UNIVERSIDADE FEREDERAL DE MINAS GERAIS

Trabalho prático 2

Redes de Computadores

Elise Guimarães de Araújo

7/6/2015

Relatório do segundo trabalho prático da disciplina Redes de Computadores, da Universidade Federal de Minas Gerais, lecionada pelo professor Luiz Filipe Menezes Vieira. Prática destinada a exercitar o uso de sockets UDP.

Introdução

Esse relatório discorre acerca da implementação de um par de programas operando conforme arquitetura cliente-servidor, exercendo comunicação uni e bidirecional sobre o protocolo de transporte User Datagram Protocol (UDP), cuja escolha já indica a demanda por desenvolvimento também de periféricos dedicados a garantir integridade, confiabilidade e disponibilidade ao funcionamento dos programas.

Implementação

Os códigos foram inteiramente desenvolvidos em linguagem C, utilizando-se o módulo de encapsulamento fornecido para a aplicação dos sockets e das funções associadas.

O cabeçalho desenvolvido para complementar o UDP é composto de 64 bits: 32 dedicados à detecção de erro, feita através de uma função similar ao checksum, mas mais simples (simplesmente somam-se todos os bytes do buffer enviado); e os demais 32 dedicados ao número de sequência do pacote (os números de sequência possíveis são o dobro do tamanho da janela).

A janela deslizante foi implementada paralelamente ao sistema de armazenamento de pacotes, de forma que as mensagens armazenadas em determinado instante durante o processamento são somente aquelas mantidas na janela. Arquitetou-se a janela como uma fila, objetivando-se explorar a funcionalidade FIFO dessa estrutura de dados.

O servidor executa a seguinte iteração: verifica se há espaço para novos envios na janela através da função windowFull; lê um buffer do arquivo com fread; calcula checksum de detecção de erro; cria cabeçalho UDP de 64 bits (checkResult + seqNumber); envia o pacote composto do cabeçalho e do buffer; armazena tudo no buffer associado à janela com a função windowInsert; incrementa o número de sequência, que indica o próximo pacote a ser enviado; incrementa a contagem de bytes para posterior medição; recebe possíveis acks (64 bits), com um timeout de 1s; caso o timeout se exceda, reenvia todos os pacotes não confirmados; indica, no array de structs que representa o buffer associado à janela, os pacotes que foram confirmados; e desliza a janela, se preciso, com a função removeAckds, que remove o pacote da saída da janela caso este já tenha sido confirmado.

O cliente, por sua vez, primeiramente guarda espaço no buffer associado à janela janela para elementos com SeqNumber de 0 a tam_janela - 1, com a função windowReserve. Em seguida itera com o seguinte funcionamento: recebe pacote do server; lê pacote e salva o que recebeu em checkResult, SeqNumber e buffer; faz detecção de erro com checksum; buferiza pacotes recebidos corretamente; atualiza janela, indicando o que foi recebido, e lastRcvd (último pacote recebido na ordem correta); desliza janela, se possível: removeRcvds - remove da janela os pacotes já recebidos em ordem - e windowReserve - reserva espaço para novos pacotes; e envia Ack com número de lastRcvd.

O fim do arquivo é indicado pelo envio da string “fim” pelo receptor. Não houve preocupação com padding, uma vez que é extremamente improvável que se repita essa exata conjuntura.

A temporização se deu através da opção de temporização do socket, seguida da verificação do erro associado à função recvfrom, de forma a, sempre que o erro no recebimento for devido a timeout, reenviam-se todos os pacotes não confirmados.

Metodologia

Os programas foram desenvolvidos, compilados e executados no sistema operacional Ubuntu versão 12.04 (Precise Pangolin). As especificações do sistema relevantes são: processador Intel Core i7-4800MQ 2.70GHz, 8GB de RAM, adaptadores de rede Killer Wireless-N 1202 e Realtek PCIe Gigabit Ethernet. Os testes foram realizados em uma única máquina.

