CÉSAR HENRIQUE KALLAS
DANIEL FIGUEIREDO MACHADO
FÁBIO HENRIQUE GENEROSO
RENATO DEMARCO

TRABALHO DE REDES DE COMPUTADORES 1
GNUTELLA

CAMPINAS/2005

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

GNUTELLA

DISCIPLINA: Redes de Computadores 1
PROFESSOR: Juan Manuel A. Coelho

GRUPO:
César Henrique Kallas RA: 02099224
cesarkallas@gmx.net

Daniel Figueiredo Machado RA: 02508224
daniel.fm@uol.com.br

Fábio Henrique Generoso RA: 02164473
fhgeneroso@gmail.com

Renato Demarco RA: 02142818
renatodemarco@gmail.com

SUMÁRIO

INTRODUÇÃO

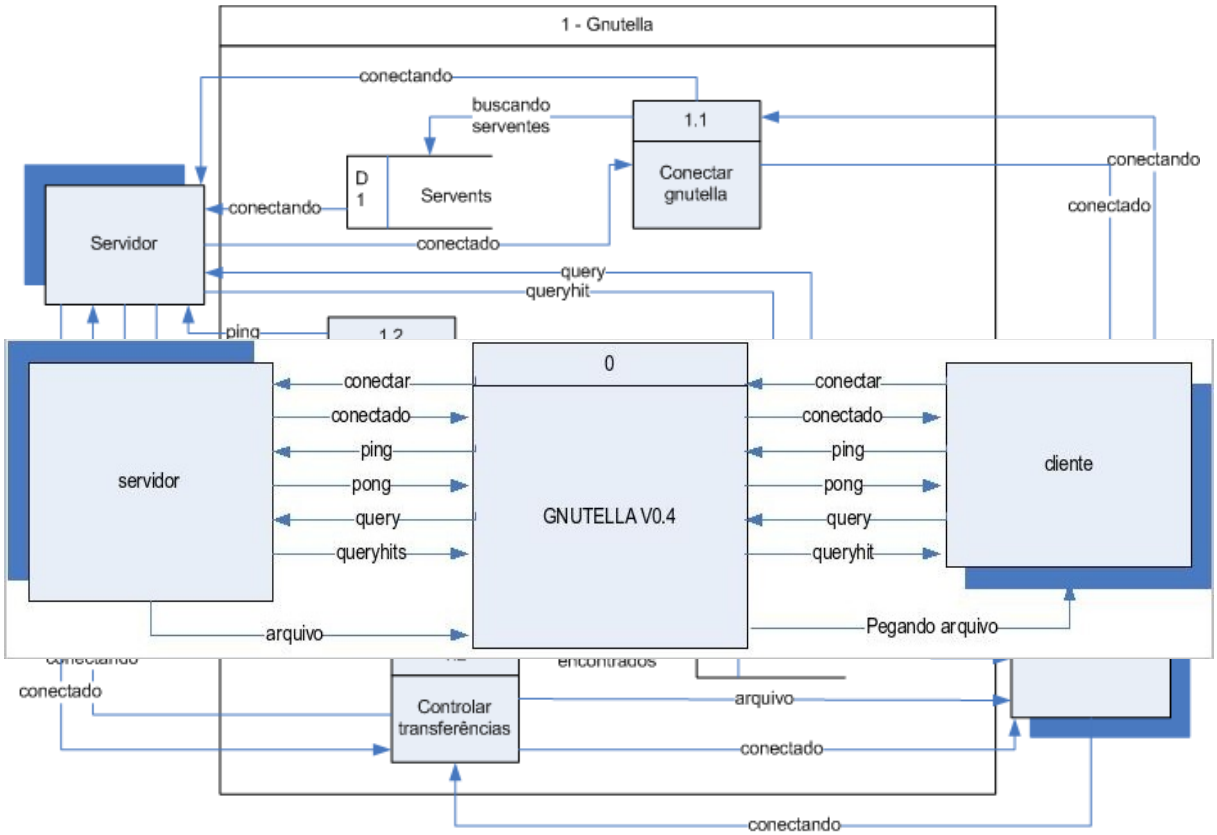
Esse projeto visa implementar um servent P2P (pear to pear) baseado numa versão simplificada do protocolo Gnutella 0.4.

