socks examples

Example for SOCKS 'associate' command

The associate command tells the SOCKS proxy server to establish a UDP relay. The server binds to a new UDP port and communicates the newly opened port back to the origin client. From here, any SOCKS UDP frame packets sent to this special UDP port on the Proxy server will be forwarded to the desired destination, and any responses will be forwarded back to the origin client (you).

This can be used for things such as DNS queries, and other UDP communicates.

Connection Steps

1. Client -(associate)-> Proxy (Tells the proxy to create a UDP relay and bind on a new port)
2. Client <-(port)- Proxy (Tells the origin client which port it opened and is accepting UDP frame packets on)

At this point the proxy is accepting UDP frames on the specified port.

3. Client –(udp frame) -> Proxy -> Destination (The origin client sends a UDP frame to the proxy on the UDP port, and the proxy then forwards it to the destination specified in the UDP frame.)
4. Client <–(udp frame) <– Proxy <– Destination (The destination client responds to the udp packet sent in #3)

Usage

The 'associate' command can only be used by creating a new SocksClient instance and listening for the 'established' event.

Note: UDP packets relayed through the proxy servers are packaged in a special Socks UDP frame format. SocksClient.createUDPFrame() and SocksClient.parseUDPFrame() create and parse these special UDP packets.

```
import * as dgram from 'dgram';
import { SocksClient, SocksClientOptions } from 'socks';

// Create a local UDP socket for sending/receiving packets to/from the proxy.
const udpSocket = dgram.createSocket('udp4');
udpSocket.bind();

// Listen for incoming UDP packets from the proxy server.
udpSocket.on('message', (message, rinfo) => {
  console.log(SocksClient.parseUDPFrame(message));
  /*
  { frameNumber: 0,
    remoteHost: { host: '8.8.8.8', port: 53 }, // The remote host that replied with a UDP pa
    data: <Buffer 74 65 73 74 0a> // The data
```

```
  }
  */
});

const options: SocksClientOptions = {
  proxy: {
    host: '104.131.124.203',
    port: 1081,
    type: 5
  },

  // This should be the ip and port of the expected client that will be sending UDP frames t
  // Most SOCKS servers accept 0.0.0.0 as a wildcard address to accept UDP frames from any s
  destination: {
    host: '0.0.0.0',
    port: 0
  },

  command: 'associate'
};

const client = new SocksClient(options);

// This event is fired when the SOCKS server has started listening on a new UDP port for UDP
client.on('established', info => {
  console.log(info);
  /*
  {
    socket: <Socket ...>,
    remoteHost: { // This is the remote port on the SOCKS proxy server to send UDP frame pac
      host: '104.131.124.203',
      port: 58232
    }
  }
  */

  // Send a udp frame to 8.8.8.8 on port 53 through the proxy.
  const packet = SocksClient.createUDPFrame({
    remoteHost: { host: '8.8.8.8', port: 53 },
    data: Buffer.from('hello') // A DNS lookup in the real world.
  });

  // Send packet.
  udpSocket.send(packet, info.remoteHost.port, info.remoteHost.host);
});
```

```
// SOCKS proxy failed to bind.
client.on('error', () => {
  // Handle errors
});

// Start connection
client.connect();
```