



**Universidade de Brasília - UnB  
Faculdade UnB Gama - FGA  
Curso de Engenharia de Software**

***Lab01 – Simulação do modelo de Referência OSI  
pela construção de um protótipo de camada  
de enlace de dados utilizando um modelo de  
comunicação Half-Duplex***

**Fundamento de Redes de Computadores - Turma A**

**Professores: Fernando William Cruz**

**Autores: Bruno Carmo Nunes - 18/0117548  
Marcos Vinícius Rodrigues da Conceição - 17/0150747**

**Brasília, DF  
2021**



## 1. INTRODUÇÃO

Na ciência da computação, mais especificamente em redes de computadores, a camada de ligação de dados, também conhecida como camada de enlace de dados ou camada de link de dados, é uma das sete camadas do modelo OSI. Esta camada detecta e, opcionalmente, corrige erros que possam acontecer na camada física. É responsável pela transmissão e recepção (delimitação) de quadros e pelo controle de fluxo. Ela também estabelece um protocolo de comunicação entre sistemas diretamente conectados. Neste laboratório foi de extrema necessidade o entendimento sobre o funcionamento e comunicação desta camada para a execução correta da atividade. Outros conceitos já previamente estabelecidos em aulas anteriores como as comunicações via protocolos de comunicação distintos TCP e UDP suas particularidades também foram utilizados.

## 2. DESCRIÇÃO DA SOLUÇÃO

1. O primeiro problema foi entender como os códigos bases em C disponibilizados pelo professor agiam. Pois algumas funções e bibliotecas eu ainda não as tinha utilizado em C.

Foi pesquisado sobre o funcionamento das bibliotecas `<sys/socket.h>`, `<netinet/in.h>` e `<arpa/inet.h>`. Entendi algumas das funções que elas fazem em relação à construção do percurso da construção das camadas socket, bind, listen, accept, send, recv e close.

2. Houve uma grande dificuldade de conseguir estabelecer a conexão com outra máquina utilizando código base disponibilizado. Após muita pesquisa e estudo, entendi juntamente dos meus outros colegas do grupo que era um problema na abertura das portas ,do endereço da rede local que o modem gerava e também com restrições com o firewall do sistema operacional. Foi necessário aprender como criar um port forwarding (Redirecionamento de portas), e desabilitar o firewall para que fosse possível o envio dos dados entre duas máquinas distintas em redes diferentes.

3. As camadas N e N-1 estão todas dentro do mesmo código (processo executado) pois não conseguimos mandar informações entre os processos muito menos

produzir uma fila de comunicações entre eles. Porém foi feito isso dentro do código já gerado para aplicação.

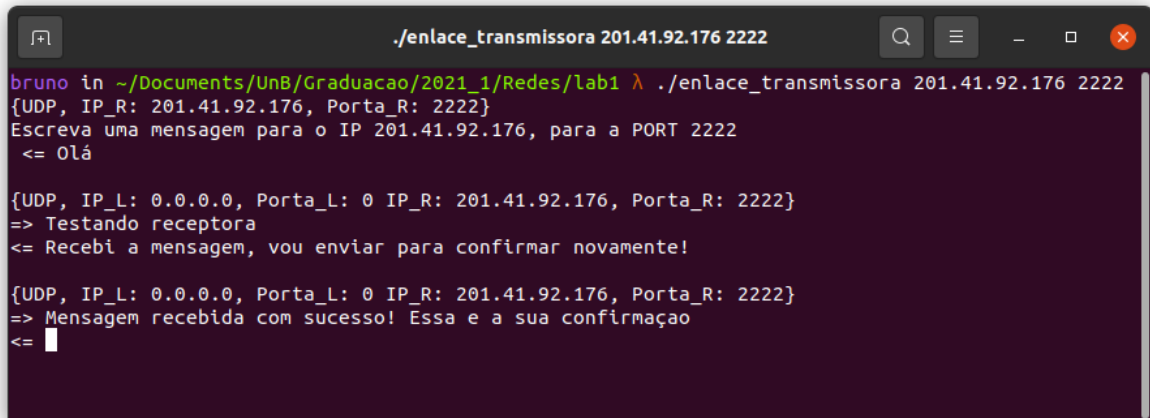
4. Outro problema era entender como seria o procedimento da fila de espera em relação a mensagens. Como cada mensagem é enviada por um final de linha ('\n') onde os caracteres de uma única mensagem podem ser de até 200, foi feito o envio de múltiplas mensagens no console para ver se ele guardava no buffer do teclado as informações passados posteriormente, e notamos que tudo que era guardado e enviados quando chegasse a vez de fala no processo.

