Comparação de Desemprenho TCP e UDP

Eduarda Elger[1], Heloisa A. Alves[1]

[1]Centro de Ciências Exatas e Tecnológicas– Universidade Estadual do Oeste do Paraná (UNIOESTE)
Rua Universitária 1619 – 85819-110 – Cascavel – PR – Brazil

eduarda.elger@unioeste.br, heloisa.alves@unioeste.br

Abstract. *This article descibes the implementation of code from TCP and UDP protocol, with and without delivery guarantee. In this way, this articles subject is to compare the performance of both protocols, where will be tested in two different computers. For each test will be used packets with different sizes, being 100 bytes, 500 bytes, 1000 bytes.*

Resumo. *Este artigo descreve a implementação de códigos dos protocolos TCP e UDP, com e sem garantia de entrega. Sendo assim, o presente trabalho tem como objetivo comparar o desemprenho em ambos os protocolos, onde serão realizados testes em dois computadores diferentes. Para cada teste será utilizado pacotes com tamanhos diferentes, sendo de 100 bytes, 500 bytes e 1.000 bytes.*

1. Introdução

A camada de transporte de uma rede é de extrema importância atualmente, este é responsável pela transferência de dados entre máquinas, reunindo diversos protocolos, entre eles, os protocolos TCP e UDP.

"O protocolo de controle de transmissão (TCP) é orientado à conexão, o que significa que, uma vez que a conexão foi estabelecida, os dados podem ser transmitidos em duas direções. Ele tem sistemas integrados para verificar se há erros e garantir que os dados sejam entregues na ordem em que foram enviados, tornando-o o protocolo perfeito para a transferência de informações como imagens estáticas, arquivos de dados e páginas da web." (SILVA, 2023).

"O User Datagram Protocol (UDP) é um protocolo de Internet mais simples e sem conexão, no qual os serviços de verificação e recuperação de erros não são necessários. Com o UDP, não há sobrecarga para abrir, manter ou encerrar uma conexão — os dados são continuamente enviados para o destinatário, quer ele os receba ou não." (SILVA, 2023).

"A principal diferença entre TCP (protocolo de controle de transmissão) e UDP (protocolo de datagramas do usuário) é que TCP é um protocolo baseado em conexão e UDP é sem conexão. Enquanto o TCP é mais confiável, ele transfere dados mais lentamente. O UDP é menos confiável, mas funciona mais rapidamente. Isso torna cada protocolo adequado para diferentes tipos de transferências de dados." (GORMAN, 2023).

Quando queremos a comunicação entre interprocessos distribuídos podemos utilizar os socktes dos protocolos de transporte. "Os sockets UDP e TCP são a interface provida pelos respectivos protocolos na interface da camada de transporte" (TACLA, 2022). Também, é possível realizar comunicações sockets multicast, onde podemos enviar um único pacote IP para um conjunto de processos.

Para melhor visualização e compreensão desse tema será analisado neste artigo o uso e o comportamento de uma aplicação TCP comparando-o com uma implementação do UDP com e sem garantia. O trabalho foi subdividido em uma descrição dos protocolos (seção 2); uma descrição computacional (seção 3), contendo informações sobre a implementação do TCP e UDP e o ambiente de teste. Na seção 4 são apresentados os resultados e na seção 5 as considerações finais.

2. Protocolos

"O User Datagram Protocol (UDP) e o TCP são os protocolos básicos de nível de transporte para fazer conexões entre os hosts da Internet. Tanto o TCP quanto o UDP permitem que programas enviem mensagens para e recebam mensagens de aplicativos em outros hosts. Quando um aplicativo envia um pedido para a camada de Transporte enviar uma mensagem, UDP e TCP quebrar as informações em pacotes, adicionar um cabeçalho de pacote incluindo o endereço de destino e enviar as informações para a camada de Rede para processamento adicional. Tanto o TCP quanto o UDP utilizam portas de protocolo no host para identificar o destino específico da mensagem. Protocolos de nível superior e aplicativos usam UDP para fazer conexões de datagramas e TCP para fazer conexões de fluxo. A interface de soquetes do sistema operacional implementa esses protocolos." (IBM CORPORATION, 2023).

