this gateway and previous.

N: 1 bit

Flag of preservation of the following label. It is used at a connection establishment only. Value is set after processing header by the gateway. If it is set, the label allocated by a gateway saves the following address object of an explicit route. If it is not set, the gateway does not save a following address object.

RESV: 4 bits

These bits should be set to zero on transmit and ignored on receipt.

O_COUNT: 6 bits

Indicates the options quantity, which concern to this header. The slave options must be located immediately behind gateway header in a packet.

GTW_NODE_ADDR : 0-256 bytes

The NODE_ADDR field of the gateway transport address. If GL = 0 and GTP = 0, this field is absent. Zero GTW_NODE_ADDR value indicates any node of the set network. After processing header, to zero GTW_NODE_ADDR field the real address must be assigned.

ZERO_PADDING : 0-3 bytes

Zero bytes for alignment of header end on border four bytes.

Error messages, which are formed by gateways (DISCARDED-PACKET, GATEWAY-ERROR, NEW-PMTU options (see sections 6.4, 6.5, 6.6), do not contain the gateway address. In order to inform about the gateway address, which has generated the option, option may be the subordinate to gateway header. The gateway MUST set in header E and W flags at forwarding.

6.1.2 LABEL-HEADER

LABEL-HEADER option contains a gateway label in length of 28 bits and has a special format which differs from general option format:

```
        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
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |0|0|0|0|                  LABEL_VALUE                          |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

LABEL_VALUE: 28 bits

The protocol does not impose any restrictions on values of this field. It is supposed, that the gateway which has allocated this label, can unequivocally identify its format.

6.1.3 GENERAL-LABEL

The GENERAL-LABEL option is a label, which is defined in General Switch Management Protocol v3 [18]. This option has a special format, which differs from general option format:

```
        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
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |0 1 0 0|       Label Type      |           Label Length        |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                                                               |
       ~                          Label Value                          ~
       |                                                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Label Type

A 12-bit field indicating the type of label.

Label Length

A 16-bit field indicating the length of the Label Value field in bytes.

Label Value

A variable length field that is an integer number of 32 bit words long. The Label Value field is interpreted according to the Label Type as described in [18].

Stacked Label Indicator is not used, as EHTP has other methods of calculation of a labels stack end.

6.1.4 LABEL-OPTION

The LABEL-OPTION contains a label of any format in length up to 1020 bytes. The option has the following format:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 0 1 0 1|E|D|G|  HEAD_LENGTH  |    OPCODE     |    RESERVED   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ~                           LABEL_VALUE                         ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields of general format have the following values:

```
E = 0
D = 0
G = 1
HEAD_LENGTH = 1 - 255
OPCODE = 14
```

RESERVED: 8 bits

This field should be set to zero on transmit and ignored on receipt.

LABEL_VALUE: 4 - 1020 bytes

The protocol does not impose any restrictions on this field values. It is supposed, that the gateway which has allocated this value, can identify a format unequivocally.

6.1.5   COOKIE-GATEWAY

The COOKIE-GATEWAY option is used at a connection establishment for checking of endpoints. It has the following format:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 0 1 0 1|E|D|G|  HEAD_LENGTH  |    OPCODE     |  C_LIFE_TIME  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ~                          COOKIE_VALUE                         ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields of general format have the following values:

```
         E = 1
         D = 1
         G = 0
         HEAD_LENGTH = 1 - 255
         OPCODE = 15
```

C_LIFE_TIME : 8 bits

The lifetime of this COOKIE in minutes. Value 0 indicates uncertain lifetime value. Value 255 is reserved.

COOKIE_VALUE : 4-1020 bytes

The protocol does not impose any restrictions on a format and value of this field. It can be hashing function of unchangeable parameters of primary connection, the endpoint and time. It is recommended, that this value was difficult for predicting.

COOKIE-GATEWAY is formed by a gateway as the subordinated option of gateway header and is sent to endpoints. The endpoint must attach received GATEWAY-COOKIE to all packets of a connection establishment if the gateway contains in an explicit route. GATEWAY-COOKIE formation is recommended. The gateway makes a decision on sending GATEWAY-COOKIE independently. At various times they can be sent in both connection sides, only aside the initiator or to not be sent.