Application Name	WAN Connection	WAN Port	LAN Port	Device Name	Internal Client	Protocol	Status
Customer settings	1_TR069_INTERNET_R_VID_210	1111~1113	1111~1113	192.168.1.109	192.168.1.109	UDP	ACTIVE
Customer settings	1_TR069_INTERNET_R_VID_210	2222~2222	2222~2222	marcos-HP-Laptop-15	192.168.1.76	UDP	ACTIVE

Figura 1 - Redirecionamento do porta realizado no modem para o envio e recebimento de dados utilizando protocolo UDP

### 3. CÓDIGO

O programa foi executado em duas máquinas diferentes, a máquina com o IP:PORT - 177.235.17.59:56865 ( Bruno ) e a máquina com o IP:PORT 201.41.92.176:2222 ( Marcos). Depois de compilados os códigos em C, na máquina do Marcos executou-se o código `./recebe 192.168.1.76 2222` onde o IP é da máquina local na LAN onde o roteador do Marcos reside, efetuando então o port forward do IP público para máquina dele, esperando a recepção de dados. Já na máquina do Bruno, executou-se o código `./enlace_transmissora 201.41.92.176 2222` onde esse IP é justamente o endereço de IP público para a casa onde está a máquina do Marcos. Como o modelo é Half-Duplex, a ideia é que um fala e outro escuta e vice versa, sendo assim, o primeiro a enviar mensagem é da máquina do Bruno, depois a máquina do Marcos recebe, e posteriormente envia para a máquina do Bruno, onde a mesma recebe e então fica nesse laço até parar com o processo.



```

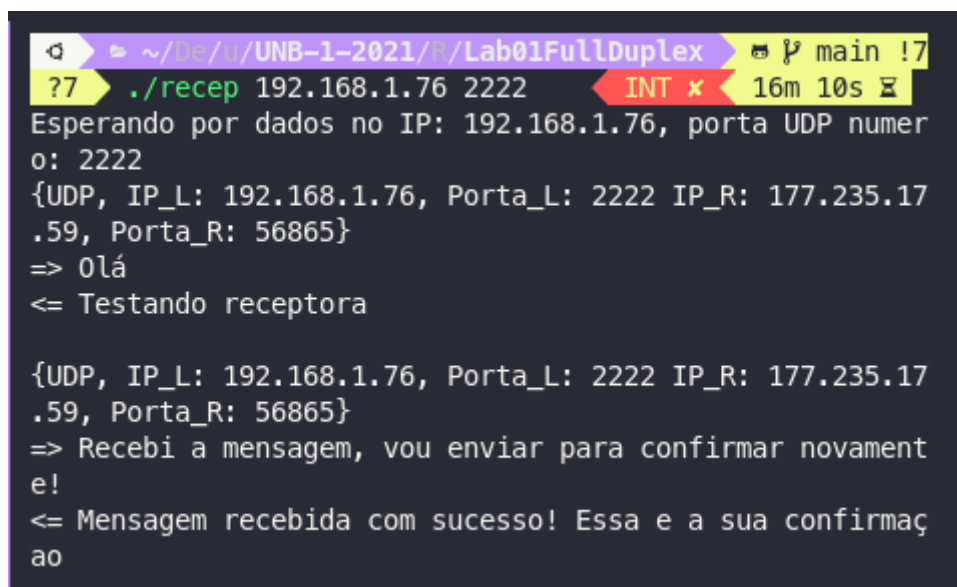
./enlace_transmissora 201.41.92.176 2222
bruno in ~/Documents/UnB/Graduacao/2021_1/Redes/lab1 λ ./enlace_transmissora 201.41.92.176 2222
{UDP, IP_R: 201.41.92.176, Porta_R: 2222}
Escreva uma mensagem para o IP 201.41.92.176, para a PORT 2222
<= Olá

{UDP, IP_L: 0.0.0.0, Porta_L: 0 IP_R: 201.41.92.176, Porta_R: 2222}
=> Testando receptora
<= Recebi a mensagem, vou enviar para confirmar novamente!

{UDP, IP_L: 0.0.0.0, Porta_L: 0 IP_R: 201.41.92.176, Porta_R: 2222}
=> Mensagem recebida com sucesso! Essa e a sua confirmação
<=

