演示

图解

代码

Erlang 问世于 1987 年，是一种面向并发的编程语言
最初是由爱立信专门为通信应用设计的，比如控制交换机或者变换协议等，因此非常适合于构建分布式，实时软并行计算系统
它的分布式机制是透明的
Erlang 运行时环境是一个虚拟机，有点像 Java 虚拟机

Erlang 在国外，主要包括爱立信的宽带， GPRS 和 ATM 交换解决方案
计算机电话，消息系统和商业银行等
国内用于游戏开发

WhatsApp 2013
4.5 亿的活跃用户，并且是史上最快达到这个数字的公司
数百个节点 , 8000+CPU, 数百 TB 内存，每秒 Erlang 消息超过 7000 万
在 2011 年 , WhatsApp 单服务器取得 100 万个 tcp 会话
在 2012 年 , tcp 会话发展到了 200 万
2013 年 , 70 亿消息入站， 110 亿消息出站，每天处理 180 亿消息

SSH SERVER

Xshell

直连

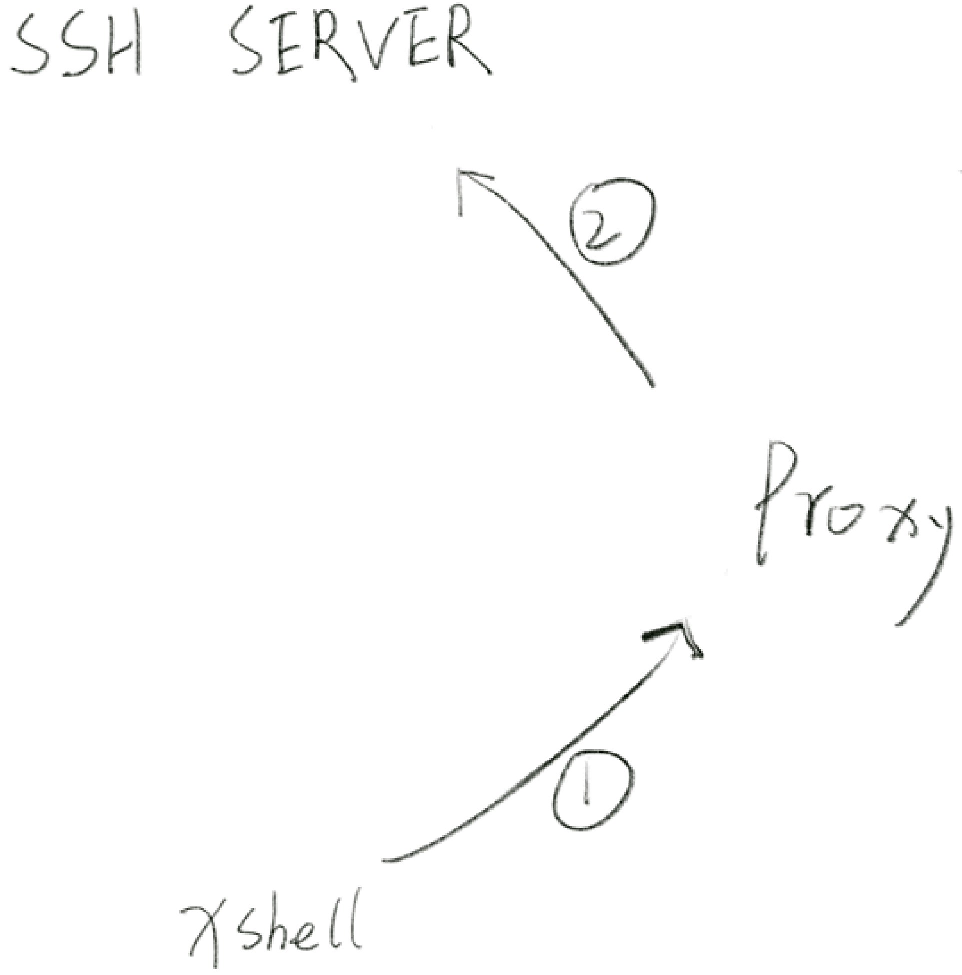
$$\begin{tikzcd} \text{SSH SERVER} \\ \text{Xshell} \arrow[u] \end{tikzcd}$$