2.1. Protocolo TCP

"A sigla TCP (Transmission Control Protocol) é o protocolo da rede de computadores responsável por toda a entrega de dados. O protocolo TCP é conhecido por ser um padrão de comunicação que permite que dispositivos e aplicativos troquem mensagens entre si através do uso de rede. Esse protocolo é totalmente orientado por conexões, de modo que uma conexão só é finalizada no momento que uma das extremidades finaliza a troca de pacotes. Assim, se o pacote de dados não é entregue para o destinatário, o processo de envio vai se repetindo constantemente até obter sucesso. Para ficar mais claro seu funcionamento, imagine um pacote de dados que precisa ser enviado para outro dispositivo. Usando o TCP, o protocolo garante a entrega desses dados ao outro dispositivo de forma bem-sucedida." (SEABRA, 2022).

2.2. Protocolo UDP

"O protocolo de datagramas do usuário (UDP) opera sobre o protocolo da Internet IP) para transmitir datagramas em uma rede. O UDP não exige que a origem e o destino estabeleçam um handshake triplo antes que a transmissão ocorra. Além disso, não há necessidade de uma conexão de ponta a ponta. Em contraste, o UDP é considerado um protocolo sem conexão porque não exige que um circuito virtual seja estabelecido antes que qualquer transferência de dados ocorra. O protocolo de comunicação apenas envia os pacotes, o que significa que ele tem muito menos sobrecarga de largura de banda e latência. Com ele, os pacotes podem seguir caminhos diferentes entre o emissor e o receptor e, como resultado, alguns pacotes podem ser perdidos ou recebidos fora de ordem. O UDP fornece dois serviços não fornecidos pela camada IP. Ele fornece números de porta para ajudar a distinguir diferentes solicitações de usuários e, opcionalmente, um recurso de soma de verificação para verificar se os dados chegaram intactos." (SEABRA, 2022).

3. Descrição Computacional

Sockets permitem a comunicação entre dois processos (programas) diferentes na mesma máquina ou em máquinas diferentes. Visto isso, foi utilizada a linguagem de programação *python* na versão 3.11.0 para a implementação dos protocolos, linguagem pelo qual tem

suporte a biblioteca de *socket* e permite que seja possível realizar o experimento de troca de mensagens entre máquinas diferentes.

O desenvolvimento dos códigos foi realizado através do ambiente de desenvolvimento integrado (IDE) Visual Studio Code. Para a troca de mensagens foi necessário que ambas as máquinas estivessem conectadas na mesma rede, o provedor da rede foi o “Conecta BR” onde conta com uma conexão de 1000 megabits por segundo.

Para a comunicação foi utilizada duas máquinas diferentes, a especificação de ambas pode ser visualizada abaixo:

- Para o cliente: foi utilizado um computador desktop com sistema operacional openSUSE Linux. Conta com um processador AMD Athlon 3000G, 8GB de RAM, 128GB SSD NVMe (somente para o sistema operacional, tem 3 HDs de 1TB para arquivos). E um IP estático: 192.168.100.150.
- Para o servidor: foi utilizado um notebook com o sistema operacional Arch Linux. Possui um processador Intel Core i5-10210U, 8GB de RAM, 256GB de SSD NVMe. E seu IP estático: 192.168.100.151.

Vale salientar que ambas as máquinas estão conectadas diretamente via cabo de rede em um switch tp-link LS105G com portas de 10/100/1000Mbps.

3.1. Implementação TCP

No processo do protocolo TCP temos quatro códigos, onde dois correspondem ao cliente e os outros ao servidor.

Para o cliente temos o código **“Cliente_TCP.py”.** O código em questão implementa um cliente TCP que envia dados para um servidor. Ele utiliza a biblioteca socket para estabelecer uma conexão com o servidor TCP em um determinado endereço IP e porta. O cliente lê dados de um arquivo binário chamado 'teste.txt' em partes, enviando cada parte para o servidor. O número total de pacotes enviados e o tempo total de comunicação são registrados e retornados pela função. O código garante que os pacotes sejam enviados em um ritmo controlado para evitar congestionamento usando uma pausa mínima entre os envios. No final, a conexão é encerrada e o arquivo é fechado. Já o código **“Main_Cliente_TCP.py”** realiza testes de comunicação TCP em diferentes tamanhos de pacotes de dados. Ele executa iterações de teste, chamando a função **Cliente_TCP** para cada tamanho de pacote e registrando os tempos de comunicação e o número de pacotes enviados. Os resultados são armazenados em um arquivo CSV chamado "**ResultadosTCP.csv**".