O projeto considera os principais aspectos da especificação do Gnutella versão 0.4, alguns aspectos relacionados a firewals e as extenções propostas na espeficicação básica que estão descritas no apêncidece do original da especificação Gnutella 0.4.

Protocolos de redes P2P todos os participantes tem os mesmos direitos e deveres, ou seja, todos contribuem para o roteamento e os servents funcionam como cliente e como servidores.

O Gnutella usa o conceito de P2P entretanto, as transferências de arquivos não são realizadas através de uma solução própria, mas sim do protocolo HTTP, ou seja, o mesmo que você usa para fazer download de arquivos em qualquer site na Web.

PROJETO



- Introdução do Projeto

Esse projeto visa a implementação de um servent da rede Gnutella, de acordo com a especificação do protocolo Gnutella versão 0.4, revisão 1.6 .
O servent é divido em duas partes, uma servidor e outra cliente. Porém, as duas partes são vistas apenas como uma pelo usuário, podendo ser implementadas separadamente.

Figura 1 - DFD nível 0

- Funções

Mensagem
Função que envia mensagem.
A mensagem pode ser enviada para um usuário ou via um socket padrão, depende do tipo passado.
Entrada: tipo da mensagem e o corpo da mensagem
Saída: status do envio da mensagem.

```
status função Mensagem(tipo, msg)
        verifique o tipo
                se tipo = USUARIO:
                        mostre uma mensagem na tela com msg
                senão
                        escreva no canal de comunicação a msg
        fim
        retorne status
fim
```

Conexão

É feita através de uma consulta na lista de servents.

Consulta-se o primeiro host declarado na estrutura e tenta fazer a conexão, caso não tenha sucesso, segue para o próximo host. Se todas tentativas falharem, uma mensagem é enviada para o usuário.

Entrada: lista_servents

Saída: status da conexão

```
função conexao(arquivo  LISTA_SERVENTS)
laço: enquanto possuir host na lista_servents faça:
pegue o host no arquivo
conectar(host, porta)
se falha: volta ao inicio do laço de repetição
se fim de arquivo:
mensagem('USUARIO', "A conexão falhou, tente outros servents.")
fim
fim
```

```
status funcao conectar(host, porta)
        conectar no host e na porta
        mensagem(CONEXAO, host, 'GNUTELLA_VERSAO');
        se 'GNUTELLA_OK':
                retorne 'CONEXAO_SUCEDIDA'
se não: retorne 'FALHA_CONEXAO'
fim
```

Ping

É enviado uma mensagem do tipo ping para um servent e espera-se que ele responda com pong.

Entrada: host

```
estrutura_payload função ping (host)
                BYTE ping[0:L-1]
                retorne ping
        fim
```

Pong

Resposta via mensagem com o descritor pong.

O host é o servent que enviou o ping, mas não o servent originou o ping, mas sim o servent que transmitiu por último o ping.

Entrada: porta do servent, host, número de arquivos no servent e tamanho total desses arquivos.

estrutura_payload função pong()
BYTE pong[L-1]
pong[0:1] = PORTA_ENTRADA
pong[2:5] = getLocalIpAddress()
pong[6:9] = ArqCompartilhados(N_ARQUIVOS, NULL)
pong[10:13] = ArqCompartilhados(TAM_ARQUIVOS, NULL)
retorne pong
fim