SSH SERVER

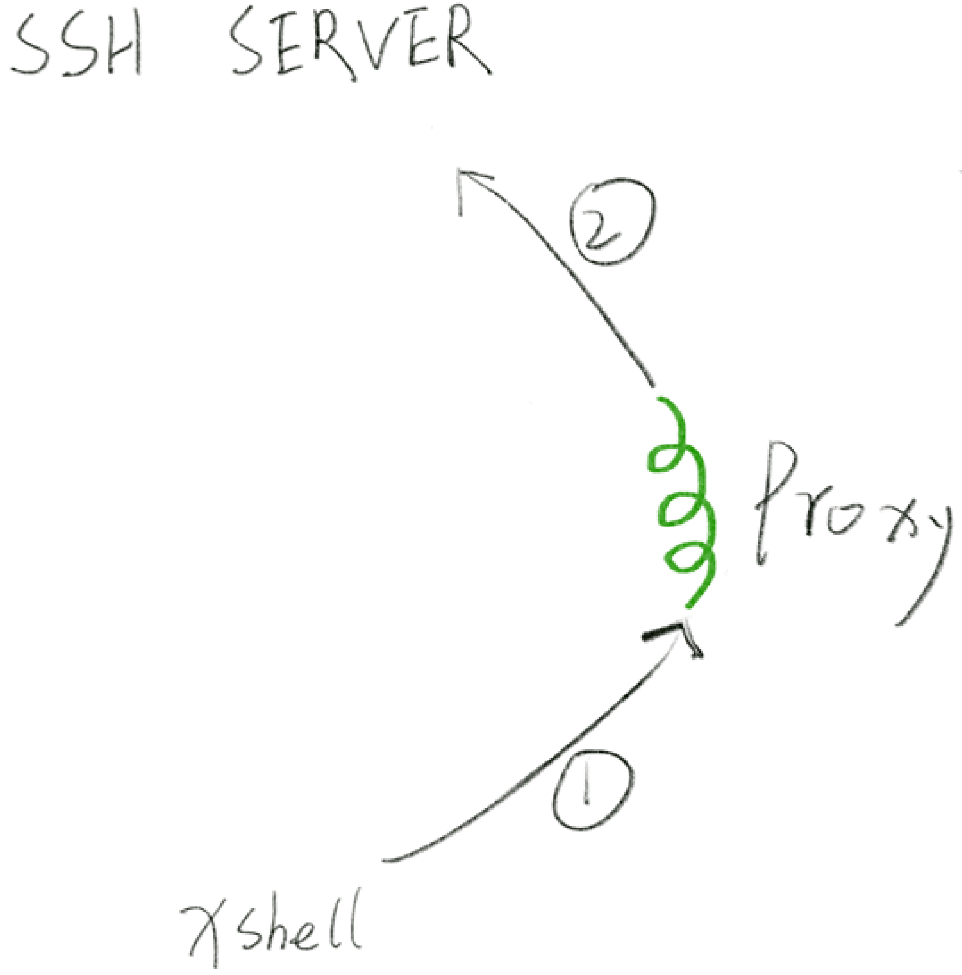
Proxy

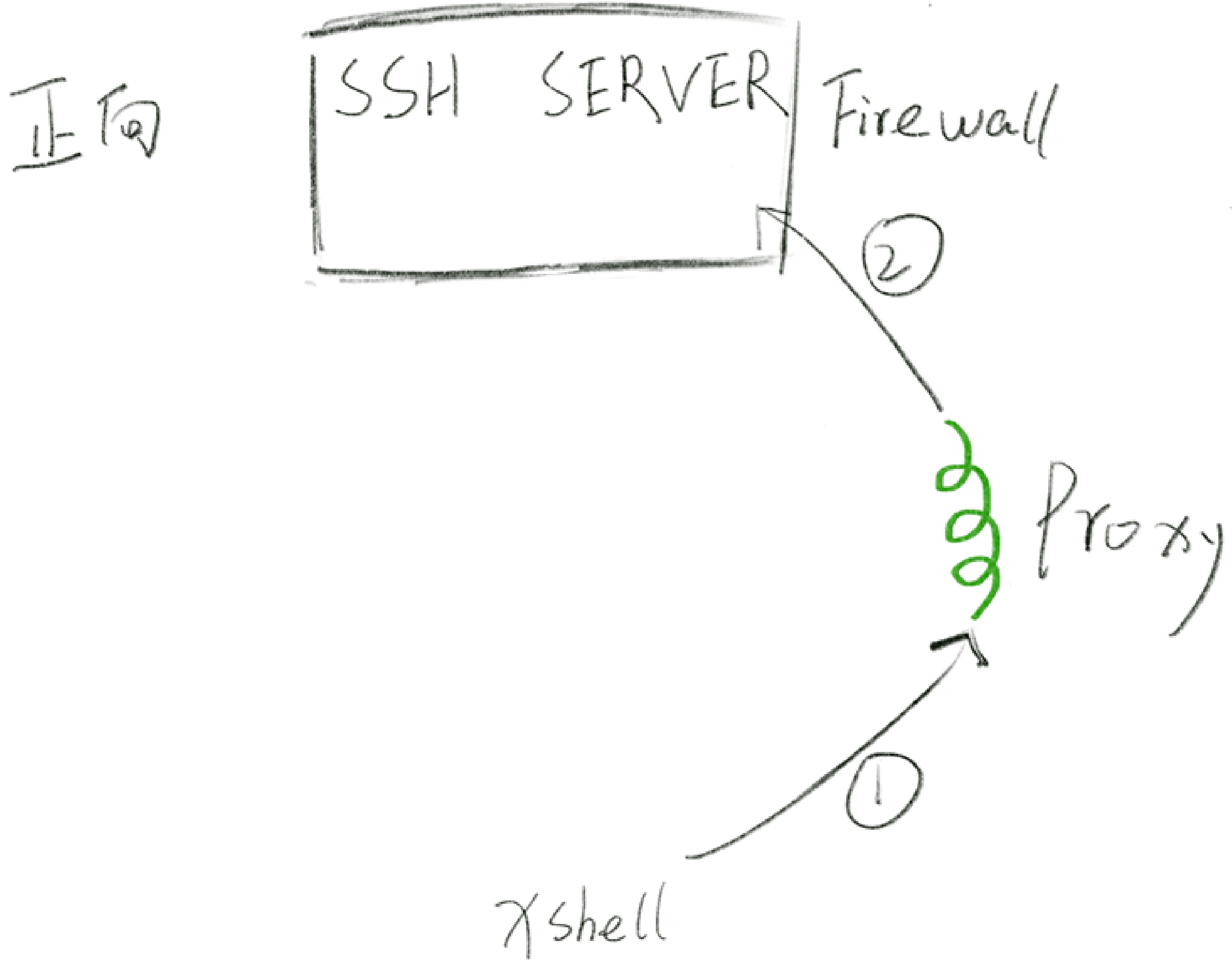
Xshell

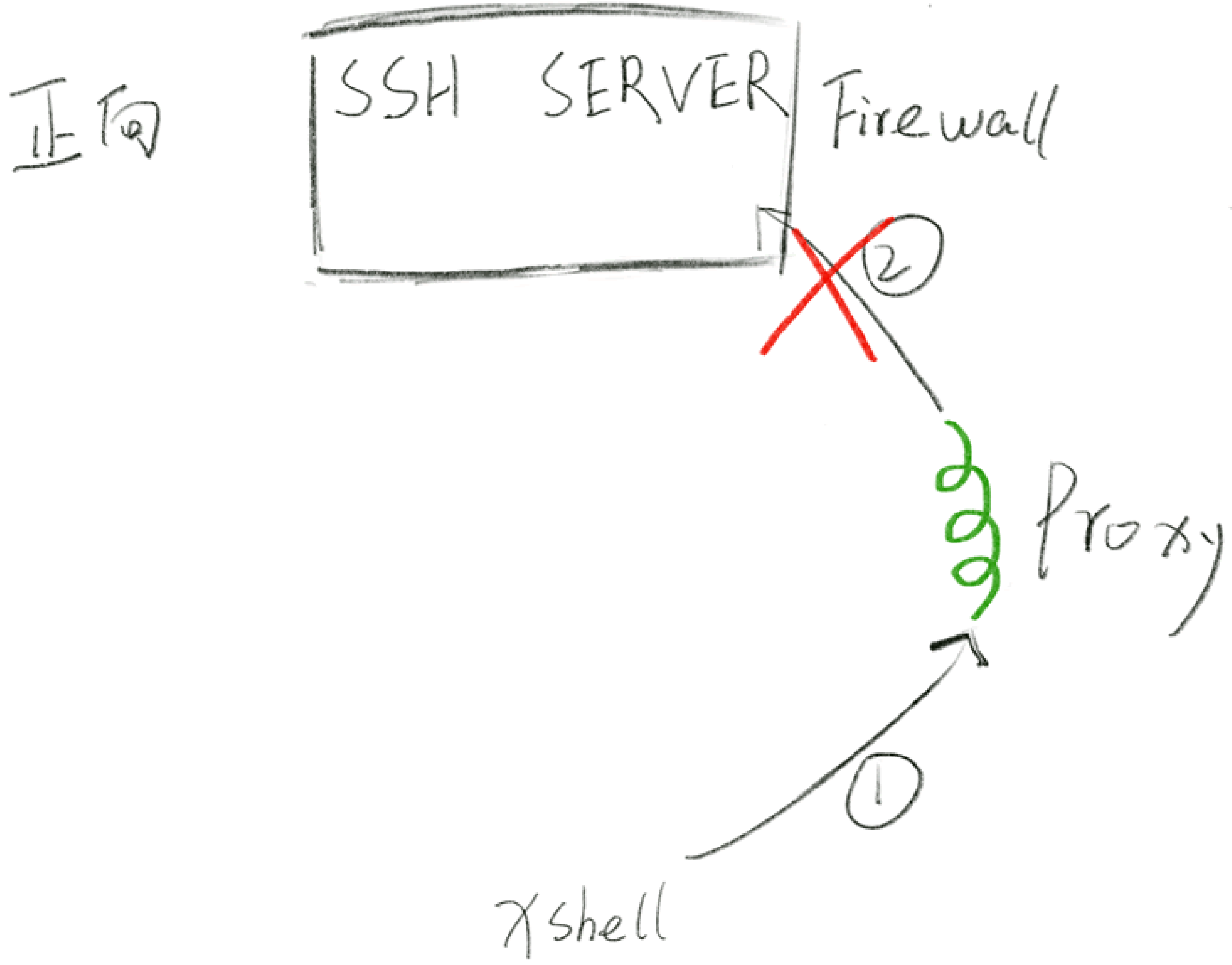
正向

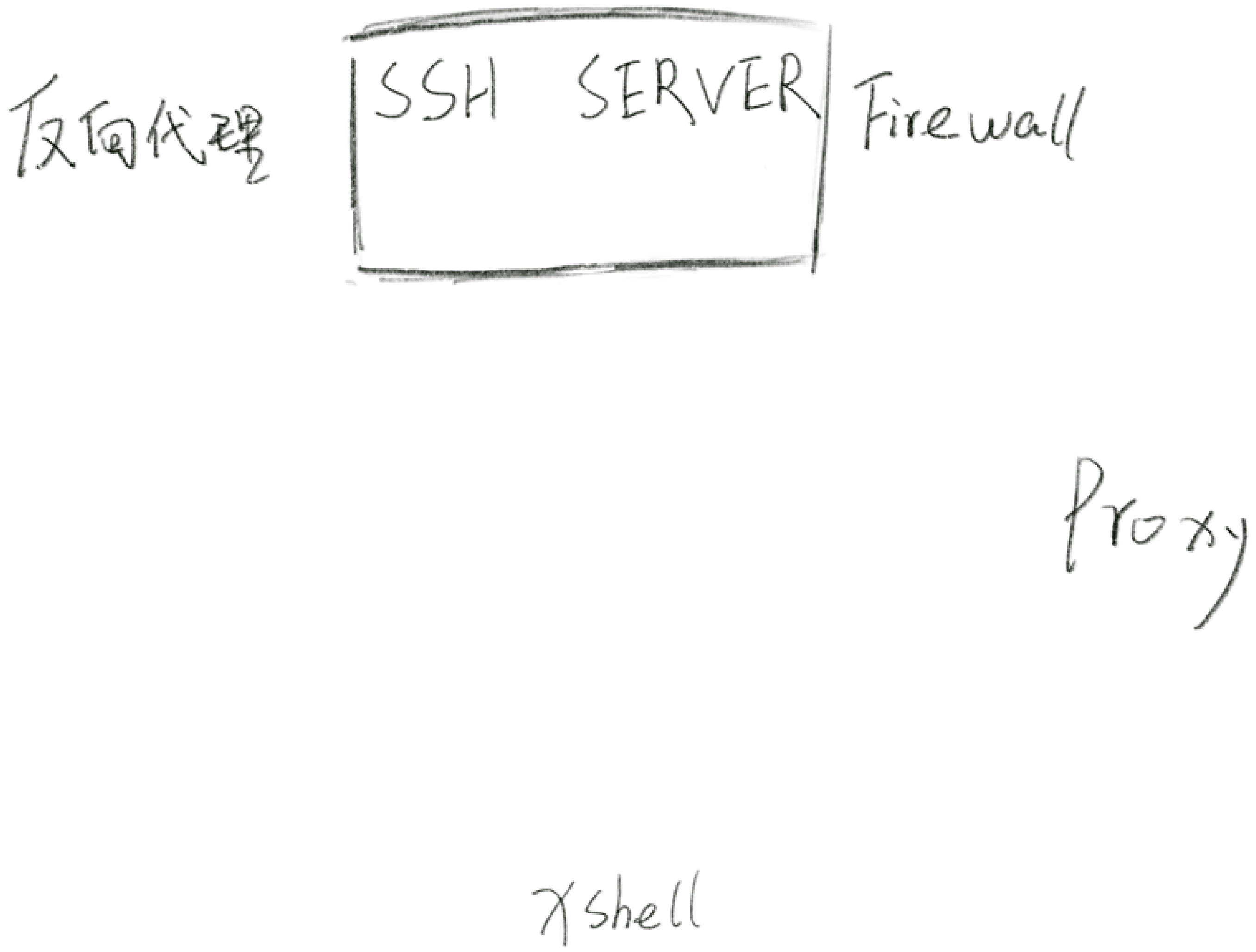


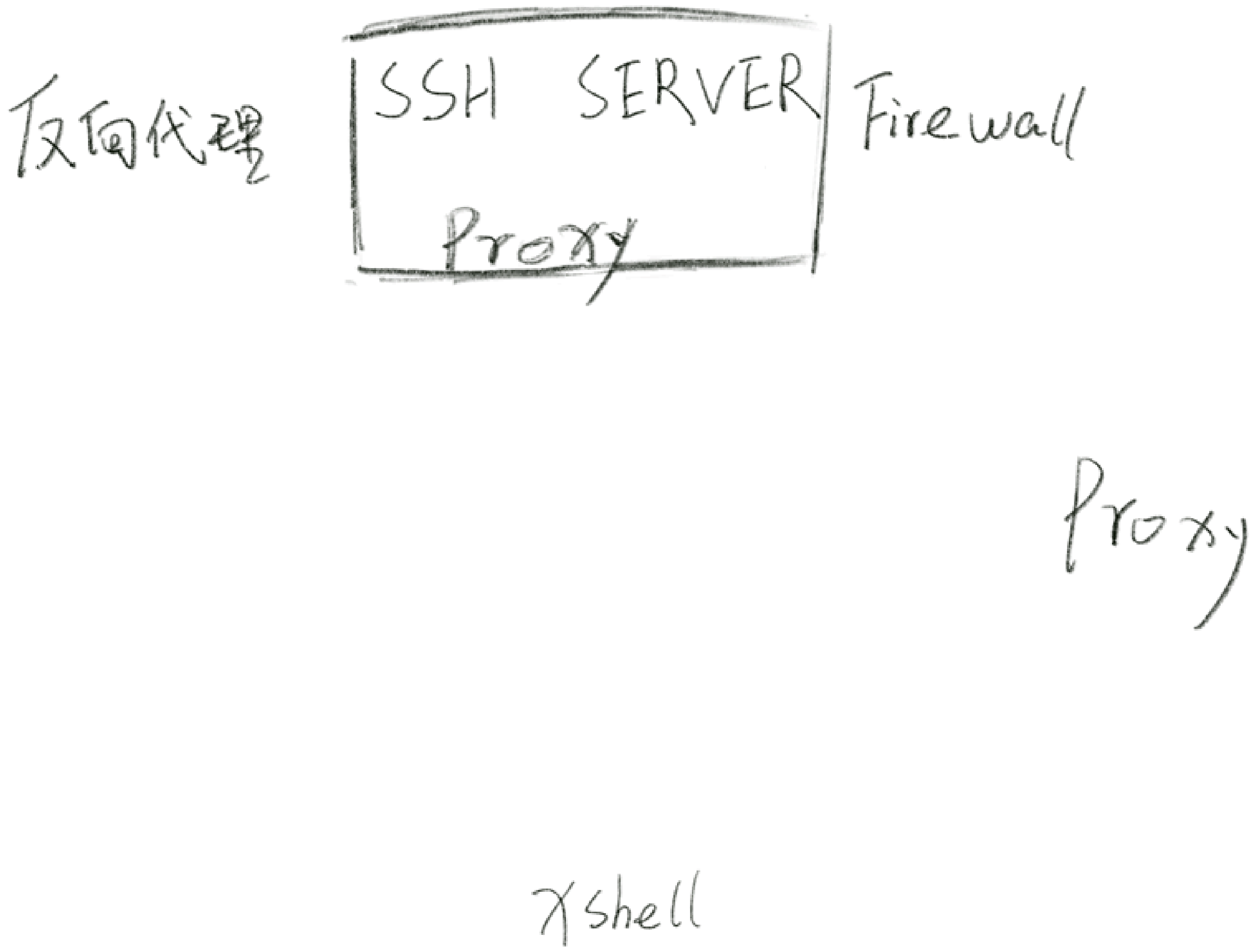
正向



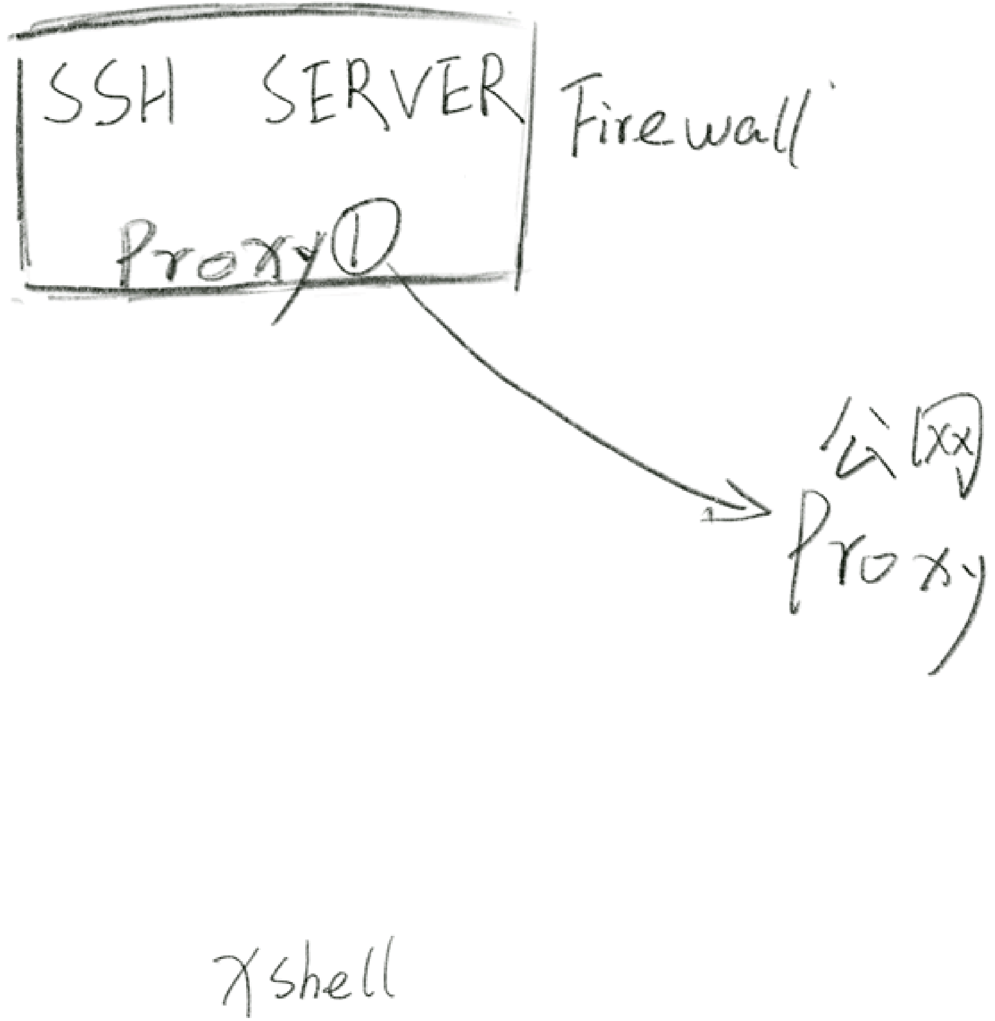

