Trabalho 3 - Laboratório de Redes de Computadores

Derick P. Garcez (13201878)
Vinícius A. dos Santos (13201941)

Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brasil

derick.garcez@edu.pucrs.br;vinicius.azevedo@edu.pucrs.br

1. Introdução

O terceiro trabalho da disciplina de Laboratório de Redes de Computadores da Graduação em Ciência da Computação consiste em implementar um programa em *Python*, usando *Socket Raw*, que implemente a funcionalidade de NAPT (*Network Address Port Translation*). O programa deverá ser validado utilizando a ferramenta de emulação Core, e testado para os tráfegos ICMP, TCP e UDP. Todo o código fonte, imagens, e conteúdo deste trabalho está disponível na plataforma GitHub através do link: *https://github.com/derickpg/T3_LabRedes* .

2. Modelo

Para este trabalho, e para a implementação realizada vamos utilizar um modelo de comunicação entre cliente e servidor. O cliente está conectado no que vamos chamar de rede interna, e o servidor está conectado no que vamos chamar de rede externa, a separação dessas redes é feita por um terceiro host que atuara como uma espécie de Roteador, ou servidor NAPT. Podemos ver na Imagem 1 como está distribuída a rede modelo.

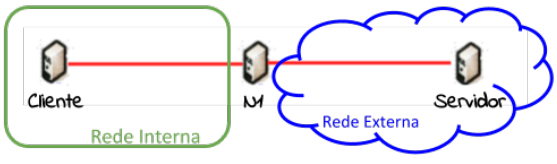

Figura 1. Modelo de Rede Utilizada para este Trabalho.

Com a funcionalidade NAPT o Cliente tem um IP e uma Porta na rede interna para a comunicação, mas o ”roteador”altera esse IP e essa Porta na saída com a comunicação externa. Assim é necessário o roteador armazenar em uma tabela as informações da comunicação para que no futuro uma comunicação externa que venha, chegue até seu destino final. A Figura 2 Exemplifica a alteração de IP e Pota em uma comunicação entre Cliente para Servidor.

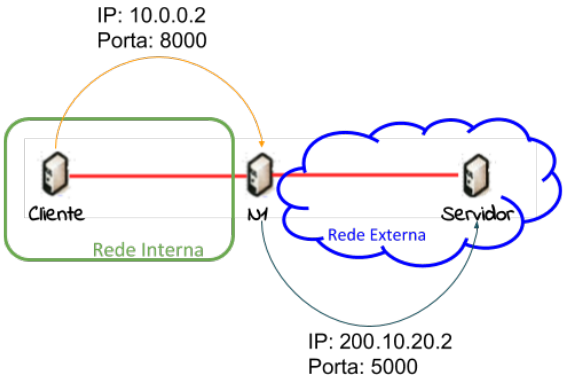

Figura 2. Modelo de comunicação entre Rede Interna e Rede Externa.

3. Implementação

Toda a implementação utilizou como base a implementação do programa *nat.py* disponibilizado pelo professor, então algumas partes do código, como a criação dos *Sockets* ou como leitura de informações dos pacotes é a mesma utilizada pelo professor.

A primeira etapa é a criação dos *Sockets* para a captura dos pacotes. Foram criados dois *Sockets* que realizam a captura dos pacotes das duas interfaces que o Servidor NAPT tem no nosso modelo, o *Socket* S0 realiza a captura da rede interna e o *Socket* S1 captura da rede externa. Após

a criação dos *Sockets* é iniciada a etapa de captura dos pacotes.