A medição do tempo de execução utilizado no cálculo do throughput foi feita com a função gettimeofday em dois pontos chave do programa considerados viáveis para se ter como início e fim. Os experimentos tiveram como parâmetros o tamanho da janela e o tamanho do buffer, e cada medição é resultado da média de três amostras. Cada teste reportado corresponde a uma execução individual - não a um loop ao redor do programa para fazer tudo um certo número de vezes.

Resultados

A tabela a seguir apresenta os resultados obtidos e as respectivas médias para valores iguais de tamanho de janela e tamanho de buffer.

Só foram feitas medições com tamanhos de buffer 100 e 500, uma vez que, por algum motivo não identificado, a operação falha para buffers superiores a 558 bytes na máquina utilizada. Um teste foi executado em uma segunda máquina com configurações máximas, apenas para certificação de êxito, e o programa funcionou normalmente, mas não foram coletadas amostras dessa segunda máquina.

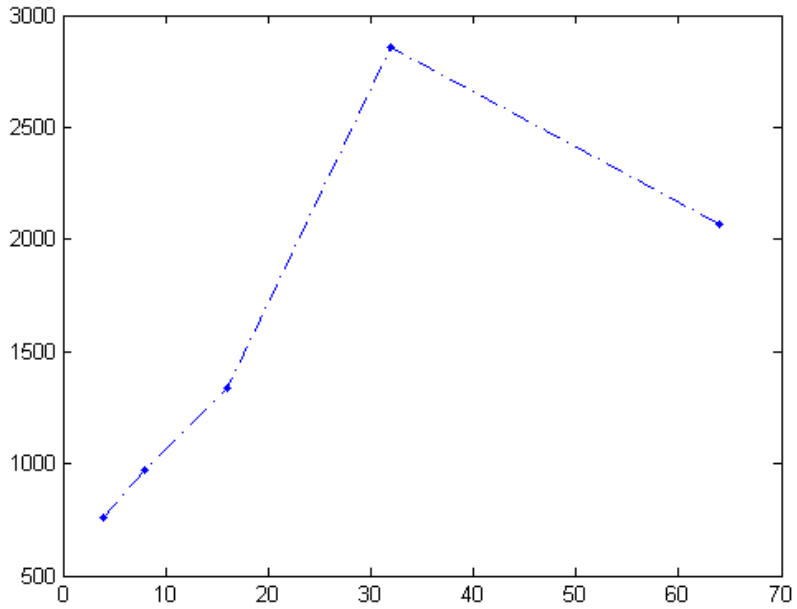
Tamanho do arquivo (bytes)	**Tamanho do buffer (bytes)**	**Tamanho da janela**	**Throughput (kbps)**	**Data + headers (bytes)**	**Tempo (s)**	**Throughput médio (kbps)**
10619	100	4	748.48	11472	0.015327	761.62
10619	100	4	733.97	11472	0.01563	
10619	100	4	802.41	11472	0.014297	
10619	100	8	985.06	11472	0.011646	968.4867
10619	100	8	915.49	11472	0.012531	
10619	100	8	1004.91	11472	0.011416	
10619	100	16	1206.31	11472	0.00951	1336.99
10619	100	16	1504.72	11472	0.007624	
10619	100	16	1299.94	11472	0.008825	
10619	100	32	2462.86	11472	0.004658	2857.943