正向
SSH SERVER
Firewall
②
Proxy
①
Xshell

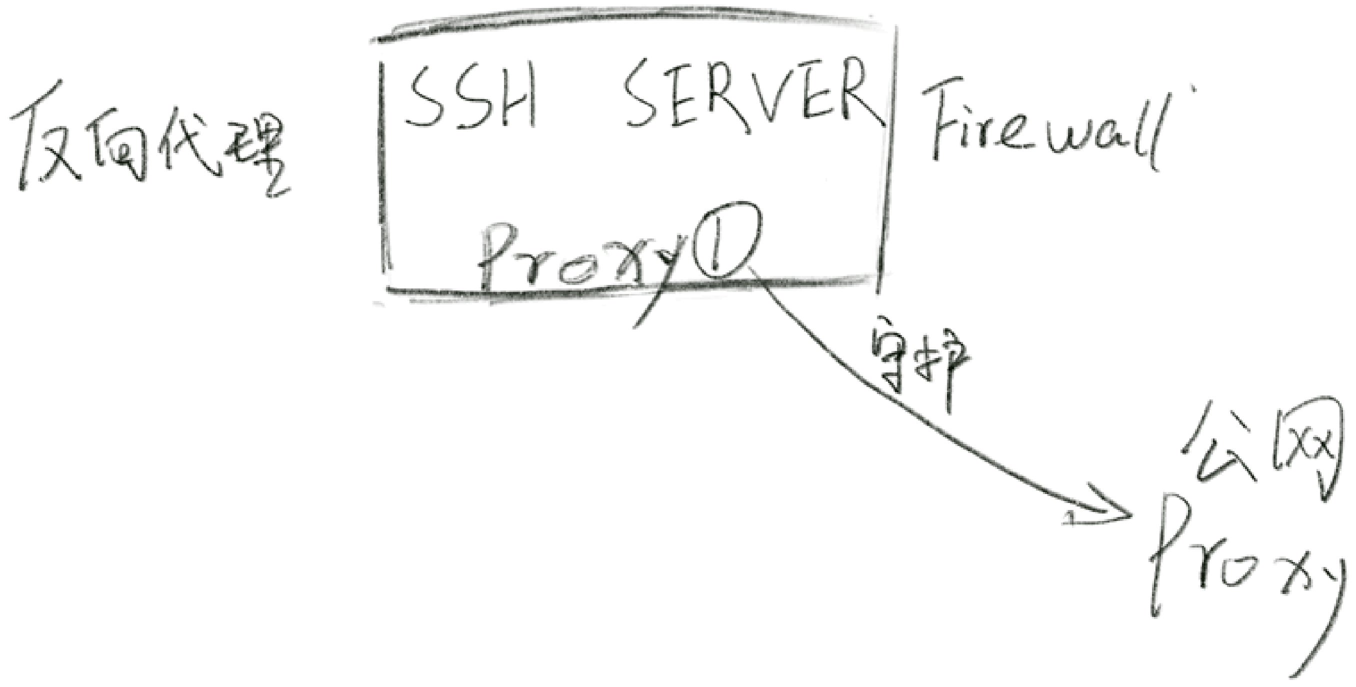

正向
SSH SERVER
Firewall
2
Proxy
1
Xshell


反向代理
SSH SERVER
Firewall
Proxy
Xshell

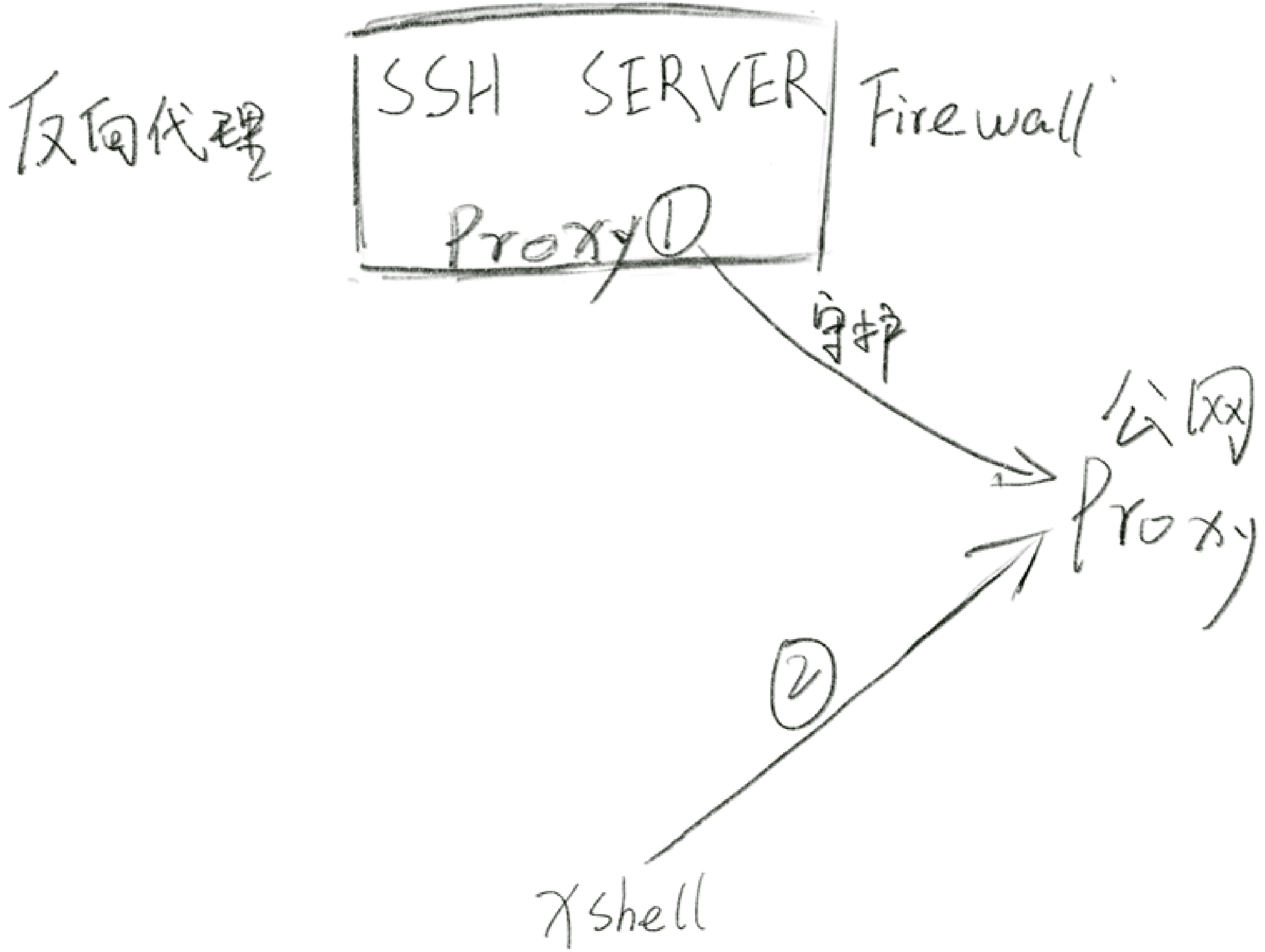

反向代理
SSH SERVER
Proxy
Firewall
Proxy
Xshell

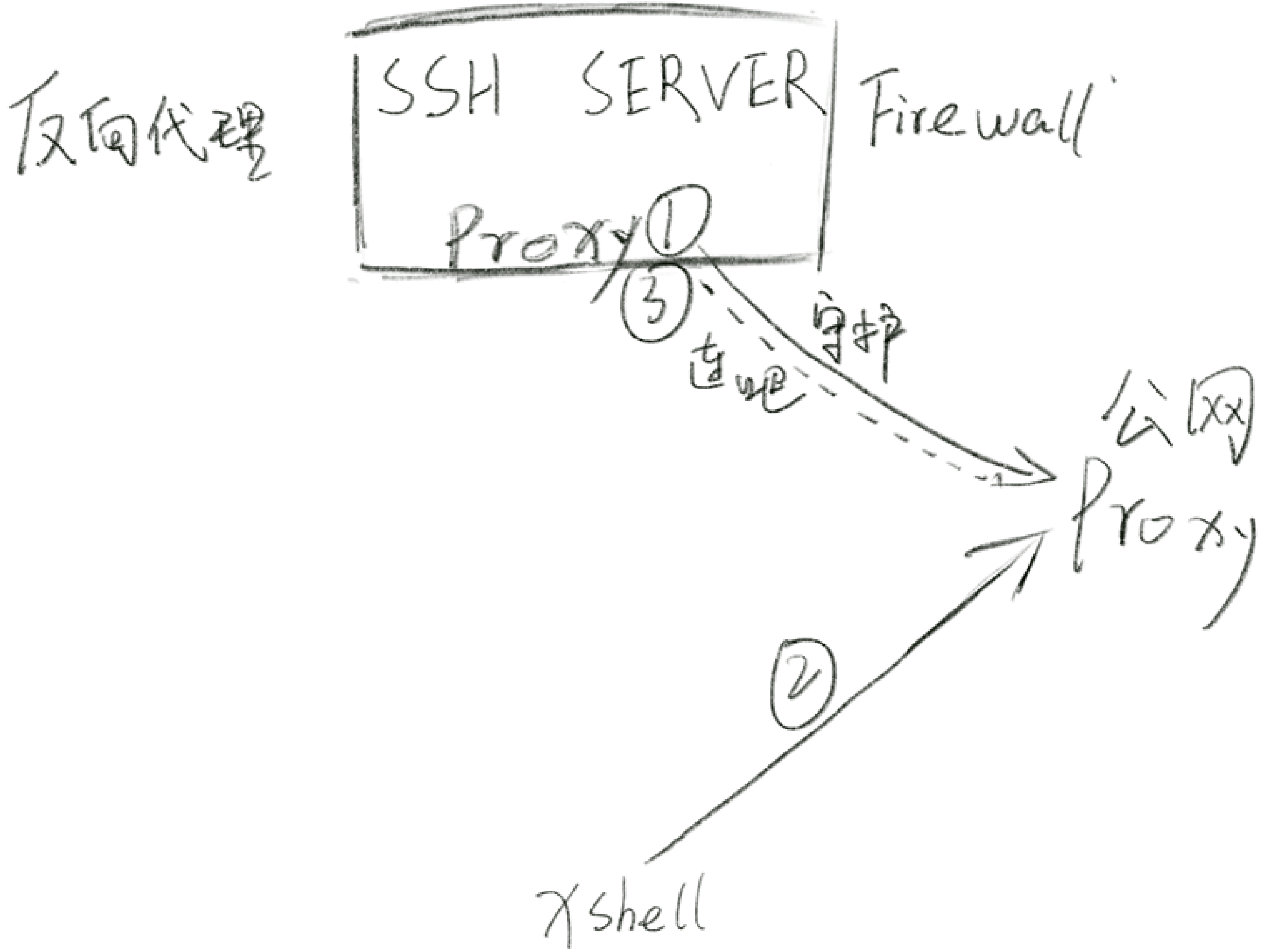
