DCC031 – Redes de computadores

Trabalho prático 1 – Protocolo SMTP

Professor: Luiz Filipe Menezes Vieira

Data de entrega: 26/09/2011

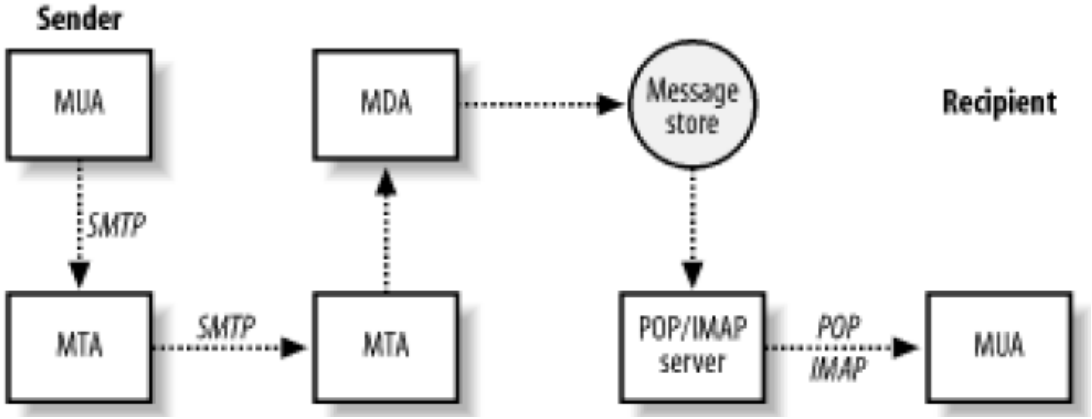
Trabalho em grupo de até dois alunos.

Valor do trabalho: 15 pontos

1. Descrição do trabalho

Neste trabalho iremos implementar um cliente e um servidor de e-mail empregando o protocolo de comunicação SMTP. Tanto o cliente quanto o servidor de e-mail implementados nesse trabalho serão capazes de se comunicar com servidores (por exemplo o Sendmail) e clientes de e-mail (Thunderbird, Outlook, ...) de produção. Iremos implementar um subconjunto mínimo do protocolo, que permita o envio de e-mails considerando que não há nenhum erro na mensagem a ser enviada.

O protocolo SMTP é um protocolo simples para envio de e-mail, baseado em texto. O SMTP é um padrão de Internet, definido na RFC 821. Ele cuida da parte de **envio** de e-mails de um cliente de e-mail para um MTA, ou de um MTA para outro MTA. Um MTA (*Mail Transfer Agent*) é uma entidade que se encarrega de repassar o e-mail até que este seja enviado ao servidor responsável pela caixa de correio do usuário, o MDA (*Message Delivery Agent*). A caixa de correio, que em geral também implementa um MTA, irá armazenar em disco as mensagens do usuário até que este venha recolhe-las. O usuário, então, irá conectar-se ao servidor empregando protocolos para acesso de mensagens (por exemplo o POP e o IMAP, definidos respectivamente nas RFCs 918 e 3501), utilizando um MUA (*Mail User Agent*, por exemplo o Outlook). A estrutura de um serviço de e-mail é mostrada na Figura abaixo.



Toda a comunicação é realizada utilizando o protocolo de transporte TCP[1]. O SMTP é um protocolo textual, ou seja, são enviados comandos em ASCII para o envio de um e-mail. O diálogo abaixo representa um e-mail sendo enviado. As linhas começando com "C:" marcam os dados enviados pelo cliente, e "S:" indicam os dados enviados pelo servidor.

```
•S: 220 hamburger.edu
•C: HELO crepes.fr
•S: 250 Hello crepes.fr, pleased to meet you
•C: MAIL FROM: <alice@crepes.fr>
•S: 250 alice@crepes.fr... Sender ok
•C: RCPT TO: <bob@hamburger.edu>
•S: 250 bob@hamburger.edu ... Recipient ok
•C: DATA
•S: 354 Enter mail, end with "." on a line by itself
 C: Date: Thu, 21 May 1998 05:33:29 -0700
 C: Subject: What do I put in your hamburger?
 C:
•C: Do you like ketchup?
•C: How about pickles?
 C: And cheese?
 C:
•C: .
•S: 250 Message accepted for delivery
•C: QUIT
•S: 221 hamburger.edu closing connection
```

O protocolo funciona da forma pedido-resposta, onde o cliente envia um pedido ou comando, e o servidor envia uma resposta indicando se o comando foi bem-sucedido ou não. O cliente envia comandos textuais, do tipo "mail from", "helo", etc. Comandos do cliente podem ser enviados em letras maiúsculas, minúsculas ou em qualquer combinação de letras maiúsculas e minúsculas. A comunicação do servidor é feita através de códigos numéricos, que são em geral seguidos por texto em ASCII indicando o significado do mesmo. Os clientes devem interpretar o comando somente pelo código numérico, não pelo texto, entretanto o texto é de grande ajuda quando mandamos um e-mail "na mão", ou seja, fazendo um telnet para o servidor. Os códigos são definidos com três números, sendo que o primeiro algarismo é sempre 2, 3, 4 ou 5. Códigos iniciados com 2 indicam uma resposta positiva. Códigos começando com 3 indicam uma resposta positiva, entretanto a ação só será completada caso sejam fornecidos mais dados. Os códigos 4 e 5 são para falhas, sendo que o 4 é para falhas temporárias (pouco espaço em disco, erro de processamento interno, etc) e o 5 é utilizado para falhas permanentes (erros de sintaxe, comandos não implementados, etc).

