Trabalho Prático 1
PPPSRT, um protocolo de enlace confiável

Arthur Pontes Nader[1]

Rita Rezende Borges de Lima[1]

[1]Universidade Federal de Minas Gerais

arthurnader@dcc.ufmg.br

ritaborgesdelima@dcc.ufmg.br

Abstract. *This report presents the main decisions for implementing a protocol that follows some elements of PPP (Point-to-Point Protocol). To carry out the work, Python language was used*

Resumo. *Este relatório apresenta as principais decisões de projeto tomadas para implementação de um protocolo que segue alguns elementos do PPP (Point-to-Point Protocol). Para realização do trabalho, utilizou-se a linguagem Python.*

1. Introdução

O Point-to-Point Protocol (PPP) é geralmente utilizado para transportar pacotes do Internet Protocol (IP) por diferentes tipos de enlaces ponto a ponto. Esse protocolo possui diversos constituintes em seus quadros, sendo que entre eles se destacam a flag, que indica o começo e o fim de um quadro, a payload, que se refere aos dados a serem transmitidos e o checksum, usado para detecção de erros.

Dessa forma, esse trabalho teve como principal objetivo realizar a implementação de um protocolo de enlace orientado a byte que possui alguns elementos em comum com o PPP. Nesse protocolo, o enquadramento é feito por sentinelas de bytes, sendo que ocorre a retransmissão de mensagens quando essas não são confirmadas.

2. Implementação

Para implementar o protocolo de comunicação, foi necessário a criação dos seguintes métodos:

- Método get_checksum(self, message)

Esse método recebe como parâmetro a mensagem a ser transmitida pelo canal e faz uma iteração sobre ela para calcular o checksum de 16 bits. Assim, a cada 2 bytes da mensagem realiza-se a soma do correspondente inteiro em uma variável inicializada com valor 0 e se retira o módulo 65536 (2^{16}). Se a mensagem tiver um número ímpar de bytes, o último byte é considerado isoladamente para somar ao checksum. Dessa forma, esse método retorna o checksum de 16 bits referente a mensagem de entrada, sendo isso utilizado para detecção de erros durante a transmissão. O checksum calculado é retornado então no formato de 2 bytes.

- Método assemble_the_frame(self, processed_frame, corrupt_frame)

Já esse método recebe o quadro e uma variável de estado que indica se ele já está corrompido. Assim, busca-se a flag para identificação do começo do quadro. Em seguida, verifica-se o endereço e o controle. Se algum deles não estiver igual ao esperado, o quadro é marcado como corrompido. O protocolo associado ao quadro é guardado em uma variável que será retornada.

Acha-se então dentre os bytes restantes a flag, que indica o fim do quadro. Dessa maneira, sobra então a mensagem e o checksum, que também serão retornados para posterior tratamento. Outra variável retornada é a que indica se o quadro está corrompido ou não após essas sucessivas verificações.

- Método remove_byte_stuffing(self, processed_frame)

Esse método visa remover o byte stuffing pelo lado do receptor. Ao receber a mensagem e o checksum, realiza-se uma iteração sobre os bytes e, sempre que se encontra uma sequência composta que indique escape e flag ou escape e escape, essa sequência é substituída pelo caractere correspondente. Assim, recupera-se a mensagem original e o checksum associado.

Outros procedimentos foram implementados diretamente em send(self,message) e recv(self). Entre eles destacam-se:

- Byte Stuffing:

Essa parte do código realiza a substituição de bytes correspondentes ao caractere de escape e a flag que ocorrem no payload e no checksum. Isso é feito definindo-se uma variável inicial correspondente ao byte stuffing vazia e iterando-se sobre os bytes da mensagem e do checksum. Ao encontrar um escape ou uma flag, adiciona-se à variável os correspondentes caracteres de substituição. Demais bytes são adicionados à variável diretamente.

Após isso, o quadro é montado com o byte stuffing feito e enviado pelo canal.

- Retransmissão:

A retransmissão é feita após o destinatário não receber uma confirmação do pacote enviado. Assim, utiliza-se de uma chamada recursiva ao método send(self,message) com a mesma mensagem inicial. Isso faz com que, caso ocorra mais de um erro em algum pacote, realize-se mais de uma tentativa de retransmissão da mensagem.

Do lado do recv(self), caso um bloco seja identificado como corrompido, esse método é retornado recursivamente, o que causará o estouro da temporização por parte do transmissor, que então terá que reenviar o bloco.

- Quadros repetidos:

Para identificação de quadros repetidos ou fora de ordem, criaram-se dois atributos para a classe. Esses atributos guardam o valor do protocolo do último quadro enviado e do último quadro recebido. Assim, para se evitar quadros repetidos ou fora de ordem, basta comparar se o protocolo somado de uma unidade é diferente do protocolo do quadro analisado atualmente. Se for diferente, será necessário realizar uma retransmissão.

3. Resultados

Inicialmente, testou-se a execução do programa para pacotes sem inserir erros nos quadros. Nesse caso, o arquivo foi devidamente recebido.

Para se testar a retransmissão de blocos com erros, a biblioteca random do Python foi utilizada. Assim, por meio desta biblioteca pode-se escolher se um quadro vai ser enviado com erro ou não, fazendo com que após diversas tentativas o quadro seja devidamente recebido em algum momento.

Observou-se que houve retransmissão adequada dos pacotes, sendo que ao final do processo o arquivo foi recebido corretamente.

Para simular a retransmissão de quadros repetidos ou fora de ordem, a mesma estratégia acima foi utilizada, só que desta vez escolhe-se um valor para o protocolo correto ou decrescido de uma unidade.

Se for escolhido o protocolo errado, o quadro está repetido ou fora de ordem, sendo que assim haverá um conflito no recv(self), causando a retransmissão do pacote.

Novamente, houve retransmissão do quadro até que o protocolo indicasse que o pacote recebido não estava duplicado ou fora de ordem. Ao final do processo, o arquivo foi recebido corretamente.

4. Conclusão

Os resultados obtidos permitem concluir que a implementação foi bem sucedida, havendo retransmissão adequada dos pacotes corrompidos e descarte de pacotes repetidos.

O desenvolvimento deste trabalho foi uma boa oportunidade de colocar em prática os conceitos e métodos vistos durante a disciplina Redes de Computadores. A sua realização possibilitou uma maior aprendizagem acerca de como realizar a comunicação eficiente e correta entre dois computadores utilizando um protocolo ponto a ponto.

5. Referências

PETERSON, L. L.; DAVIE, B. S. Redes de Computadores - Uma Abordagem de Sistemas. Quinta Edição. Rio de Janeiro. Elsevier Editora Ltda. 2013.