
SSH Tunneling Recipes
Developer Toolbox Series
Rafael Luque, OSOCO
osoco software company

Contents

Protocol tunneling

One network protocol —the delivery protocol— encapsulates a different payload protocol.

— Wikipedia

SSH tunneling

A secure shell (SSH) tunnel consists of an encrypted tunnel created through a SSH protocol connection.

— Wikipedia

Common uses

To securely connect to a remote host and have your network traffic encrypted

Common uses

To securely connect to a remote host and have your network traffic encrypted

- You are on a public, non secure, non trusted or unencrypted network.
- You use an insecure protocol like POP3, IMAP, SMTP, FTP, telnet, etc.

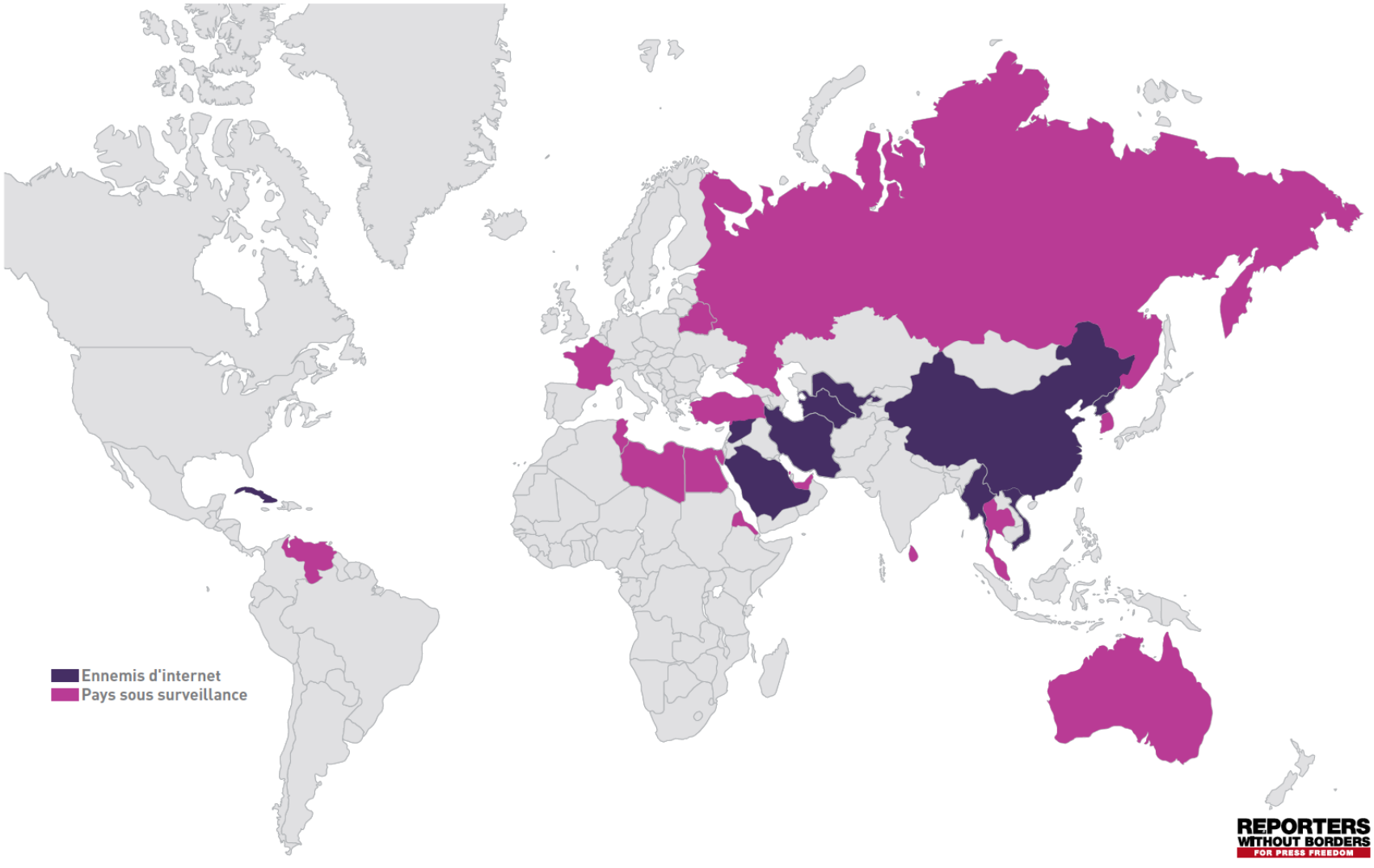
Common uses

To bypass local network restrictions and monitoring services

Common uses

Internet censorship circumvention

Map of cyber-censorship



Common uses

Open *backdoors* to allow outbound connections to hosts behind a firewall

Common uses

X11 forwarding

Common uses

Access services bound to the loopback interface

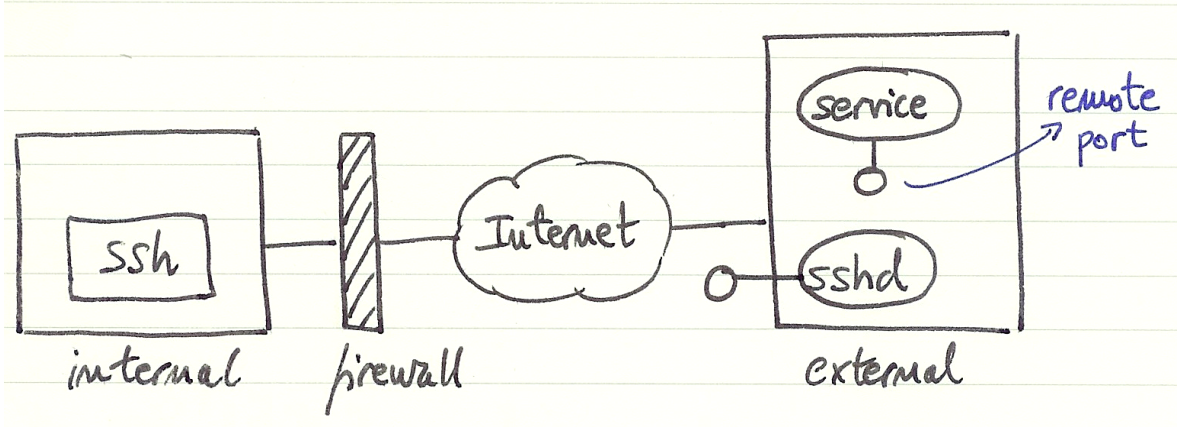
Contents

Local port forwarding

Local port forwarding (aka outgoing tunneling) forwards traffic coming to a local port to a specified remote port

Local port forwarding

Recipe #1: Access a remote service behind a firewall

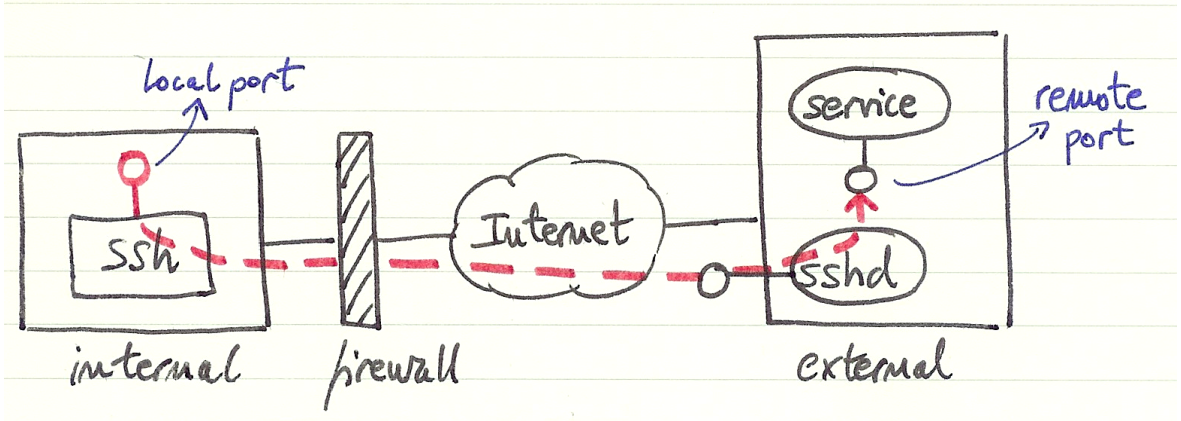



```
ssh -fN -L <localport>:localhost:<remoteport> user@external
```

The service is available on the loopback interface only.