Para o servidor, o código ”**Servidor_TCP.py**” recebe dados enviados por um cliente e os salva em um arquivo. Ele utiliza a biblioteca socket para estabelecer a conexão e receber os dados. O servidor fica escutando por conexões em um endereço IP e porta específicos. Quando uma conexão é estabelecida, o servidor recebe blocos de dados do cliente e os escreve em um arquivo. O código também permite que o servidor faça uma pausa opcional antes de encerrar a conexão. E o código “**Main_Servidor_TCP.py**” realiza testes do servidor TCP para diferentes tamanhos de pacotes de dados. Ele executa múltiplas iterações de teste para cada tamanho de pacote, chamando a função **Servidor_TCP** e registrando os resultados. Após cada iteração, há uma pausa antes de prosseguir para a próxima.

Para execução de ambos os códigos basta executar o código *main* no terminal da máquina específica. É importante mencionar que o código do cliente deve conter o endereço IP da máquina do servidor. Abaixo segue o comando que deve ser utilizado em cada caso:

- Cliente: ***python3 Main_Cliente_TCP.py***
- Servidor: ***python3 Main_Servidor_TCP.py***

3.2. Implementação UDP com e sem garantia

Para o protocolo UDP temos apenas dois códigos, onde ambos correspondem ao UDP com e sem garantia.

O código "**udp.py**" implementa uma comunicação cliente-servidor usando sockets UDP. A classe Cliente envia arquivos para o servidor, enquanto a classe Servidor recebe os arquivos enviados. Ambas as classes utilizam sockets UDP e possuem opções de garantia de entrega. O cliente envia os dados do arquivo em partes de tamanho fixo e pode reenviar pacotes se necessário. O servidor recebe os pacotes e os escreve em um arquivo. O código usa a biblioteca socket para as operações de comunicação e a biblioteca time para medir o tempo de execução e introduzir pausas.

Agora, o código "**main.py**" realiza testes de comunicação UDP em um cenário cliente-servidor. Ele permite escolher entre executar como cliente ou servidor através de argumentos na linha de comando. Os testes são realizados com diferentes tamanhos de pacotes e número de iterações. Os resultados são armazenados em um arquivo CSV.

Para execução de ambas as condições (com garantia ou sem), basta executar o código *main* no terminal da máquina específica. É importante mencionar que o código do cliente deve conter o endereço IP da máquina do servidor. Abaixo segue o comando que deve ser utilizado em cada caso:

- Cliente sem garantia: ***python3 main.py rec.txt servidor***
- Servidor: ***python3 main.py env.txt cliente***
- Cliente com garantia: ***python3 main.py rec.txt servidor garantia***
- Servidor com garantia: ***python3 main.py env.txt cliente garantia***

4. Análise e Resultados

Nesta etapa os testes aplicados nos protocolos contaram com um arquivo texto com um tamanho de 300 MB, onde o utilizamos de teste para diferentes tamanhos de pacotes, sendo eles: 100 bytes, 500 bytes e 1000 bytes. Foram realizadas cerca de dez execuções, todos os resultados foram armazenados em arquivo CSV para análise posterior. A fim de garantir que o servidor está tendo troca de mensagens um arquivo de texto é criado para receber todo o conteúdo do arquivo de teste a cada interação entre o servidor e o cliente. Com base nestas informações foi realizado os seguintes cálculos: média, e desvio padrão com intervalo de confiança de 95% a fim de realizar uma análise de desempenho entre os protocolos. Abaixo seguem as fórmulas dos respectivos cálculos.

- Fórmula da média:

$$Média = \frac{\Sigma x_i}{n}$$

- Fórmula do desvio padrão:

$$S = \sqrt{\frac{\Sigma(x_i - \overline{x})^2}{n-1}}$$

- Fórmula do intervalo de confiança:

$$CI = \overline{x} \pm z\frac{s}{\sqrt{n}}$$

Todos os resultados dos cálculos e testes acima foram sinalizados em colunas e/ou linhas das tabelas subsequentes.