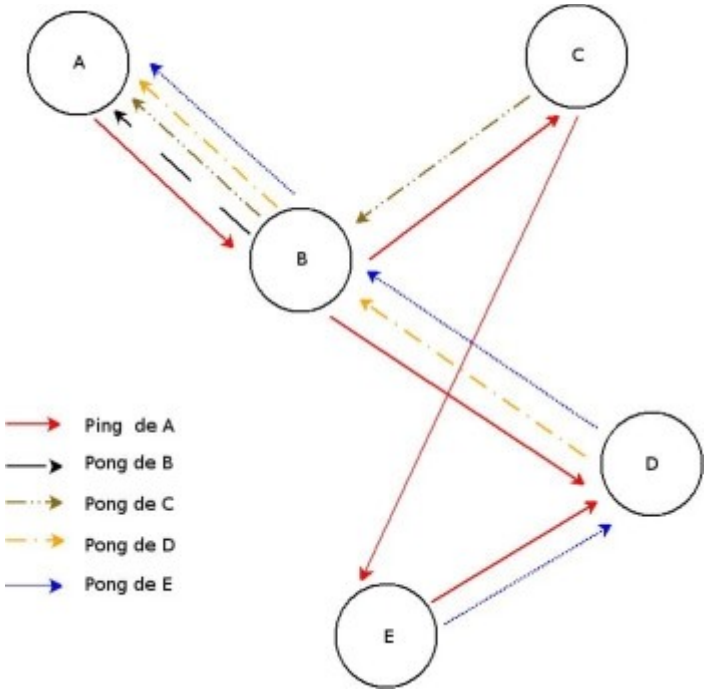


Figura 3 - Percursso de um PING / PONG

Query
Função de busca de arquivos no servents.

estrutura_payload função query(string)
BYTE query[L-1]
query[0:1] = QUERY_LIVRE | QUERY_MIN | QUERY_MED
inteiro n = tamanho(string)
query[2:n] = string
query[n+1] = NULL
retorne query
fim

QueryHits
Função que retorna o número de arquivos encontrados de acordo com uma query passada.

estrutura_payload função queryhits(string)
BYTE queryhits[L-1]
queryhits[1:2] = PORTA_ENTRADA
queryhits[3:6] = getlocalipaddress()
queryhits[7:10] = getSpeed()

queryhits[11:N] = ArqCompartilhados(PROC_ARQUIVOS, string)
retorne *estrutura_payload*
fim

GeraDescritor
Função que encapsula o payload no descritor.
Array de bytes:

Descritor ID	Payload	TTL	Hops	Tamanho do Payload

Entrada:
Payload - PING / PONG / QUERY / QUERYHITS / BYE
Estrutura correspondente ao descritor payload

descritor funcao GeraDescritor(payload, estrutura_payload)
BYTE descritor[n]
descritor[0:15] = GeraDescriptorID()
descritor[16] = payload
descritor[17] = TTL
descritor[18] = 0
inteiro t = tamanho(estrutura_payload)
descritor[19-22] = t
descritor[23:t] = estrutura_payload
retorne descritor
fim

GeraDescriptorID
Função que gera um número de 16 bytes (128 bits) aleatório (e possivelmente único).

numero função GeraDescriptorID()
inteiro n = gera_id_randomico(128 bits)
retorne n
fim

ArqCompartilhados
Retorna o número de arquivos compartilhados ou o tamanho total desses arquivos. Entrada:
String: nome de um arquivo
Opção: N_ARQUIVOS (número de arquivos)
TAM_ARQUIVOS (tamanho total dos arquivos)

numero ArqCompartilhados(opcao, string)
inteiro n = some o número de arquivos no diretório
verifique opcao
caso TAM_ARQUIVOS:
laço: enquanto n != 0
string arquivo = prox_arq_diretório(DIRETÓRIO)
tam = tam + tamanho(arquivo)
n - -
retorne tam
caso N_ARQUIVOS:
retorne n
caso PROC_ARQUIVOS:
estrutura dados_arquivo (inteiro index,

```
                                                    inteiro tamanho, string nome)
                        procure_no_disco(string, arquivo)
                        retorne dados_arquivo
            fim
        fim
```