Local port forwarding

Recipe #1: Access a remote service behind a firewall

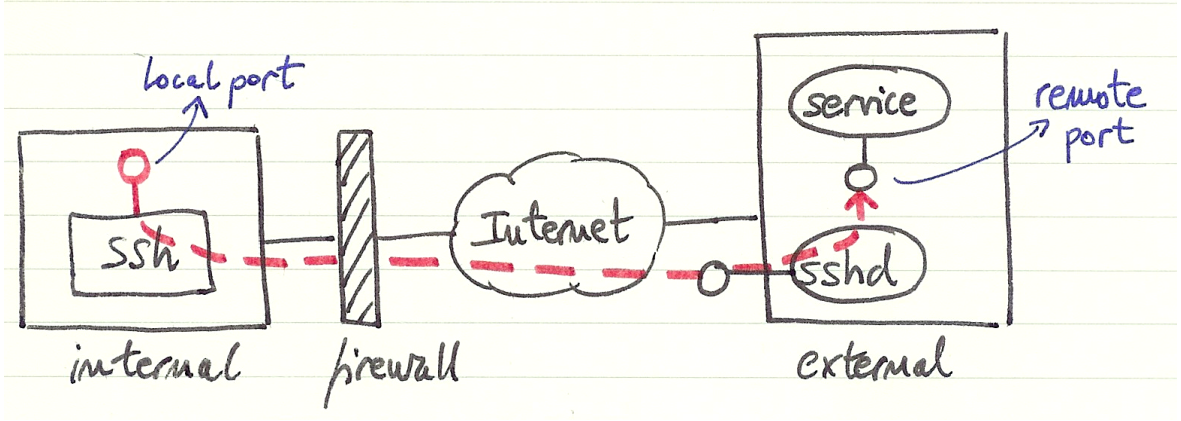



```
ssh -fN -L <localport>:localhost:<remoteport> user@external
```

The service is available on the loopback interface only.

Local port forwarding

Recipe #1: Access a remote service behind a firewall

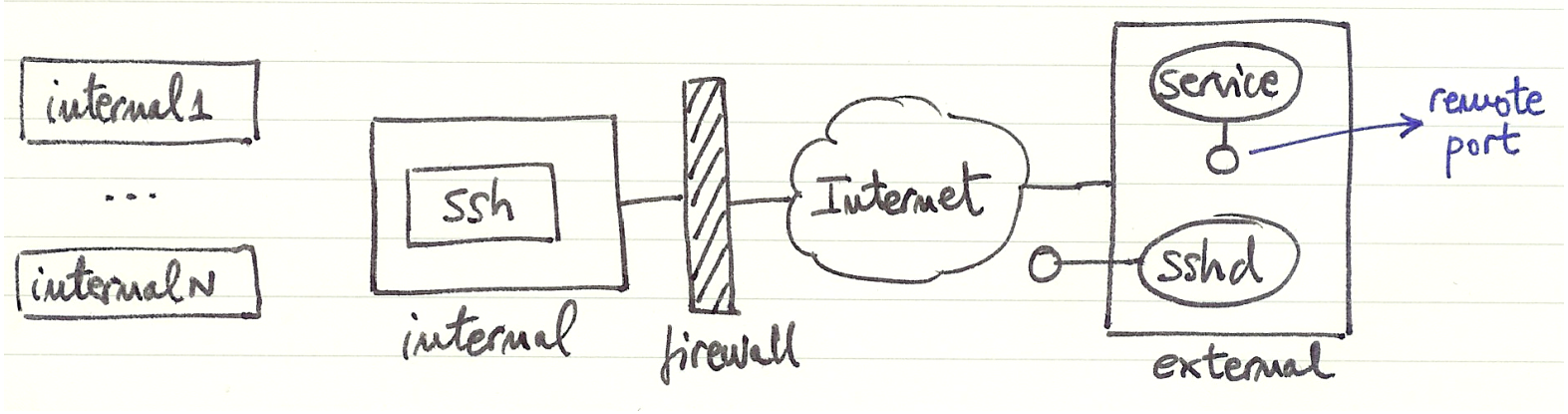



```
ssh -fN -L <localport>:localhost:<remoteport> user@external
```

The service is available on the loopback interface only.

Local port forwarding

Recipe #2: Access a remote service from any host behind the firewall



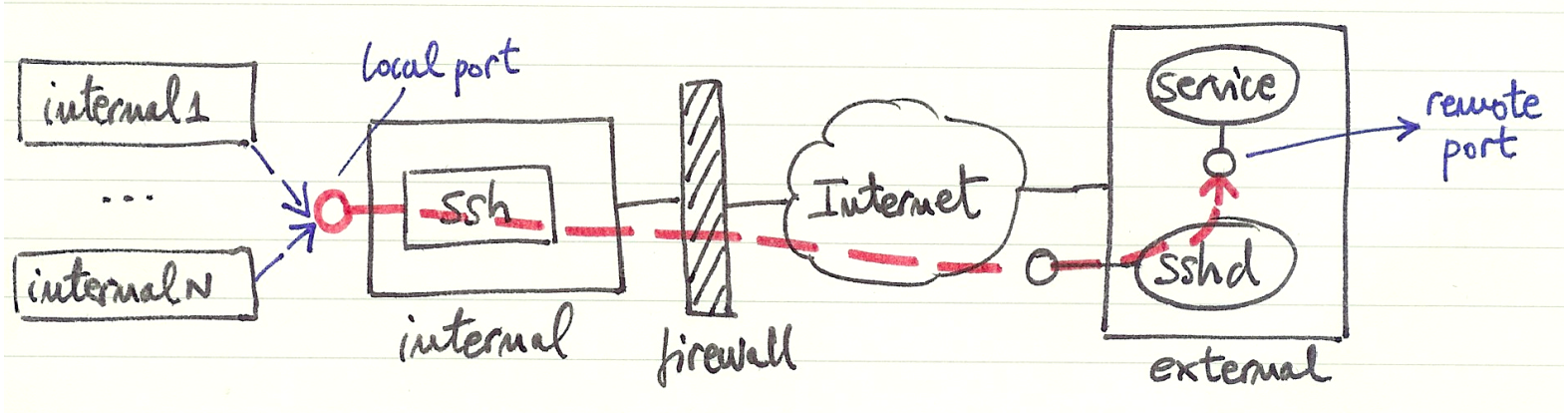

```
ssh -fN -L 0.0.0.0:<localport>:localhost:<remoteport> user@external
```

