**RPC protocol**

**Data encoding(mã hóa dữ liệu)**

All integral data is encoded in little endian format.

Tất cả dữ liệu quan trọng được mã hóa theo format little endian.

**Protocol negotiation (negotiation các giao thức)**

**The negotiation works by exchanging negotiation frame immediately after connection establishment. The negotiation frame format is:**

Negotiation hoặc động bằng cách hoán đổi frame negotion ngay lập tức sau khi thiết lập kết nối. Format của frame negotion là:

uint8\_t magic[8] = SSTARRPC

uint32\_t len

uint8\_t data[len]

**The negotiation frame data is itself composed of multiple records, one for each feature number present. Feature numbers begin at zero and will be defined by later versions of this document.**

Dữ liệu của frame negotiation bao gồm nhiều records, mỗi record là cho mỗi number\_feature ở hiện tại. feature\_number bắt đầu từ zero và sẽ được định nghĩa bởi các phiên bản sau của tài liệu này.

struct negotiation\_frame\_feature\_record {

uint32\_t feature\_number;

uint32\_t len;

uint8\_t data[len];

}

**A negotiation\_frame\_feature\_record signals that an optional feature is present in the client, and can contain additional feature-specific data.**

Một negotiation\_frame\_feature\_record ám chỉ rằng một tính năng tùy chọn có trong client, và có thể chứa dữ liệu feature-specific bổ sung.

**The feature number will be omitted in a server response if an optional feature is declined by the server.**

Number\_feature sẽ được bỏ qua trong một phản hồi của server nếu một tính năng tùy chọn bị từ chối bởi server.

**Actual negotiation looks like this:**

Negotiation thực tế trông giống như là:

Client Server

--------------------------------------------------------------------------------------------------

send negotiation frame

recv frame

check magic (disconnect if magic is not SSTARRPC)

send negotiation frame back

recv frame

check magic (disconnect if magic is not SSTARRPC)

**Supported features( tính năng được hỗ trợ)**

**Compression (nén)**

feature\_number: 0

**data : opaque data that is passed to a compressor factory**

**provided by an application. Compressor factory is**

**responsible for negotiation of compression algorithm.**

Data : dữ liệu mờ đó là cái mà được truyền tới một **compressor factory** cung cấp bởi một ứng dụng. Compressor factory chịu trách nhiệm cho negotiation của thuật toán nén.

**If compression is negotiated request and response frames are encapsulated in a compressed frame.**

Nếu compression là request được thương lượng và frames phản hồi được đóng gói trong frame được nén.

**Timeout propagation**

feature\_number: 1

data : none

If timeout propagation is negotiated request frame has additional 8 bytes that hold timeout value

for a request in milliseconds. Zero value means that timeout value was not specified.

If timeout is specified and server cannot handle the request in specified time frame it my choose

to not send the reply back (sending it back will not be an error either).

**Connection ID**

feature\_number: 2

uint64\_t conenction\_id : RPC connection ID

Server assigns unique connection ID for each connection and sends it to a client using

this feature.

**Stream parent**

feature\_number: 3

uint64\_t connection\_id : RPC connection ID representing a parent of the stream

If this feature is present it means that the connection is not regular RPC connection

but stream connection. If parent connection is closed or aborted all streams belonging

to it will be closed as well.

Stream connection is a connection that allows bidirectional flow of bytes which may carry one or

more messages in each direction. Stream connection should be explicitly closed by both client and

server. Closing is done by sending special EOS frame (described below).

**Isolation**

feature number: 4

uint32\_t isolation\_cookie\_len

uint8\_t isolation\_cookie[len]

The `isolation\_cookie` field is used by the server to select a

`seastar::scheduling\_group` (or equivalent in another implementation) that

will run this connection. In the future it will also be used for rpc buffer

isolation, to avoid rpc traffic in one isolation group from starving another.

The server does not directly assign meaning to values of `isolation\_cookie`;

instead, the interpretation is left to user code.

**Compressed frame format**

uint32\_t len

uint8\_t compressed\_data[len]

after compressed\_data is uncompressed it becomes regular request, response or streaming frame

**Request frame format**

uint64\_t timeout\_in\_ms - only present if timeout propagation is negotiated

uint64\_t verb\_type

int64\_t msg\_id

uint32\_t len

uint8\_t data[len]

msg\_id has to be positive and may never be reused. data is transparent for the protocol and serialized/deserialized by a user

**Response frame format**

int64\_t msg\_id

uint32\_t len

uint8\_t data[len]

if msg\_id < 0 enclosed response contains an exception that came as a response to msg id abs(msg\_id) data is transparent for the protocol and serialized/deserialized by a user

**Stream frame format**

uint32\_t len uint8\_t data[len]

len == 0xffffffff signals end of stream data is transparent for the protocol and serialized/deserialized by a user

**Exception encoding**

uint32\_t type

uint32\_t len

uint8\_t data[len]

**Known exception types**

USER = 0

UNKNOWN\_VERB = 1

**USER exception encoding**

uint32\_t len

char[len]

This exception is sent as a reply if rpc handler throws an exception. It is delivered to a caller as std::runtime\_error(char[len])

**UNKNOWN\_VERB exception encoding**

uint64\_t verb\_id

This exception is sent as a response to a request with unknown verb\_id, the verb id is passed back as part of the exception payload.

**More formal protocol description**

request\_stream = negotiation\_frame, { request | compressed\_request }

request = verb\_type, msg\_id, len, { byte }\*len

compressed\_request = len, { bytes }\*len

response\_stream = negotiation\_frame, { response | compressed\_response }

response = reply | exception

compressed\_response = len, { byte }\*len

streaming\_stream = negotiation\_frame, { streaming\_frame | compressed\_streaming\_frame }

streaming\_frame = len, { byte }\*len

compressed\_streaming\_frame = len, { byte }\*len

reply = msg\_id, len, { byte }\*len

exception = exception\_header, serialized\_exception

exception\_header = -msg\_id, len

serialized\_exception = (user|unknown\_verb)

user = len, {byte}\*len

unknown\_verb = verb\_type

verb\_type = uint64\_t

msg\_id = int64\_t

len = uint32\_t

byte = uint8\_t

negotiation\_frame = 'SSTARRPC' len32(negotiation\_frame\_data) negotiation\_frame\_data

negotiation\_frame\_data = negotiation\_frame\_feature\_record\*

negotiation\_frame\_feature\_record = feature\_number len {byte}\*len

feature\_number = uint32\_t

Note that replies can come in order different from requests, and some requests may not have a reply at all.