Reflection note: Socks5 proxy servers

Isuru Edirisinghe

15TH September 2025

Introduction

I refined my knowledge about proxy servers and additionally i learned about the Socks5 proxy servers. I researched and identified that the socks5 proxies supports various authentication methods and can handle any type of traffic, including TCP and UDP, making it the best choice for users who need a robust proxy solution.

So, as the advantages of using this Socks5 proxy i can mention the below points.

- Socks5 proxies have extra privacy and speed (mask their IP addresses)
- lends IP addresses gives access to Geo-restricted contents
- suitable for diverse applications
- anonymity: tracking protection
- Secure Data Transfers: SOCKS5 proxies can be used to secure data transfers between clients and servers, making them useful for businesses.

This SOCKS protocol makes a tunnel via a proxy server, through which the client requests are directed. It reaches the proxy server, which gets the web page for the client, and hands it back to him. It is not limited to HTTP traffic only, like Squid proxy server, but can proxy any TCP traffic. It has older version 4 and new version 5, which support some new capabilities, like client authentication, UDP packages and server-side named resolution.

for the Testing of Proxy servers, we can use curl

While writing the Node.js code, I had to learn how the SOCKS5 handshake actually works at the byte level - reading RFC1928 and RFC1929 to understand the authentication negotiation and CONNECT request format. I also picked up how to use Node's built-in `net` module to create raw TCP sockets and forward data without any HTTP helpers.

Debugging

For debugging, I logged every incoming buffer in hex and compared it to the RFC diagrams. Wireshark and simple `console.log` statements helped me catch mistakes in the handshake bytes and authentication logic quickly.

Improvements if given more time

If I had more time, I'd add full RFC support—UDP ASSOCIATE, IPv6, better error handling—and write automated tests. I'd also layer in TLS and stronger authentication for a production-ready proxy.

I got the idea of developing a proxy using SSH and PuTTY on Linux, since I already have experience using PuTTY to log in to AWS EC2 instances. I'm interested in trying that approach as well and find it very exciting.

References

- https://www.privateproxyguide.com/building-a-socks5-proxy-server-with-node-js/
- https://www.npmjs.com/package/socks5server
- https://en.iplau.com/nodejs-socks5.html
- https://www.pyproxy.com/information/Building-a-Pure-Source-SOCKS5-Proxy-
- https://www.proxyrack.com/blog/create-your-own-socks5-proxies-using-ssh-and-putty/