Redes de Computadores

Relatório TP3

Sockets UDP, janela deslizante
Prática para exercitar o uso de sockets UDP e temporizações

Cleves Henrique Ilarino Palhares Matrícula: 2012021543
Vinícius de Oliveira Silva Matrícula: 2013007820

Data de entrega: 03/12/2017

Introdução

O trabalho a seguir consistiu em implementar um par de programas que operem no modelo cliente-servidor e exercitem tanto a transmissão unidirecional quanto a comunicação do tipo requisição-resposta sobre o protocolo UDP, utilizando um protocolo de janela deslizante. A implementação deve utilizar a biblioteca de sockets do Unix (Linux).

Como foi utilizado UDP foi necessário criar um protocolo confiável para garantir a ordem dos pacotes, para essa tarefa utilizamos o sistema da janela deslizante.

O protocolo criado foi unidirecional, isto é, focou no envio de dados na direção do servidor para o cliente. Também foi proposto mensagens de confirmação em diferentes momentos além de técnicas de detecção de erros.

Na implementação foi utilizada a biblioteca tp_socket.h na criação dos sockets e envio dos dados.

Execução

O sistema cliente-servidor proposto para este trabalho foi implementado utilizando a linguagem de programação C99 através da IDE Jetbrains CLion e compilado através da ferramenta GCC. Todo o ambiente de desenvolvimento e execução foi baseado em Linux. Para compilar e executar os programas, deve se utilizar o comando make seguido dos seguintes comandos:

./clienteFTP <IP_Servidor> <Porta_de_Acesso> <Nome_Arquivo> <Tamanho_Buffer> <Tamanho_janela> (para o programa cliente);

./servidorFTP <Porta_de_Acesso> <Tamanho_Buffer> <Tamanho_janela> (para o programa servidor).

É importante também salientar os seguintes pontos:

- O programa servidor deve ser iniciado antes do programa cliente;

- O argumento **<IP_Servidor>** refere-se ao endereço IPv4 da máquina que executa o programa servidor. O formato esperado para esse argumento é: ###.###.###.###;

- O argumento **<Porta_de_Acesso>** é um número inteiro escolhido pelo usuário que deve ser (estritamente) maior que 1024 e (estritamente) menor que 65536. Este argumento deve ter necessariamente o mesmo valor tanto na entrada do programa cliente tanto do servidor;

- O argumento **<Tamanho_janela>** é um número inteiro escolhido pelo usuário;

- O argumento **<Tamanho_Buffer>** deve ser maior que 1 e também deverá ter o mesmo valor para os programas cliente e servidor;

- O argumento **<Nome_Arquivo>** é um string cujo número de caracteres deve ser menor que o tamanho do buffer;

- Os computadores que executam os programas cliente e servidor devem ser visíveis um ao outro. Isto é se o computador que executa o programa cliente tentar um ping, uma resposta válida deverá ser recebida.

- O arquivo requisitado deverá estar no mesmo diretório que o executável do programa servidor.

Implementação

Conforme mencionado anteriormente, a comunicação entre os programas Cliente e Servidor que compõem este trabalho é baseada na estratégia janela deslizante que é uma característica de alguns protocolos que permite que o remetente transmita mais que um pacote de dados antes de receber uma confirmação. Depois de recebê-lo para o primeiro pacote enviado, o remetente desliza a janela do pacote e manda outra confirmação. O número de pacotes transmitidos sem confirmação é conhecido como o tamanho da janela; aumentando o tamanho da janela melhora-se a vazão.

