SISTEMA MULTITALK

Lúcia Maria Oliveira Marcelo Migueletto de Andrade
Osvaldo S. F. Carvalho

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Caixa Postal 702, CEP 30161 Belo Horizonte - MG
Tel.: (031) 443-4088 - Fax: (031) 443-4352
lucia@dcc.ufmg.br - miguel@dcc.ufmg.br *

Sumário

Este artigo descreve o sistema **Multitalk**, *que permite a conversação simultânea entre vários usuários de uma rede de computadores. A implementação foi feita utilizando funções da interface de transporte (TLI) do sistema Uniz, através da qual é possível criar aplicações independentes de protocolo e características físicas da rede. Apresentamos aqui os principais recursos do sistema, bem como os aspectos básicos de sua implementação.*

Abstract

This paper describes the **Multitalk** *system, which allows simultaneous conversation among users of a computer network. Its implementation was carried out using functions of the Unix transport level interface (TLI), by which it is possible to create applications that are independent of protocol and network physical characteristics. We present here the main features of the system, besides the basic aspects of its implementation.*

1 Introdução

O sistema **Multitalk** permite estabelecer diálogos simultâneos entre vários usuários de uma rede. Tais diálogos são organizados sob a forma de sessões. Cada sessão pode ser referente, por exemplo, a um assunto. Os usuários podem criar sessões livremente, as quais podem ou não ter senha, de forma a controlar o acesso às mesmas. Existem, ainda, recursos que possibilitam que o criador e os usuários de uma sessão permaneçam anônimos, garantindo total privacidade.

A implementação foi feita utilizando funções da interface da camada de transporte do Unix *System V* (TLI), através da qual é possível se abstrair de características físicas da rede, o que leva a implementações independentes de protocolo. A interface com o usuário foi construída visando a possibilidade de se usar o sistema a partir de qualquer terminal *tty*, desde que corretamente configurado.

*Este trabalho conta com o apoio da Telebrás – contrato no. 416/91 e do CNPq.

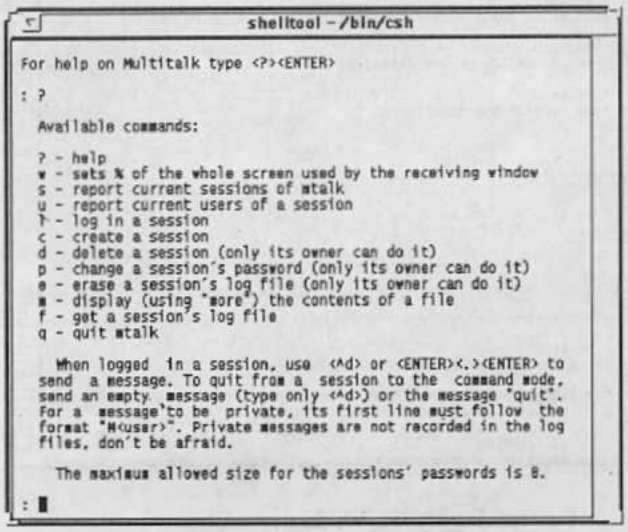

Figura 1: *Help* do sistema **Multitalk**

Quanto às aplicações, o sistema é útil, por exemplo, para a realização de reuniões e manutenção de grupos de debate onde os membros estejam fisicamente distantes uns dos outros. As sessões podem opcionalmente ser gravadas, proporcionando o registro de tudo o que nelas for dito.

2 Descrição do Sistema

Após entrar no sistema **Multitalk**, o usuário se encontra no modo de comando, caracterizado pelo *prompt* ":". Teclando "*?*", é exibida a lista dos comandos disponíveis, como mostra a figura 1.

Segue uma breve descrição de cada um desses comandos.

1. *w - (window)*

   Este comando estabelece o tamanho relativo das janelas de envio e recepção de mensagens. O valor *default* é 75% da janela total destinada à recepção de mensagens e 25% para a de envio.

2. *s - (sessions)*

   O comando *s* permite saber quais são as sessões existentes, o número de usuários em cada uma delas e os proprietários das mesmas (usuários que as criaram). É mostrado ainda, para cada sessão, se há necessidade ou não de senha para se ter acesso a ela e se possui ou não um arquivo de *log*, para o registro da conversação.