反向代理



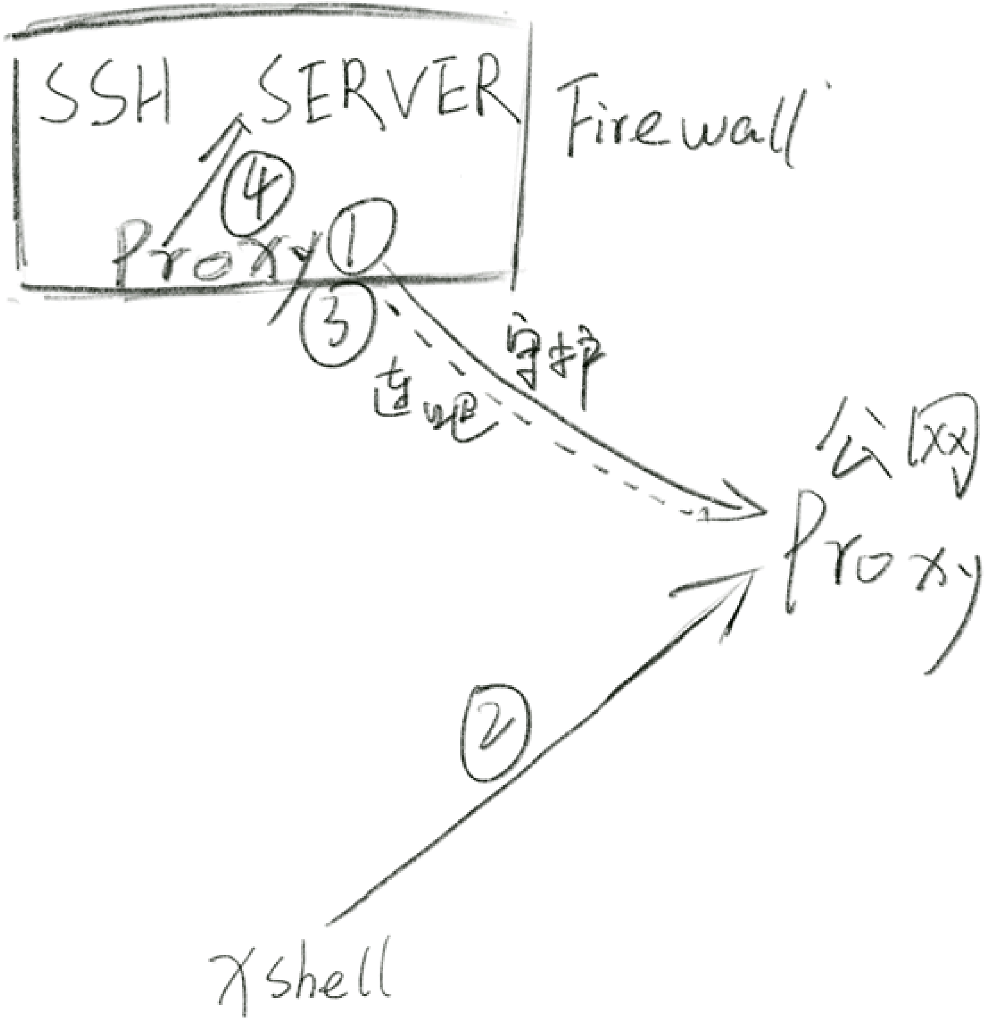


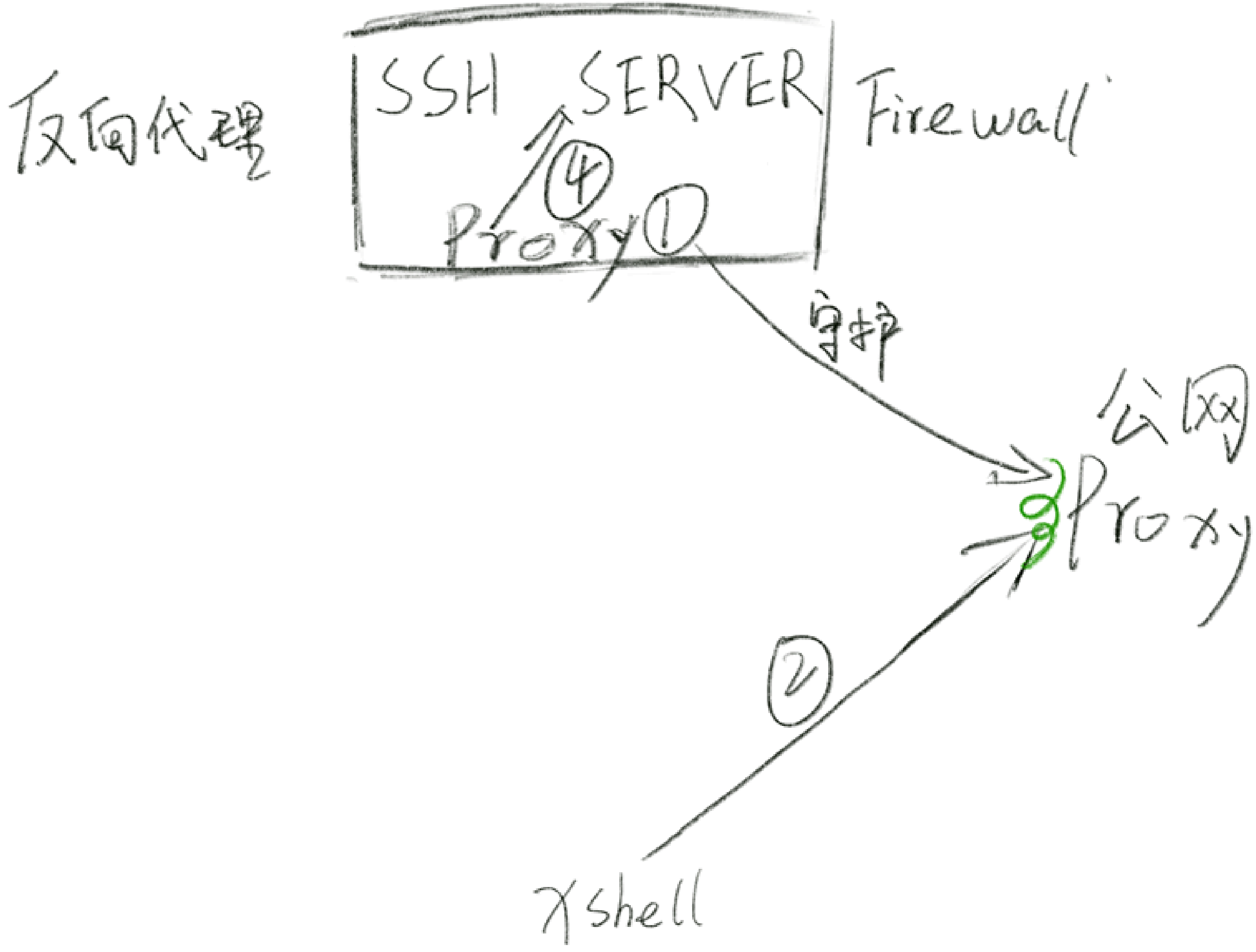
Xshell


反向代理
SSH SERVER
Proxy①
Firewall
守护
公网
Proxy
②
Xshell

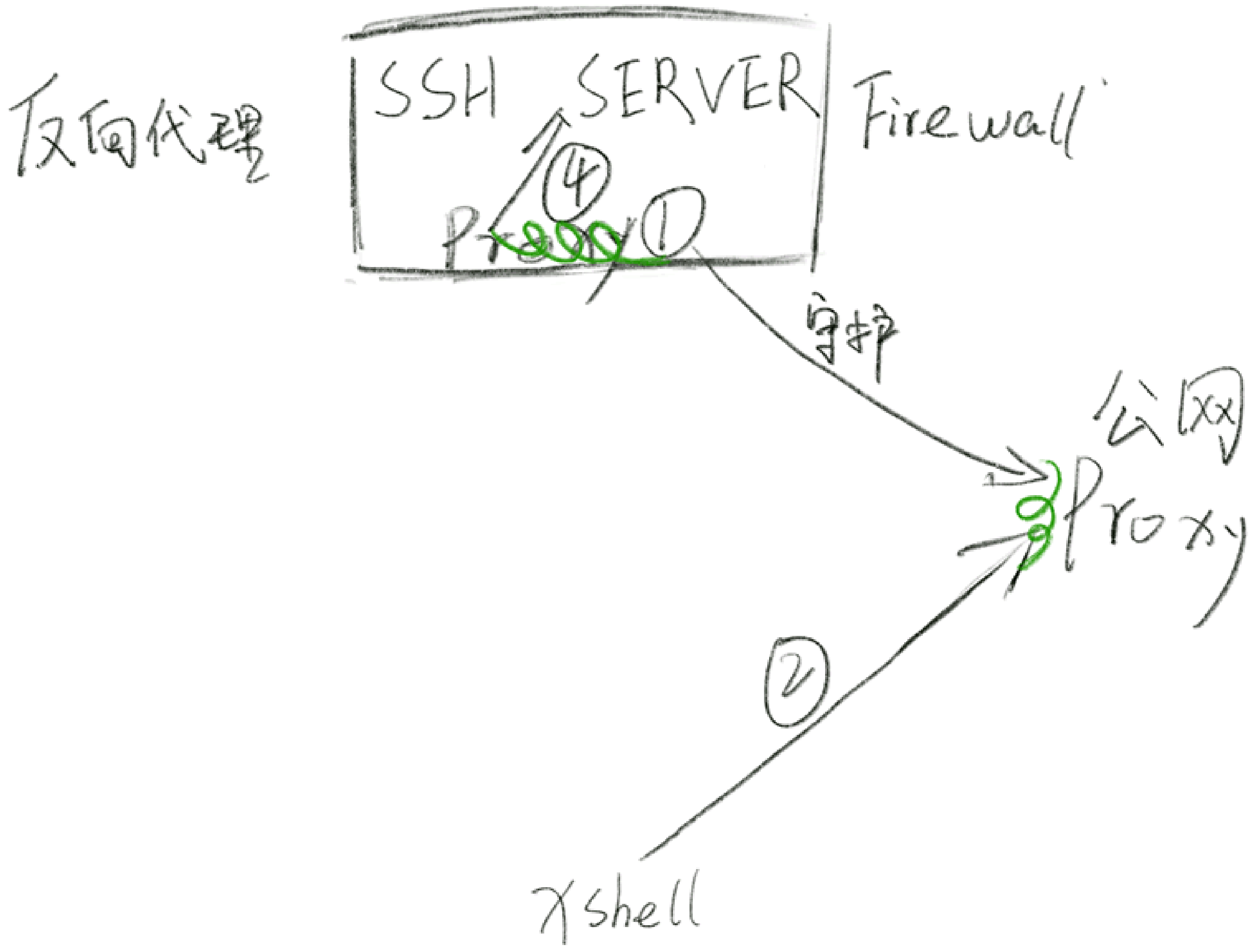

反向代理
SSH SERVER
Proxy ①
Firewall
③
连接
守护
公网
Proxy
②
Xshell

反向代理