4.1. TCP:

Abaixo será demonstrado os resultados obtidos do protocolo TCP (tabela 1). Como o esperado conforme foi aumentado o tamanho dos pacotes a sua quantidade de pacotes enviados caíram drasticamente, isto ocorreu para que a rede não sobrecarregasse. Além disto, o TCP possui desempenho maior que os demais protocolos testados no quesito de entrega dos pacotes, já que a perda de pacotes é menor.

Tamanho	***Tempos***	***Pacotes***
100	1091,41	12046977
100	1062,63	12046977
100	1061,69	12046977
100	1063,90	12046977
100	1064,24	12046977
100	1065,43	12046977
100	1076,51	12046977
100	1060,63	12046977
100	1053,90	12046977
100	1057,49	12046977
500	83,33	943773
500	81,75	943773
500	84,01	943773
500	82,31	943773
500	83,36	943773
500	84,32	943773
500	82,31	943773
500	82,97	943773
500	82,92	943773
500	8,33	943773
1000	41,32	438541
1000	40,26	438541
1000	43,04	438541
1000	40,68	438541
1000	44,12	438541
1000	43,28	438541
1000	43,32	438541
1000	41,11	438541

1000 1000	40,50 42,09	438541 438541
Média	911,61	10.011.243.33
Desvio Padrão	478.86	1.306.992.060.689.372.8
Intervalo de confiança	740.43 a 1082.81	9.544.476.816.977.999 a 10.478.009.846.998.99

Tabela 1 – Tabela TCP: Dados

4.1. UDP (sem garantia):

Devido a tentativa de entrega mais rápida de pacotes, o número de pacotes enviados pelo UDP (tabela 2) acaba sendo muito maior do que a do TCP, já que muitos destes pacotes acabam sendo perdidos devido ao protocolo não possuir garantia de entrega.

Testes	***Tamanhos***	***Tempos***	***Pacotes***	***Reenvios***
0	100	15.701205968856812	4095973	0
1	100	15.702195167541504	4095973	0
2	100	17.21751379966736	4095973	0
3	100	15.488649606704712	4095973	0
4	100	15.488712549209595	4095973	0
5	100	15.506744146347046	4095973	0
6	100	15.563997507095337	4095973	0
7	100	15.552164554595947	4095973	0
8	100	15.51793885231018	4095973	0
9	100	15.52293848991394	4095973	0
0	500	4.771906137466431	819196	0
1	500	4.752185821533203	819196	0
2	500	4.753793954849243	819196	0
3	500	4.760676860809326	819196	0
4	500	4.74602484703064	819196	0
5	500	4.78772497177124	819196	0
6	500	4.768585205078125	819196	0
7	500	4.759012937545776	819196	0
8	500	4.774017095565796	819196	0
9	500	4.71122407913208	819196	0
0	1000	6.834809064865112	409599	0
1	1000	7.498744010925293	409599	0
2	1000	7.116926908493042	409599	0
3	1000	7.323587656021118	409599	0
4	1000	7.0773255825042725	409599	0
5	1000	8.564354181289673	409599	0
6	1000	8.167242050170898	409599	0
7	1000	4.379950523376465	409599	0
8	1000	3.59610652923584	409599	0
9	1000	9.095508098602295	409599	0
Média		6.91	1.365.324,33	0

Desvio padrão	5.99	1.056.703,401	0
Intervalo de confiança	4.6743 a 9.1457	970.815 a 1.759.833	0

Tabela 2 – Tabela UDP (sem garantia): Dados

4.1. UDP (com garantia):

Conforme novamente esperado, o tempo de envio do UDP com garantia foi maior que o de sem garantia, já que agora ele possui mais regras a serem seguidas. Além disto a média de envio de pacotes também foi maior, já que neste protocolo a perda de pacotes é menor. Esse detalhes podem ser observados na tabela 3.