A base para o desenvolvimento do algoritmo criado é descrita nas figuras abaixo:

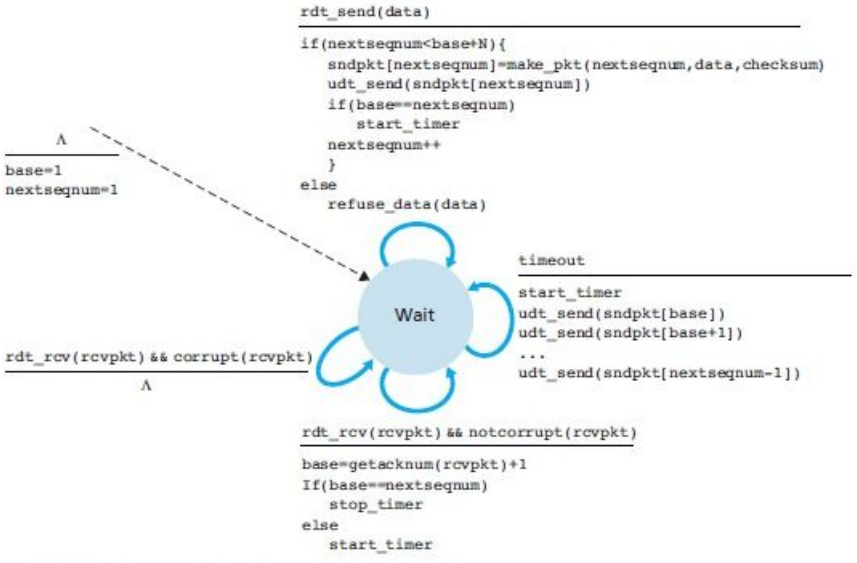


Figura 1- Funcionamento base do Servidor

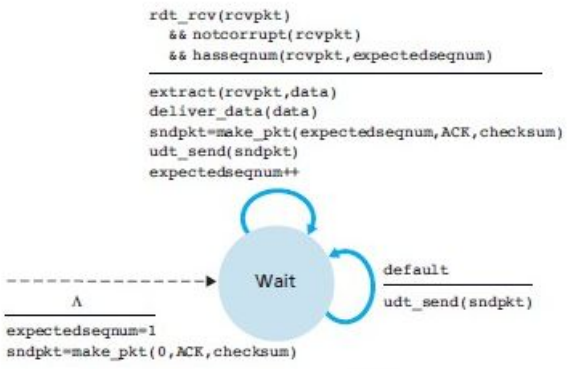


Figura 2 - Funcionamento Base do Cliente

A estrutura do pacote ficou definida da seguinte forma:

1 byte para identificar o tipo do pacote:

Cada pacote pode ser de 5 tipos:

0 - Pacote especial ACK;

1 - Pacote comum carregando dados;

2 - Pacote contendo os últimos bytes de uma transmissão;

3 - Pacote especial utilizado para requests ao servidor.

4- Pacote indicando que o cliente recebeu todos os pacotes válidos

Como pode-se perceber, poderiam ter sido utilizados apenas 3 bits para fazer essa identificação, porém optou-se por utilizar um byte completo (3 bits que variam + 5 bits constantes 0) para facilitar a implementação.

8 bytes para identificar o número de sequência do pacote:

Cada pacote pode ter um número de sequência que pode ser entre: 0 e 9223372036854775807

Esse campo do pacote faz parte de um mecanismo de sincronização entre cliente e servidor que funciona através do aceite ou recusa de pacotes, dependendo do valor do número de sequência do pacote esperado. Esta lógica pode ser descrita da seguinte forma:

“Uma janela de transmissão é, para o emissor, o conjunto de quadros que podem ser enviados independentemente de ter recebido suas confirmações. Por exemplo, um emissor com janela tamanho N pode enviar desde o quadro 0 até o quadro N-1 sem que tenha recebido confirmação, no entanto, para enviar o quadro N é necessário que o primeiro quadro enviado (o) seja confirmado. Quando a confirmação ocorre, a janela passa a incluir os quadros numerados de 1 até N, mantendo portanto o tamanho N. Da mesma forma, quando o quadro 1 é confirmado, a janela passa a incluir os quadros de 2 a N+1. Esse fato caracteriza o termo janela deslizante. Normalmente, a numeração de sequência é calculada sempre módulo M, onde M é o número máximo de sequência utilizado. É interessante também observar que caso o servidor receba uma confirmação de recebimento com um número de sequência X, este considerará que todos os pacotes com número de sequência menores que X também estão confirmados.