3. *u - (users)*

   Através do comando *u* consegue-se saber os nomes dos usuários que estão participando de uma determinada sessão. Tal informação só é acessível aos usuários que possuem a senha da mesma.

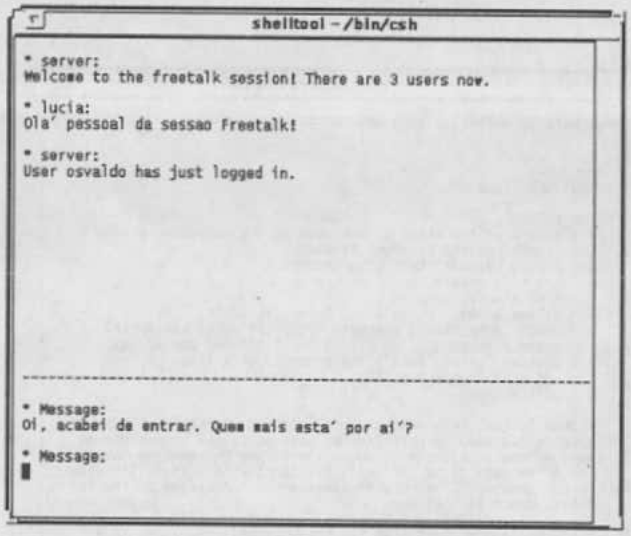

Figura 2: Modo de diálogo

4. *l - (login)*

   Este é o comando que permite que o usuário entre em alguma sessão. Além do nome da sessão, o usuário deve fornecer a senha da mesma, quando for o caso, e o nome pelo qual deseja ser identificado perante os outros usuários da sessão. O sistema, então, sai do modo de comando e entra no chamado modo de diálogo, mostrado na figura 2, onde são enviadas e recebidas as mensagens. Estando no modo de diálogo, o usuário pode compor a sua mensagem, que pode possuir uma ou mais linhas. Para enviá-la, basta digitar *control-d* ou *enter.enter*, assim como no sistema de correio eletrônico do Unix. Para abandonar uma sessão, o usuário deve enviar uma mensagem vazia, ou seja, digitar apenas *control-d*, ou enviar a mensagem "quit". Após abandonar uma sessão, o programa retorna ao modo de comando. Existe, ainda, o recurso de envio de mensagens particulares, cuja primeira linha deve ter o formato *#destinatário*, onde se especifica o nome do único usuário participante da sessão ao qual a mensagem é destinada. O usuário que recebe uma mensagem desse tipo é informado de que ela é de caráter particular. Mensagens com essa característica não são gravadas no arquivo de *log* da sessão, caso a mesma o tenha.

5. *c - (create a session)*

   Usando o comando *c*, sessões podem ser criadas livremente por qualquer usuário do sistema. O usuário que cria uma sessão é considerado o proprietário da mesma. No momento da criação é especificado se a sessão é pública (sem senha) ou não, se possui um arquivo de *log* e se o proprietário deve ser oculto ou não. Se a opção for pela ocultação, o seu nome não é exibido no relatório de sessões (comando *s*).

6. *d - (delete a session)*

   Este comando permite acabar com uma sessão. Só o proprietário de uma sessão pode removê-la, desde que não haja usuários participando dela.

7. *p - (password)*

Com o uso do comando *p*, o proprietário de uma sessão pode alterar sua senha.

8. *e - (erase a log file)*

   Este comando apaga o conteúdo do arquivo de *log* de alguma sessão. O arquivo, no entanto, continua a existir. Esta também é uma operação reservada ao proprietário da sessão.

9. *f - (file)*

   O comando *f* corresponde à solicitação do arquivo de *log* de alguma sessão. Se a sessão possuir senha, apenas os usuários que a conhecem podem ter acesso ao arquivo.

10. *m - (more)*

    Este comando permite ao usuário ver o conteúdo de um arquivo texto qualquer. É útil, por exemplo, para o usuário examinar o conteúdo de um arquivo de *log* que ele tiver buscado.

