# DYNAMIC SSH PORT FORWARDING (SSH+SOCKS)

## The Problem to be Solved

Regular SSH port forwarding requires that the local port, the destination IP and the destination port all be specified at the moment the SSH connection is created. This means it can only be used when all that information is known in advance, and does not need to be changed while the connection is open.

This limitation makes it effectively impossible to route applications that make many network connections to many destinations, like a web browser, through regular SSH port forwarding.

Dynamic Port Forwarding makes it possible for any app that can use the standard SOCKS protocol to route traffic though an SSH connection, that includes apps like web browsers, chat clients, and email clients.

## Description

Dynamic port forwarding is a relatively recent addition to SSH, and one of SSH’s little-known gems.

The SOCKS protocol can be used to proxy a TCP connection from any port to any port on behalf of any client that supports the protocol.

It is normally used at the permitter of corporate networks to regulate external internet access. All computers inside the corporation that need to make out-going network connections use the SOCKS proxy, which can then apply any rules to those connection requests the corporation desires. All network connections effectively get broken in two. The clients talk to the SOCKS proxy, and the SOCKS proxy talks to the destination server.

When using SSH dynamic port forwarding, what happens is that a SOCKS server is started on your local computer, running on a port you specify, and it sends all the traffic it proxyies through the SSH connection, and out onto the internet from the remote end of the SSH connection. While the traffic is encapsulated within the SSH connection it's encrypted. Once it leaves the SSH connection it is un-encrypted for the remainder of it's journey.

This really is analogous to a VPN, with the caveat that only traffic sent to the locally running SOCKS proxy is secured.

The good news is that the SOCKS standard is very widely implemented. All the major browser can use SOCKS, and there is OS-level support for SOCKS in Windows and OS X.

The down-side over a real VPN is that you MUST be sure all apps are configured to use the SOCKS proxy before you start to use them, and you must

remove the SOCKS configuration once the SSH connection is closed or all your apps will lose internet access.

# Instructions

To instruct SSH to behave as a SOCKS proxy, use the **-D** flag. The **-D** flag requires that the local port the SOCKS server should listen on be specified. The default SOCKS port is 1080, so that's a good choice. To set up a SOCKS proxy on the default port use a command of the following form:

```
ssh -D 1080 user@computer
```

# Example Use Cases

- **Access local-only web servers on remote servers** – if X11 forwarding is not a viable option for what ever reason, dynamic port forwarding can be used as an alternative to access local-only web interfaces like those for CUPS or webmin. Simply configure your locally running browser to use the SOCKS server provided by SSH, and then browser to the local URL (be sure the browser is not configured to bypass the proxy for local addresses).
- **Securely browse the web in coffee shops/hotels** – if you set up an SSH server in your home, you can use SSH dynamic port forwarding to route all your browser traffic through an SSH connection to your home, safely getting you through the hostile coffee shop or hotel network.

- **Bypass geographic restrictions** – some websites are only available from some countries. If you set up an SSH server in your home, you can use dynamic port forwarding to browse the web from anywhere and make it appear you are at home. This is a great way to keep up with your favourite sports ball matches while travelling. Assuming you have no moral objections to doing so, you could also rent a cheap virtual server in a country who's TV you like better than the TV in your own country, and use dynamic SSH port forwarding to watch streaming TV from that country from anywhere in the world.

Source: https://www.bartbusschots.ie/s/2015/04/06/taming-the-terminal-part-32-of-n-ssh-tunnelling/