1. 一个主要的接口

|  |
| --- |
| type Interface interface {    AddMapping(protocol string, extport, intport int, name string, lifetime time.Duration) error  DeleteMapping(protocol string, extport, intport int) error  ExternalIP() (net.IP, error)  String() string  } |

这个接口包含了实现NAT穿透相关的所有功能

1. 三个实现上述Interface接口的结构体

|  |
| --- |
| type autodisc struct {  what string // 需要被检测的NAT映射协议类型  once sync.Once //sync.Once 也是 Go 官方的一并发辅助对象，它能够让函数方法只执行一次，达到类似 init 函数的效果  doit func() Interface //在等待期间需要调用的用于进行协议检测的函数  mu sync.Mutex  found Interface //存放搜索获得的端口映射器(采用UPnP协议或者NAT-PMP协议)  } |

|  |
| --- |
| type upnp struct {  dev \*goupnp.RootDevice //支持UPnP的根设备 ？？  service string  client upnpClient //请求UPnP的客户端，也就是内网下的主机节点的客户端程序  } |

|  |
| --- |
| type pmp struct {  gw net.IP //网管的IP地址  c \*natpmp.Client //客户端  } |

还有一种extIP类型变量也实现了该接口

|  |
| --- |
| type extIP net.IP |

1. 函数

|  |
| --- |
| func Parse(spec string) (Interface, error)  对输入的字符串指令进行解析   1. 指令为："", "none", "off" 不进行任何操作 2. 指令为："any", "auto", "on" 调用Any()函数，本函数负责在NAT设备上搜索任意的支持实现NAT映射的协议(UPnP或PMP),返回第一个查询到的端口映射器 3. 指令为："extip", "ip" 调用ExtIP()函数对传入的IP地址进行有效性判断，并转换为extIP类型(此指令适用的情况是本节点的IP地址就是公网IP，不需要其他协议进行映射) 4. 指令为”upnp” 调用UPnP()函数，在局域网中广播查询支持UPnP协议的NAT网管并返回相应的端口映射器 5. 指令为"pmp", "natpmp", "nat-pmp"，调用PMP()函数，查询支持NAT-PMP协议的NAT网管并返回相应的端口映射器(根据是否传入NAT网管IP地址决定是定点查询还是广播查询) |

|  |
| --- |
| func ExtIP(ip net.IP) Interface  对传入的IP地址检查(是否为空)，并将此IP地址转化为extIP类型(实现了Interface接口) |

|  |
| --- |
| func Any() Interface  本函数负责在NAT设备上搜索任意的支持实现NAT映射的协议(UPnP或PMP),返回第一个查询到的端口映射器  底层调用startautodisc()函数进行查询(此函数分别使用discoverUPnP()和discoverPMP()在局域网中搜索支持UPnP和PMP协议的路由器) |

|  |
| --- |
| func UPnP() Interface  查询NAT设备是否支持UPnP协议，若支持返回一个使用UPnP协议的端口映射器。查询过程将以UDP广播的方式发送到整个局域网的设备上  底层调用startautodisc()函数进行查询(仅使用discoverUPnP()查询支持UPnP的路由器) |

|  |
| --- |
| func PMP(gateway net.IP) Interface  查询NAT设备是否支持PMP协议，若支持返回一个使用PMP协议的端口映射器。参数中提供的网关地址应该是局域网中的路由器的IP地址，如果给定的网关地址为零，PMP将尝试自动发现路由器。  若指定了路由器IP地址，则直接与该路由设备连接，在本地建立PMP客户端。  否则需要通过广播查询路由器，底层调用startautodisc()函数进行查询(仅使用discoverPMP()查询支持UPnP的路由器) |

|  |
| --- |
| func startautodisc(what string, doit func() Interface) Interface  开启单独协程负责执行指定的NAT映射协议搜索函数  参数1指定要查询的NAT映射协议，参数2指定要执行的协议搜索函数  返回值为autodisc类型结构体，包含了搜索获得的端口映射器(采用UPnP协议或者NAT-PMP协议)，实现了Interface接口 |

|  |
| --- |
| func Map(m Interface, c chan struct{}, protocol string, extport, intport int, name string) 核心  利用搜索获取的端口映射器(实现了Interface接口的结构体)进行实际的NAT端口映射。  m即为端口映射器,protocol指定对TCP还是UDP端口进行映射， extport和intport分别为NAT映射后的内外端口  Map函数为m添加一个端口映射，并保持此映射关系直到管道C被关闭 。通常由一个协程单独运行此函数 |

|  |
| --- |
| func discoverUPnP() Interface  本函数负责在局域网中搜寻支持UPNP的网管设备，并返回它在本地网络上可以找到的第一个设备(搜索所有支持UPnP IGD V1/V2协议的路由设备)。  //返回值是upnp结构体类型的c，该结构体实现了nat.go中的Interface接口  底层调用了discover()函数进行实现 |

|  |
| --- |
| func discover(out chan<- \*upnp, target string, matcher func(\*goupnp.RootDevice, goupnp.ServiceClient) \*upnp)  暂未分析 |

|  |
| --- |
| func discoverPMP() Interface  在局域网通过广播的方式查询支持NAT-PMP协议的网管路由器  通过调用potentialGateways()搜索获取本网段下可能是路由器的IP地址。接着调用natpmp.NewClient()向这些可能的IP地址发送探测请求，成功的话返回pmp结构体 |

|  |
| --- |
| func potentialGateways() (gws []net.IP)  获取所有可能的局域网中NAT网管的IP地址(在这里总是假设路由器的IP地址是每个网段下的1号主机的IP地址🡪 X.X.X.1) |

