Universidade Federal de Minas Gerais (UFMG)
Departamento de Ciência da Computação
Redes de Computadores

TRABALHO PRÁTICO 1

Servidor POP

Dalmo Melo Pereira da Rocha
dalmomelo@dcc.ufmg.br

Lucas de Oliveira Silvestre
lsilvs@dcc.ufmg.br

Resumo: *O "protocolo de agência de correio" versão 3 (POP3 – Post office Protocol) é um protocolo padrão para recuperação de e-mail. O protocolo POP3 controla a conexão entre um cliente de email POP3 e um servidor onde o email fica armazenado. O serviço POP3 usa o protocolo POP3 para recuperar emails de um servidor de email para um cliente de email POP3.No presente trabalho apresentaremos uma implementação de um servidor POP, assim como de um cliente de leitura de e-mails. As implementações estão disponíveis em versões para IPv4 e IPv6.*

Introdução

POP - "Protocolo de agência de correio"

O protocolo POP3 tem três estados de processos para tratar a conexão entre o servidor de email e o cliente de email POP3: o estado de autenticação, o estado de transação e o estado de atualização.

Durante o estado de autenticação, o cliente de email POP3 que estiver conectado ao servidor deve ser autenticado antes que os usuários possam recuperar seus emails. Se o nome de usuário e senha que são fornecidos pelo cliente de email corresponder àqueles no servidor, o usuário será autenticado e continuará com o estado de transação. Caso contrário, o usuário receberá uma mensagem de erro e não terá permissão para conectar para recuperar os emails.

Para evitar qualquer dano ao armazenamento de email depois que o cliente é autenticado, o serviço POP3 bloqueia a caixa de correio do usuário. Os novos emails entregues na caixa de correio depois que o usuário foi autenticado (e que a caixa de correio foi bloqueada) não estarão disponíveis para download até que a conexão tenha sido encerrada. Além disso, somente um cliente pode conectar-se a uma caixa de correio de cada vez; as solicitações de conexão adicionais são rejeitadas.

Durante o estado de transação, o cliente envia comandos POP3 e o servidor recebe e responde a eles de acordo com o protocolo POP3. Qualquer solicitação do cliente recebida pelo servidor que não esteja de conformidade com o protocolo POP3, é ignorada e uma mensagem de erro é enviada de volta.

O estado de atualização encerra a conexão entre o cliente e o servidor. É o último comando transmitido pelo cliente.

Após o encerramento da conexão, o armazenamento de email é atualizado para refletir as alterações feitas enquanto o usuário estava conectado ao servidor de email. Por exemplo, depois que o usuário recupera seus emails com êxito, os emails recuperados são marcados para exclusão e são excluídos do armazenamento de email, a menos que o cliente de email do usuário esteja configurado para fazer o contrário.

O detalhamento técnico completo do protocolo POP pode ser encontrado no documento online RFC-1939.

Solução proposta

A solução proposta no presente trabalho segue as regras definidas na documentação do protocolo. Para efeito de simplificação, o estado de autenticação foi implementado sem verificação real de usuário, ou senha, sendo que o servidor simplesmente toma como verdade as informações recebidas.

Inicialmente o servidor estabelece uma conexão por socket para ficar “ouvindo” uma porta do computador local em que está sendo executado, à espera de uma conexão de um cliente. Quando uma conexão “cliente” chega ao servidor, este espera as mensagens enviadas pelo cliente referentes ao estado de autenticação, ou seja, USER e PASS. Após autenticar o usuário, passa-se ao estado de transação, em que dados sobre os e-mails do usuário disponíveis no servidor são enviados para o cliente. O servidor lê todas as mensagens do usuário presentes no disco, e grava em memória algumas informações sobre as mesmas, como nome de arquivo, tamanho total, código identificador.

No lado cliente, o aplicativo se conecta automaticamente ao servidor após receber como argumentos informações de servidor, usuário e senha, e espera que o usuário navegue nas mensagens disponíveis no servidor, caso haja alguma.

Implementação

O programa *servidor* pode receber a conexão de um cliente de cada vez. Após receber os dados de autenticação de usuário do cliente conectado, o servidor entra em espera até que seja recebida uma nova mensagem do cliente.
O tipo de dados Cliente é um struct que guarda as informações do usuário, como nome de usuário, senha, lista de e-mails lidos do disco, número de e-mails, etc. A lista de e-mails é do tipo Email, e guarda como informações nome do arquivo, código identificador do e-mail, tamanho em bytes.
A cada nova mensagem recebida do cliente o servidor valida o comando e executa as ações correspondentes.
A seguir mostramos uma representação em pseudo-código dos estados de execução do programa servidor.

```
inicio;
inicializaServer();
enquanto (1)
        {
        // Aguarda conexão de um cliente
        Socket = accept(SocketServer);

                // Processa as mensagens do cliente, e executa as ações referentes a cada uma das
                mensagens:USER, PASS,  STAT, LIST, RETR, DELE, QUIT
        processaMensagens();

                // Libera memória de alocações dinâmicas

                // Ao final do processamento, fecha o socket
        }
        fim;
```

Arquivos

- *servidor4.c* - Chamada principal do programa que implementa o servidor POP versão IPv4.