Testes	***Tamanhos***	***Tempos***	***Pacotes***	***Reenvios***
0	100	618.1606502532950.	4095973	0
1	100	603.5811774730680.	4095973	0
2	100	609.5175127983090.	4095973	0
3	100	595.5952620506280.	4095973	0
4	100	589.8877689838400.	4095973	0
5	100	596.8328363895410.	4095973	0
6	100	599.4635345935820.	4095973	0
7	100	588.215991973877.	4095973	0
8	100	581.2298521995540.	4095973	0
9	100	603.5599157810210.	4095973	0
0	500	122.3948187828060.	819196	0
1	500	124.09078431129400.	819196	0
2	500	124.52579498291000.	819196	0
3	500	129.88308835029600.	819196	0
4	500	126.86087846755900.	819196	0
5	500	126.82047629356300.	819196	0
6	500	136.74588680267300.	819196	0
7	500	133.11480498313900.	819196	0
8	500	131.69846987724300.	819196	0
9	500	124.04574489593500.	819196	0
0	1000	91.4771876335144.	409599	0
1	1000	91.18001866340630.	409599	0
2	1000	91.24642992019650.	409599	0
3	1000	90.81075382232660.	409599	0
4	1000	91.05000948905940.	409599	0
5	1000	90.86288690567010.	409599	0
6	1000	91.83081126213070.	409599	0
7	1000	91.28479552268980.	409599	0
8	1000	91.43878889083860.	409599	0
9	1000	91.16267395019530.	409599	0
Média		290.460	1.773.922.67	0
Desvio padrão		237.504	1,589,304.8704935366	0
Intervalo de confiança		201.98681270572334 a 378.93384020581704	1,181,934.34 a 2,365,910.00	0

Tabela 3 – Tabela UDP (com garantia): Dados

5. Conclusão

Pelos conceitos que foram vistos a respeito dos protocolos de transporte TCP e UDP, podemos ver de forma prática como é feito o processo de comunicação entre cliente-servidor na rede, como observado nas execuções dos códigos podemos ver como é a interface dos *sockets* quando é feita as trocas de mensagens.

Observou-se que o protocolo TCP é mais confiável e eficiente em termos de transmissão de pacotes, visto que seu tempo de resposta é um pouco maior que outros protocolos, e a sua entrega é mais garantida. Embora o protocolo UDP tenha um tempo de resposta menor para disparar vários pacotes, ele não possui garantia de entrega, o que resulta na perda de vários pacotes, o que reduz sua confiabilidade.

Com isso, concluímos a importância do TCP e UDP na rede. O TCP pelo qual é dito como o confiável devido a sua qualidade de serviço de comunicação, sendo útil em aplicações que realizam transações bancárias, devido a sua garantia de entrega durante a transmissão onde os dados e valores ficam livrs de errros em sequência e sem perdas ou duplicação. Mesmo o UDP sendo descrito como não confiável, ele é muito útil em serviços em que a velocidade é mais essencial e a perda mínima dos dados não gere desvantagens. Um exemplo ideal seria em questões de jogos online, onde é necessário que a aplicação siga rodando mesmo que alguns dados se percam na comunicação, evitando assim os famosos lags (atrasos na comunicação)

6. Referências

PROTOCOLOS de Transporte Internet-Protocolos de Nível. [*S. l.*], 24 mar. 2023. Disponível em: https://www.ibm.com/docs/pt-br/aix/7.3?topic=protocols-internet-transport-level. Acesso em: 14 jul. 2023.

QUAL a diferença entre TCP e UDP? Entenda o que são esses protocolos!. [*S. l.*], 27 abr. 2023. Disponível em: https://gaea.com.br/diferenca-entre-tcp-e-udp/. Acesso em: 14 jul. 2023.

SEABRA, Giulianna. **O que é UDP e quais as diferenças com o TCP?**. [*S. l.*], 20 dez. 2022. Disponível em: https://blog.betrybe.com/desenvolvimento-web/udp-diferencas-tcp/. Acesso em: 14 jul. 2023.

TACLA, C. A. SOCKETS UDP, TCP E MULTICAST. Universidade Tecnológica Federal do

Paraná. Apresentação do Power Point. Disponível em: < http://www.inf.ufsc.br/~bosco.sobral/ensino/ine5645/2014-1/0050-Sockets.pdf > Acesso em: 14 jul. 2023.

TCP ou UDP: qual é a diferença e qual é o melhor protocolo?. [*S. l.*], 23 fev. 2023. Disponível em: https://www.avast.com/pt-br/c-tcp-vs-udp-difference. Acesso em: 14 jul. 2023.