GABRIEL DE OLIVEIRA PONTAROLO
GRR20203895
RODRIGO SAVIAM SOFFNER
GRR20205092

RELATÓRIO - JANELA DESLIZANTE

Redes de Computadores - Trabalho 3

Relatório apresentado à disciplina Redes de Computadores, do curso de Bacharelado em Ciência da Computação da Universidade Federal do Paraná – UFPR – como requisito parcial para aprovação na disciplina, sob a orientação do prof. Dr. Eduardo Todt.

CURITIBA

2023

SUMÁRIO

1. INTRODUÇÃO

O relatório a seguir descreve brevemente a implementação de uma janela deslizante seletiva para envio de mensagens em uma rede de computadores, junto com estruturas e algoritmos para seu funcionamento. O projeto está implementado em uma versão não finalizada do trabalho 1, utilizando a linguagem C++, o principal algoritmo se encontra no arquivo “githyanki.cpp”.

2. JANELA

Foi implementado um algoritmo de janela deslizante seletiva. Nossa implementação utiliza janelas de 16 de tamanho, pois o frame implementado no trabalho 1 possui 4 bits para indicar o número da sequência. Para o uso mais eficiente o número máximo de frames enviados por vez é a metade do tamanho máximo da janela, no nosso caso 8.

Dentro da implementação, as janelas do client e do server possuem uma janela física em forma de vetor, que tem apenas o tamanho da janela de envio, que no caso do nosso protocolo tem o tamanho 8. Então, sempre que for mencionado a janela dentro dos algoritmos, essa janela apenas possui tamanho 8, sendo assim apenas o envio e o recebimento pode ser de 8 frames, assim removendo a possibilidade de receber e enviar frames de outra janela de envio.

3. ACKNOWLEDGE

3.1 RECEBIMENTO DE ACKNOWLEDGE

Apresenta dois algoritmos similares, um para o server e outro para o client, com a função de aplicar as instruções de ACK e NACK, junto com o seu número da sequência. As duas instruções confirmam o recebimento de todos os frames anteriores, frames que possuem uma sequência anterior a do ACK ou NACK recebido, possuindo uma diferença onde o ACK também reconhece a si próprio.

O acknowledge é importante tanto para o server quanto para o client, porém os dois possuem funções diferentes.

O server receberá o acknowledge a partir da rede e possui a função para atualizar a sua janela de envio, removendo os frames que foram confirmados, e atualizando sua janela com os próximos frames que devem ser enviados.

O client processa o acknowledge da mesma forma que o server, removendo os frames confirmados e adicionando na janela os próximos frames que serão aceitos.

Tudo é direcionado utilizando o número da sequência do frame.

3.2 ENVIO DE ACKNOWLEDGE

O envio de acknowledge apenas acontece do client para o server. O client escolhe dar ack/nack quando sua janela de recebimento estiver completa ou tiver acontecido um timeout no seu socket. Apesar de o client não receber ack, ele processa a mesma instrução que foi enviada para o server, com o objetivo de manter uma sincronização de janelas entre o client e o server. Manter as janelas do client e do server iguais é essencial para o melhor funcionamento da janela.

4. TIMEOUTS

Timeouts são utilizados em diversos locais para que a comunicação não venha a uma parada e para detectar uma possível queda da transmissão. No caso da nossa implementação, a todo timeout o server retransmite todos os frames e caso o client tenha um timeout será enviado para o server um NACK independente se sua janela tenha andado ou não.

5. FRAMES DUPLICADOS

Na implementação feita, o servidor prepara os frames e sabe a quantidade de frames que possuem dados e envia essa quantidade na ordem que está sua janela. O seu acknowledge se responsabiliza em manter a janela com os frames nas

posições corretas em referência a suas sequências, com isso pode enviar o mesmo frame múltiplas vezes causando frames duplicados no lado do client.

No client apenas é aceito frames que estão na sua janela e que ainda não tenham sido recebidos, assim rejeitando frames com sequências não presentes na janela e frames duplicados. Todos os frames aceitos pela janela são marcados como recebidos assim apenas o primeiro frame recebido se uma determinada sequência é válida e duplicatas são rejeitadas.

6. STRUCTS

O algoritmo utiliza de diversas structs, tanto para representar a janela do client como a do server, que possuem implementações diferentes, como diversas outras structs para o auxílio do o seu funcionamento. Principais requisitos dessas structs é salvar variáveis que representam última sequência que foi dada ack, sequência que representa o final da transmissão e representações da janela, sendo a do server apenas um vetor de int que indica as sequências atuais da janela e no client uma struct com dois valores int uma para sequência da janela, igual ao server, e outro int o index da posição do dado daquela determinada frame.

7. FRAMES LOST

Para simular a perda de pacotes que existem no mundo real, foi implementado um random processamento de frames recebidos no lado do client.

8. FINAL DA JANELA

No nosso protocolo usamos um frame com tipo END para o final da transmissão. Junto desse frame na parte de dados é enviado o nome do arquivo, quando for o caso.

O server quando realiza o preparamento dos frames percebe quando todos os dados foram enviados e então cria e adiciona a sua janela um frame de final de transmissão e então espera que o mesmo seja acknowledged pelo client, assim terminando a transmissão.

O client quando receber o frame END irá resgatar o nome do arquivo caso seja o caso e vai aceitar a transmissão terminada quando for enviado o acknowledge do frame final.

A confirmação será feita da mesma forma que as outras confirmações da janela. Client e server sabem a sequência referente ao frame final, assim sabendo que ao confirmá-lo, a partir de um ACK ou NACK, determina a finalização da transmissão. No caso de nossa implementação o server pode não receber o frame de confirmação caso o ACK ou NACK enviado pelo client seja perdido, fazendo com que o server fique em um estado de espera e o client finalizado.