No modelo de janela deslizante Go Back N não existe o conceito de janela de recepção, ou seja, o cliente deve receber os pacotes na ordem correta e estes são imediatamente entregues à camada superior (neste caso, têm seu conteúdo salvo no arquivo). Se os pacotes recebidos não tiverem o número de sequência esperados pelo cliente, eles são descartados e o cliente envia um ack ao servidor informando qual foi o último pacote aceito com sucesso. Ao receber esta mensagem, o servidor retransmite todos os pacotes com número de sequência maior que o valor informado pelo cliente.

4 bytes para identificação do número de bytes úteis enviados:

Este campo identifica quantos caracteres válidos devem ser gravados pelo cliente ao receber um pacote do servidor. Essa informação é útil especialmente em pacotes do tipo 2, que na transmissão de um arquivo cujo tamanho em bytes não é múltiplo do tamanho do buffer,

carregarão um número de caracteres válidos menor que os demais pacotes, uma vez que carregarão apenas os últimos bytes do arquivo.

Ex.: Arquivo de 360 bytes, Buffer de 100 bytes:

1o Pacote: Carrega 100 bytes;

2o Pacote: Carrega 100 bytes;

3o Pacote: Carrega 100 bytes;

4o Pacote: Carrega 60 bytes úteis + 40 bytes de preenchimento, adicionados apenas para manter todos os pacotes do mesmo tamanho.

<Tamanho_do_buffer> bytes para armazenar informação útil

Este campo carrega porções do arquivo solicitado pelo cliente.

1 byte para CHECKSUM.

Este campo serve para armazenar o valor do CHECKSUM de cada mensagem calculado pelo emissor do pacote. Essa informação é útil para detectar erros de transmissão, uma vez que o receptor pode recalcular este número e comparar com o número original. Se ambos forem iguais, as chances de ter ocorrido um erro de transmissão são baixíssimas.

Além da adição de metadados às porções de dados úteis, também foi implementado um esquema de temporizações de recepção de mensagens, que consiste em retransmitir a última mensagem enviada caso a confirmação de recebimento (ACK) desta não chegue em um certo tempo pré-determinado, que no caso deste trabalho foi definido em 1 segundo.

Metodologia

Além de contribuir para o aprendizado de técnicas de programação de sistemas comunicantes via redes, o trabalho prático proposto tem também o objetivo de fazer com que os alunos abordem o problema comunicação via rede de uma maneira analítica. Para se obter essa perspectiva, foi pedido que variados testes fossem executados com o sistema a fim de observar e analisar seu comportamento em uma rede real. Para cumprir esse requisito, o sistema implementado foi submetido a diversos testes e a metodologia destes é apresentada nessa seção.

4.1 Dados sobre o experimento:

Para a criação e execução do TP foi utilizado o sistema operacional LINUX na versão mint-18.2-cinnamon-64bits. Foi executado em uma máquina virtual configurada para utilizar 1 núcleo de processador, 4 GB de Memória RAM e 20GB de HD placa de rede funcionando em modo NAT.

Tanto o cliente quanto o servidor foram executados sempre na mesma máquina. O teste foi executado basicamente com arquivo um arquivo de texto de 200Kbytes. Devido as simulações de erros de rede inseridos no código não foi viável utilizar um arquivo maior, pois o tempo de cada execução atingia valores consideravelmente altos.

4.2 Medições

De forma simples e direta, as medições requeridas para o desenvolvimento deste trabalho incluem a aferição do tempo decorrido entre o início e o fim do recebimento do arquivo por parte do programa cliente, a velocidade de transmissão da rede e o número total de bytes transferidos entre cliente e servidor. Apesar de algumas dessas medições parecerem triviais a primeira vista existem algumas nuances a se considerar, por exemplo: O número de bytes transferidos não é, nesse caso, igual ao número de bytes que constituem o arquivo, devendo-se considerar também os bytes de cabeçalho, perdas de pacotes na rede, possíveis corrupções de dados, timeouts nos canais de comunicação (definidos como 1s no contexto deste trabalho), etc. Desse modo, para se obter tais números, foi preciso interferir diretamente no código do programa cliente para que esse recordasse o número de bytes transferidos, o tempo gasto com o recebimento, etc. Do ponto de vista técnico, foi utilizada a função clock_gettime da biblioteca time.h da linguagem C para medir o tempo gasto na transferência e uma variável que controla os bytes recebidos, quer eles estejam corrompidos ou não.

