Simuladoção de Rede em Anel
Redes de Computadores

Bruno Mazardo, Thiago Rocha

[1]Faculdade de Informática – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brazil

Resumo. *Este artigo tem como objetivo apresentar e esclarecer a solução encontrada para a atividade proposta, estrutura de dados envolvendo a solução, tecnologias utilizadas e mecanismos de sincronização.*

1. Redes em Anéis

A topologia de rede em anel consiste em uma série de estações de trabalhos conectadas em série formando um circuito fechado. O anel não interliga as estações diretamente, mas sim consiste em uma lógica de repetidores ligados por um meio físico. Redes em anel são capazes de transmitir e receber dados em configuração unidirecional. Como vantagem por sua simplicidade o projeto torna menos sofisticado os protocolos de comunicação, assegurando a entrega corretamente dos pacotes em sequência para o destino. Contudo, cada computador conectado a rede é um ponto crítico de falha, uma vez que a rede é circular e unidirecional a queda de um único ponto compromete toda a rede. Como acontece em qualquer topologia, cada estação, ou nó, atende por um endereço que, ao ser reconhecido por uma estação, aceita a mensagem e a trata. Os maiores problemas desta topologia são relativos a sua pouca tolerância a falhas. Qualquer que seja o controle de acesso uitlizado, ele pode ser perdido por problemas de falha e pode ser difícil determinar com certeza se este controle foi perdido ou decidir qual nó deve recriá-lo.

2. Enunciado do Problema

A implementação do trabalho 2 das disciplinas de redes de computadores e laboratórios de redes de computadores consiste em simular na rede local o funcionamento de uma rede em anel. O programa deverá implementar a transmissão de mensagens entre as máquinas que compõem a rede em anel. O programa deve possuir dois tipos de mensagens, token e dados. Todas ás máquinas devem possuir uma fila para gerenciamento das mensagens de dados. Uma máquina só poderá enviar uma mensagem for possuidor do token. Ao enviar uma mensagem o usuário deverá identificar o usuário através do apelido.

2.1. Inicialização da Rede

Ao iniciar o programa, o usuário deverá informar o endereço de IP da máquina que encontra-se diretamente a sua esquerda, um apelido e um tempo do token.

2.2. Estrutura do Pacote de Dados e Token

A estrutura para o envido do token é um código de valor 1234, enquanto o a mensagem de dados deve ser enviada com um sequência numérica de valor 2345 seguida de ';' e da mensagem que será composta pelos campos controle de erro, apelido de origem, apelido de destino e mensagem todos separados pelo caracter ':'. Exemplificando: 2345;controle:origem:destino:mensagem.

2.3. Controle de Erro

Existem três valores para o campo de controle de erro: OK, naocopiado e erro.

2.4. Lógica de Funcionamento

Se uma máquina iniciar com o token esta deve verificar sua lista de mensagens, caso possui mensagens deve enviá-la a máquina diretamente a sua direita. Caso á maquina não contenha nada em sua lista esta deve enviar o token para a próxima máquina. Ao receber um mensagem de dados a máquina em questão deve validar se a mensagem é dela mesma e verificar o controle de erro no cabeçalho. Caso a mensagem possui 'erro' está deve tentar enviar novamente a mensagem para a rede, caso já tenha realizado este processo deve desistir de enviar a mensagem e enviar o token para a máquina seguinte. Em caso do controle de erro possuir OK ou naocopiado esta deve apenas enviar o token para a próxima máquina. Em caso da origem não ser a mesma máquina abrindo a mensagem esta deve verificar se o destino da mensagem é para a máquina em questão, caso seja é necessário mostrar a mensagem enviada e sua origem, também deve-se modificar o cabeçalho de controle para erro ou OK e devolve-la a rede. Caso a mensagem não seja para a máquina esta deve apenas reenviar a mensagem para a máquina logo a sua direita.

3. Tecnologia e Linguagem

Para a realização do projeto foi utiliza a plataforma NodeJS. Esta plataforma é um interpretador de código JavaScript que funciona do lado do servidor/desktop e seu objetivo é ajudar programadores na criação de aplicações de alta escalabilidade. O Node.js é baseado no interpretador V8 JavaScript Engine (interpretador de JavaScript open source implementado pelo Google em C++ e utilizado pelo Chrome). O Node.js utliza programação orientada a eventos, sendo assim os desenvolvedores podem criar aplicações altamente escaláveis sem usar o threading, usando um modelo simplificado de programação orientada a eventos que usa possui funcões de retorno de chamada para sinalizar a conclusão de uma tarefa. Operando em uma única thread e usando chamadas de E / S não bloqueantes, a plataforma suportar dezenas de milhares de conexões concorrentes sem necessitar do uso da troca de contexto de thread. O design de compartilhar uma única thread entre todas as solicitações que usam o padrão de callback destina-se a criar aplicativos altamente concorrentes, onde qualquer função que executa E / S deve usar um retorno de chamada.

4. Estrutura da Solução

A estrutura da aplicação está dividida em cinco arquivos principais. App, Config, Message, Network e Terminal.

4.1. Terminal

O arquivo terminal possui a implementação básica para entrada de dados através do console apresentado para o usuário. Dentro de sua criação é possivel setar uma lista para a salvar um novo padrão de mensagem na fila de mensagens do nó da rede.

4.2. Config

O arquivo de Config exporta uma classe para ler e setar as variáveis de configuração da aplicação, dentre elas estão nome da máquina, ip e porta para a máquina logo a direita na rede, se possui token e tempo do token.

4.3. Message

O arquivo de Message possui uma classe para mapear as mensagens recebidas através da rede. Possuindo o construtor de um parametro, sendo ele uma string, a classe separa e constrói os dados da mensagem. Esta classe também possui métodos para verificar se a mensagem é do usuário, veio do usuário e manipulações em geral da mensagem. Sempre que uma mensagem é enviada pela rede é possível utilizar o método toString da classe para criar uma mensagem pré-formatada.

4.4. Network

Classe responsável por manipular e aplicar a lógica da aplicação. Está classe possui um datagram baseado no protocolo UDP4 e dois métodos principais. O primeiro deles é o appLogic, que possui todas as regras citadas no enunciado do trabalho. E o segundo método principal é o run que possui a capacidade de iniciar o datagram e fazer um bind do mesmo com uma porta.

4.5. App

A classe app é apenas um ponto de junção de todas as classes anteriores, criando todas as classes necesarias e iniciando a aplicação.

5. Fluxo de Execução

A aplicação tem como ponto de partida a classe App. Seu construtor declara e constrói três atributos, config, terminal e network. Após a criação da classe App o objeto instanciado pode iniciar a aplicação através do método run. Neste método o atributo network é configurado passando como parametro o objeto de config e uma referência para a lista de mensagens. Logo após a configuração a rede é inicialiada. Nela é vínculada uma ação ao evento de receber uma mensagem, esta ação é o método appLogic, toda vez que o servidor receber uma mensagem este faz a chamada do método appLogic. Após a configuração e inicialização da rede o atributo terminal é configurado com uma referência para a lista de mensagem e com o nome do usuário desta estação.