11. *q - (quit)*

    O comando *q* finaliza o programa **Multitalk**.

3 Aspectos da Implementação

A implementação utiliza a Interface de Transporte do sistema Unix, TLI, que fornece uma interface para a camada de transporte do modelo OSI (*Open Systems Interconnection*). A TLI é uma biblioteca de funções implementada usando-se o mecanismo de STREAMS para E/S. É um enfoque diferente do dos soquetes, que são implementados dentro do núcleo do Unix, e são utilizados através de chamadas de sistema. Referências à TLI e aos soquetes podem ser encontradas em [1] e [2].

A importância da camada de transporte consiste no fato de que ela é o nível mais baixo do modelo OSI que fornece um serviço confiável, com ligação fim-a-fim, utilizado por aplicações e protocolos dos níveis superiores. Esse serviço oculta a topologia e características próprias da rede subjacente, permitindo a implementação de aplicações independentes do meio e do protocolo.

É importante enfatizar que TLI não implementa os serviços oferecidos pela camada de transporte, mas apenas constitui a interface através da qual se tem acesso a esses serviços. Cada canal entre o provedor de transporte e o processo usuário é chamada *transport endpoint* pela TLI. A figura 3 mostra dois processos se comunicando.

O sistema foi implementado em uma rede de estações de trabalho, em ambiente Unix, composta de várias sub-redes físicas, onde é usado o protocolo TCP/IP. Essa família de protocolos provê duas opções equivalentes à camada de transporte do modelo OSI: TCP (*Transmission Control Protocol*), orientado a conexão, e UDP (*User Datagram Protocol*), que utiliza datagramas. O trabalho apresentado utiliza o protocolo TCP, para o qual a interface TLI do sistema operacional utilizado está implementada.

A interface do modo de diálogo foi feita usando-se a biblioteca *curses* do Unix *System V*. Assim, o sistema **Multitalk** pode ser usado em qualquer *shell*, seja dentro de uma interface gráfica ou num terminal de vídeo comum. O código foi todo escrito em C padrão.

4 Estrutura do Sistema

O sistema **Multitalk** é composto por cinco módulos:

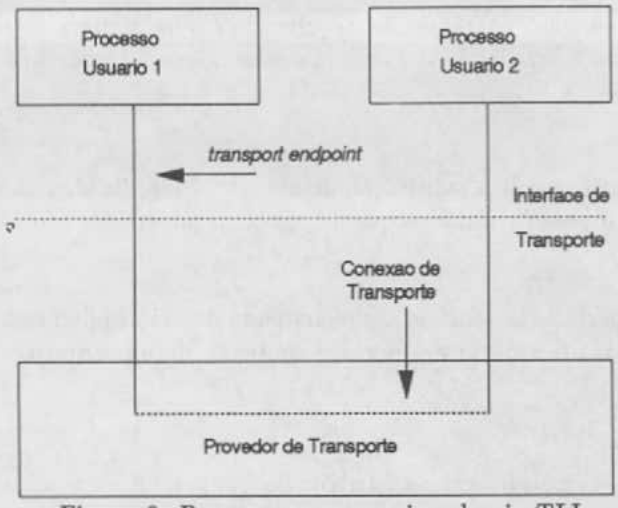

Figura 3: Processos se comunicando via TLI

- servidor de mensagens
- servidor de arquivos
- cliente de mensagens
- cliente mestre de mensagens
- cliente de arquivos

Há uma única instância dos servidores de mensagens e de arquivos. Os servidores ficam residentes numa única máquina, sendo executados em *background*. Nos seus estados normais, estão bloqueados à espera de eventos, que são, tipicamente, a chegada de uma mensagem oriunda de algum cliente de mensagens, no caso do servidor de mensagens, ou uma requisição de conexão por parte de algum cliente de arquivos, no caso do servidor de arquivos.

