

Implementação de uma rede *Token Ring*

Antonio Carlos Salzvedel Furtado Junior e Tiago Rodrigo Kepe
GRR20080946,GRR20084630

17 de novembro de 2010

Universidade Federal do Paraná
CI058 Redes I Bacharelado em Ciência da Computação
For: Luiz Carlos Pessoa Albini

1 Implementação

O nosso programa responsável pela rede se chama tokenring. Ele é iniciado com dois parâmetros obrigatórios. O primeiro deles é o nome da próxima máquina contida no anel, o segundo é a porta usada, que deve ser a mesma para toda a rede.

Com base nos parâmetros, nosso programa cria um servidor e um cliente. O servidor é responsável por escutar mensagens na porta especificada. O cliente é responsável tanto por repassar as mensagens escutadas pelo servidor, como enviar suas próprias mensagens.

Para que pudessemos ler mensagens produzidas por uma determinada máquina ao mesmo tempo em que repassávamos mensagens da rede, tais como o bastão, decidimos usar diferentes threads adicionais para realizar este trabalho. Então usamos a biblioteca PThreads e baseamos nosso trabalho no esquema produtor-consumidor.

1.1 Descrição das Threads

Ao todo, usamos três threads em nossa aplicação, descreverei brevemente a utilidade de cada uma.

A primeira delas é a `insert_buffer`, ela é a thread produtora. Ela apenas lê linhas da entrada padrão e cada linha é colocada em uma entrada de um *bounded buffer*. Cada linha deve ter no máximo `MAX_LINE` caracteres e o buffer tem `MAX_BUFF_SIZE` espaços.

A segunda thread é a `remove_buffer`. Ela possui um conjunto maior de tarefas, além de ser a thread consumidora. Ela deve escutar mensagens na rede, e caso ela tenha recebido o bastão ela deve verificar se há

algo a ser consumido do buffer. Ela deve mandar o bastão imediatamente se não houver nada no buffer, ou mandar a mensagem caso contrário. Ela deve também retirar as mensagens que mandou, ela deve também reconstruir o bastão caso o tenha perdido, ou seja, o timeout estourou e ela não recebeu o bastão. Uma outra tarefa dela é criar o bastão inicial caso seja designada a fazê-lo.

A última thread é o waitTimeout. Ao contrário das outras threads, ela só é iniciada quando o bastão é criado ou quando o bastão é recebido pela primeira vez pelo processo. Sua única função é incrementar uma variável chamada time. Esta variável é compartilhada entre esta thread e a remove_buffer, caso esta variável seja maior que uma constante TIMEOUT, remove_buffer deve começar o processo para recriar o bastão. Cabe a remove_buffer zerar este time quando receber o bastão. As operações sobre esta variável são protegidas.