A etapa de captura dos pacotes, primeiro verifica em qual *Socket* foi capturado o pacote para identificação de qual a direção o pacote está transitando. O pacote é aberto utilizando o código abaixo e assim é vista as informações contidas dentro dele:

```
struct.unpack("!6s6sH",
   eth_header)
```

Após isso é realizada uma verificação se o pacote capturado é um pacote IP ou não. E por útlimo são realizadas três verificações, a ponto de saber se o pacote capturado é um pacote ICMP, TCP ou UDP. Nos três casos o procedimento realizado é praticamente o mesmo, apenas alterando a forma de captura das informação dentro do pacote. No caso do ICMP não é realizada a captura da porta, pois o mesmo protocolo não utiliza a porta. Utilizamos o código abaixo para abrir o Cabeçalho do IP:

```
struct.unpack("!BBHHHBBH4s4s",
   ip_header)
```

Com a utilização do *Socket Raw* é necessário a identificação dos tipos de pacotes que estamos trabalhando, dessa forma a primeira identificação do pacote realizada é se o mesmo é um pacote IP, caso seja começamos a realizar a identificação do pacote, para a criação de um novo. Como precisamos no ”Roteador”ou no nosso servidor NAPT, armazenar as informações entre a comunicação entre a rede interna e a rede externa, foi criada uma matriz onde são guardadas as seguintes informações: Protocolo, IP Interno, Porta Interna, IP Externo, Porta Externa, IP Destino e Porta Destino. A Figura 3 apresenta um exemplo de preenchimento desta Matriz interna que funciona como uma tabela de informações.

Qualquer comunicação entre a Rede interna e Rede Externa em ambas as direções irá passar por verificação ou inclusão na Matriz, para que não seja utilizada mesma porta para a comunicação, no caso de saída da rede interna, ou então para uma nova tradução no caso de entrada da rede interna. Uma comunicação da rede externa que não tenha mapeamento na tabela será desconsiderada pois a rede externa não pode iniciar uma comunicação com um host na rede interna.

PROTOCOLO	IP INTERNO	PORTA INTERNA	IP EXTERNO	PORTA EXTERNA	IP DESTINO	PORTA DESTINO
TCP	10.0.0.2	8000	201.10.10.2	5000	142.68.52.2	4000
...	...	...	...	...	...	...

Figura 3. Exemplo de Tabela para Armazenamento das Informações.

3.1. ICMP

O pacote ICMP utiliza o IP apenas para a comunicação, dessa forma não é necessária abertura específica de seu cabeçalho, utilizamos as informações do cabeçalho IP. Então ao receber um pacote do tipo ICMP, o nosso NAPT utiliza as informações do cabeçalho de Ethernet e IP (Protocolo, IP origem, IP público e IP destino) para preencher a tabela. Quanto está transitando uma resposta, verifica-se na tabela a existência dessas informações para permitir o encaminhamento ao host correto.

3.2. TCP

A Abertura do pacote TCP é através do seguinte código:

```
struct.unpack("!HHLLBBHHH",
   tcp_header)
```

No caso de um pacote do tipo TCP, devemos desempacota-lo para acessar as portas utilizadas. Então acontece uma verificação em relação ao host destino e portas utilizadas, uma vez que dois hosts da rede interna utilizam o mesmo IP público, é necessário validar que não aconteça dois encaminhamentos para o mesmo destino utilizando a mesma porta. Por exemplo,

os hosts da rede interna A e B enviam pacotes TCP para o mesmo destino host C, utilizando sempre a mesma porta 12203 e o mesmo IP público. Neste caso é como se o destino recebesse o pacote do mesmo host, portanto verificamos na nossa tabela de tradução se já existe uma entrada para o host destino com a porta em questão, caso exista será atribuída ao novo pacote uma porta diferente. Optamos por acrescentar 1 ao número da porta original, então voltando ao exemplo, se a porta original for 12203, o segundo pacote a ser enviado para o host C, irá com a porta 12204. As informações (Protocolo, IP e Porta origem, IP e porta públicos e IP e porta Destino) serão adicionados como novas entradas na tabela de tradução.

Ao receber o pacote através da interface 1, nosso sistema refaz todo o processo de desempacotar os cabeçalhos para obter acesso à informação. Verifica-se se existe entrada correspondente na tabela de tradução e remonta o pacote alterando a porta e o IP Destino (neste momento está setado como o IP público) para a porta o IP interno do host presente em nossa rede. Então encaminha o pacote através da interface 0 para nossa rede interna.