Principal
Função principal do programa.

```
função principal()
        conexao(LISTA_SERVENTS)
        nova_linha_execucao(servidor(PORTA_ENTRADA))
        ...
        servidor:
        laço:
                BYTEreq[] = escute_porta(PORTA_ENTRADA)
        se req[16] = PING
                nova_linha_execucao(
                GeraDescriptorID(PONG, pong()))
        se req[16] = QUERY
                nova_linha_execucao(
                GeraDescriptorID(QUERYHITS, queryhits(req[2:N])))
...
                ...
                cliente
        Mensagem(PING,GeraDescriptorID(PING, ping()))
Mensagem(QUERY,GeraDescriptorID(QUERY,query(string)))
                ...
        fim
```

- Estrutura de dados

Constantes
Contém a definição de todos as constantes do protocolo, declarados de forma estática em um função do programa separada ou na função principal.

```
String GNUTELLA “GNUTELLA CONNECT/0.4\n\n”
String GNUTELLA_OK “GNUTELA OK\n\n”
String DIRETORIO “/tmp/gnutella/arquivos”
inteiro         CONEXAO                 0xa0
inteiro         CONEXAO_SUCEDIDA        0xa1
inteiro         FALHA_CONEXAO           0xa2
inteiro         USUARIO                 0xa3
inteiro         N_ARQUIVOS              0xa4
inteiro         TAM_ARQUIVOS            0xa5
inteiro         PROC_ARQUIVOS           0xa6
inteiro         PING            0x00
inteiro         PONG                    0x01
inteiro         QUERY                   0x80
inteiro         QUERYHITS               0x81
inteiro         BYE                     0x02
inteiro         TTL                     10
```

inteiro	QUERY_LIVRE	0x00
inteiro	QUERY_MIN	0x01
inteiro	QUERY_MED	0x02
inteiro	NULL	0x00
string	LISTA_SERVENTS	“servents.ini”

Variáveis

inteiro	PORTA_ENTRADA	2000 (padrão)

Lista de servents: lista_servents
Arquivo texto com hosts que já estão conectados a rede, lista obtida através de sites na internet.

Estrutura do arquivo:
<nome do host>:<porta>
gnet.ath.cx:6346
gnet2.ath.cx:6346
gnet3.ath.cx:6346
gnet4.ath.cx:6346
gnet5.ath.cx:6346
gnotella.fileflash.com:6346
gnutellahosts.com:6346

Diretório de arquivos
Diretório em disco que contém os arquivos a serem compartilhados na rede.

Lista de arquivos sendo baixados
Estrutura de dados com a lista de arquivos sendo baixos pelo cliente do servent.

IMPLEMENTAÇÃO

Ambiente de programação:
j2sdk – Java for Student Development 1.4 ou superior
Linux – Suse Linux 9.3 – kernel 2.6.11.4-21.9-default

Código
O software funciona com três threads iniciais:

1. - Interface gráfica, interação com o usuário
2. - Servidor de conexões de outro servent
3. - Servidor HTTP para requisições

1. - Interface gráfica
   Toda comunicação com os outros servents é requisitada atráves dessa interface. Os descritores (ping ,pong , query, queryhits) são enviados por solicitação e recebidos os resultados nessa interface.

2. Servidor de conexões
   Trata requisições de servents que conectados a ele

3. Servidor web
   Trata pedido de downloads de arquivos

TESTE

Ambiente com três computadores:

1. A Endereço: 10.0.0.1 porta: 6346
2. B Endereço: 10.0.0.10 porta: 6346
3. C Endereço: 10.0.0.12 porta: 6346

1. CONEXÃO

B conecta em A
A conecta em C
C conecta em B

Resultado: conexões bem sucedidas.

2. ENVIANDO PINGS E REPASSANDO PINGS / PONGS

B envia ping para A
A recebe ping de B
A envia pong para B
A envia ping para C
C recebe ping de A
C envia pong para A
C envia ping para B
B recebe ping de C
B envia pong para C
C envia pong para A

Resultado: Testes bem sucedidos.