Há uma instância de cliente de mensagens para cada usuário. É através delas que eles interagem com o sistema. O sistema pode possuir um administrador, que tem disponível para si um cliente mestre de mensagens, com todas as características do cliente comum, mais recursos adicionais de manutenção. Esses recursos são o comando para matar o servidor de mensagens (que, por sua vez, antes de morrer, mata o servidor de arquivos) e a possibilidade de executar todos os comandos existentes com quaisquer sessões, independentemente das senhas ou propriedades das mesmas. O administrador pode, por exemplo, usar o cliente mestre de mensagens para entrar em, mudar a senha de, ou remover qualquer uma das sessões.

Quando um usuário requisita a transferência do arquivo de *log* de alguma sessão, o seu cliente de mensagens dispara um cliente de arquivos, que, por sua vez, tenta conectar-se ao servidor de arquivos. Este, ao receber um pedido de conexão, cria, usando a chamada de sistema *fork*, um outro servidor para transmitir o arquivo desejado pelo cliente. Fica, assim, livre para atender a outros pedidos de conexão. Ao fim da transferência, o servidor e o cliente nela envolvidos morrem. O cliente de mensagens, que ficara esperando, retoma então sua execução.

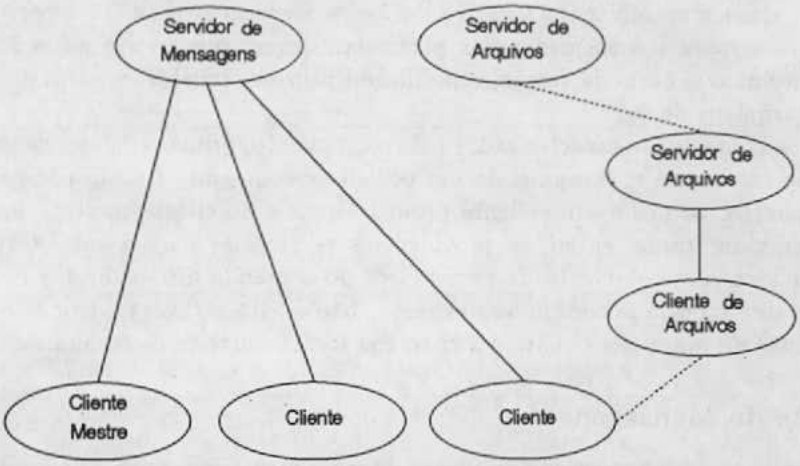

Figura 4: Interação entre os módulos

A figura 4 ilustra a interação destes módulos. Nela podemos ver três usuários utilizando o sistema, sendo que um deles está recebendo um arquivo de *log*. As linhas cheias representam conexões para a comunicação entre clientes e servidores, enquanto que as linhas pontilhadas representam chamadas *fork*. Os clientes podem estar usando qualquer máquina interligada via rede à máquina onde estão os servidores de mensagens e de arquivos.

5 Descrição dos Módulos

5.1 Servidor de Mensagens

As principais estruturas de dados utilizadas por este módulo são as tabelas de informações sobre sessões, de informações sobre os usuários participando de alguma sessão e de *endpoints* usados para a comunicação com os clientes de mensagens dos usuários.

Na primeira, armazena-se, para cada sessão existente, o seu nome, senha, arquivo de *log*, nome do proprietário e a indicação se o nome deste deve ser ocultado ou não. Na inicialização do programa, estes dados podem ser lidos de um arquivo de configuração, que é atualizado quando o cliente mestre mata o servidor.

Na segunda, para cada usuário, são guardados a sessão da qual ele participa e o nome que o identifica dentro da mesma.

Toda a comunicação com os clientes de mensagens é feita através da tabela de *endpoints*. O *endpoint* do servidor de mensagens ocupa uma posição desta tabela, e seu endereço é fixo e conhecido por todos os seus clientes. As demais posições são usadas para a conexão com os clientes.

A comunicaccão entre servidor e clientes de mensagens se dá de duas formas: assíncrona ou síncrona. A forma assíncrona ocorre quando o servidor está bloqueado, esperando por algum evento. Os eventos, excluindo-se os que indicam erros, podem ser de três tipos: pedidos de conexão, recebimento de dados e pedidos de desconexão. A forma síncrona será vista logo adiante.

Os dados recebidos pelo servidor podem ser de dois tipos: de controle ou não (comuns).