or

```
ssh -fN -g -L <localport>:localhost:<remoteport> user@external
```

Local port forwarding

Recipe #2: Access a remote service from any host behind the firewall



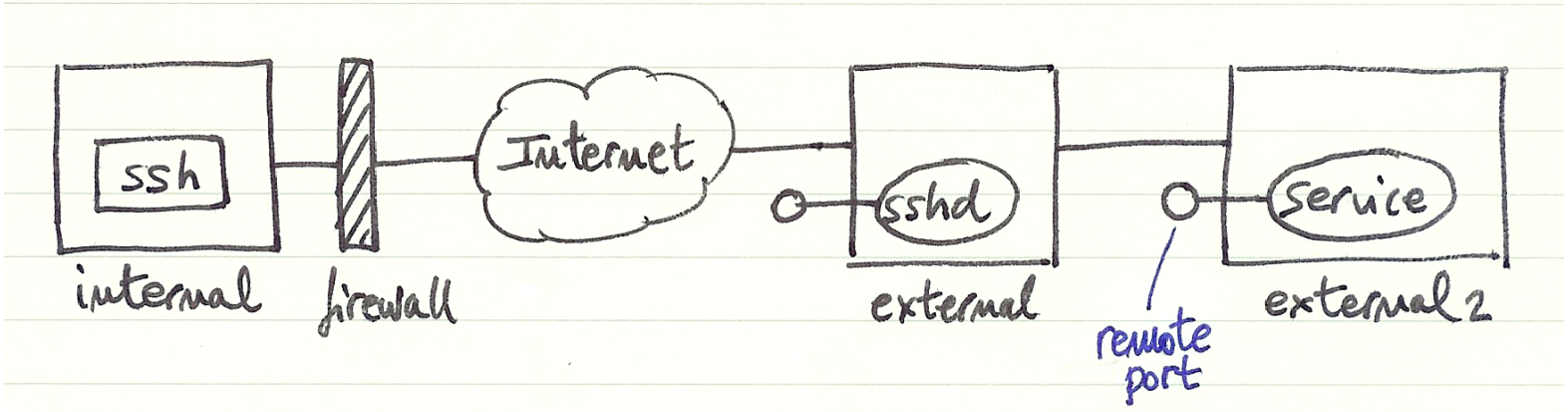

```
ssh -fN -L 0.0.0.0:<localport>:localhost:<remoteport> user@external
```

or

```
ssh -fN -g -L <localport>:localhost:<remoteport> user@external
```

Local port forwarding

Recipe #3: Access a remote service visible from the ssh server

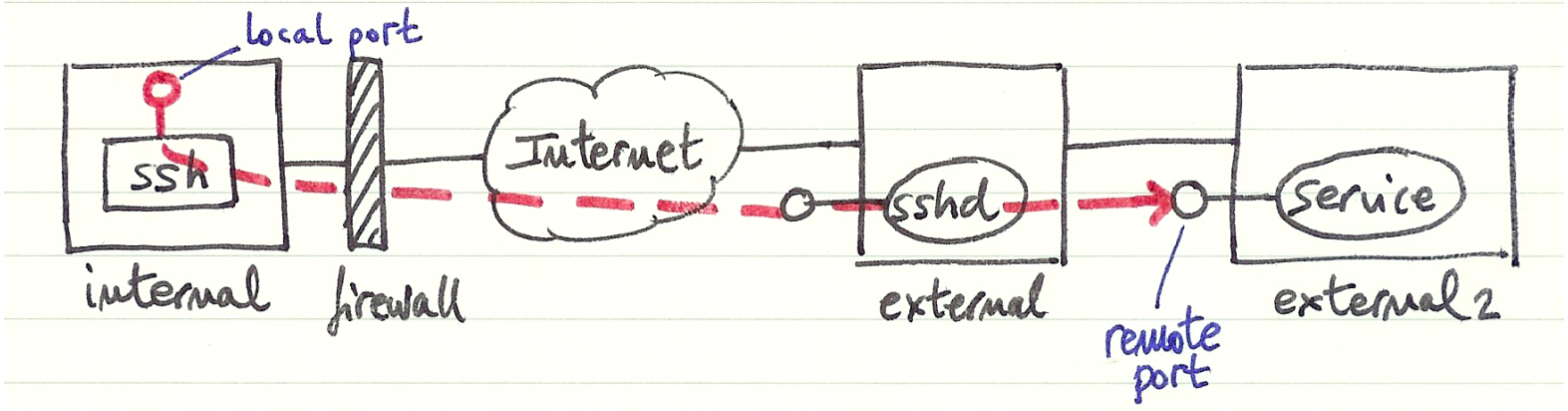



```
ssh -fN -L <localport>:external2:<remoteport> user@external
```

The service is available on the loopback interface only.

Local port forwarding

Recipe #3: Access a remote service visible from the ssh server

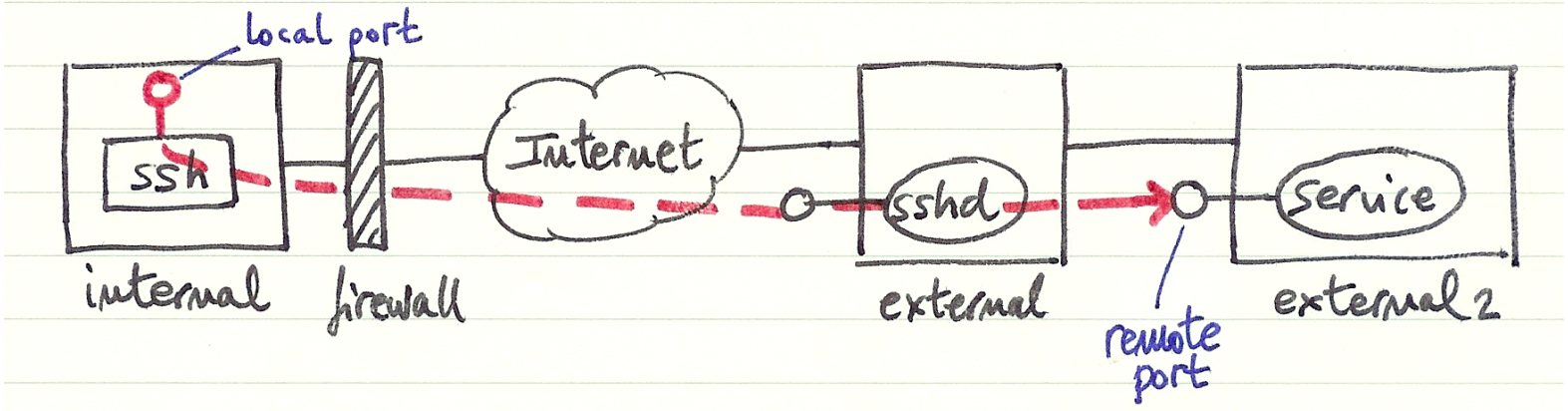



```
ssh -fN -L <localport>:external2:<remoteport> user@external
```

The service is available on the loopback interface only.

Local port forwarding

Recipe #3: Access a remote service visible from the ssh server

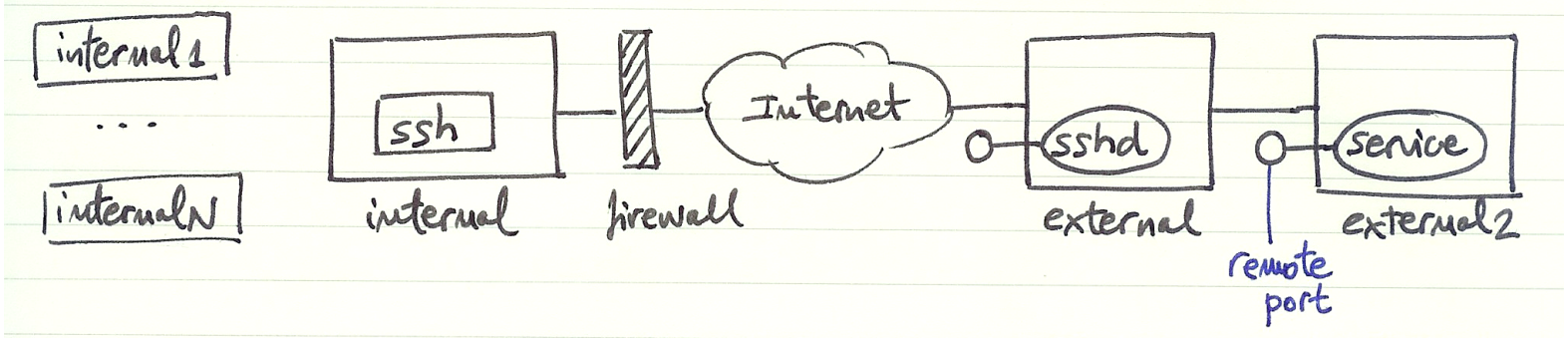



```
ssh -fN -L <localport>:external2:<remoteport> user@external
```

The service is available on the loopback interface only.