4.3 Simulações de erros

Visando simular eventuais perdas e/ou danos dos pacotes transmitidos foram inseridas na biblioteca tp_socket algumas modificações.

Basicamente as modificações alteram o conteúdo de alguns pacotes e em outros casos simplesmente simula a perda do pacote completo. Cada um desses eventos ocorre com uma probabilidade de 5%.

Tais mecanismos visam simular perdas no canal e atestar a confiabilidade do protocolo criado para transmissão de dados.

4.4 Procedimento para aquisição de dados

O procedimento utilizado consistiu em executar em uma mesma máquina, porém em diretórios diferentes, o servidor e o cliente para realizar a transmissão do arquivo. Então registrou-se o tempo gasto na transmissão, a taxa de transmissão e a quantidade de dados enviados.

O tamanho do arquivo foi sempre 200Kbytes. E para cada transmissão foram alterados os parâmetros do tamanho do buffer e tamanho da janela conforme descrito a seguir:

Tamanhos do buffer: 100 bytes

Tamanho da Janela: 2, 4, 8, 16, 32, 64, 100 bytes

Tamanhos do buffer: 500 bytes

Tamanho da Janela: 2, 4, 8, 16, 32, 64, 128, 256, 500 bytes

Tamanhos do buffer: 1000 bytes

Tamanho da Janela: 2, 4, 8, 16, 32, 64, 128, 256, 512, 1000 bytes

Resultados

Com base nos procedimentos foram obtidos os seguintes dados.

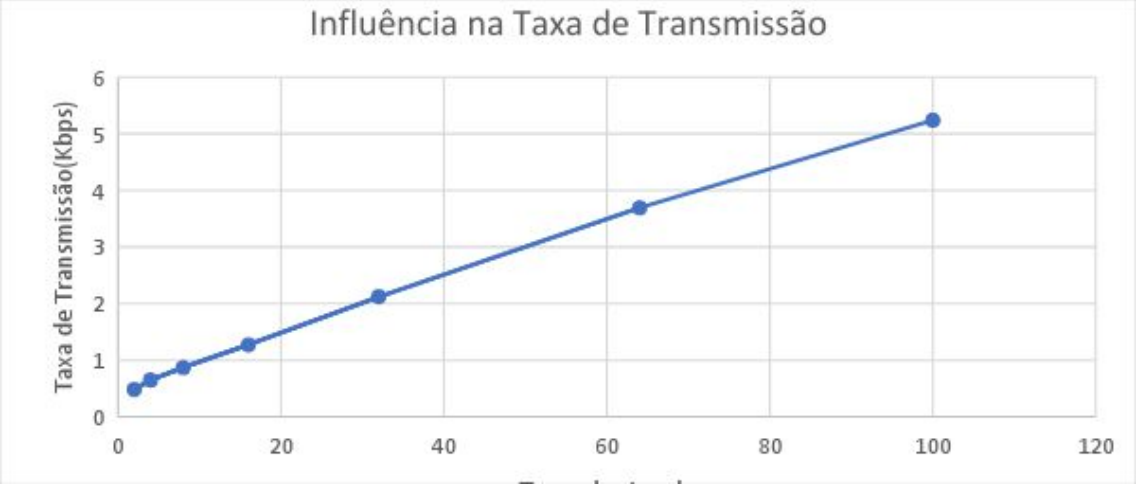
Buffer com tamanho de 100 Bytes:

Tamanho Arquivo	Tamanho Buffer	Tamanho Janela	Tempo de Transmissão	Taxa de Transmissão	Bytes Transmitidos
200k	100	2	563,214	0,48	269006
200k	100	4	486,456	0,65	316266
200k	100	8	483,346	0,87	419555
200k	100	16	492,545	1,27	625993
200k	100	32	491,61	2,12	1040387
200k	100	64	480,255	3,69	1774096
200k	100	100	480,568	5,24	2518406

Analisando esses dados percebemos que o aumento do tamanho da janela diminui o tempo de transmissão, porém quando o tamanho da janela aumenta demais esse ganho acaba não compensando uma vez que no caso de um pacote recebido estar corrompido, a quantidade de informações a serem retransmitidas é consideravelmente grande. Para o extremo da janela de transmissão ser do mesmo tamanho que o próprio buffer, a quantidade de bytes transmitido foi 12 vezes o tamanho efetivo do arquivo.