Dados comuns são mensagens que os usuários enviam. Toda mensagem, após ser recebida,

é então difundida, com o nome do usuário que a enviou acrescentado ao seu início, para todos os demais usuários que estejam participando, naquele instante, da sessão na qual foi enviada a mensagem. Caso a sessão tenha arquivo de *log*, a mensagem também é registrada nele. A exceção a essa regra são as mensagens particulares, que, como visto na seção 2, só são enviadas a um único usuário da sessão, especificado pelo seu emissor, e, além disso, não são gravadas nos arquivos de *log*.

Os dados de controle são caracterizados pelo recebimento, primeiramente, de uma seqüência especial de caracteres e, a seguir, de um código de comando. Há um código para cada comando disponível ao usuário no cliente de mensagens e no cliente mestre. Recebido um comando, o servidor toma, então, as providências necessárias para a sua execução, entre as quais se incluem o recebimento de parâmetros do comando e o envio dos resultados da verificação da consistência dos mesmos ao cliente. Isto é feito através da troca de seqüências pré-estabelecidas de mensagens. Esta é a chamada forma síncrona de comunicação.

5.2 Cliente de Mensagens

Assim que é iniciada a sua execução, o cliente de mensagens conecta-se ao servidor de mensagens, entra no modo de comando e fica interpretando os comandos que o usuário digita.

Como descrito na subseção anterior, a cada comando válido que recebe, o cliente executa protocolos de comunicação com o servidor, lhe enviando dados de controle para a identificação do comando assim como informações adicionais fornecidas pelo usuário, tais como nomes e senhas de sessões, por exemplo. Como resultado, o cliente recebe do servidor ou uma mensagem de erro, caso os dados fornecidos pelo usuário não tenham passado nos testes de consistência, ou a confirmacão da execução do comando, ou, ainda, o próprio resultado do comando, como nos casos dos relatórios de sessões e usuários de sessão (comandos *s* e *u*, respectivamente).

5.3 Cliente Mestre de Mensagens

O cliente mestre de mensagens foi feito a partir do cliente comum, acrescentado-se a este o comando *k* para matar o servidor de mensagens e o dotando de um código de usuário e de uma senha especiais que são sempre aceitos como válidos pelo servidor em todos os comandos nos quais são conferidas tais informações. Assim, o cliente mestre pode executar todos os comandos em quaisquer circunstâncias, à exceção da remoção de sessão na qual haja algum usuário.

5.4 Servidor e Cliente de Arquivos

O servidor e o cliente de arquivos são programas relativamente simples. Além do que já foi dito sobre eles anteriormente, merece ser mencionado que, assim como o servidor de mensagens, o servidor de arquivos instala-se num *endpoint* com endereço conhecido pelo seu cliente. Quando é estabelecida a conexão entre os mesmos, o cliente envia o nome do arquivo que deseja receber e, aí, é feita a sua transmissão.

6 Conclusões

O sistema **Multitalk** já se encontra implementado e em uso. Sua principal utilidade é permitir a conversação simultânea entre várias pessoas, que podem estar fisicamente distantes. Pode ser usado, por exemplo, para a realização de reuniões e debates. O diálogo se dá em sessões, de acordo com o assunto. As sessões podem ser criadas por qualquer usuário e podem

ter acesso restrito através de senha. É possível, também, registrar em arquivo tudo o que foi dito na sessão.

A implementação foi feita visando a portabilidade do sistema, que pode ser instalado em qualquer máquina com sistema operacional compatível com o Unix *System V* e que tenha a biblioteca da interface da camada de transporte implementada. O **Multitalk** pode ser usado em qualquer tipo de terminal de vídeo devidamente configurado.

Como perspectivas futuras, pretende-se desenvolver um cliente de mensagens com interface gráfica para ser usado especificamente em estações de trabalho e fazer um estudo do desempenho do sistema, em termos de número de usuários suportados e tempos de resposta obtidos.

Referências

[1] W. Richard Stevens. *UNIX Network Programming.* Prentice Hall, 1990.

[2] Sun Microsystems, Inc. *Network Programming Guide*, Março 1990.