socks examples

Example for SOCKS 'bind' command

The bind command tells the SOCKS proxy server to bind and listen on a new TCP port for an incoming connection. It communicates the newly opened port back to the origin client. Once a incoming connection is accepted by the SOCKS proxy server it then communicates the remote host that connected to the SOCKS proxy back through the same initial connection via the origin client.

This can be used for things such as FTP clients which require incoming TCP connections, etc.

Connection Steps

1. Client -(bind)-> Proxy (Tells the proxy to bind to a new port)
2. Client <-(port)- Proxy (Tells the origin client which port it opened)
3. Client2 –> Proxy (Other client connects to the proxy on this port)
4. Client <–(client2's host info) (Proxy tells the origin client who connected to it)
5. Original connection to the proxy is now a full TCP stream between client (you) and client2.
6. Client <–> Proxy <–> Client2

Usage

The 'bind' command can only be used by creating a new SocksClient instance and listening for 'bound' and 'established' events.

```
const SocksClient = require('socks').SocksClient;

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

  // This should be the ip and port of the expected client that will connect to the SOCKS p
  // Most SOCKS servers accept 0.0.0.0 as a wildcard address to accept any client.
  destination: {
    host: '0.0.0.0',
    port: 0
  },

  command: 'bind'
};
```

```
const client = new SocksClient(options);

// This event is fired when the SOCKS server has started listening on a new port for incomi
client.on('bound', (info) => {
  console.log(info);
  /*
  {
    socket: <Socket ...>,
    remoteHost: { // This is the remote ip and port of the SOCKS proxy that is now acceptin
      host: '104.131.124.203',
      port: 49928
    }
  }
  */
});

// This event is fired when the SOCKS server has accepted an incoming connection on the new
client.on('established', (info) => {
  console.log(info);
  /*
  {
    socket: <Socket ...>,
    remoteHost: { // This is the remote ip and port that connected to the SOCKS proxy on th
      host: '1.2.3.4',
      port: 58232
    }
  }
  */

  // At this point info.socket is a regular net.Socket TCP connection between client and cl

  console.log(info.socket);
  // <Socket ...>  (this is a raw net.Socket that is established to the destination host th
});

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