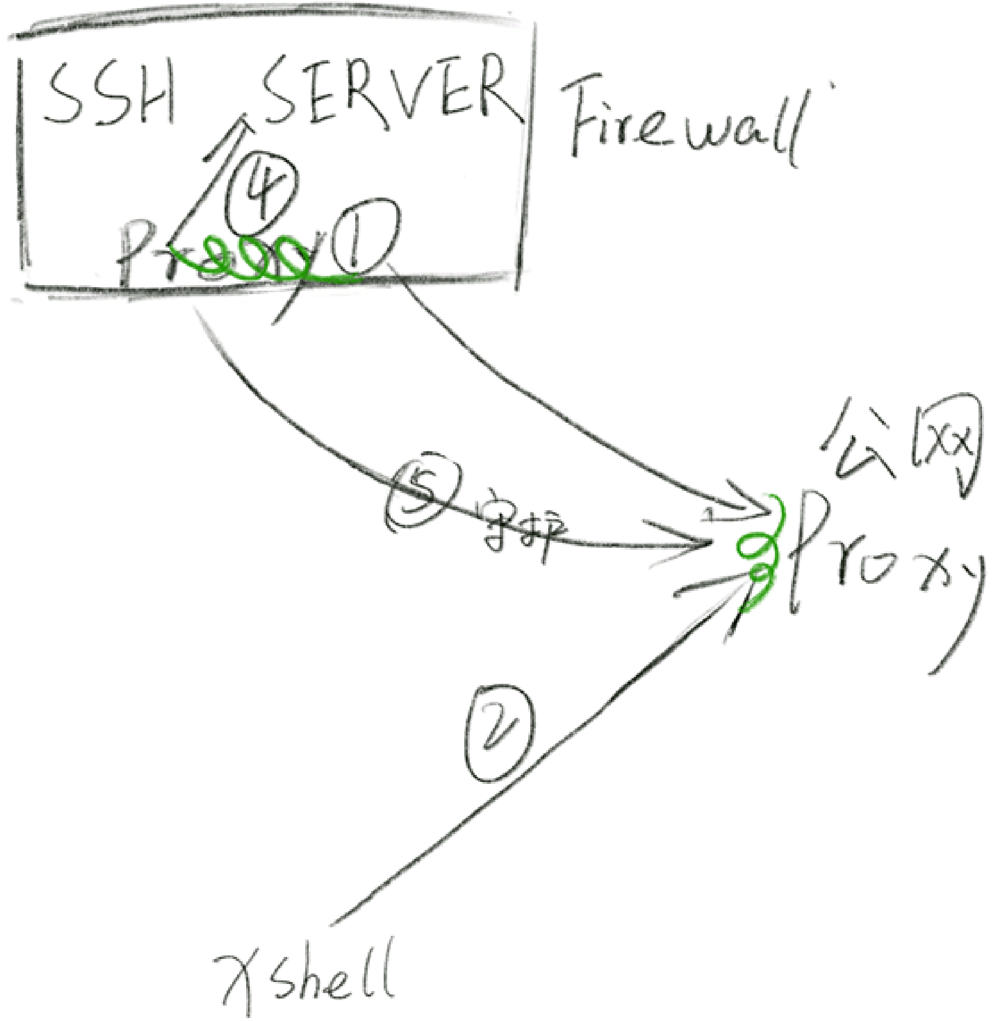


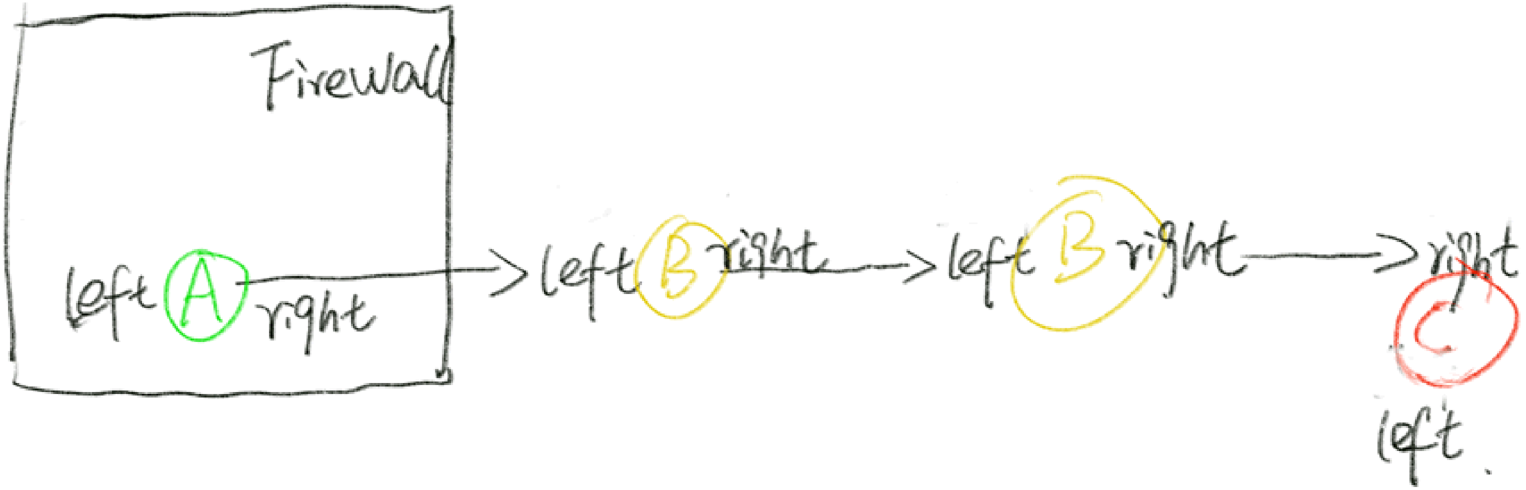


反向代理
SSH SERVER
Firewall
④
Proxy①
守护
公网
Proxy
②
Xshell


反向代理
SSH SERVER
Firewall
④
Proxy
①
守护
公网
Proxy
②
Xshell

反向代理





公网 B

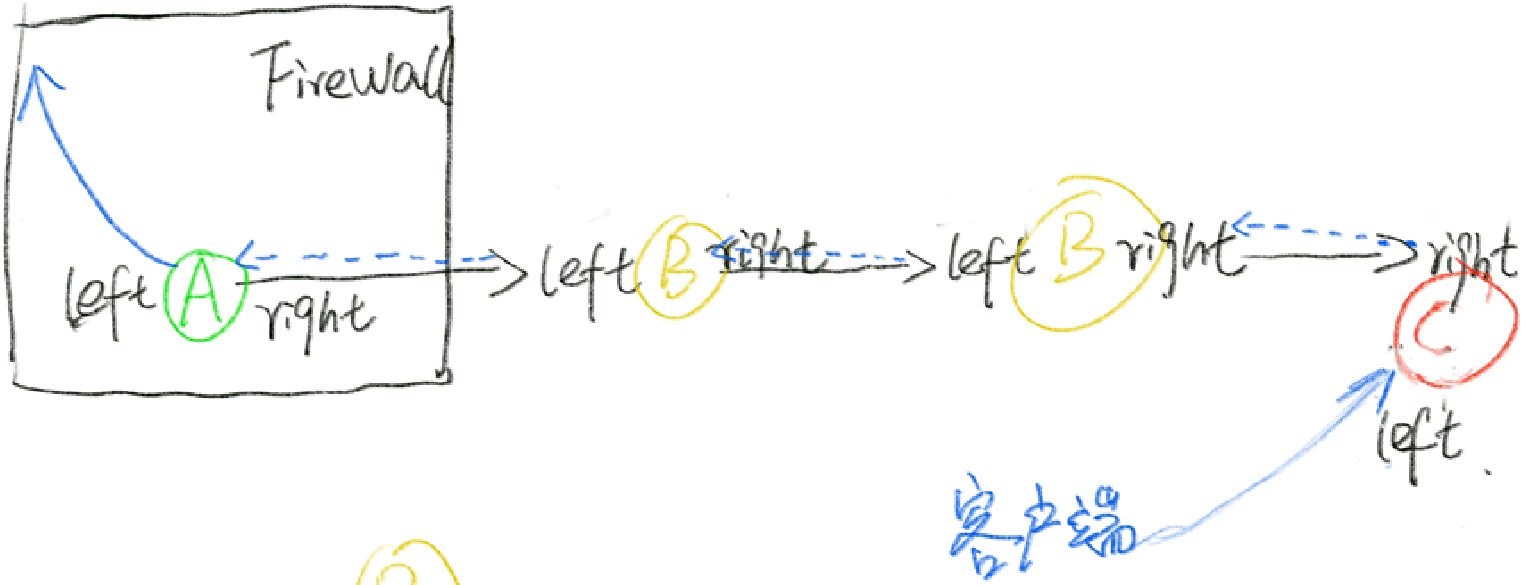
正向代理: Proxy -listen-left 2222 -connect-right IP:端口