Endpoints must save COOKIE value for use in reconnection commands. New COOKIE-GATEWAY replaces an old. COOKIE-GATEWAY option MUST be sent only in packets with commands of a connection establishment and reconnection.

6.2 Connections Control through Gateways

6.2.1   Connection Establishment

Gateways participate only in a primary connections establishment. Secondary connections use a route and reservation of resources of the primary connection. The gateways of explicit route participate in a connections establishment only. If the gateway is free, it does not participate in a connections establishment.

At a connection establishment the gateway MAY execute the following procedures:

- o checking of COOKIE from the initiator and the responder,
- o fixing of a connection route,
- o labels assignment and distribution.

The explicit route is set at a connection establishment and may be changed only at reconnection. The gateways options defining a route, MUST follow connection establishment commands in a packet. Gateways of all types may present at one connection simultaneously. The gateway, which forms COOKIE in two sides, must be fixed in the direct and return traffic.

Labels are distributed upstream. Labels allocation for connection is executed on the third step (CONNECT command from initiator to responder) for the traffic from responder to initiator and on the fourth step (CONNECT-ACK command from responder to initiator) for the traffic from initiator to responder. The label is allocated by a gateway after its COOKIE checking. For acceleration of a connection establishment, the protocol allows to distribute initial labels on the first and second step of a connection establishment.

At a stage of a connection establishment, the explicit route is set in a packet by sequence of GATEWAY-HEADER options. After a connection establishment, it is possible to replace gateways headers with labels. All packets with commands of a connection establishment must contain addresses header of endpoints and transport addresses of gateways in GATEWAY-HEADER.

For a gateway, the following processing rules of the explicit routing information in packets with of a connection establishment commands (on steps) are defined:

1 step - sending of INIT command from the initiator to the responder:

- o If the gateway header does not contain in a packet, it may be inserted.
- o The gateway may allocate a temporary stateless label for the second step of a connection establishment (from the responder to the initiator). By forwarding, the gateway header must contain the transport address and the label.
- o It is possible to add gateways in not traversed area of a route.
- o It is possible to add gateways on the traversed area of a route directly ahead of this gateway, if it is known, that the added gateway must not send COOKIE aside the responder.
- o The responder, at reception of INIT command, forms a return packet with another or with the same route, having changed the order of GATEWAY-HEADER options on back.

2 step - sending of INIT-ACK command from the responder to the initiator:

- o If the gateway header is not contained in a packet, it may be inserted. In this case, the gateway cannot send COOKIE aside the responder.
- o If the gateway header in the received packet contains a label, routing of this packet is carried out on this label. By forwarding, the label must be deleted or replaced with other temporary label of this gateway allocated for the third step of a connection establishment (from the initiator to the responder).
- o If there is no gateway label in the received packet, it may be allocated for the third step of a connection establishment.
- o If the temporary label is allocated, the gateway header must contain the transport address and the label by forwarding.
- o It is possible to add gateways on the not traversed area of a route directly ahead of this gateway, if it is known, that the added gateway must not send COOKIE aside the responder.
- o It is possible to add gateways on the traversed area of a route directly ahead of this gateway, if it is known, that the added gateway must not send COOKIE in both sides of connection.
- o The initiator may silently refuse a connection establishment, if it does not accept a route in the received packet with INIT-ACK command.

3 step - sending of the CONNECT command from the initiator to the responder:

- o If the gateway header does not contain in a packet, it may be inserted. In this case, the gateway cannot send COOKIE in both sides of connection.
- o If the gateway header in the received packet contains a label, routing of this packet is carried out on this label.
- o On the third step, the gateway allocates a constant label, which will be used for the traffic of connection from the responder to the initiator. If the gateway must be fixed in a connection route without distribution of a label, the packet must contain header of this gateway with the transport address.
- o If the label allocated by this gateway defines direct hop up to the previous gateway from an explicit route, I flag must be set.
- o If the label allocated by this gateway saves the previous gateway from a route of this packet, N flag must be set.
- o It is possible to add gateways on the traversed area of a route directly ahead of this gateway if it is known, that the added gateway must not send COOKIE in both sides of connection and must not allocate a label.
- o The gateway may generate or replace existing COOKIE option. Value of COOKIE field from this option will be used on the fourth step and in the following reconnection under the

initiative of the responder (see section 6.2.3.4).

