Introduction

ShadowQUIC is 0-RTT QUIC based proxy with SNI camouflage.

Shadowquic doesn't provide any authentication layer. Authentication is provided by JLS protocol.

Command

Each proxy request is started with a command carried by a bistream.

```
+-------+--------------+
|  CMD  |   SOCKSADDR  |
+-------+--------------+
|   1   |   Variable   |
+-------+--------------+
```

There are three types of command indicated by `CMD` field:

- `0x01` : TCP Connect
- `0x03` : UDP Association over datagram extension
- `0x04` : UDP Association over unistream

`SOCKSADDR` field is socks address format:

```
+---------+-----------+--------+
|  ATYP   |   ADDR    |  PORT  |
+---------+-----------+--------+
|    1    | Variable  |    2   |
+---------+-----------+--------+
```

- ATYP address type of following address
  - IP V4 address: `0x01`
  - DOMAINNAME: `0x03`
  - IP V6 address: `0x04`
- ADDR desired destination address
- PORT desired destination port in network octet order

TCP Proxy

TCP Connect command is supported to proxy forward TCP connection.

TCP Connect

TCP proxy task is directed followed by TCP Connect command like *socks5*

UDP Proxy

The UDP proxy scheme is greatly different from common protocol like TUIC/hysteria. The principle of design is to decrease datagram header size and reaches the **maximum MTU size**.

The design has heavily considerred RFC 9298

```
+--------------+--------------+
|   SOCKSADDR  |  CONTEXT ID  |
+--------------+--------------+
|   Variable   |      2       |
+--------------+--------------+
```

UDP Associate command is carried by bistream called **control stream**. For each datagram received from local socket or remote socket a control header consists of `SOCKSADDR` and `CONTEXT ID` is sent. If `CONTEXT ID` has been sent in the past which indicates the destination address has been cached, then this header could been skipped.

For each connection, implementation must maintain two `CONTEXT ID` spaces. One is for client to server direction. The other is for server to client direction. These two id spaces are independent.

control stream doesn't send payload. The payload is carried by unistream or datagram extension chosen by user. Control stream **MUST** remain alive during udp association task.

If control stream is terminated, the udp association task **must** also be terminated.

UDP Associate command associates a remote socket to local socket. For each destination from a local socket the datagram will be asigned a `CONTEXT ID` which is in **one-to-one corespondance** to 4 tuple (local udp ip:port, destination udp ip:port).

Each datagram payload will be prepended with a 2 bytes context ID.

For each datagram from local socket or remote socket the `SOCKSADDR` and `CONTEXT ID` pair will be sent in the control stream. And `SOCKSADDR` and `CONTEXT ID` pair will be sent **at least once** for each new `CONTEXT ID`.

Associate Over Stream

```
+---------------+--------------+--------------+--------------+--------------+-----+
|  CONTEXT ID   |     LEN      |    PAYLOAD   |     LEN      |    PAYLOAD   | ... |
+---------------+--------------+--------------+--------------+--------------+-----+
|      2        |      2       |   Variable   |      2       |   Variable   | ... |
+---------------+--------------+--------------+--------------+--------------+-----+
```

If the datagram is carried by QUIC unistream, a 2 byte length tag is prepended to the payload. For the following datagram with the same context id, unistream could be reused, and context id is not needed to be sent. Only LEN field and PAYLOAD will be sent. Namely for each unistream, CONTEXT ID is sent only once right after this stream is opened,

Associate Over Datagram

```
+---------------+--------------+
|  CONTEXT ID   |    PAYLOAD   |
+---------------+--------------+
|      2        |   Variable   |
+---------------+--------------+
```

If datagrams are carried by QUIC datagram extension, the payload is sent directly without length field (only with `Context ID`).

SunnyQUIC

SunnyQUIC is the twin protocol of **ShadowQUIC**. It is nearly the same as **ShadowQUIC**. The only difference is that **SunnyQUIC** gives up JLS layer and provide a QUIC layer authentication. The underlying connection is native QUIC connection.

Authentication

SunnyQUIC adds a new *authentication* command. The command is carried by the bistream.

```
+-------+---------------+
|  CMD  |   AUTH_HASH   |
+-------+---------------+
|   1   |       64      |
+-------+---------------+
```

The `CMD` field is `0x5` for authentication command, The `AUTH_HASH` field is truncated 64byte hash: `SHA256(username:password)[0..64]`

For client, the authentication command can be issued in parallel with other proxy commands.

For server, it should block any commands until authentication is finished. If authentication fails, server should terminate the connection.