- *socket4.c* - Estabelecimento de conexão socket versão IPv4.
- *servidor6.c* - Chamada principal do programa que implementa o servidor POP versão IPv6.
- *socket6.c* - Estabelecimento de conexão socket versão IPv6.
- *arquivo.c* - Implementa rotinas de leitura de e-mails, envio de informações de e-mails via socket.
- *funcoes.c* - Implementa rotinas de recebimento e tratamento de mensagens vindas pelo socket do cliente.
- *cliente4.c* - Aplicação versão IPv4 que conecta ao servidor e permite ao usuário que leia seus e-mails recebidos.
- *cliente6.c* - Aplicação versão IPv6 que conecta ao servidor e permite ao usuário que leia seus e-mails recebidos.

Compilação

Os programas *cliente* e *servidor*, nas versões para IPv4 e IPv6, podem ser compilados de duas formas distintas, ambas baseadas no compilador GCC. A primeira é através do arquivo makefile, e para proceder à compilação basta digitar o comando abaixo, que gerará 4 arquivo executáveis, referentes aos programas cliente e servidor em duas versões, IPv4 e IPv6:

make

Outra forma de compilar os programas *cliente* e *servidor* é através dos comandos diretos:

Servidor4

gcc -Wall -g servidor4.c socket4.c funcoes.c arquivo.c -o servidor4

Cliente4

gcc -Wall -g cliente4.c

Servidor6

gcc -Wall -g servidor6.c socket6.c funcoes.c arquivo.c -o servidor4

Cliente6

gcc -Wall -g cliente6.c

Execução

A aplicação inicia com a execução do servidor, que permanece em execução aguardando novas conexões. O cliente então é iniciado passando como argumentos o nome do usuário, a senha e o endereço do servidor (nesta ordem). O cliente inicia a conexão com o servidor e envia o usuário e senha. O servidor valida os dados e retorna mensagem de sucesso. Logo em seguida, informa quantas mensagem há na caixa do cliente e o espaço ocupado por elas.

A partir desse momento, o usuário dispõe de quatro opções para interagir com o servidor. Ele poderá ler a próxima mensagem (acessa a primeira caso nenhuma tenha sido acessada ainda), ler a mensagem anterior (retorna mensagem informativa caso esteja na primeira mensagem), deletar a mensagem atual (necessida logoff para excluir de fato) e sair da aplicação cliente.

Toda essa interação entre o cliente e o servidor é realizada baseada nos padrões estabelecidos pelo protocolo POP. Ou seja, o usuário interage por uma interface simplificada e a aplicação se comunica com o servidor através do protocolo específico.

Para ter acesso à lista de e-mails do cliente, enviamos o comando LIST ao servidor que responde com a lista em questão. A lista contem apenas o id e o espaço ocupado de cada e-mail. O usuário não vê essa lista, mas pode escolher ver a próxima ou a mensagem anterior. Quando o usuário navega pelas mensagens, o comando RETR {códido do e-mail} é enviado ao servidor que responde com a mensagem em questão (que é exibida para o usuário). Uma vez que a mensagem está sendo visualizada pelo usuário, ele pode excluí-la através do comando "e". Esse comando é traduzido pelo comando POP DELE e enviado ao servidor. O servidor marca o e-mail como deletado e retorna um OK para o cliente.

Por fim, quando o comando QUIT é enviado ao servidor, a conexão é finalizada, o cliente se desconecta e o servidor permanece "ouvido" novas conexões.

Resultados

Como resultado desse trabalho, tivemos uma noção bem mais ampla sobre o funcionamento de uma aplicação cliente/servidor e a importância dos padrões e protocolos definidos para garantir a compatibilidade entre as várias camadas do processo em questão. Sem esses padrões, não seria possível desenvolver aplicações independentes que se comunicassem com servidores de terceiros.

Conclusão

Ao utilizar o protocolo POP para desenvolver uma aplicação de e-mail, agilizamos o desenvolvimento, uma vez que o protocolo já nos guia sobre as funcionalidades básicas de um serviço de e-mail. A ordem de execução da aplicação se resume em: executar o servidor e aguardar até que a conexão seja estabelecida, executar o cliente e aguardar a mensagem de conexão estabelecida, o cliente então enviar comandos ao servidor solicitando a leitura ou exclusão de mensagens. Esse é o fluxo geral da aplicação em questão.

Nos testes efetuados, percebemos que a execução local era rápida demais para o correto processamento das mensagens enviadas em sequência pelo servidor ao cliente, motivo por que inserimos comandos *sleep()* entre funções *send()* consecutivas.

Os programas *servidor4* e *servidor6* são rotinas simples, e por este motivo não são capazes de atender a mais de um cliente simultaneamente.

Referências

Ziviani, N. (2007). Projeto de Algoritmos com implementações em Pascal e C. Pioneira Thomson Learning.

Post Office Protocol. Disponível em <http://pt.wikipedia.org/wiki/Post_Office_Protocol>

Post Office Protocol version 3. RFC1939. Disponível em <http://www.faqs.org/rfcs/rfc1939.html>