De toda forma a estratégia da janela deslizando é consideravelmente mais rápida que o método stop – wait.

Vejamos graficamente abaixo os resultados obtidos:


Influência na Taxa de Transmissão
Taxa de Transmissão(Kbps)
0
1
2
3
4
5
6
0
20
40
60
80
100
120

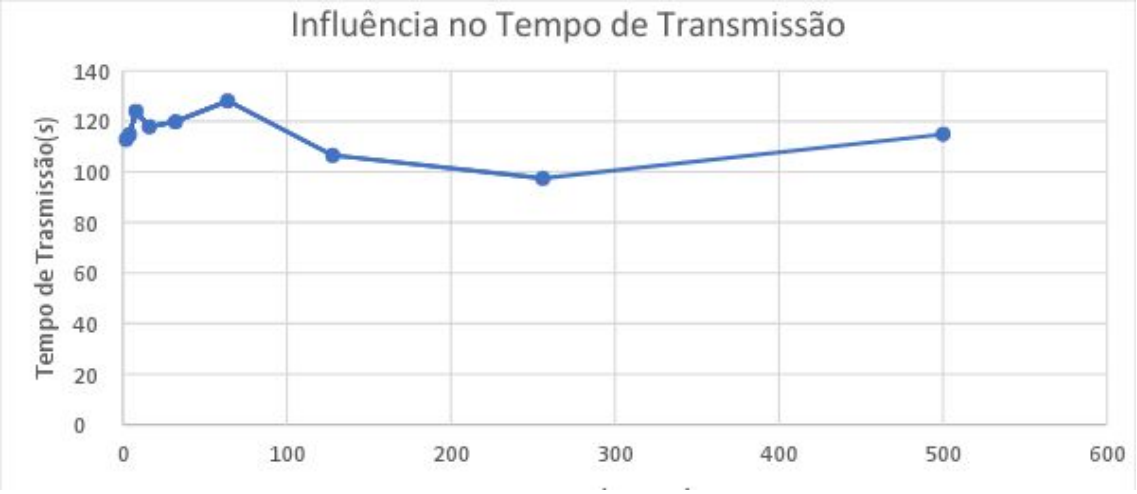

Influência Volume de Dados Transferidos
Qtd de Bytes Transmitidos
0
500000
1000000
1500000
2000000
2500000
3000000
0
20
40
60
80
100
120
Tamanho Janela

