Trabalho Prático 2 – Redes de Computadores

Pedro Araujo Pires

Protoclo UDP

Para trocar informações entre computadores, além de estarem conectados, eles precisam entender o que representam os dados enviados através da rede. Este é o papel dos protocolos de rede. A Internet foi projetada através de camadas de protocolos. Deste modo os desenvolvedores teriam liberdade para criar e mudar as formas de comunicação protocolo UDP - User Datagram Protocol - fica na camada de transporte, como visto na figura. Isto significa que ele é responsável por passar uma mensagem vinda da rede para uma aplicação um nível acima. O UDP utiliza a forma de datagramas para enviar mensagens e não garante a confiabilidade do canal. Desta forma a aplicação fica responsável pelo tratamento de possíveis erros na transmissão de pacotes entre o cliente e o servidor.

Janela Deslizante

Existem várias formas de estabelecer a confiabilidade da rede, que não é fornecida pelo protocolo UDP. Uma delas é o método Stop-and-wait. Basicamente, o transmissor envia cada pacote e espera sua confirmação para então poder enviar o próximo. A desvantagem deste método é que o canal permanece ocioso enquanto ele funciona com base na premissa de que antes de enviar um novo pacote, o transmissor deve ter certeza que o receptor recebeu o pacote anterior. Uma forma de permitir que o canal não fique ocioso é através da utilização de janelas deslizantes. Para isso são criados buffers para armazenar temporariamente os pacotes enviados e recebidos. Assim pode-se enviar pacotes simultaneamente.

O objetivo deste trabalho foi desenvolver uma aplicação no modelo cliente-servidor, em cima do protocolo UDP, utilizando o protocolo de janela deslizante Go-Back-N (janela do receptor tem tamanho de um pacote). O cliente envia ao servidor o nome do arquivo que deseja, e recebe este arquivo. Para isto deveria ser usada a biblioteca {\it sockets} do Unix.

Implementação

Para a temporização foi utilizado um tempo de *timeout* de 1 segundo, conforme sugerido na especificação do trabalho. Caso não chegue a confirmação do pacote detro desse tempo, o mesmo é retransmitido.

São quatro tipos de mensagens previstas para o protocolo implementado: mensagens de dados com cabeçalho 'MORE', indicando que ainda exitem mais pacotes a serem enviados, e com cabeçalho 'END', indicando que a transmissão do arquivo terminou; mensagens de confirmação, com cabeçalhos ACK, que indica que o pacote foi recebido em problemas, e e FAIL, que indica que houve um erro na transmissão. Entretanto, adotou-se um formato único de mensagem a ser usado nesse protocolo, cuja composição do cabeçalho é dada a seguir:

- Byte 1: Tipo da mensagem (ACK, FAIL, MORE ou END)
- Byte 2: Número de sequência do pacote
- Último Byte: Checksum

Detecção de Erros

O mecanismo de detecção de erros é baseado na temporização e na checagem do *checksum* pelo cliente.

Temporização

Ao determinar um tempo de *timeout* para os pacotes, caso não chegue a confirmação de um pacote esperado (o pacote necessário para deslizar a janela), ele é enviado novamente. Caso a temporização ocorra devido a uma perda do ACK, o cliente simplesmente reenvia o ACK quando o pacote chega novamente, e o descarta.

Caso o pacote recebido não seja o esperado, o cliente o armazena na posição correspondente da janela para posterior processamento. Esse comportamento corresponde ao algoritmo *selective repeat*.

Checksum

Um *checksum* é anexado ao final do pacote antes do envio. Quando um pacote chega no cliente, este faz a checagem do *checksum*, e caso exista algum erro, o pacote é descartado e é enviado um pacote de confirmação que indica esse erro. Quando o servidor recebe esse cabeçalho, o pacote é reenviado independentemente do timeout.

Metodologia

Ambiente de testes

Os testes feitos foram realizados com o cliente executando numa máquina com as seguintes configurações:

- Processador: Intel(R) Core(TM)2 CPU T7400 @ 2.16GHz
- Cache: 32KB(L1) e 4MB(L2)
- Memória: 2GB
- Kernel: 2.6.32-41-generic

O servidor foi executado na máquina jaguar, do DCC. A conexão à rede foi feita através da rede *wireless* UFMG, com o acesso à máquina jaguar sendo feito através da VPN do DCC.

Execução

Cada teste foi executado quatro vezes, e os valores apresentados abaixo são a média dos resultados dos testes. Após cada teste, o arquivo fo verificado com o programa *md5sum*.

Resultados

A tabela abaixo contém as velocidades obtidas nos testes para diferentes tamanhos de arquivos e de janelas.

	100 bytes	1000 bytes	4000 bytes	8000 bytes
2	0.04	1.53	3.60	31.68
4	0.01	2.34	4.31	58.80
8	0.34	8.31	5.05	97.07
16	0.58	2.64	4.0	140.0
32	-	-	5.70	157.67

Foi observado que os resultados obtidos atenderam às nossas espectativas nos casos testados. A velocidade é máxima a partir do momento em que a janela é grande o suficiente para guardar a mensagem inteira.

Conclusão

O protocolo da janela deslizante apresenta um maior desempenho na rede, pois consegue enviar mais pacotes em um intervalo de tempo menor. O protocolo Stop-and-Wait não consegue o mesmo desempenho pelo fato de esperar por cada confirmação do pacote que acabou de enviar, só enviando o próximo quando receber a confirmação do atual, o que resulta em baixa taxa de utilização da rede e consequente ineficiência na transmissão. Porém, como a função de recebimento causa uma espera no programa, esse resultado não é visto no tempo de execução.

O trabalho foi de grande importância para conhecer a complexidade na implementação de protocolos de envio de arquivos. A teoria envolvida neste trabalho é bem simples, entretanto a implementação possui vários detalhes de funcionamento que a torna complexa de se codificar.