内网 A

反向代理: Proxy -connect-left IP:端口 -connect-right IP:端口

公网 C

Proxy -listen-left 2222 -listen-right 3333



公网 B

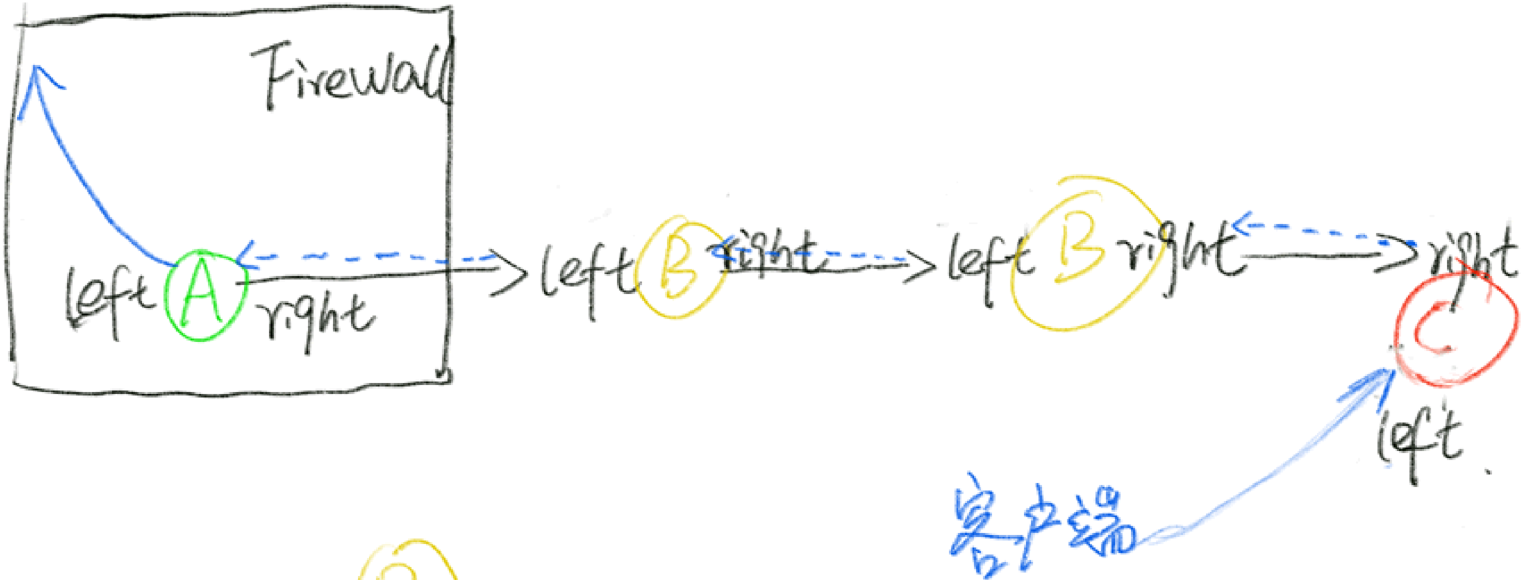
正向代理: Proxy -listen-left 2222 -connect-right IP:端口

内网 A

反向代理: Proxy -connect-left IP:端口 -connect-right IP:端口

公网 C

Proxy -listen-left 2222 -listen-right 3333



公网 B

正向代理: Proxy -listen-left 2222 -connect-right IP:端口

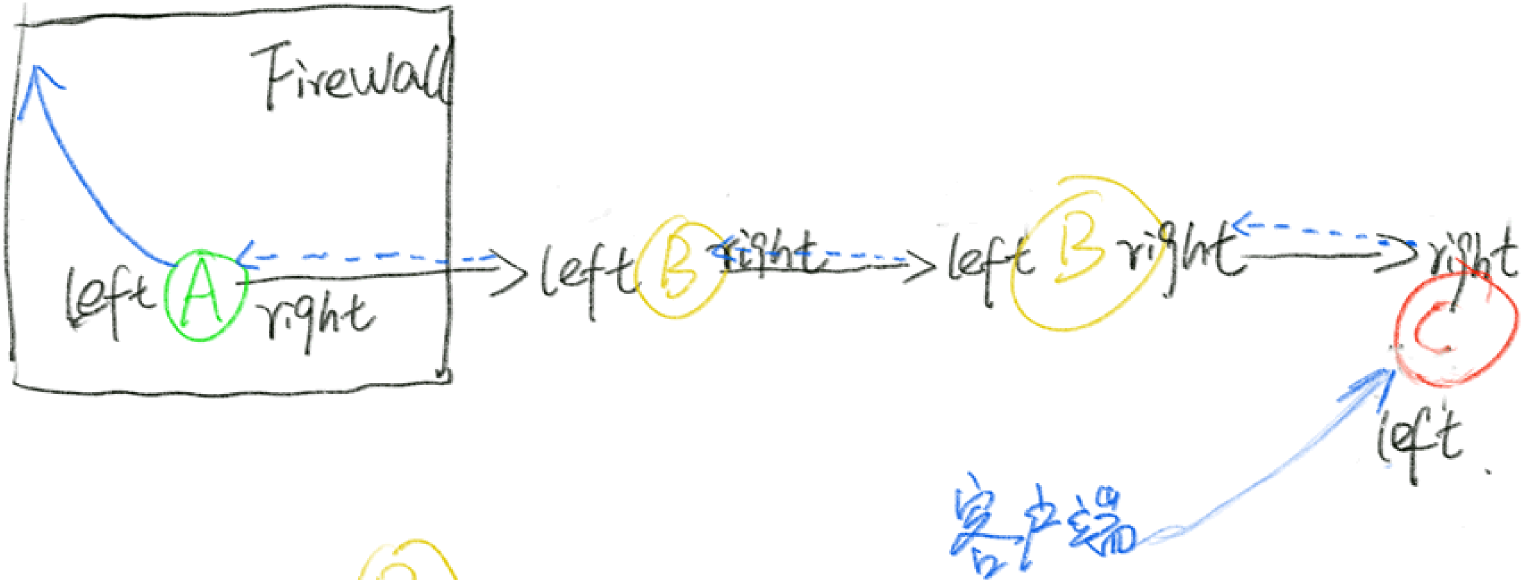
内网 A

反向代理: Proxy -connect-left IP:端口 -connect-right IP:端口

公网 C

Proxy -listen-left 2222 -listen-right 3333

配对



公网 B

正向代理: Proxy -listen-left 2222 -connect-right IP:端口

内网 A

反向代理: Proxy -connect-left IP:端口 -connect-right IP:端口

公网 C

Proxy -listen-left 2222 -listen-right 3333

[illegible]

sock5 协议

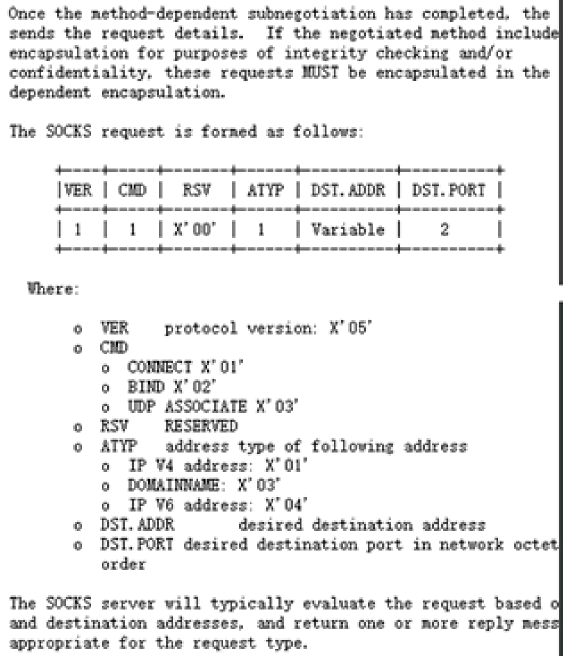
The SOCKS request is formed as follows:

VER	CMD	RSV	ATYP	DST.ADDR	DST.PORT
1	1	X’00’	1	Variable	2

Where:

- o VER   protocol version: X’05’
- o CMD
  - o CONNECT X’01’
  - o BIND X’02’
  - o UDP ASSOCIATE X’03’
- o RSV   RESERVED
- o ATYP   address type of following address
  - o IP V4 address: X’01’
  - o DOMAINNAME: X’03’
  - o IP V6 address: X’04’
- o DST.ADDR   desired destination address
- o DST.PORT desired destination port in network octet order

The SOCKS server will typically evaluate the request based on source and destination addresses, and return one or more reply messages, as appropriate for the request type.



Once the method-dependent subnegotiation has completed, the
sends the request details. If the negotiated method include
encapsulation for purposes of integrity checking and/or
confidentiality, these requests MUST be encapsulated in the
dependent encapsulation.

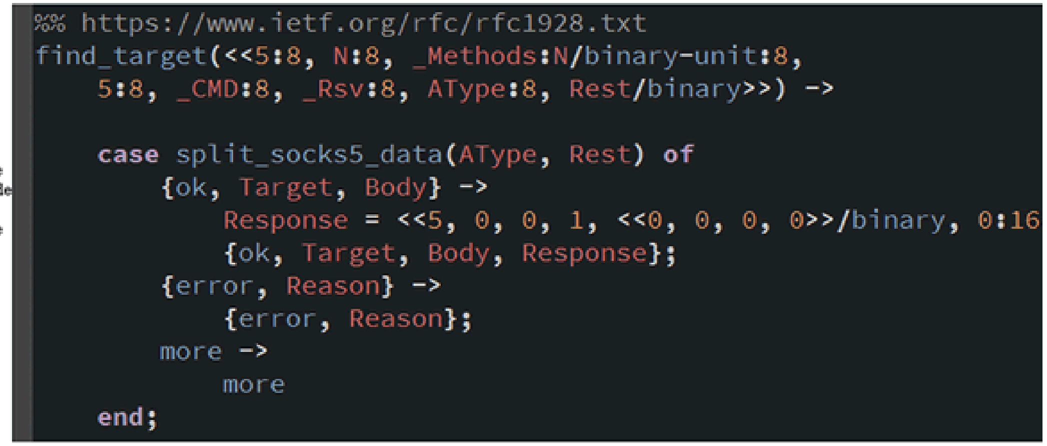
The SOCKS request is formed as follows:

VER	CMD	RSV	ATYP	DST.ADDR	DST.PORT
1	1	X'00'	1	Variable	2

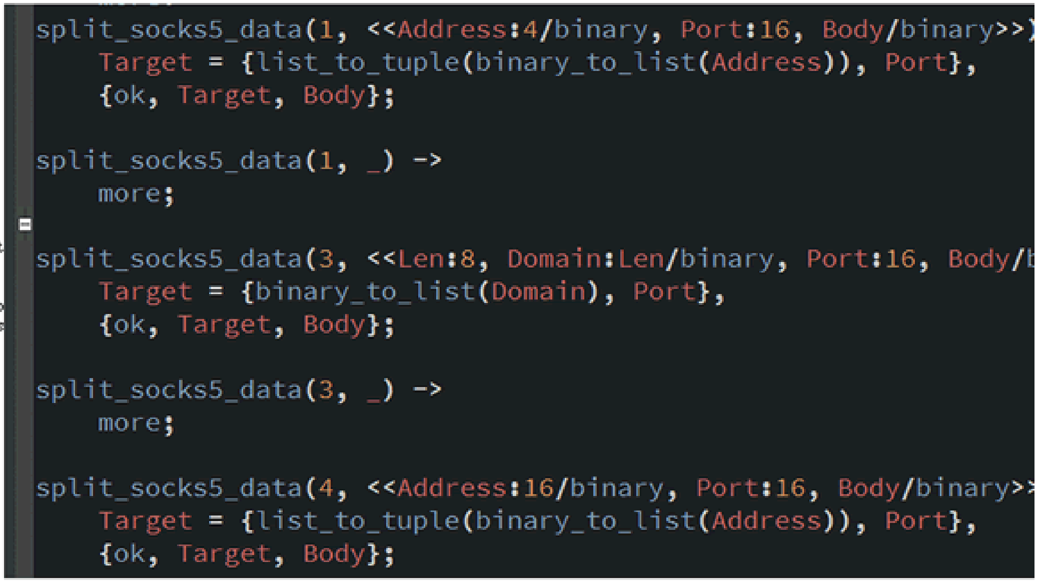
Where:

- o VER protocol version: X'05'
- o CMD
  - o CONNECT X'01'
  - o BIND X'02'
  - o UDP ASSOCIATE X'03'
- o RSV RESERVED
- o ATYP address type of following address
  - o IP V4 address: X'01'
  - o DOMAINNAME: X'03'
  - o IP V6 address: X'04'
- o DST.ADDR desired destination address
- o DST.PORT desired destination port in network octet
  order

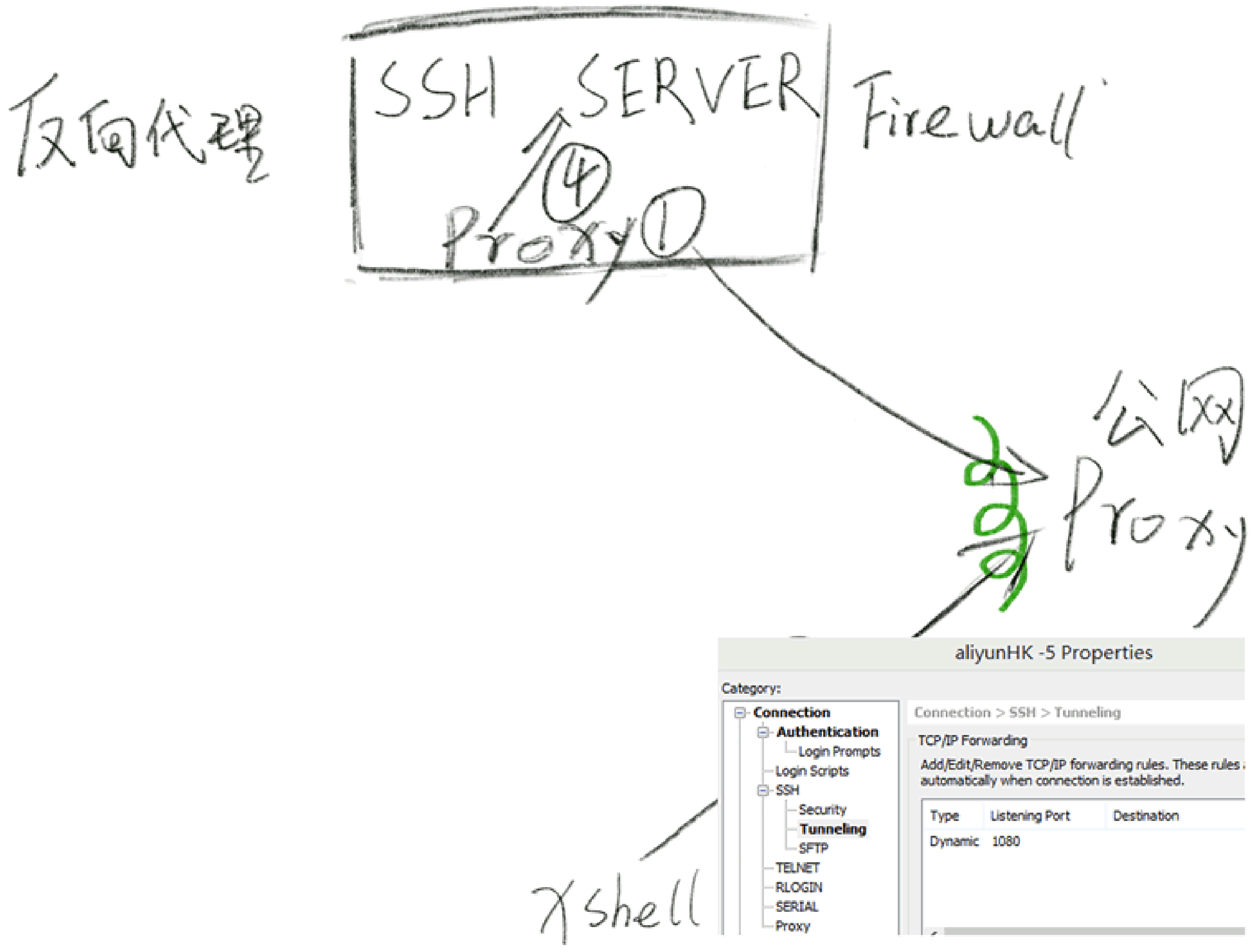
The SOCKS server will typically evaluate the request based o
and destination addresses, and return one or more reply mess
appropriate for the request type.

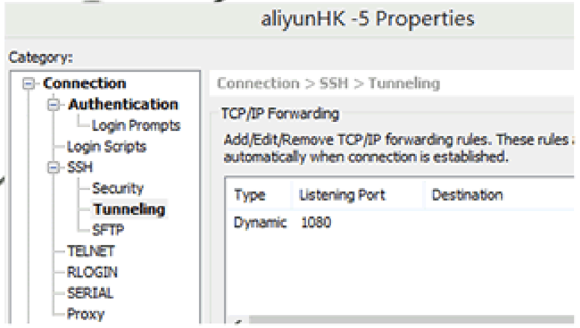


```
%% https://www.ietf.org/rfc/rfc1928.txt
find_target(<<5:8, N:8, _Methods:N/binary-unit:8,
    5:8, _CMD:8, _Rsv:8, AType:8, Rest/binary>>) ->

    case split_socks5_data(AType, Rest) of
        {ok, Target, Body} ->
            Response = <<5, 0, 0, 1, <<0, 0, 0, 0>>/binary, 0:16
            {ok, Target, Body, Response};
        {error, Reason} ->
            {error, Reason};
        more ->
            more
    end;
```



```
split_socks5_data(1, <<Address:4/binary, Port:16, Body/binary>>)
    Target = {list_to_tuple(binary_to_list(Address)), Port},
    {ok, Target, Body};

split_socks5_data(1, _) ->
    more;

split_socks5_data(3, <<Len:8, Domain:Len/binary, Port:16, Body/b
    Target = {binary_to_list(Domain), Port},
    {ok, Target, Body};

split_socks5_data(3, _) ->
    more;

split_socks5_data(4, <<Address:16/binary, Port:16, Body/binary>>
    Target = {list_to_tuple(binary_to_list(Address)), Port},
    {ok, Target, Body};
```




aliyunHK -5 Properties

Category:

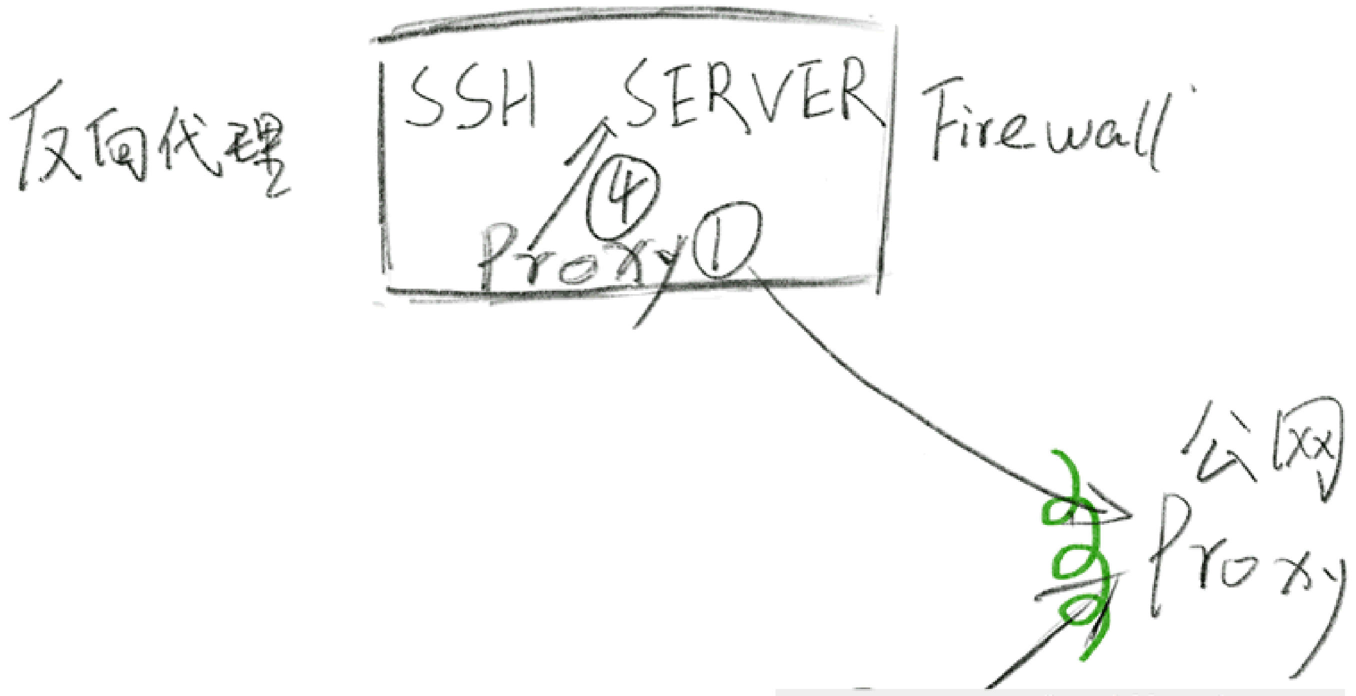
- Connection
  - Authentication
    - Login Prompts
  - Login Scripts
  - SSH
    - Security
    - Tunneling
    - SFTP
  - TELNET
  - RLOGIN
  - SERIAL
  - Proxy

Connection > SSH > Tunneling

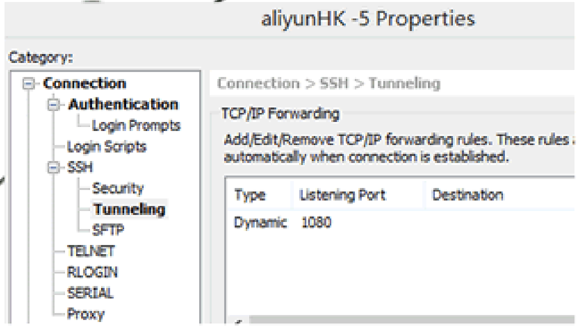
TCP/IP Forwarding

Add/Edit/Remove TCP/IP forwarding rules. These rules
automatically when connection is established.