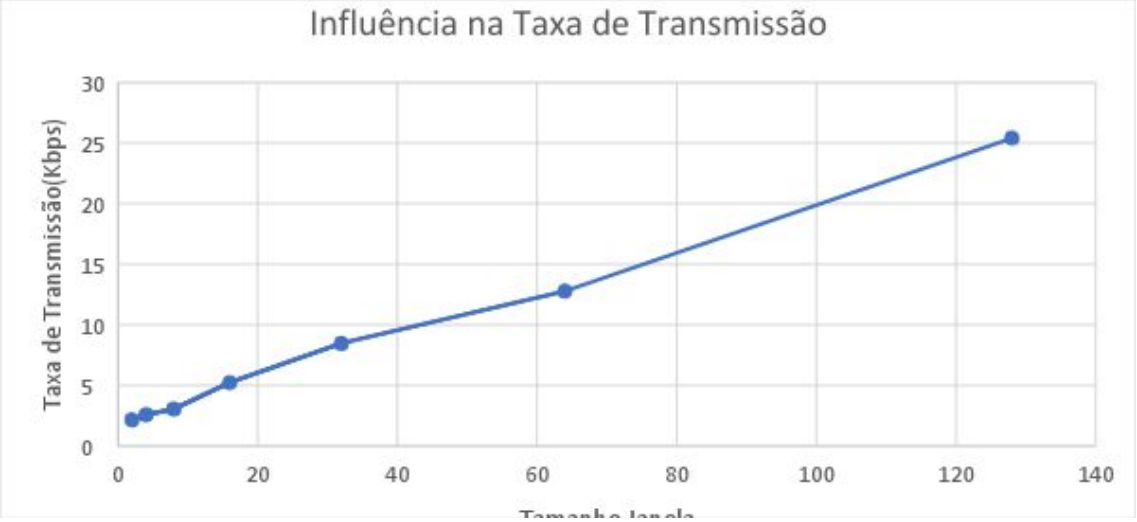
Tamanho Arquivo	Tamanho Buffer	Tamanho Janela	Tempo de Transmissão	Taxa de Transmissão	Bytes Transmitidos
200k	500	2	112,728	2,16	244567
200k	500	4	114,753	2,59	297508
200k	500	8	123,885	3,06	379738
200k	500	16	117,828	5,24	617215
200k	500	32	119,817	8,47	1015046
200k	500	64	128,021	12,79	1638518
200k	500	128	106,494	25,42	2707662
200k	500	256	97,46	34,3	3343487
200k	500	500	114,87	44,68	5132202

Buffer com tamanho de 500 Bytes:

Aumentando o tamanho do buffer para 500 bytes percebemos uma melhora significativa no tempo necessário para transmitir o mesmo arquivo de 200 Kbytes. O tempo chegou a ser 5 vezes menor que no caso do buffer com tamanho limitado a 100 bytes. Contudo foi observado que o aumento do tamanho da janela deslizante reduzia o tempo de transmissão até ela ser aproximadamente metade do tamanho do buffer. Isso acontece porque a quantidade de bytes descartados depois disso começou a aumentar demasiadamente. Então mesmo para uma taxa de transmissão maior o tempo para transmissão completa do arquivo volta a crescer.

Vejamos graficamente abaixo os resultados obtidos:




Influência na Taxa de Transmissão
Taxa de Transmissão(Kbps)
30
25
20
15
10
5
0
0
20
40
60
80
100
120
140


Influência Volume de Dados Transferidos
Qtd de Bytes Transmitidos
3000000
2500000
2000000
1500000
1000000
500000
0
0
20
40
60
80
100
120
140
Tamanho Janela

Buffer com tamanho de 1000 Bytes:

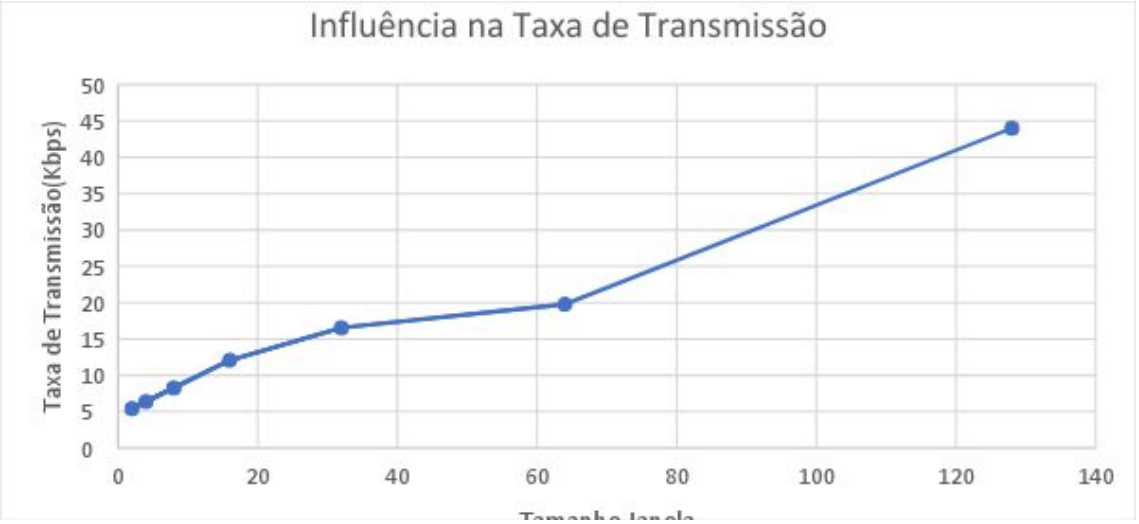
Tamanho Arquivo	Tamanho Buffer	Tamanho Janela	Tempo de Transmissão	Taxa de Transmissão	Bytes Transmitidos
200k	1000	2	43,073	5,43	234198
200k	1000	4	41,986	6,41	269686
200k	1000	8	39,944	8,26	330532
200k	1000	16	37,891	12,09	458297
200k	1000	32	50,17	16,54	830423
200k	1000	64	76,837	19,78	1520937
200k	1000	128	48,163	44	2119222
200k	1000	256	56,325	36,98	2083727
200k	1000	512	68,594	43,65	2994288
200k	1000	1000	50,612	35,56	1800828