4 step - sending of CONNECT-ACK command from the responder to the initiator:

- o The third step actions, but for the opposite traffic of connection are executed. For sending packets, the labels distributed on the third step can be used.

On any step of connection establishment, the gateway may delete from an explicit route the addresses of other gateways at observance of all following rules:

- o The deleted gateway MUST NOT is obligatory (B = 0).
- o It is possible to delete addresses of gateways on the traversed area of a route if they are located directly ahead of this gateway. For example, it can be demanded at label allocation.
- o It is possible to delete any gateway on the not traversed area of a route.
- o The gateway may delete its header and send a packet forward.

Temporary labels, which are distributed on the first and second step of a connection establishment, MUST be stateless labels.

On the third and fourth step, the endpoints must save an explicit route of the received packets for use at reconnection and in sending the subsequent traffic of data. The sequence from the received packet must be changed on back.

One gateway in each direction may delete the transport address of the sender from addresses header. For this purpose, the gateway must save in state variables the transport address or a route up to an endpoint and relate this information with the allocated label. The endpoint must know about removal of the source address and compute the packet checksum without it. The interaction question of endpoint and a gateway, which deletes the address of the sender, is beyond the scope of this document. At removal of the address, it is necessary to take into account, that not all services of the upper layer provide an opaque addressing.

6.2.2 Data Transfer

Packets contain the short-cut routing information after a connection establishment. Address objects from packets with last connection establishment commands must be optimized by the following rules:

- o The label without the gateway address MUST NOT follows the gateway address without label (the gateway without label can forward packets only on the basis of transport addresses)

- o After a label with I = 0 (the label does not define direct hop up to the following gateway), the object with the transport address (gateway header or addresses header) must be located.
- o If all route consists of labels with I = 1, the packet must not contain transport addresses of gateways and endpoints.
- o If last gateway in a route has set I = 1 (i. e. the label allocated by it defines direct hop up to the endpoint), the packet must not contain addresses header. The packet MUST contain addresses header in all other cases. If the packet does not contain an explicit route, the sender endpoint is considered as last gateway of an explicit route. If it sends a packet immediately to receiver endpoint, the packet MUST NOT contains addresses header.
- o If at current label N = 1 (the gateway saves the following address object), the following address object deletes from a stack.

At processing entering packets after a connection establishment, the gateway must execute the following logic:

- o Gateways Options are looked through, since the first.
- o If the label is the first unprocessed address object the packet is forwarded on value of this label.
- o If the gateway header with label (without the transport address or with the address of this gateway) is the first address object, the label is processed also, as in the previous item. In addition, the header can contain the subordinated options, which must be processed.
- o If the label has incorrect value or the first unprocessed object is the label and a gateway does not allocate labels, the packet is silently discarded.
- o If the first unprocessed address object is the header of other gateway with the transport address (with or without label) or the address of the endpoint from addresses header, forwarding will be realized based on transport addresses.
- o The processed address object in stack top deletes at forwarding.
- o If the gateway saves address object of the following gateway, this object must be added in packet front.
- o Gateways headers with W = 1 are not processed, if they concern to other gateway.

6.2.3 **Reconnection**

The reconnection reason is:

- o Route restoration after the long inactivity period of endpoints.

- o Necessity of change of a route through the gateways fixed in a route by breakdown one of them.
- o Necessity of change of resources reservation parameters.
- o Dynamic reduction of PMTU value.

Reinstalled connection MUST have the same constant values of connection identifiers, as initial. Primary connection is reinstalled only. Secondary connections begin to use new labels and/or a route without sending special primitives. Reconnection commands are formed as gateway options. In a packet, they are located before options of an explicit route. All packets contain next sequence number. Reconnection does not change packets numbers sequence of connection.