Local port forwarding

Recipe #4: Access a remote service visible from the ssh server for any host behind the firewall



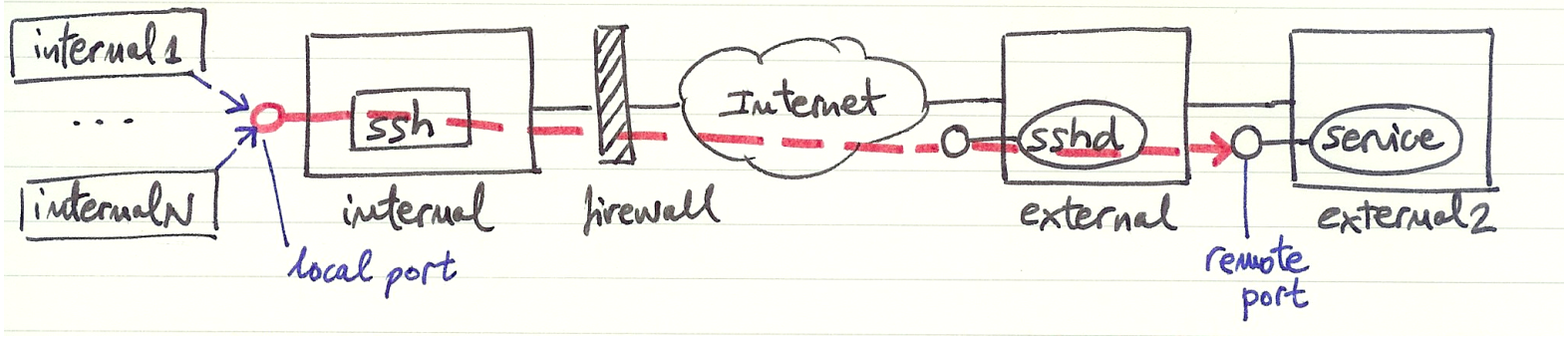

```
ssh -fN -L 0.0.0.0:<localport>:external2:<remoteport> user@external
```

or

```
ssh -fN -g -L <localport>:external2:<remoteport> user@external
```

Local port forwarding

Recipe #4: Access a remote service visible from the ssh server for any host behind the firewall




```
ssh -fN -L 0.0.0.0:<localport>:external2:<remoteport> user@external
```

or

```
ssh -fN -g -L <localport>:external2:<remoteport> user@external
```

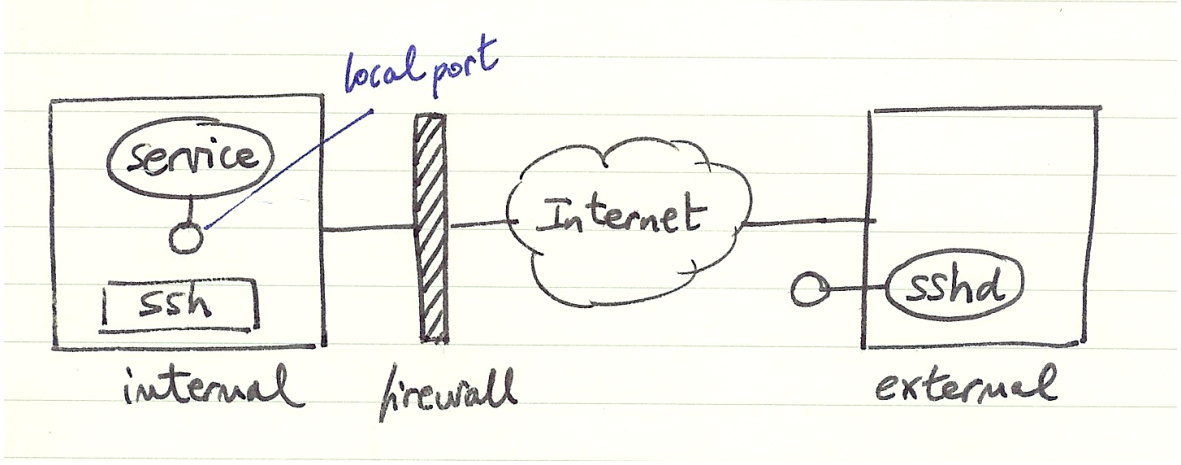
Contents

Remote port forwarding

Remote port forwarding (aka incoming tunneling) forwards traffic coming to a remote port to a specified local port

Remote port forwarding

Recipe #5: Access a service behind a firewall from the ssh server

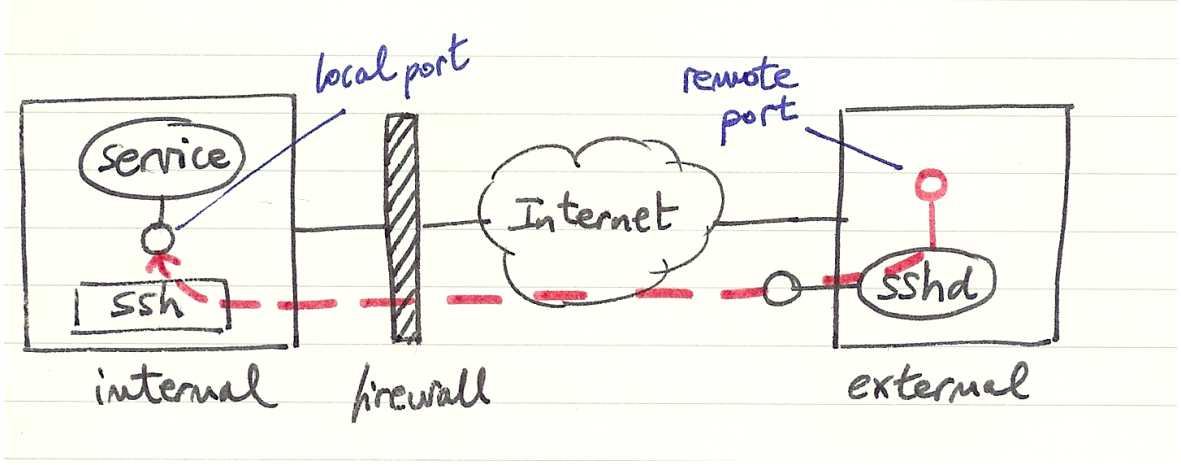



```
ssh -fN -R <remoteport>:localhost:<localport> user@external
```

The service is available on the loopback interface only.

Remote port forwarding

Recipe #5: Access a service behind a firewall from the ssh server

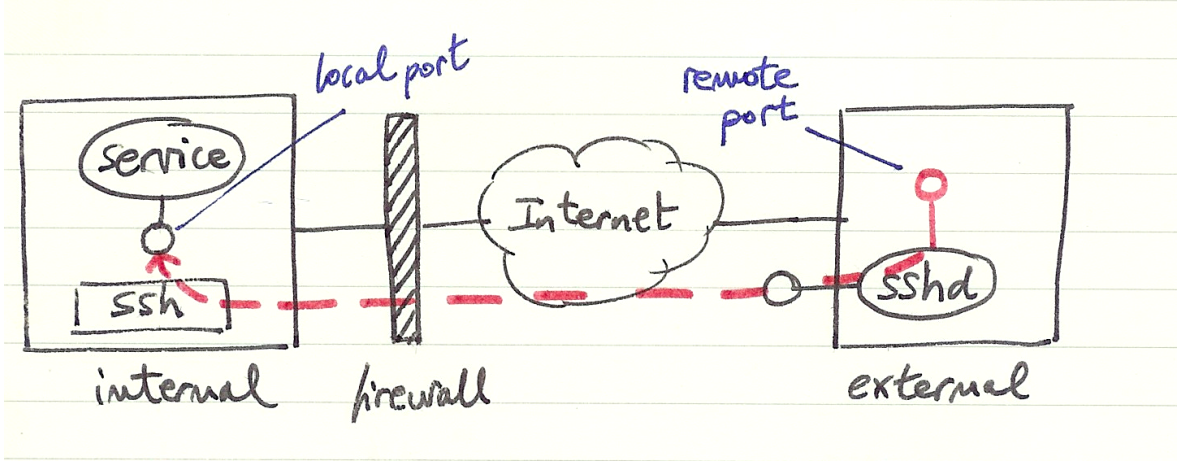



```
ssh -fN -R <remoteport>:localhost:<localport> user@external
```

The service is available on the loopback interface only.