3. ENVIANDO QUERYS E REPASSANDO QUERYS

B envia query para A
A responde com queryhits para B (se tiver arquivos)
A envia query para C
C responde com queryhits para A
A repassa o queryhits para B

Resutlado: Testes bem sucedido.

4. CONEXÃO PARA TRANSFERÊNCIA DE ARQUIVO

B conecta em A e pede arquivo
A responde com OK
A envia arquivo
B recebe arquivo

Resultado: transferência bem sucedida.

COMPILAÇÃO E USO DO PROGRAMA

O projeto foi desenvolvido em JAVA utilizando o Sistema Operacional Linux para o seu desenvolvimento. A versão do compilador java usado é javac 1.5.0_03.

Para compilação foi feio um makefile simples, basta digitar make no diretório onde estão os arquivos para que a compilação seja efetuada.

No makefile linhas de instrução são feitas para que o java além de compilar os arquivos java crie diretórios para os .class entre outros diretórios para melhor organização do programa e ainda copie alguns arquivos nescessários para que o programa rode com sucesso.

Outro fator importante do makefile é a criação da documentação .html através do javadoc, essa criação é feita por linhas de instrução no makefile.

Na documentação criada pelo javadoc que pode ser acessada no diretório doc, tem a especificação de todas as classes e as importantes funções de cada uma delas sendo explicadas dando detalhes de variáveis que são passadas, o objetivo da função e o retorno dela.

Para rodar o programa basta executar os arquivos Estella.class e EstellaServidor.class que são gerados pela compilação através do makefile no diretório Estella.

Para compilar e rodar o programa é nescessário que o sistema operacional tenha instalado uma máquina virtual java para que o programa possa ser compilado e executado com sucesso.

export CLASSPATH=.:modules/org-jdesktop-layout.jar

Código fonte

```
rm -rf estella
mkdir estella
mkdir estella/gui
cp -R icones/ estella/gui/ > /dev/null
javac -d . *.java -Xlint -sour
```

Documentação

```
javadoc -d doc *.java
```

Iniciar programa

$ sh Estella.sh

Ver documentação

cd doc
abrir Index.html

CONCLUSÃO

De acordo com a opinião do grupo os objetivos da atividade foram alcançados. O programa desenvolvido é bastante interessante para quem está estudando redes e demonstra na prática aspectos aprendidos em sala de aula.

Foi aprendido bastante sobre como programar usando tecnologias para utilização de redes para comunicação entre computadores usando sempre clientes e servidores de arquivos como base dessa comunicação.

A principal dificuldade encontrada pelo grupo foi o tempo dado para o desenvolvimento do projeto que foi bastante curto dada a complexibilidade do trabalho.

Uma sugestão que o grupo daria para as próximas turmas atingirem os objetivos da disciplina com mais clareza seria aumentar o tempo dado para o desenvolvimento do semestre, levando em consideração que a matéria é dada no sexto semestre do curso onde outras disciplinas com projetos de complexibilidade semelhantes mas sobre assuntos diferentes são cursadas pela maioria dos alunos. Para melhorar a qualidade dos projetos e o aprendizado dos alunos, aumentar o tempo para o desenvolvimento principalmente na parte de projeto e implementação seria bastante interessante.

REFERÊNCIAS E BIBLIOGRAFIA

- http://rfc-gnutella.sourceforge.net/
- http://www.gnutelliums.com/
- http://gnutella.da.ru/ (lista de hosts.)
- http://java.sun.com/j2se/1.4.2/docs/api/

- http://java.sun.com/docs/books/tutorial/uiswing/components/table.html
- http://www.codetoad.com/java_JTable.asp
- http://rfc-gnutella.sourceforge.net/developer/stable/index.html

LICENÇA DE USO

GPL – Gnu Public License - Version 2, June 1991 (em anexo)