A primeira etapa da comunicação é a saudação (até os computadores gostam de cortesia...), onde o cliente e o servidor dão um alô. O cliente envia um comando "HELO servidor", onde servidor é o nome do mesmo na Internet. Em seguida, o cliente pode começar a enviar o e-mail, indicando de quem é o e-mail "mail from", e para quem o e-mail será enviado "rcpt to". Caso o servidor aceite, ele irá enviar um comando 354, indicando para o cliente que o texto do e-mail pode ser enviado[2]. O fim do e-mail é

---

[1] Sim, o TCP! Não usamos o SSL ou outro tipo de protocolo de sessão ou transporte seguros. Ou seja, o SMTP normal não tem segurança! Procure sempre verificar com o seu provedor se ele suporta o SMTPS (SMTP+SSL ou SMTP+TLS) para garantir que os seus e-mails não possam ser lidos enquanto em trânsito.

[2] Servidores de e-mail bem configurados só enviam e-mails de contas válidas do seu domínio. Isso serve para evitar spams. Se alguém tentar enviar um e-mail de

delimitado por uma linha contendo somente um ponto "."[3]. Nesse momento, o servidor processa o e-mail, e envia uma resposta positiva ou negativa. O cliente pode enviar mais e-mails, ou desconectar usando o comando "quit".

O formato do e-mail é bem característico. Ele começa com os cabeçalhos, que são sempre da forma "Nome: valor", onde o valor é considerado depois do espaço seguido aos dois pontos até o fim da linha. Um exemplo de campo é o "Date:", que o cliente pode definir (isso é útil para e-mails que ficam travados em um MTA intermediário). Um campo importante é o "Subject", que é empregado nos softwares de e-mail para indicar o assunto da mensagem. Depois que todos os cabeçalhos foram enviados, o cliente envia uma linha vazia para indicar o início do texto da mensagem, e em seguida o seu texto.

Arquivos binários são enviados utilizando a codificação Base64, contendo antes do mesmo um marcador com um identificador. O marcador são dois caracteres "-", e o nome é uma string. O conteúdo é terminado por um novo marcador de fim, tendo a forma "--nome--". É ainda usual termos atributos, sendo os mais comuns o tipo do conteúdo e o seu nome. Estes ficam separados do arquivo por uma linha vazia. Um exemplo segue abaixo:

```
--0015174c4610934d3f0488ace53e
Content-Type: application/pdf; name="Article.pdf"
Content-Disposition: attachment; filename="Article.pdf"
Content-Transfer-Encoding: base64
X-Attachment-Id: f_ga9m7ty70

TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz
IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu
dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4=
--0015174c4610934d3f0488ace53e--
```

2. Programas a serem desenvolvidos

O trabalho prático irá consistir de dois programas, um cliente e um servidor de e-mails. Ambos irão implementar um sub-conjunto do protocolo SMTP, tal como descrito na seção anterior, de forma que um cliente de e-mail comercial possa se conectar no servidor desenvolvido, ou que o cliente de e-mail desenvolvido no trabalho possa se conectar em um servidor de produção. O cliente e o servidor devem assumir as seguintes premissas:

1. Tanto o destinatário como o emissor do e-mail podem ser considerados corretos, ou seja, eles existem e são válidos.

2. As mensagens de e-mail acabam de forma correta

3. Podem ocorrer linhas no e-mail começando com ".", ou linhas onde tenhamos somente um ponto na mesma.

4. Podem haver arquivos anexados, empregando a codificação Base64.

Viagra@bestprice.com ou de viagra@ufmg.br no servidor da UFMG, ele será negado (se o usuário 'viagra' não existir no último caso).

[3] Para podermos enviar uma mensagem que possui uma linha com somente um ponto, o cliente usa um sistema de escape, enviando dois pontos. O servidor deve ser esperto o suficiente para eliminar um ponto sempre que uma linha começar por dois pontos.

5. Será empregado o TCP como protocolo de transporte.

6. O servidor deve imprimir uma mensagem após cada código de retorno, para facilitar a leitura da saída do mesmo quando conectado com telnet.

a. Programa cliente

O programa cliente, chamado cliente, terá quatro parâmetros na linha de comando: o endereço IP ou o nome do servidor em que devemos, o e-mail do emissor, o e-mail do destinatário, e o caminho de um arquivo a ser enviado (opcional). Ao executar o programa, ele deve imprimir uma mensagem "Assunto:", onde o usuário poderá digitar o assunto do seu e-mail. Em seguida, o programa irá exibir uma mensagem dizendo: "Digite a sua mensagem. Termine a escrita com uma linha terminada por FIM". O usuário poderá digitar a sua mensagem, que terminará com uma linha onde ele irá escrever FIM. Ao final deste procedimento, o programa terminará a sua execução. Apresentamos a seguir uma execução exemplo do cliente (itálico indica entrada escrita pelo usuário do programa):

```
[~/]> cliente 127.0.0.1 damacedo@dcc.ufmg.br aluno@dcc.ufmg.br tp0.pdf
Assunto: Especificação do TP0
Digite a sua mensagem. Termine a escrita com uma linha terminada por FIM
Caros alunos,
Estou enviando a especificação do TP0 de redes.
Abraços,

Daniel

FIM
```