Remote port forwarding

Recipe #5: Access a service behind a firewall from the ssh server

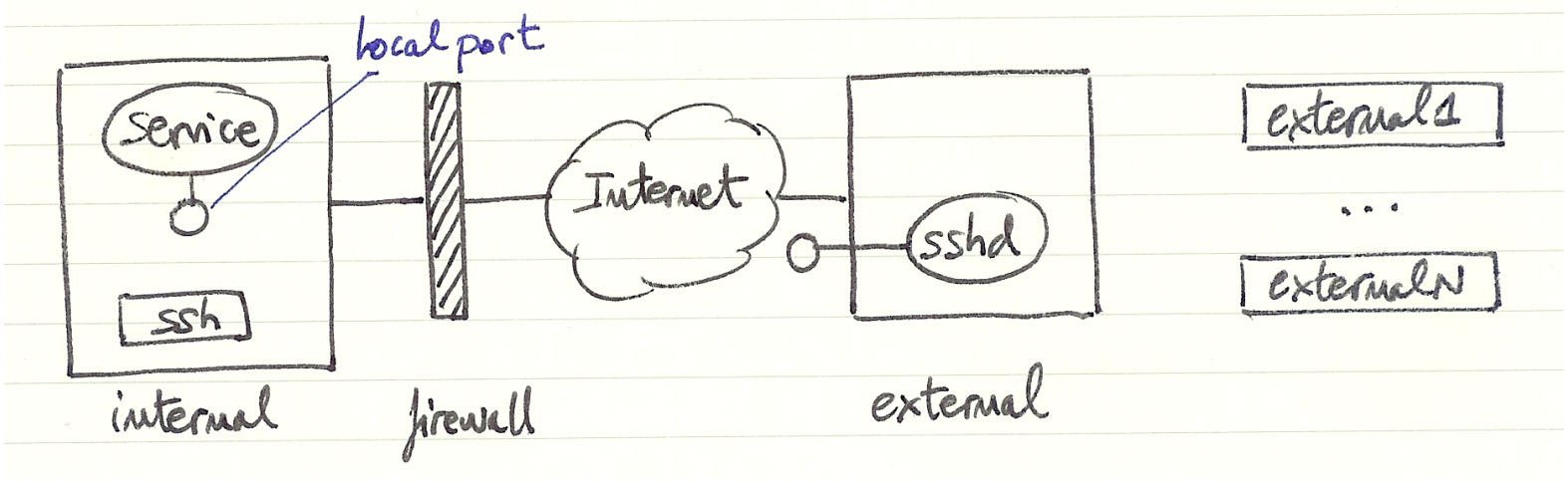



```
ssh -fN -R <remoteport>:localhost:<localport> user@external
```

The service is available on the loopback interface only.

Remote port forwarding

Recipe #6: Access a service behind a firewall from any external host with access to the ssh server



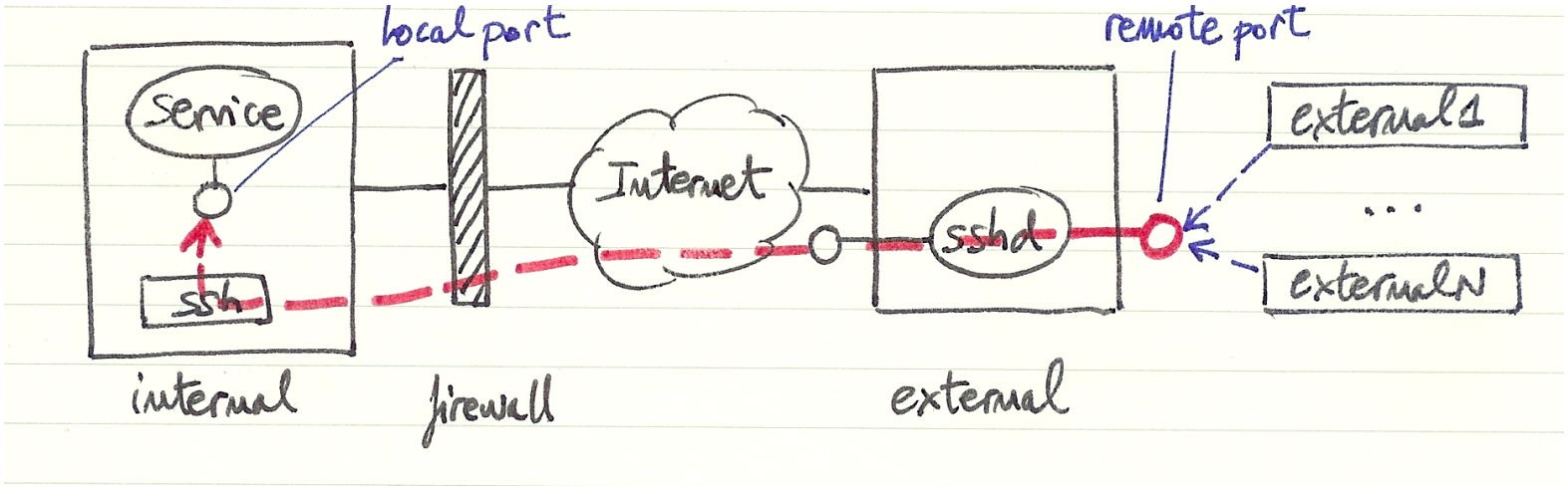

```
ssh -fN -R 0.0.0.0:<remoteport>:localhost:<localport> user@external
```

Edit `/etc/ssh/sshd_config` at ssh server to allow the client to select the address to which the forwarding is bound:

```
GatewayPorts clientspecified
```

Remote port forwarding

Recipe #6: Access a service behind a firewall from any external host with access to the ssh server



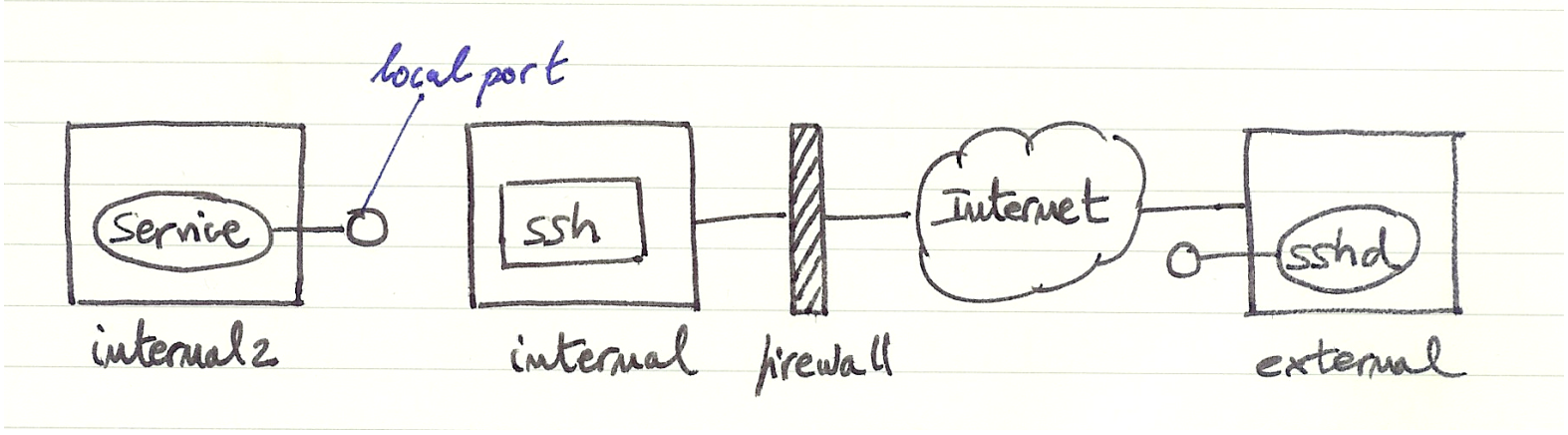

```
ssh -fN -R 0.0.0.0:<remoteport>:localhost:<localport> user@external
```