```

figura 2 - Recebimento dos dados enviados do PC1(client-Bruno) para o PC2(servidor-Marcos)



```

~/Documents/UnB-1-2021/1/Lab01FullDuplex main !7
?7 ./recep 192.168.1.76 2222 INT x 16m 10s
Esperando por dados no IP: 192.168.1.76, porta UDP numero: 2222
{UDP, IP_L: 192.168.1.76, Porta_L: 2222 IP_R: 177.235.17.59, Porta_R: 56865}
=> Olá
<= Testando receptora

{UDP, IP_L: 192.168.1.76, Porta_L: 2222 IP_R: 177.235.17.59, Porta_R: 56865}
=> Recebi a mensagem, vou enviar para confirmar novamente!
<= Mensagem recebida com sucesso! Essa e a sua confirmação

```

figura 3 - Recebimento dos dados enviados do PC1(Client-Bruno) para o PC2(servidor-Marcos)

## 5. CONCLUSÃO

Neste projeto tivemos o primeiro baque sobre como é necessário o entendimento claro dos processos, camadas, protocolos entre outros para que fosse possível executar o laboratório de maneira satisfatória. Enfrentamos bastante dificuldade em entender os motivos do porque não estarmos conseguindo estabelecer a conexão e depois essa mesma ser recusada pelo servidor. Esse laboratório foi muito enriquecedor pois conseguimos aprender várias outras habilidades que não estavam necessariamente no foco da atividade, como configurar no modem local, um redirecionamento de portas para que a mensagem do cliente conseguisse chegar no servidor e vice-versa. Aprendemos a utilizar um pouco do Uncomplicated Firewall também conhecido como UFW para permitir a chegada dos dados ao processo que mantinha o servidor aberto. Também foi possível notar que o servidor consegue receber dados de vários clientes como exemplificado na figura 3 onde eu, Marcos e o Bruno, cada um em máquinas distintas, conseguimos enviar os dados sem problema. Como é possível notar nas figuras 2 e 3, criamos um chat rudimentar UDP com a funcionalidade de buffer (que funciona de certa forma como uma fila de espera) ao enviar uma mensagem quando em modo de recepção ele espera a mensagem ser recebida para o envio automático quando entra em modo de envio.


Vemos como é interessante essa abstração da comunicação entre máquinas com os protocolos utilizando uma camada de enlace de dados. Inferimos que o UDP é limitado para muitas chamadas no programa pois podem vir resultados misturados ou até mesmo fazendo a mesma operação várias vezes pela falta de confirmação com o cliente. Num cenário de um site que precisa de muitas requisições, isso seria complicado pois poderiam acarretar a problemas na solução computacional, levando então prejuízo ao dono e os usuários da solução.

## 6 - Referências

Andrew S. Tanenbaum - Redes de Computadores, Quarta Edição 2003

Tanenbaum, Andrew S. Redes de Computadores, Editora: Elsevier, Quarta Edição 2003

 TCP vs. UDP - Acessado em 05/07/2021

 UDP Programming in C - Acessado em 05/07/2021

<https://docente.ifrn.edu.br/tadeuferreira/disciplinas/2012.1/redes-i-eja/Aula12.pdf> -

Acessado em 05/07/2021