Everywhere in the text of this document, the reconnection commands are equivalent to connection establishment commands if other logic is not defined separately.

Any endpoint of connection can be the reconnection initiator. In case of counter reconnection, the initiator of the initial (first) connection has advantage. Counter packets of reconnection from the responder are silently discarded.

6.2.3.1 REINIT, REINIT-ACK

The REINIT and REINIT-ACK commands are used for reconnection and correspond to INIT and INIT-ACK commands of initial primary connection establishment. They are formed as a gateway options, have an identical format and differ only in OPCODE value. Packets with REINIT and REINIT-ACK commands must not include payload. Fields of base header must have the following values:

SEQUENCE_NUM -	Contains the current sequence number of reinstalled connection.
SERVICE_ID -	Contains the service identifier of the initial primary connection.
DATA_LENGTH =	0
CONNECTION_ID -	This field contains the identifier of initial primary connection allocated by the packet receiver.

The option of REINIT and REINIT-ACK commands has the following format:

```
  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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |1 0 1 0 1|E|D|G|  HEAD_LENGTH  |    OPCODE     |RES_VER_FLAGS|F|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |RSV|         MIN_HL_PMTU       |RSV|            PMTU           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields of general format have the following values:

```
   E = 1
   D = 1
   G = 1
   HEAD_LENGTH = 1
   OPCODE =
            16 for REINIT
            17 for REINIT-ACK
```

RES_VER_FLAGS: 7 bits

The reserved versions flags. Values of these bits are set to zero by sending. If at receiving even one of them is set to 1, the packet must be silently rejected.

F: 1 bit

Flag of the protocol first version. This bit MUST be set to 1. It sets the protocol version, which is defined in this document.

RSV: 2 + 2 bits

Reserved. These fields should be set to zero on transmit and ignored on receipt.

MIN_HL_PMTU: 14 bits

Indicates the upper layer PMTU minimal size in 32-bit words (see section 5.1.1.1).

PMTU: 14 bits

Indicates PMTU size in 32-bit words (see section 5.1.1.1).

6.2.3.2 RECONNECT, RECONNECT-ACK

RECONNECT and RECONNECT-ACK commands correspond to CONNECT and CONNECT-ACK commands of an initial connection establishment. The packet containing these commands MAY contain an upper layer data. RECONNECT and RECONNECT-ACK are formed as gateways options, have an identical format and differ only in OPCODE value. Fields of base header must have the following values:

```
   SEQUENCE_NUM -  Contains the current sequence number of
                   reinstalled connection.
   SERVICE_ID -    This field contains the service identifier of the
                   initial primary connection.
   DATA_LENGTH =   0 - 65535
   CONNECTION_ID - This field contains the identifier of initial
                   primary connection allocated by the packet
                   receiver.
```

The option of RECONNECT and RECONNECT-ACK commands has the following format:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 0 1 0 1|E|D|G|  HEAD_LENGTH  |    OPCODE     | INACTION_TIME |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields of general format have the following values:

```
      E = 1
      D = 1
      G = 1
      HEAD_LENGTH = 0
      OPCODE =
               18 for RECONNECT
               19 for RECONNECT-ACK
```

INACTION_TIME: 8 bits.

The reserved inactive time. This field contains time interval in seconds at which expiration gateways will deallocate connection resources at absence of traffic (see section 6.8).

6.2.3.3 2-way Handshake

2-way handshake reconnection is used, if it is necessary to renew data transmission after the expiration of reserved inactive time (see section 6.8) or if it is necessary to change reservation parameters. The primary goal is redistribution of labels with state. 2-way handshake reconnection repeats the third and fourth steps of a primary connection establishment.

The reconnection initiator sends RECONNECT command. The packet with this command contains an initial route, which is formed in accordance with the following rule:

- For the initiator of initial connection, the explicit route from a packet with CONNECT-ACK command of initial connection (upside-down) takes.
- For the responder of initial connection, the explicit route from a packet with CONNECT command of initial connection takes.
- If reconnection is not the first, the explicit route from the previous accepted reconnection command takes.
- All labels with state delete from options of an explicit route.
- COOKIE-GATEWAY options and stateless labels are saved.

All packets with reconnection commands must contain addresses header. The packet with RECONNECT command is processed on gateways the same as packet with CONNECT command. Receiver of RECONNECT processes the received explicit route the same as a route from a packet with CONNECT command (see section 6.2.1). For reconnection acknowledgement the packet with RECONNECT-ACK command is sent. This command is processed by gateways as CONNECT-ACK command.

The reconnection commands can contain payload. The reconnection initiator can send the following packets with data before RECONNECT-ACK reception. These packets must include an initial route and addresses header. Required QoS cannot be guaranteed before RECONNECT-ACK reception.

If lifetime even one gateway COOKIE from an explicit route of connection has expired, reconnection from two steps MUST NOT is used. In this case, reconnection from four steps is used.

6.2.3.4 4-way Handshake

4-way handshake reconnection is used for a choice of a new connection explicit route through gateways for which the endpoint does not know current COOKIE values. 4-way handshake reconnection completely repeats algorithm of an establishment of starting primary connection. Commands correspondence are shown in the following table:

Primary Connection Command	Corresponding Reconnection Command
INIT	REINIT
INIT-ACK	REINIT-ACK
CONNECT	RECONNECT
CONNECT-ACK	RECONNECT-ACK

Reconnection packets use current sequence numbers of connection and current SPI.

6.2.4 Connections Termination through Gateway

If the gateway has allocated to connection a stateless label, it must not supervise connection closing. If a gateway has allocated a label with state, it may:

(1) Not trace connections closing and deallocate resources on the timer of reserved inactive time (see section 6.8). As the gateway cannot check up the packet authentication data, it is RECOMMENDED to apply this way to the connections using an authentication. It allows excluding a fake of connection closing commands.

(2) The gateway can trace connections termination. Computing loading at a gateway grows in this case, as connections termination commands are formed as endpoint options and their analysis requires viewing of all packet headers. Advantage is fast free of the resources. If the gateway has allocated independent labels into everyone sides of connection, it deallocates resources immediately after reception any of DISCONNECT or DISCONNECT-ACK commands. If the gateway has allocated one or the connected labels, resources are deallocated at reception of DISCONNECT-ACK command.

If connection has ceased to use a gateway because of reconnection procedure, the gateway deallocates resources on the timer of reserved inactive time.

At using of initial connection, reconnection is allowed. Endpoints must not use simultaneously more than four explicit routes in one connection.

6.3 Use of symbolical addresses

The transport address in symbolical representation (see section 3.3) may be written down to destination address only in a packet with the INIT command (first command of a connection establishment). All other packets of the connection MUST have the destination address in a binary format. The source address always MUST be in a binary format.

Protocols, which allow gateways to define binary addresses from symbolical representations, lay beyond the scope of this document. Gateways must not modify packet addresses header. By transferring a packet with the receiver symbolical address, the following logic MAY be used:

(1) The packet is sent to address of the defined gateway. It can be a default gateway or the special gateway responsible for routing of packets with symbolical addresses.
(2) If the gateway can calculate the address binary format of the endpoint, it forms a packet explicit route. Last gateway header contains the endpoint transport address in binary format.
(3) If the gateway cannot calculate the endpoint address, it adds its header in a packet and set W flag. Addition is necessary to exclude routing loops. After that, the packet is sent the following gateway. If the following gateway address is unknown, the packet is silently discarded.

At processing entering packets with INIT command, the endpoint must take into account that its address can be in last gateway header of an explicit route if the receiver address in addresses header has symbolical representation.

Use of symbolical addresses in gateways headers is left opened for discussion.

6.4 DISCARDED-PACKET

There are several reasons on which a gateway or an endpoint may reject connection packets. As the discarded packet stops advance of confirmed cumulative sequence number, DISCARDED-PACKET option must be sent instead of the discarded packet. A gateway or an endpoint may send this option. A gateway sends an option in the same streamline in which it was necessary to send the rejected packet. The gateway forms DISCARDED-PACKET as gateway option. The endpoint must form DISCARDED-PACKET as endpoint option. The option has the following format:

```
  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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |1 0 1 0 1|E|D|G|  HEAD_LENGTH  |    OPCODE     |    ERRCODE    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     DISCARDED_BLOCK_1_START   |     DISCARDED_BLOCK_1_END     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 ~                                                               ~
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     DISCARDED_BLOCK_n_START   |     DISCARDED_BLOCK_n_END     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields of general format have the following values:

E = 1
D = 1
G = 0
HEAD_LENGTH = 0 - 255
OPCODE = 20

ERRCODE: 8 bits

Error code of the discarded packet. This value concerns to all packets numbers of this option. The following values are defined:

17 - connection is closed. This error is formed by endpoint at closing secondary connections. After S-DISCONNECT sending, the endpoint does not receive entering packets of secondary connection. Instead of acknowledgements, DISCARDED-PACKET option is sent.
133 - Packet size exceeds MTU.
134 - SERVICE_ID value is not connected with active service. Such error may be generated for a packet with BIND command or by sending through 0 - connection of the connectionless data. A gateway (firewall) or an endpoint may form an option with this error.
135 - The packet is deleted because of gateway congestion. The option with this error code may not contain a range of the discarded packets. In this case, it is the warning of a possible congestion or the rejected packet has not required acknowledgement. The sending of DISCARDED-PACKET option with this error code is OPTIONAL. This option NOT RECOMMENDED to be sent more

often, than once during RTT.

DISCARDED_BLOCK_n_START + DISCARDED_BLOCK_n_END: 16 + 16 bits

These fields contain the beginning and the end (inclusive) the block of the discarded packets. Their value is negative displacement from sequence number of the packet to which this option is attached. Zero value is allowable. If the block contains number of one packet, both fields MUST have one value.

The receiver endpoint of the discarded packet after reception of DISCARDED-PACKET option sends this option back to the sender of rejected packet. Exception is inadmissible SERVICE_ID value in a packet with BIND command. This command is confirmed by special packet. Therefore the receiver of DISCARDED-PACKET with ERRCODE = 134 only includes number of the discarded packet in cumulative acknowledgement number of the following packet.

The sender of the discarded packet must send the same packet repeatedly after reception of DISCARDED-PACKET option. If repeated sending is impossible, DISCARDED-PACKET option is ignored. At reception DISCARDED-PACKET for packets of the closed connection (ERRCODE = 17), repeated sending is not carried out.

If the message on congestion is received, it is necessary to execute procedures of congestions control (see section 5.3).

If the data receiver before reception of DISCARDED-PACKET option has received even one correct packet from a range specified in the option, option is considered erroneous and is ignored.

6.5 NEW-PMTU

NEW-PMTU option is used for measurement of current PMTU. It is formed as gateway option and has the following format:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0 1 0 1|E|D|G|  HEAD_LENGTH  |    OPCODE     |    RESERVED   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|RSV|         MIN_HL_PMTU       |RSV|            PMTU           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields of general format have the following values:

E = 1
D = 1
G = 1
HEAD_LENGTH = 1

      OPCODE = 21

   RESERVED + RSV + RSV: 8 + 2 + 2 bits

      Set to zero on transmit and ignored on receipt

   MIN_HL_PMTU: 14 bits

      Indicates the upper layer PMTU minimal size in 32-bit words
      (see section 5.1.1.1).

   PMTU: 14 bits

      Indicates PMTU size in 32-bit words (see section 5.1.1.1).

If the gateway cannot transfer on connection a packet because of dynamic reduction PMTU, it forms simultaneously two options: NEW-PMTU and DISCARDED-PACKET with ERRCODE = 133. Then these options will be sent together as it is described in section 6.4.