Edit `/etc/ssh/sshd_config` at ssh server to allow the client to select the address to which the forwarding is bound:

```
GatewayPorts clientspecified
```

Remote port forwarding

Recipe #7: Access a service in a host accesible by the ssh client from the ssh server

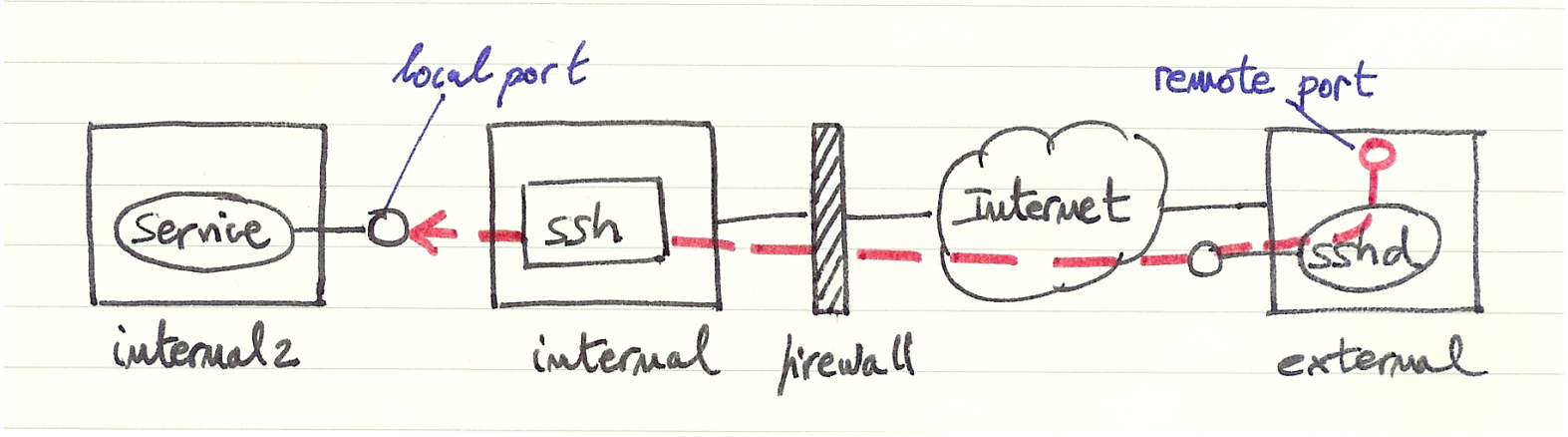



```
ssh -fN -R <remoteport>:internal2:<localport> user@external
```

The service is available on the loopback interface only.

Remote port forwarding

Recipe #7: Access a service in a host accesible by the ssh client from the ssh server

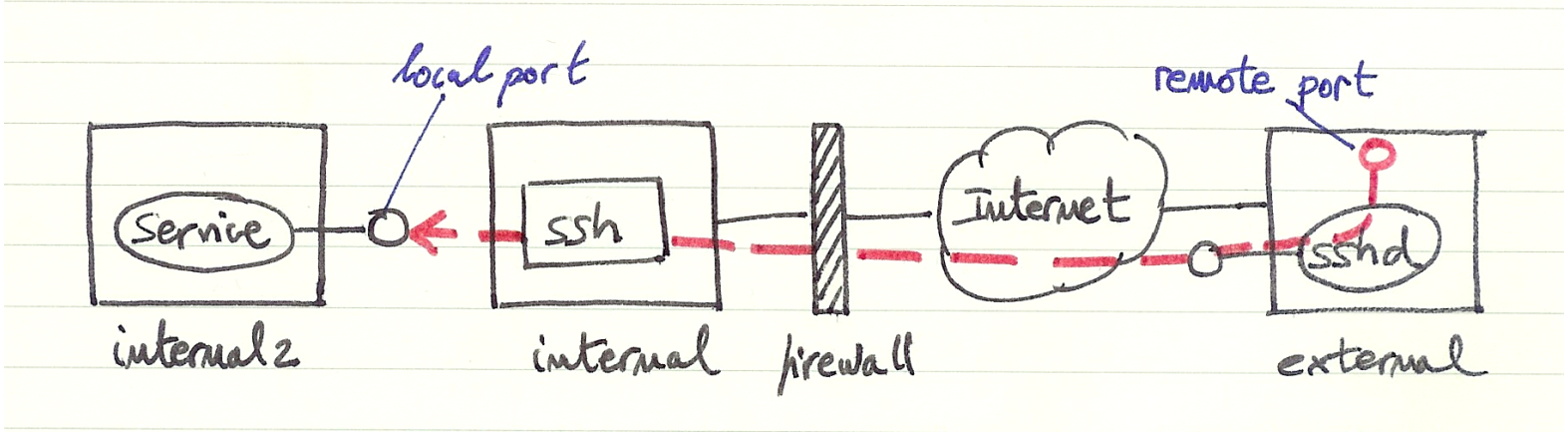



```
ssh -fN -R <remoteport>:internal2:<localport> user@external
```

The service is available on the loopback interface only.

Remote port forwarding

Recipe #7: Access a service in a host accesible by the ssh client from the ssh server

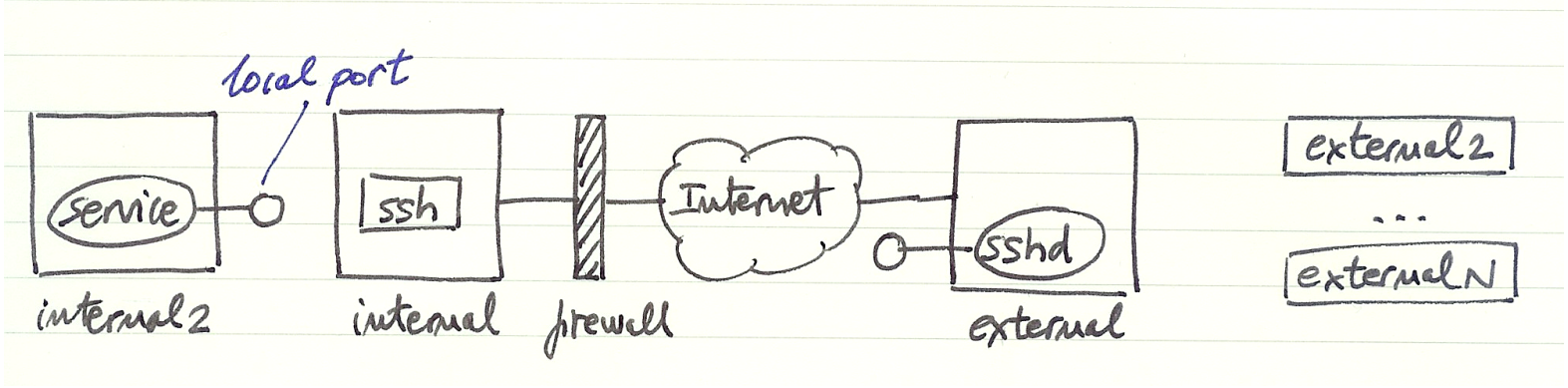



```
ssh -fN -R <remoteport>:internal2:<localport> user@external
```

The service is available on the loopback interface only.