Type	Listening Port	Destination
Dynamic	1080	



chrome.exe -proxy-server=socks5://127.0.0.1:1080


aliyunHK -5 Properties

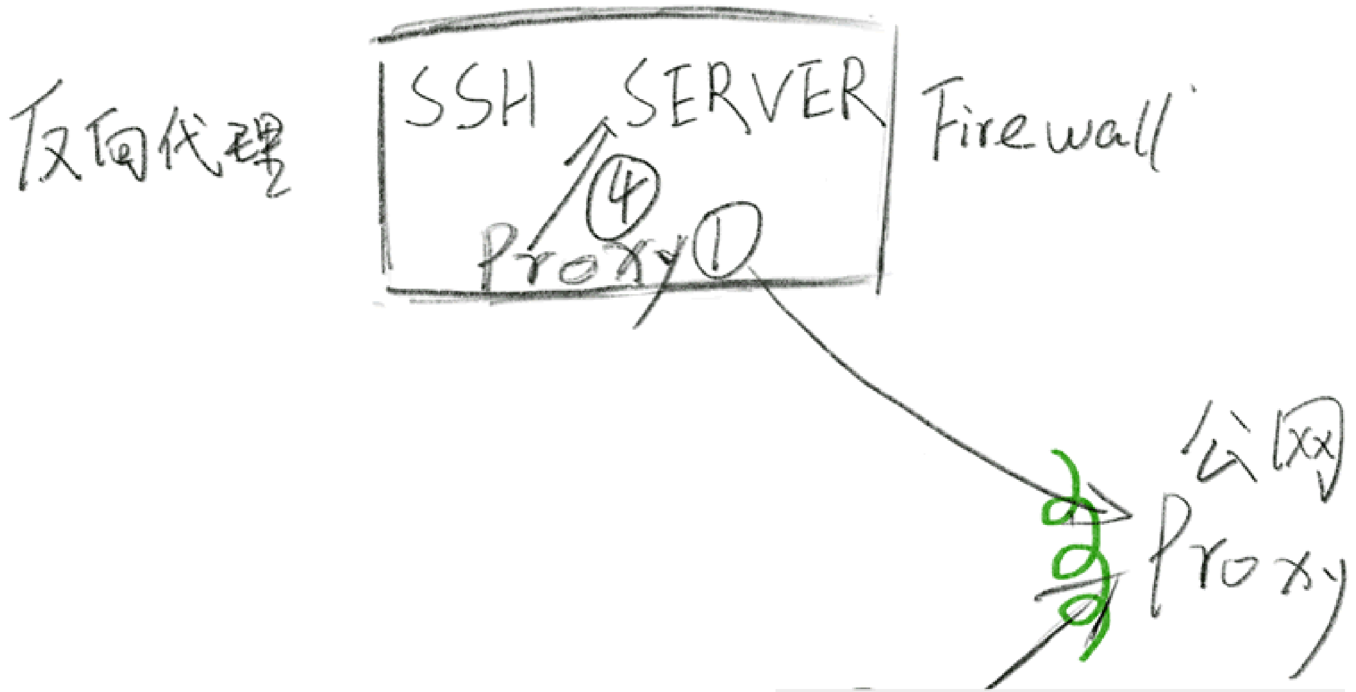
Category:

Connection > SSH > Tunneling

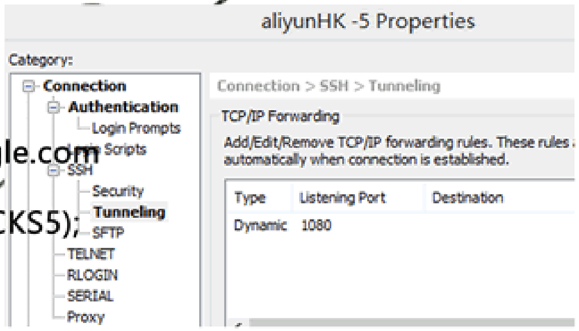
TCP/IP Forwarding

Add/Edit/Remove TCP/IP forwarding rules. These rules
automatically when connection is established.

Type	Listening Port	Destination
Dynamic	1080	



```
chrome.exe -proxy-server=socks5://127.0.0.1:1080

curl --socks5-hostname 127.0.0.1:1080 https://www.google.com

curl_setopt($ch, CURLOPT_PROXYTYPE, CURLPROXY_SOCKS5);
curl_setopt($ch, CURLOPT_PROXY, "127.0.0.1:1080");
```



aliyunHK -5 Properties

Category:

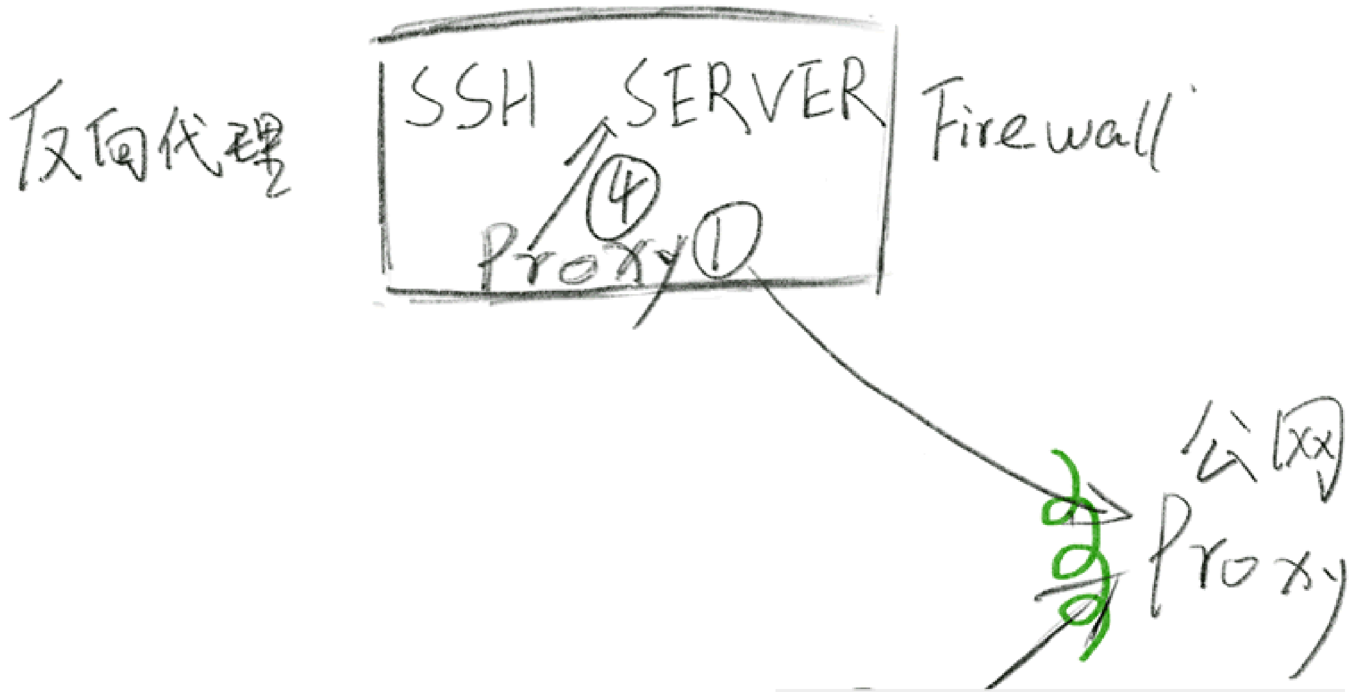
- Connection
  - Authentication
    - Login Prompts
  - Scripts
  - SSH
    - Security
    - **Tunneling**
    - SFTP
  - TELNET
  - RLOGIN
  - SERIAL
  - Proxy

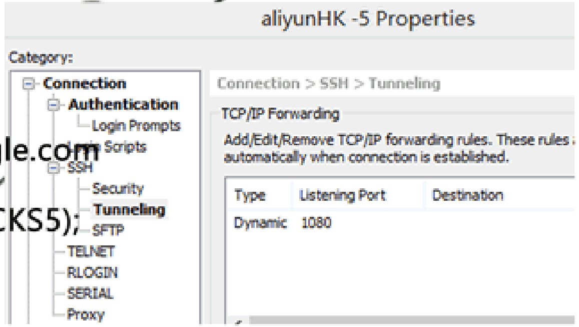
Connection > SSH > Tunneling

TCP/IP Forwarding

Add/Edit/Remove TCP/IP forwarding rules. These rules automatically when connection is established.

Type	Listening Port	Destination
Dynamic	1080	





aliyunHK -5 Properties

Category:

- Connection
  - Authentication
    - Login Prompts
  - Scripts
  - SSH
    - Security
    - **Tunneling**
    - SFTP
  - TELNET
  - RLOGIN
  - SERIAL
  - Proxy

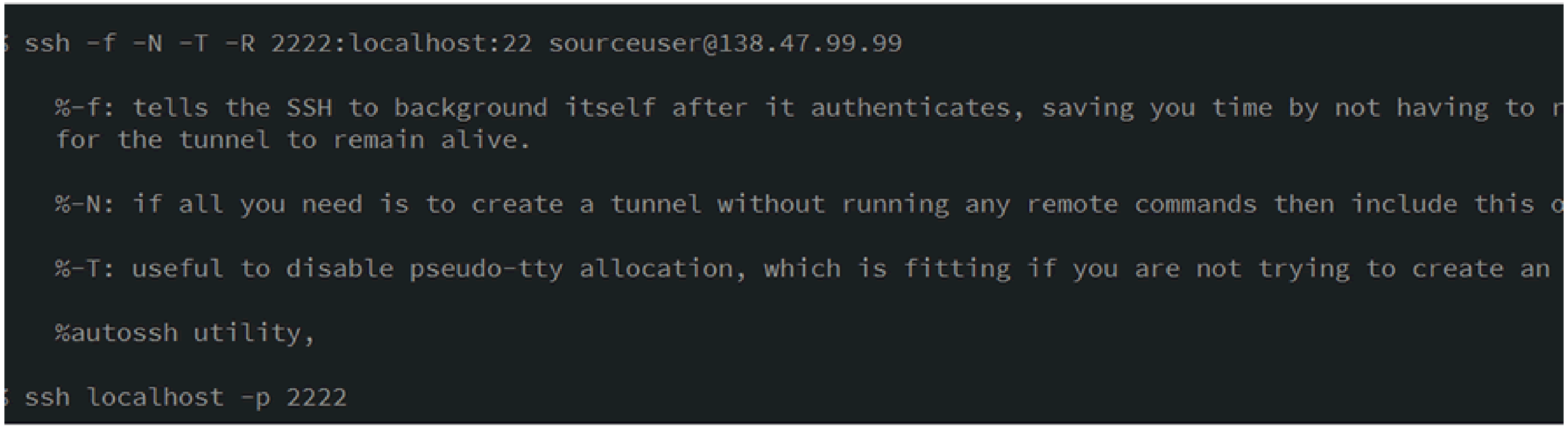
Connection > SSH > Tunneling

TCP/IP Forwarding

Add/Edit/Remove TCP/IP forwarding rules. These rules automatically when connection is established.

Type	Listening Port	Destination
Dynamic	1080	

```
chrome.exe -proxy-server=socks5://127.0.0.1:1080

curl --socks5-hostname 127.0.0.1:1080 https://www.google.com

curl_setopt($ch, CURLOPT_PROXYTYPE, CURLPROXY_SOCKS5);
curl_setopt($ch, CURLOPT_PROXY, “127.0.0.1:1080”);

git config --global http.proxy ‘socks5://127.0.0.1:1080’
git config --global https.proxy ‘socks5://127.0.0.1:1080’
```



```
 ssh -f -N -T -R 2222:localhost:22 sourceuser@138.47.99.99

  %-f: tells the SSH to background itself after it authenticates, saving you time by not having to r
  for the tunnel to remain alive.

  %-N: if all you need is to create a tunnel without running any remote commands then include this o

  %-T: useful to disable pseudo-tty allocation, which is fitting if you are not trying to create an

  %autossh utility,

 ssh localhost -p 2222
```