O arquivo *tp0.pdf* deverá ser lido e codificado em Base64, empregando a implementação feita no TP0. O mesmo será enviado no fim da mensagem, logo após o texto ASCII. O identificador do arquivo no momento do envio pode ser uma string qualquer. O "Content-Type" pode ser "application/octet-stream", que é o equivalente a um arquivo binário genérico.

b. Programa servidor

O programa servidor, chamado *servidor*, irá escutar conexões de clientes de e-mail. Ele não tem nenhum parâmetro para a sua execução. Vocês podem assumir que somente um e-mail será enviado por execução, e que somente um cliente por vez estará conectado ao servidor. Os e-mails recebidos serão gravados em arquivos, um para cada usuário, e novos e-mails serão adicionados ao fim do arquivo (*append*), precedidos por uma linha contendo vinte caracteres "=". Para o exemplo do e-mail anterior, iremos criar um arquivo "aluno.mbox". Desconsiderem tudo depois da arroba na hora para definir o nome do arquivo, por exemplo o servidor ao receber um e-mail para aluno@dcc.ufmg.br e outro para contato@www.google.com, irá criar um arquivo "aluno.mbox" e outro chamado "contato.mbox". Os arquivos devem ser criados no diretório atual onde o programa servidor está sendo executado.

O formato de cada e-mail é parecido ao formato mbox (definido nas RFCs 2822 e 4155, e ainda empregado em alguns programas comerciais de e-mail). Um exemplo após o envio de dois e-mails igual ao especificado acima criaria um arquivo "aluno.mbox" da seguinte forma. Reparem que existem os atributos do e-mail, e em seguida vem o corpo do e-mail, separado por uma linha vazia.

```
Delivered-to: aluno@dcc.ufmg.br
From: damacedo@dcc.ufmg.br
Subject: Especificação do TP0
```

```
Caros alunos,
Estou enviando a especificação do TP0 de redes.
Abraços,

Daniel

--0015174c4610934d3f0488ace53e
Content-Type: application/octet-stream; name="tp0.pdf"
Content-Disposition: attachment; filename="tp0.pdf"
Content-Transfer-Encoding: base64

TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz
IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu
dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4=
--0015174c4610934d3f0488ace53e--
====================
Delivered-to: aluno@dcc.ufmg.br
From: damacedo@dcc.ufmg.br
Subject: Especificação do TP0

Caros alunos,
Estou enviando a especificação do TP0 de redes.
Abraços,

Daniel

--0015174c4610934d3f0488ace53e
Content-Type: application/octet-stream; name="tp0.pdf"
Content-Disposition: attachment; filename="tp0.pdf"
Content-Transfer-Encoding: base64

TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz
IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu
dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4=
--0015174c4610934d3f0488ace53e--
====================
```

3. Entrega do código

O código deve ser entregue em um arquivo Zip (não pode ser RAR nem .tgz nem .tar.gz) para **vieira.submission@gmail.com**, com assunto [REDES-TP1] "seu nome", contendo um arquivo **readme.txt** com o nome dos integrantes. Incluam todos os arquivos (.c, .h, makefile, não incluam executáveis ou arquivos objeto) EM UM ÚNICO DIRETÓRIO. Um makefile deve ser fornecido para a compilação do código. Este makefile, quando executado sem parâmetros, irá gerar os dois programas, cliente e servidor, EXATAMENTE com esses nomes. Submissões onde os programas não seguem as especificações de parâmetros e nomes, makefiles não funcionando ou arquivos necessários faltando não serão corrigidos. O julgamento do trabalho será feito em **Linux**, portanto verifiquem se o seu trabalho é compatível com Linux.

4. Documentação

A documentação deve ser breve, descrevendo as decisões de implementação mais importantes e o funcionamento do código, de forma que o professor possa entender o que foi feito no código sem ter que entender linha a linha dos arquivos. Implementações modularizadas deverão mencionar quais funções são implementadas em cada módulo ou classe.

5. Desconto de nota por atrasos

Trabalhos entregues com atraso sofrerão as seguintes penalidades:

- **Um dia de atraso**: correção valendo somente 50% da nota.
- **Dois dias de atraso**: correção valendo somente 25% da nota.
- **Atrasos superiores a dois dias**: o trabalho não será corrigido.

6. Referências

- RFC 821 – Simple Mail Transfer Protocol. http://tools.ietf.org/html/rfc821
- RFC 2822 - Internet Message Format. http://tools.ietf.org/html/rfc2822
- RFC 4155 - The application/mbox Media Type. http://tools.ietf.org/html/rfc4155