Remote port forwarding

Recipe #8: Access a service in a host accesible by the ssh client from any host with access to the ssh server



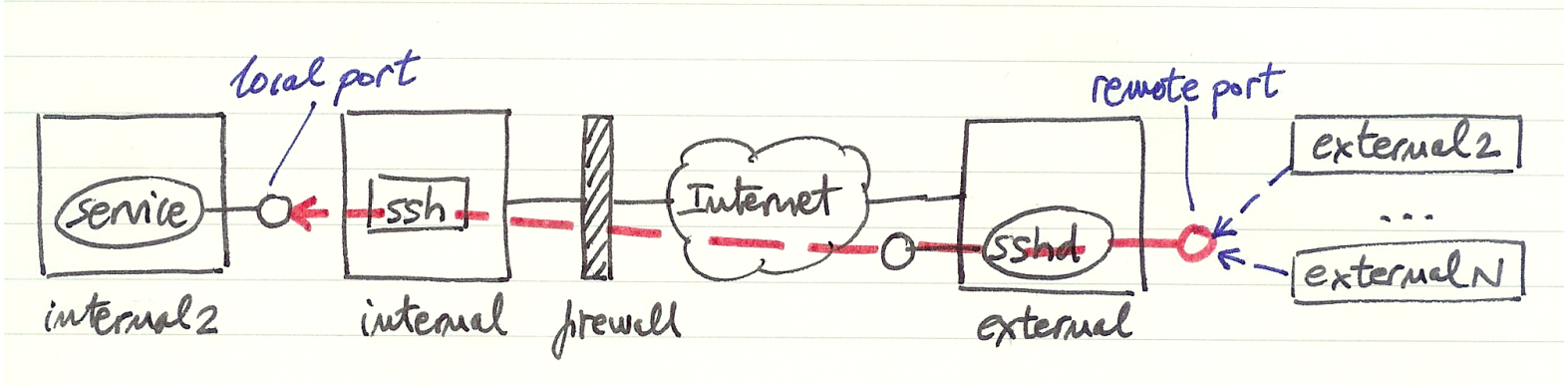

```
ssh -fN -R 0.0.0.0:<remoteport>:internal2:<localport> user@external
```

Edit `/etc/ssh/sshd_config` at server to allow the client to select the address to which the forwarding is bound:

```
GatewayPorts clientspecified
```

Remote port forwarding

Recipe #8: Access a service in a host accesible by the ssh client from any host with access to the ssh server




```
ssh -fN -R 0.0.0.0:<remoteport>:internal2:<localport> user@external
```

Edit `/etc/ssh/sshd_config` at server to allow the client to select the address to which the forwarding is bound:

```
GatewayPorts clientspecified
```

Contents

SOCKS

SOCKS is an Internet protocol that routes network packets between a client and server through a proxy server

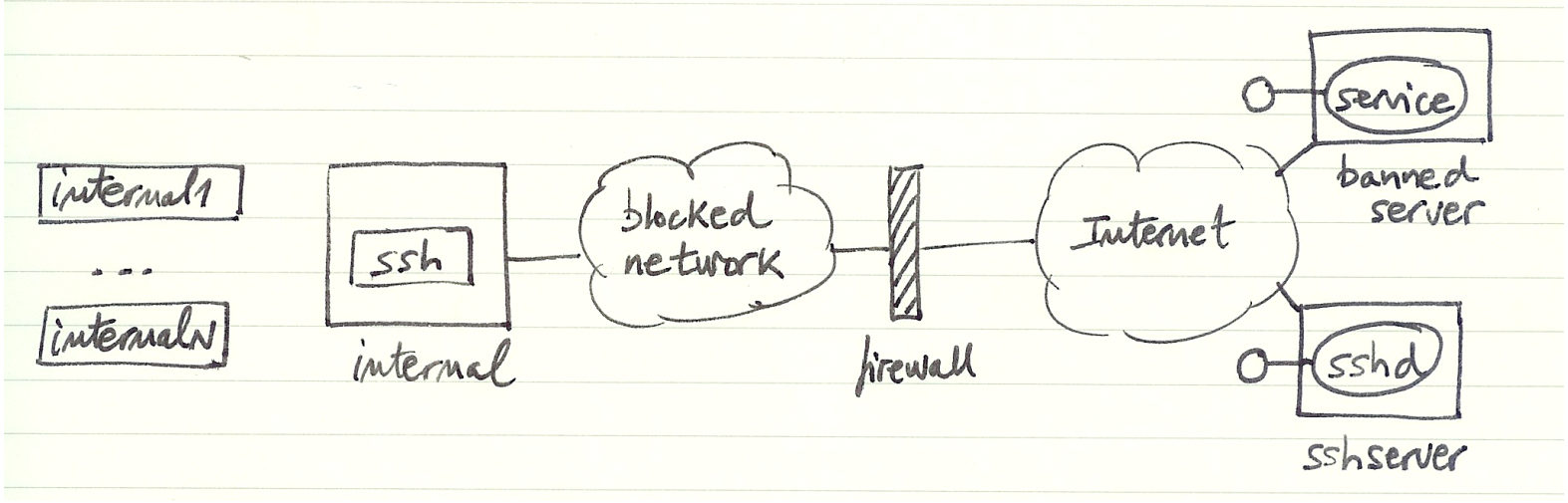
— Wikipedia

SSH dynamic port forwarding

- SSH dynamic port forwarding allows the user to create a local SOCKS proxy.
- Free the user from the limitations of connecting only to a predefined remote port and server.
- Circumvention tool allowing to bypass Internet filtering to access content otherwise blocked by governments, workplaces and schools.

Dynamic port forwarding with SOCKS

Recipe #9: Setup a SOCKS proxy



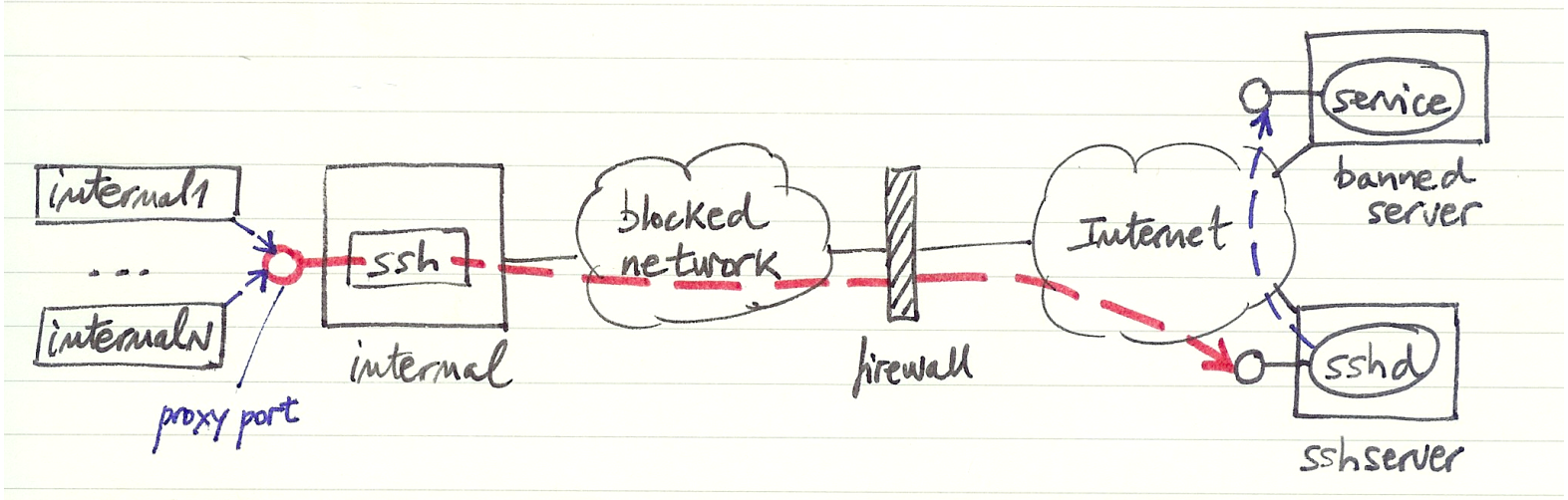

```
ssh -fN -D <proxyport> user@sshserver
```