6.6 GATEWAY-ERROR

GATEWAY-ERROR option is used by gateways for sending the message on a critical error by sending of a connection packet. GATEWAY-ERROR is formed as gateway option and has the following format:

```
     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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1 0 1 0 1|E|D|G|  HEAD_LENGTH  |    OPCODE     |    ERRCODE    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Fields of general format have the following values:

      E = 1
      D = 1
      G = 0
      HEAD_LENGTH = 0
      OPCODE = 22

   ERRCODE: 8 bits

      Contains an error code (see section 6.7).

The gateway must send GATEWAY-ERROR in a direction of sending a packet, which has caused an error. Back message error sending is not stipulated. If the gateway cannot send a packet forward, it must discard it silently.

GATEWAY-ERROR option concerns to primary connection and everything connected with it secondary connections. It can be transmitted on any of these connections.

The gateway at packet receiving with GATEWAY-ERROR option executes simple forwarding. If the packet contains a connection establishment command, the gateway must not reserve resources for connection and allocate a label.

The endpoint at receiving GATEWAY-ERROR option carries out the following actions:

- o If the option is received in a packet with INIT command, the packet containing INIT-ACK and DISCONNECT commands is sent. The receiver of the packet with INIT-ACK + DISCONNECT considers connection not established.
- o GATEWAY-ERROR option in a packet with INIT-ACK command does not finish connection immediately if it is expected, that the packet with INIT-ACK should contain an authentication data. In this case, the same packet with INIT command is repeatedly sent and during the set timeout, INIT-ACK command is expected. If correct INIT-ACK command is received, the packet with GATEWAY-ERROR is silently discarded. If the authentication is not used, connection is closed immediately at reception of GATEWAY-ERROR option.
- o GATEWAY-ERROR option received with any packet since the third step of connection establishment initiates connection termination (see section 5.1.6), or reconnection (see section 6.2.3.4).
- o If GATEWAY-ERROR is received in the packet duplicate, the option is not processed and a packet is silently discarded.

6.7 Gateway Error Codes

The following errors codes, which can be contained in ERROR-CODE field of GATEWAY-ERROR option are defined:

15 - Not supported protocol version.
129 - A gateway label cannot be allocated.
130 - The gateway cannot provide the requested QoS.
131 - Erroneous gateway label.
132 - Erroneous gateway COOKIE.

6.8 Activity Control

Reserved inactive time (RIT) is used for dynamic deallocating of the network resources allocated for EHTP connections. RIT value is set by the connection initiator in CONNECT command. The responder can reduce this time in CONNECT-ACK command. If changed RIT does not

suit to the initiator, it must begin connection closing procedure with ERRCODE = 6.

RIT is used only if there is even one gateway in the connection route, which has allocated a label to this connection and has set S flag. As direct and back route can differ, endpoints must agree upon RIT value irrespective of gateways presence. At the activity control, primary connections and everything connected with it secondary connections are considered as one connection.

Gateway reset RIT timer at receipt of any connection packet from any direction. Traffic in everyone side is traced separately, if the gateway has allocated independent labels in each side of connection. The endpoint uses the following logic:

(1) The value = RIT - 4 * max (RTT) is used. This value must
be   corrected at change RTT.
(2) Acknowledgements of packets are taken into account only.
(3) RIT timer is corrected for the time of sending last confirmed packet.

If RIT timer has expired, the gateway must deallocate resources occupied with connection. If the allocated label is used only for this connection, it must be deleted or marked unused. The gateway can use this label for other connection or a route at once. At deallocation connection resources, the gateway must not send network primitives.

If the gateway has allocated a stateless label (S = 0), it MUST NOT supervise RIT.

The endpoint at end of corrected RIT timer must note the connection route as closed. At that connection is not terminated and network primitives are not sent. If it is necessary to send data on connection in the subsequent, procedure of reconnection (see section 6.2.3.3) should be executed.

The procedures connected with the activity control, resources deallocating on gateways and reconnection are transparent for the upper layer and do not result in connections closing.

The endpoint for maintenance of route activity at upper layer traffic absence sends a packet of cumulative acknowledgement (see section 5.2.1) with the set P pushing flag. The receiver of such packet must immediately send a cumulative acknowledgement packet with not set P flag. The connection initiator must respond activity through an time interval = RIT - 8 * max(RTT), and the responder through an interval   = RIT
- 6 * RTT.