1. extIP类型变量实现的Interface接口的各方法

|  |
| --- |
| func (n extIP) ExternalIP() (net.IP, error) { return net.IP(n), nil }  返回net.IP类型的IP地址 |

|  |
| --- |
| func (n extIP) String() string { return fmt.Sprintf("ExtIP(%v)", net.IP(n)) }  返回字符串类型的IP地址 |

|  |
| --- |
| func (extIP) AddMapping(string, int, int, string, time.Duration) error { return nil }  func (extIP) DeleteMapping(string, int, int) error { return nil } |

1. upnp结构体实现的Interface接口的各方法

这些方法会使用到一个新的接口下的函数(本接口由package internetgateway1包实现)

|  |
| --- |
| type upnpClient interface {  GetExternalIPAddress()(string,error) //获取NAT设备对外的公网IP地址(多层NAT就不是公网IP地址)  AddPortMapping(string, uint16, string, uint16, string, bool, string, uint32) error //添加端口映射  DeletePortMapping(string,uint16,string)error //删除端口映射  GetNATRSIPStatus() (sip bool, nat bool, err error) //检查SIP协议的状态？  } |

|  |
| --- |
| func (n \*upnp) ExternalIP() (addr net.IP, err error)  获取运行在内网主机上的客户端程序在NAT网管上的对外的映射IP地址  底层调用n.client.GetExternalIPAddress()实现 |

|  |
| --- |
| func (n \*upnp) AddMapping(protocol string, extport, intport int, desc string, lifetime time.Duration) error  在NAT网管上实现upnp协议的NAT地址映射   1. 首先调用n.internalAddress()方法获取本主机的内网IP地址 2. 调用n.client.AddPortMapping()方法实现NAT地址映射 |

|  |
| --- |
| func (n \*upnp) internalAddress() (net.IP, error)  获取内网主机的内网IP地址   1. net.ResolveUDPAddr("udp4", n.dev.URLBase.Host)的参数二是什么意思？获取谁的UDP地址？ 2. net.Interfaces()获取本主机所有的网络接口(一台计算机一般有若干网络接口，IP的，蓝牙的。。。) 3. iface.Addrs() //获取上述获得的所有网络接口的地址 4. 通过类型断言，获取所有的IP地址(蓝牙等地址不要) 5. 查看这些IP地址中是否存在步骤①中获取的UDP地址的IP，若存在则返回该IP |

|  |
| --- |
| func (n \*upnp) DeleteMapping(protocol string, extport, intport int) error  删除NAT网管上的保留的指定映射记录 |

|  |
| --- |
| func (n \*upnp) String() string {  return "UPNP " + n.service  } |

1. pmp结构体实现的Interface接口的各方法

|  |
| --- |
| func (n \*pmp) ExternalIP() (net.IP, error)  获取客户端在NAT网管上的映射地址  底层调用n.c.GetExternalAddress()方法实现 |

|  |
| --- |
| func (n \*pmp) AddMapping(protocol string, extport, intport int, name string, lifetime time.Duration) error  采用PMP协议实现在NAT设备上的端口映射(指定传输层协议、端口的映射关系，并设置生命周期)  底层调用n.c.AddPortMapping()方法进行实现 |

|  |
| --- |
| func (n \*pmp) DeleteMapping(protocol string, extport, intport int) (err error)  删除NAT设备上指定的端口映射关系  底层调用n.c.AddPortMapping()发放进行实现，删除的方法比较特殊，就是让映射记录的生命周期变为0 |

|  |
| --- |
| func (n \*pmp) String() string {  return fmt.Sprintf("NAT-PMP(%v)", n.gw)  } |

1. autodisc结构体实现的Interface接口的各方法

|  |
| --- |
| func (n \*autodisc) AddMapping(protocol string, extport, intport int, name string, lifetime time.Duration) error  根据搜索获取的端口映射器(upnp结构体或pmp结构体)实现NAT设备上的端口映射  底层调用n.found.AddMapping()方法，found就是获取的端口映射器(upnp结构体或pmp结构体) |

|  |
| --- |
| func (n \*autodisc) DeleteMapping(protocol string, extport, intport int) error  取消指定的端口映射  底层调用n.found.DeleteMapping()方法 |

|  |
| --- |
| func (n \*autodisc) ExternalIP() (net.IP, error)  获取本主机在NAT设备上的对外映射IP地址  底层调用n.found.ExternalIP() |

|  |
| --- |
| func (n \*autodisc) String() string  以字符串形式返回使用的NAT映射协议(UPnP协议和PMP协议会返回不同的具有各自特征的字符串) |