Aumentando o tamanho do buffer para 1000 bytes percebemos uma melhora significativa no tempo necessário para transmitir o mesmo arquivo de 200Kbytes. O tempo chegou a ser 3 vezes menor que no caso do buffer com tamanho limitado a 500 bytes. Contudo com esse maior tamanho de buffer a variação do tamanho da janela não resultou em uma mudança linear no tempo de transmissão do arquivo.

A janela com tamanho de 16 bytes apresentou o melhor desempenho no teste executado, após isso o aumento desse parâmetro chega a causar uma piora no tempo para transmissão do arquivo. Além disso a quantidade de bytes enviados cresce cada vez mais congestionando cada vez mais a rede.

Nessa configuração dos parâmetros o sistema se mostrou extremamente sensível quanto a qualidade da rede para obter um bom desempenho.

Podemos ver mais facilmente esses resultados nos gráficos apresentados abaixo:


Influência na Taxa de Transmissão
Taxa de Transmissão(Kbps)
50
45
40
35
30
25
20
15
10
5
0
0
20
40
60
80
100
120
140


Influência Volume de Dados Transferidos
Qtd de Bytes Transmitidos
2500000
2000000
1500000
1000000
500000
0
0
20
40
60
80
100
120
140
Tamanho Janela

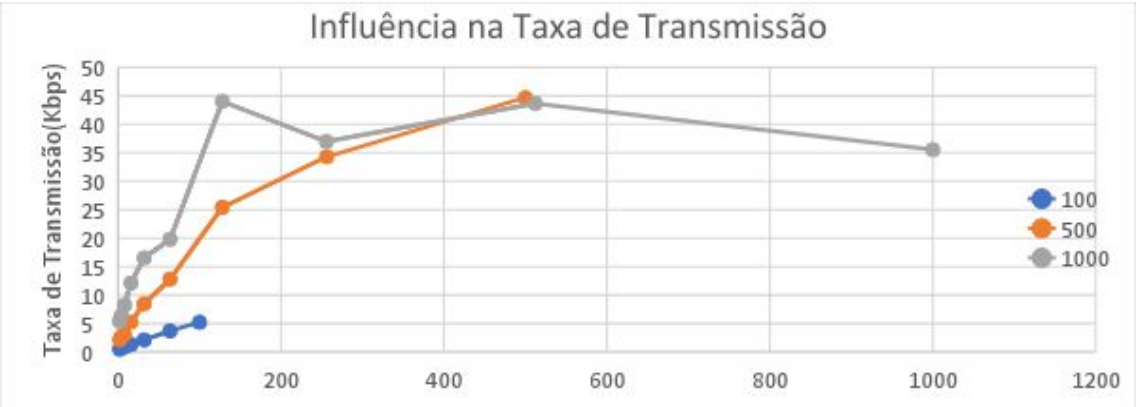
Análise

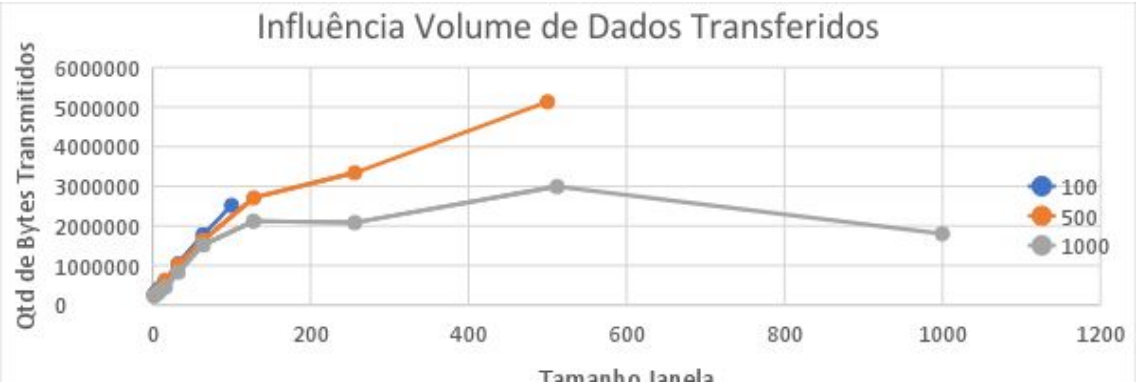
Os resultados apresentados nos experimentos são condizentes com os teóricos. Foram variados tanto o tamanho do buffer quanto o tamanho da janela de transmissão.

O aumento do tamanho do buffer gera um ganho extremamente significativo na questão do tempo necessário para se transmitir o arquivo, porém, devido a casos de perda e danificação dos pacotes enviados, o tamanho dos dados a serem transmitidos crescem significativamente. Logo geram uma rede mais congestionada.

Quanto ao tamanho da janela de transmissão também foi observado um ganho de desempenho, porém quanto maior o tamanho desse parâmetro maior será a quantidades de dados que podem ter de ser retransmitidos em caso de problemas no envio do pacote.