It is not recommended to set RIT less, than 16 * RTT. For a global network, 6-second value is recommended. For best-effort traffic, it is RECOMMENDED to set minimal RIT value and it is NOT RECOMMENDED to provide a route in an active state at absence of the upper layer traffic.

7 Security Consideration

EHTP defines connection establishment with two parameters, which can have casual value: the connection identifier and the initial sequence number. To participate in connection, the node should know both values allocated by other node. It is recommended, that even initial sequence number have difficultly predicted value.

For security increase from DoS attacks, EHTP does not provide data buffering at its layer. The decision on denial of service is taken on the applications layer. Thus, the endpoint can have the increased stability to refusals for priority-driven connections. Refusal of buffering does not exclude attacks to gateways. Nevertheless, the gateway can consider separate connections, as prioritized (for example, connections with authorization or with QoS). Besides, the gateway with state can trace a policy of congestions control which endpoints adhere.

COOKIE mechanism is used for protection an overload of a gateway with state. If the attacking node can listen the traffic and form packets with the forged source address, this protection is inefficient. Presence of the protected channel between a gateway and an endpoint can be effective only. If the gateway and endpoint have exchanged with SPI values, they can use it within the frameworks of EHTP protocol. For this purpose, the protocol should be added with special gateway header or COOKIE option. This question is not considered in the document and left opened for discussion.

EHTP does not provide independent formation of packets by gateways. The gateway can only add unsigned options to the packets generated by endpoints, and to send these packets only forward. It has the principle meaning for the connections using an authentication in networks, where the attacking node can form packets with the forged source address, but cannot modify packets. It is supposed, that the correct packet will achieve the receiver earlier, than the forged duplicate and the duplicate will be rejected. This document defines, that all duplicates and incorrect packets must be silently discarded without any further action. Security from attacks is considered as priority-driven problem in comparison with algorithm optimality.

Primary connection can be used as transit for the forged packets of secondary connections. The authentication is the protection against this kind of attacks. In any case, at using of the checksum instead of authentication, the protocol has no effective protection against the attacking node, which is in CSMA network, through which packets of connection pass.

For protection against "man-in-the-middle" attacks, the following variants are possible:

- Using authentication at EHTP layer. At that, existing keys control protocols can be transferred on EHTP transport without changes. It will give a possibility of end-to-end protection even if endpoints are located in the networks incompatible at a network layer.
- Protection at applications layer.
- Using IPsec, if endpoints are in IP network. Application of EHTP gateway functions, which require modification a packet, is impossible in this case.
- EHTP can be added with end-to-end enciphering means with definition of a new format of basic header.

8 References

[1] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[3] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[4] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", STD 7, RFC 793, September 1981.

[5] Bogdanov, A., "Unified Memory Space Protocol Specification", RFC 3018, December 2000.

[6] Allman, M., Paxson, V. and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[7] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[8] Karn, P. and W. Simpson, "Photuris: Session-Key Management Protocol", RFC 2522, March 1999.

[9] Mogul, J. and S. Deering, "Path MTU Discovery", RFC 1191, November 1990.

[10] McCann, J., Deering, S. and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[11] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, August 1996.

[12] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[13] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[14] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[15] Dawkins, S., Montenegro, G., Kojo, M., Magret, V. and Vaidya, N., "End-to-end Performance Implications of Links with Errors", RFC 3155, August 2001.

[16] ITU-T Recommendation V.42, "Error-correcting procedures for DCEs using asynchronous-to-synchronous conversion", October 1996.

[17] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[18] Doria, A., Sundell, K., Hellstrand, F. and T. Worster, "General Switch Management Protocol (GSMP) V3", RFC 3292, June 2002.

[19] Awduche, et al, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[20] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000.

[21] Larzon, Lars-Ake, Degermark, Mikael and Pink, Stephen, "The UDP Lite Protocol", Work in Progress.

9 Author's Address

Alexander Bogdanov

23-126, Generala Kuznetsova St.
Moscow, Russia 109153
RU

Phone: +7 095 706 1002
Email: a_bogdanov@s-mail.com

10 Full Copyright Statement

Copyright (C) The Internet Society (2002). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.