10619	100	32	2478.29	11472	0.004629	
10619	100	32	3632.68	11472	0.003158	
10619	100	64	1667.93	11472	0.006878	2065.693
10619	100	64	1498.24	11472	0.007657	
10619	100	64	3030.91	11472	0.003785	
10619	100	64	1255.83	11472	0.009135	
10619	500	4	4378.09	10792	0.002465	4378.09
10619	500	8	4677.94	10792	0.002307	4677.94
10619	500	16	2511.52	10792	0.004297	3847.5
10619	500	16	5183.48	10792	0.002082	
10619	500	32	4737.49	10792	0.002278	4970.11
10619	500	32	5143.95	10792	0.002098	
10619	500	32	5028.89	10792	0.002146	
10619	500	64	13065.38	10792	0.000826	8391.485
10619	500	64	14218.71	10792	0.000759	
10619	500	64	5390.61	10792	0.002002	
10619	500	64	6922.39	10792	0.001559	
10619	500	64	1713.29	10792	0.006299	
10619	500	64	9038.53	10792	0.001194	
10619	500	128	4428.4	10792	0.002437	8275.758
10619	500	128	12017.82	10792	0.000898	
10619	500	128	8307.93	10792	0.001299	
10619	500	128	9161.29	10792	0.001178	
10619	500	128	7463.35	10792	0.001446	

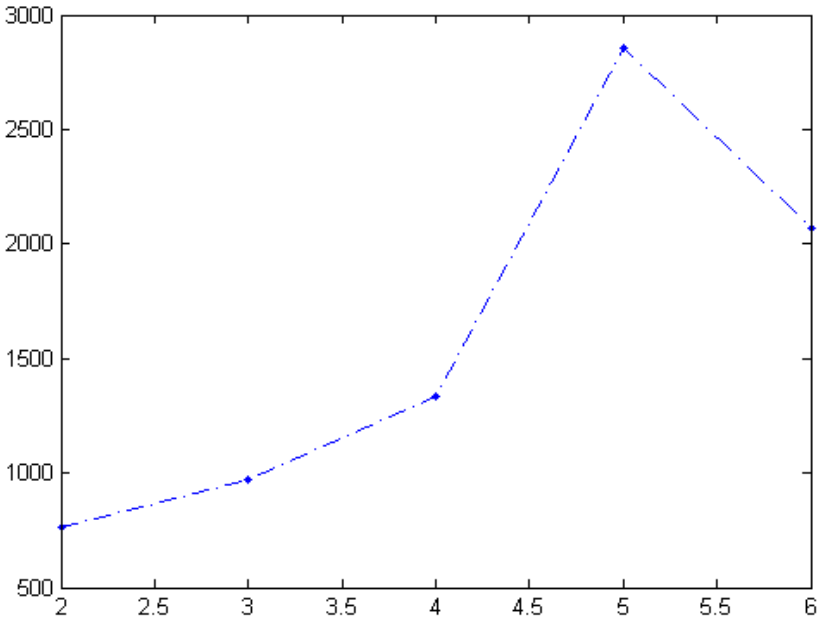
Representação gráfica das médias obtidas:

Buffer de 100 bytes: melhores resultados obtidos para janela de tamanho 64.