To allow any internal host to use the proxy:

```
ssh -fN -D 0.0.0.0:<proxyport> user@sshserver
```

Dynamic port forwarding with SOCKS

Recipe #9: Setup a SOCKS proxy




```
ssh -fN -D <proxyport> user@sshserver
```

To allow any internal host to use the proxy:

```
ssh -fN -D 0.0.0.0:<proxyport> user@sshserver
```

Contents

X forwarding

- Using X, you can run remote X applications that open their windows on your local display.
- The X protocol is insecure and wide open to snoopers.
- SSH X forwarding makes the communication secure by tunneling the X protocol:

```
ssh -X user@server xclock
```

Contents

autossh

autossh is a program to start a copy of ssh and monitor it, restarting it as necessary should it die or stop passing traffic.

```
autossh -M <port>[:echo_port] [-f] [SSH OPTIONS]
```

sslh

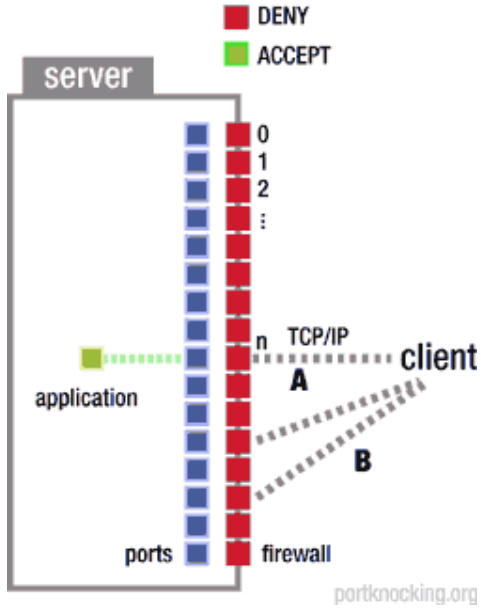
sslh makes it possible to connect to an SSH server or an OpenVPN on port 443 while still serving HTTPS on that port.

Port knocking

port knocking is a method of externally opening ports on a firewall by generating a connection attempt on a set of prespecified closed ports. Once a correct sequence of connection attempts is received, the firewall rules are dynamically modified to allow the host which sent the connection attempts to connect over specific port(s).

— Wikipedia

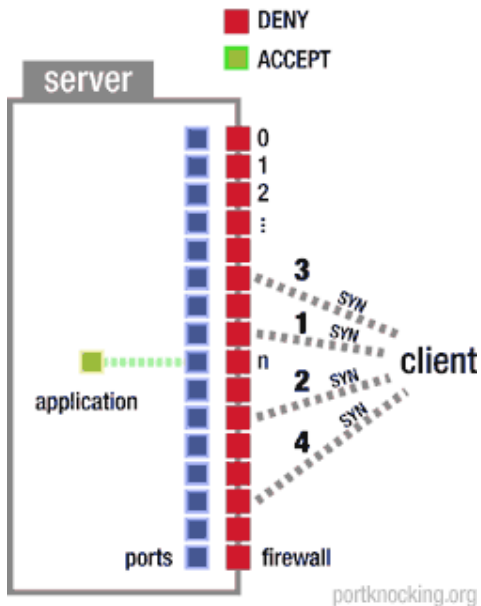
Port knocking



(A) client cannot connect to application listening on port n

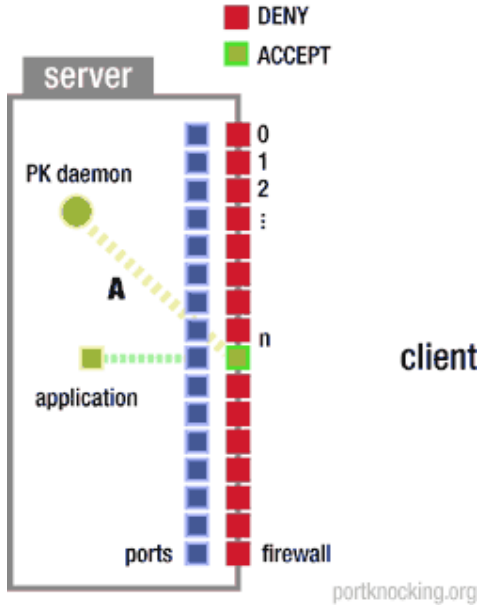
(B) client cannot establish connection to any port

Port knocking



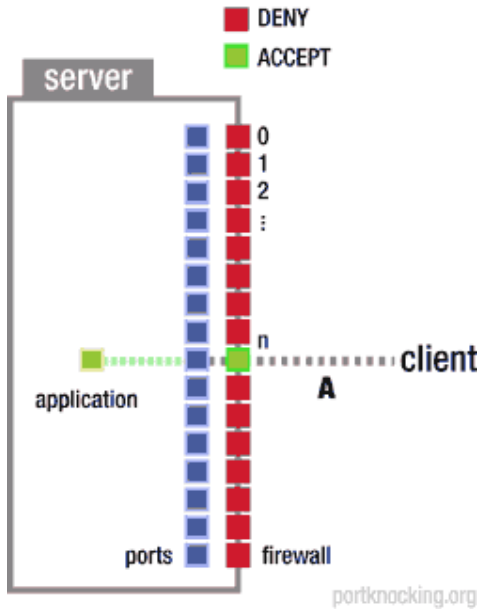
(1,2,3,4) client connects to a well-defined set of ports in a sequence that contains an encrypted message by sending SYN packets; client has a priori knowledge of the port knocking daemon and its configuration, but receives no acknowledgement during this phase because firewall rules preclude any response

Port knocking



(A) server process (a port knocking daemon) intercepts connection attempts and interprets (decrypts and decodes) them as comprising an authentic "port knock"; server carries out specific task based on content of port knock, such as opening port n to client

Port knocking



(A) client connects to port n and authenticates using applications regular mechanism

knockd

knockd is a port-knock server. It listens to all traffic on an ethernet interface, looking for special "knock" sequences of port-hits.

References

- SSH: The Secure Shell:

  `http://docstore.mik.ua/orelly/networking_2ndEd/ssh/index.htm`
- autossh:

  `http://www.harding.motd.ca/autossh/`
- sslh:

  `http://www.rutschle.net/tech/sslh.shtml`
- Port knocking:

  `http://www.portknocking.org/`
- knockd:

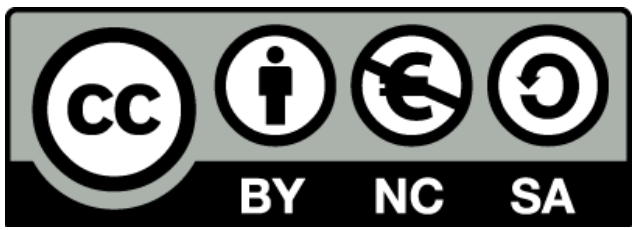
  `http://www.zeroflux.org/projects/knock`

Picture credits

- Cover photo by twicepix:

  `http://www.flickr.com/photos/twicepix/2825051329/`
- The map of the cyber-censorship by Reporters Without Borders:

  `http://march12.rsf.org/en/`



SSH Tunneling Recipes

Developer Toolbox Series

OSOCO

Rafael Luque