Portanto percebemos que um para buffers maiores em geral é ideal uma janela maior, enquanto para buffers menores pode ser viável o tamanho da janela ser maior. Tomando sempre o devido cuidado para não congestionar excessivamente a rede com pacotes de dados gigantescos.





Conclusão

Com esse trabalho foi possível aprender um pouco mais sobre o protocolo UDP e sobre a técnica de Janela Deslizante. Quando o protocolo de comunicação utilizado como base não garante a entrega dos pacotes e a ordem correta é necessário que a aplicação faça esse trabalho para garantir a transmissão confiável dos dados.

A disponibilidade de diversos materiais para consulta gratuita na internet e a discussão de questões técnicas entre os alunos da disciplina através da plataforma Moodle ajudaram de forma considerável na superação dos desafios encontrados, viabilizando assim o desenvolvimento de um trabalho à primeira vista bastante complicado.

Os parâmetros de tamanho do buffer e tamanho da janela quando alterados implicam na mudança de desempenho na transmissão do arquivo. Através dos inúmeros testes foi possível

concluir que devemos buscar uma relação entre essas duas variáveis para podermos otimizar o sistema. Essa relação pode ser afetada por vários fatores externos como a qualidade da rede, fluxo de dados, taxa de transmissão que o hardware suporta, etc e, dependendo dessas condições, podemos aumentar um parâmetro ou outro.

Contudo o trabalho agregou aos alunos conhecimentos extremamente valiosos tanto para a Academia quanto para o Mercado de Trabalho.

Referências

[1] Calculating Sum of Bytes (checksum) for char []
http://forums.codeguru.com/showthread.php?293045-Calculating-Sum-of-bytes-(checksum)-for-char

Acessado em 20/11/2017

[2] KUROSE, J.; ROSS, K. Computer Networking A Top Down Approach: 6. ed. New Jersey: Editora Pearson, 2013.

[3] Protocolo de janelas deslizantes

https://pt.wikipedia.org/wiki/Protocolo_de_janelas_deslizantes

Acessado em 20/11/2017