Escala decimal

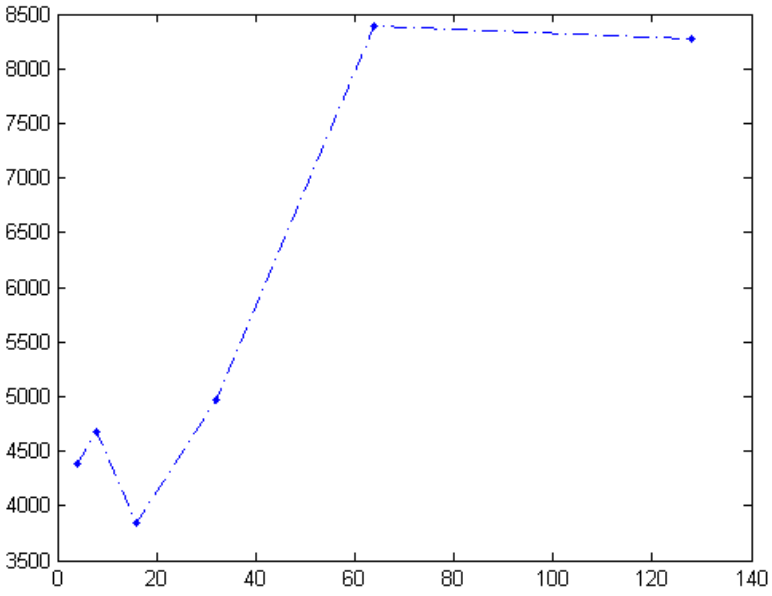


Escala logarítmica

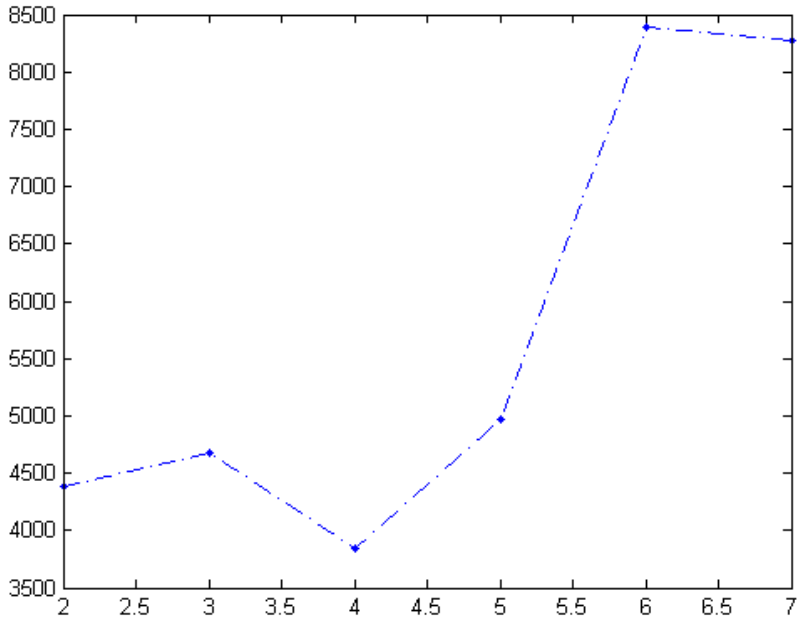


Buffer de 500 bytes: melhores resultados também obtidos para janela de tamanho 64.

Escala decimal



Escala logarítmica



Análise

Conforme esperado, as curvas apresentam valores crescentes de vazão conforme aumenta-se o tamanho da janela, até atingir um máximo - o que corresponde ao preenchimento do "cano" de vazão, com aproveitamento ótimo da largura de banda para um certo tempo de propagação. O ideal é que o servidor não fique ocioso esperando por acks, injetando sempre o máximo possível de pacotes na rede. Algumas amostragens apresentaram pontos bastante fora da curva, o que pode se justificar pelo fato de que os testes foram executados na mesma máquina, ambiente não ideal para um par de programas cujo funcionamento depende em grande parte do intervalo de propagação - e portanto da distância - entre as máquinas enviando e recebendo. O buffer de 500 bytes deveria apresentar melhores resultados para um tamanho de janela menor que o buffer de 100 bytes, o que não ocorreu.

Conclusão

Esse relatório apresentou os detalhes da implementação e resultados estatísticos de um par de programas cliente-servidor para transmissão de arquivos, cujo ponto chave foi a criação de um protocolo confiável com janela deslizante sobre o UDP, o que possibilitou ampla compreensão do funcionamento da janela, da necessidade por segurança em geral em protocolos de transporte e da organização das camadas de aplicação, transporte e rede, já que complementamos um protocolo de transporte por meio da aplicação.

Houve ainda aprendizado acerca de métodos de serialização e de temporização, de detecção de erro - ainda que a detecção implementada tenha sido um tanto rudimentar - de aritmética modular - estudaram-se alguns métodos de criptografia que a utilizam - de estruturas de dados consolidadas, da linguagem C e de programação em geral.