3.3. UDP

A Abertura do pacote UDP é através do seguinte código:

```
struct.unpack("!HHHH",
   udp_header)
```

Aqui o processo é semelhante ao que realizamos nos pacotes do tipo TCP, primeiramente será feito um desempacotamento dos headers Ethernet, IP e logo em seguida UDP, para obter acesso às informações lá contidas. Quando está saindo da rede interna, o servidor NAPT remonta o pacote alterando o IP origem para o IP público, também é verificado a necessidade de alterar a porta origem, caso exista algum host interno enviando pacotes para o mesmo destino e porta. As informações (Protocolo, IP e Porta origem, IP e porta públicos e IP e porta Destino) serão adicionados como novas entradas na tabela de tradução.

Ao receber o pacote vindo da rede externa, é necessário validar as entradas na tabela de tradução, para isso refazemos todo o processo de desempacotar os cabeçalhos para obter acesso aos dados, e verificar as informações batem com alguma linha da tabela de tradução, caso encontre, remove essas entradas da tabela e altera o IP Destino (público) deste pacote para o IP do host interno e porta se for necessário. Reempacota os cabeçalhos e envia o pacote para a rede interna.

3.4. Comunicação entre as Redes

Como foi descrito anteriormente, a comunicação através do servidor NAPT pode ser feita entre a Rede Interna e a Rede externa, e vice e versa, porém o processo inicial de comunicação deve ser iniciado pela Rede interna, e será rejeitada a tentativa de comunicação inicial de qualquer host da rede externa com a rede interna. Todo o processo explicado nas seções anteriores são para a abertura dos pacotes, leitura das informações e a criação dos novos pacotes. Porém no existe uma diferenciação entre o envio dos pacotes para a rede externa e para rede interna.

3.4.1. Rede Externa para Rede Interna

Quando um host tenta a comunicação com a rede interna ele utiliza um IP e Porta que não é o IP e Porta real da rede interna, dessa forma é realizada uma varredura na Tabela citada na Figura 3 procurando pelo IP e Porta enviados pelos host externo, afim de descobrir qual é o IP e Porta Internos que o Servidor NAPT deve criar o novo pacote.

3.4.2. Rede Interna para Rede Externa

Diferente do processo citado na Seção 3.4.1, a rede interna recebe o IP e Porta internos e atribui essas informações a um IP e Porta Externos, tudo isso é anotado na Tabela citada na Figura 3, assim caso a comunicação tenha um retorno da rede externa o procedimento citado na Seção 3.4.1 pode ser realizado.

4. Execução

Toda a execução está descrita e comentada no vídeo que está em anexo a este relatório, com os exemplos de comunicação solicitados pelo professor. Tendo em vista este vídeo em anexo não será feita nenhuma explicação da execução do programa implementado neste relatório.

5. Considerações

O trabalho foi implementado, porém apresentou erro após a retirada do *IpForwarding* do Servidor NAPT, os pacotes não retornam do host destino da rede externa para a rede interna apresentando um erro o erro apresentado está na Figura 4.

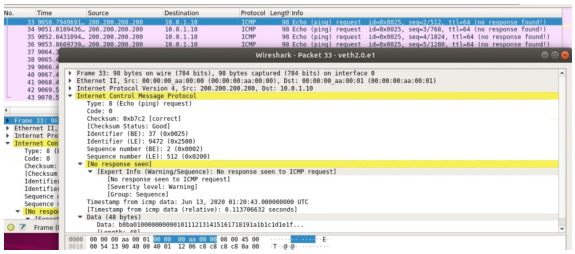

Figura 4. Exemplo de Tabela para Armazenamento das Informações.

Depois de muitas tentativas e inclusive auxílio dos colegas (uma das sugestões foi verificar o checksum dentro do cabeçalho dos pacotes, então investigamos isso dentre outras coisas, apesar de que no wireshark consta como correto e válido a informação do checksum), não conseguimos solucionar o problema, porém o trabalho utilizando *IpForwarding* funciona até certo ponto no envio e recebimento dos pacotes.