Processamento distribuído em ambiente peer-to-peer

Alexandre Helfrich
Orientando
Prof. Paulo Fernando da Silva
Orientador

Roteiro

- Introdução e Objetivos
- Fundamentação Teórica, Conceitos e Contexto Atual
- Desenvolvimento, Requisitos e Especificação
- Técnicas e Ferramentas Utilizadas
- Apresentação da Especificação
- Implementação, Técnicas e Ferramentas Utilizadas
- Operacionalidade da Implementação
- Resultados
- Conclusão e Extensões

Introdução

- Desenvolvimento Técnologico
- Aumento de Banda
- Ociosidade dos Computadores
- Processamento Distribuído

Objetivos

- Desenvolver um protótipo de um aplicativo servidor de nomes.
- Desenvolver um protótipo de um aplicativo *peer* que se conectará ao servidor de nomes .
- Utilizar vários computadores de uma rede local para formar o processamento distribuído.

Fundamentação Teórica

- Neste capítulo serão apresentados os principais conceitos a respeito das seguintes tecnologias: redes *peer-to-peer*; agentes móveis; *aglets* e XML. Também serão relatados os trabalhos correlatos.

Conceitos Básicos

- Redes Peer-to-Peer
- Agentes Móveis
- Aglets
- XML

Contexto Atual

- José Voss Junior – Protótipo de software em redes peer-to-peer para compartilhar arquivos.
- Cristiano M. Borchardt – Funcionamento de sistema de processamento de transação em ambientes distribuídos.
- Fernando Liesenberg - Protótipo de software que utiliza agentes móveis para fazer monitoramento de computadores em uma rede.

Desenvolvimento

- Neste capítulo serão apresentados os requisitos do trabalho, bem como a sua especificação e ferramentas utilizadas para a implementação, o teste de operacionalidade e a discussão dos resultados obtidos.

Requisitos Principais do Problema

- Servidor: o protótipo deverá possuir um programa servidor que será responsável pela manutenção da lista de *peers* conectados a ele e pela distribuição desta lista de *peers* aos *peers* que a ele se conectar.

Requisitos Principais do Problema

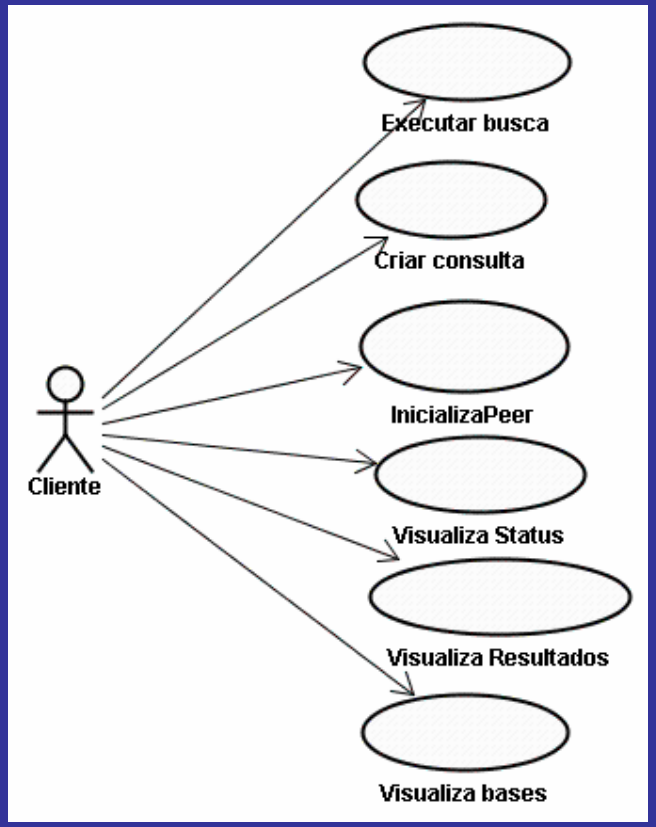
- Peer: o protótipo deverá possuir um programa *peer* que fará uma leitura dos arquivos XML que estão em um determinada pasta do sistema, os quais disponibilizará aos outros *peers* para consulta, ainda receberá a lista de *peers* do servidor de nomes e de posse desta lista de *peers* fará consultas aos arquivos XML que os *peers* da lista de *peers* possuírem.

Requisitos Principais do Problema

- Tarefas: o aplicativo *peer* fará consultas a outros *peers* da lista de *peers* nos arquivos XML que estes possuírem.
- Resposta: o programa peer deverá informar ao peer que solicitou a tarefa, a resposta à tarefa que lhe foi enviado para processamento, ou seja, após o término da pesquisa do arquivo XML, deve-se enviar o resultado da pesquisa ao *peer* que solicitou.

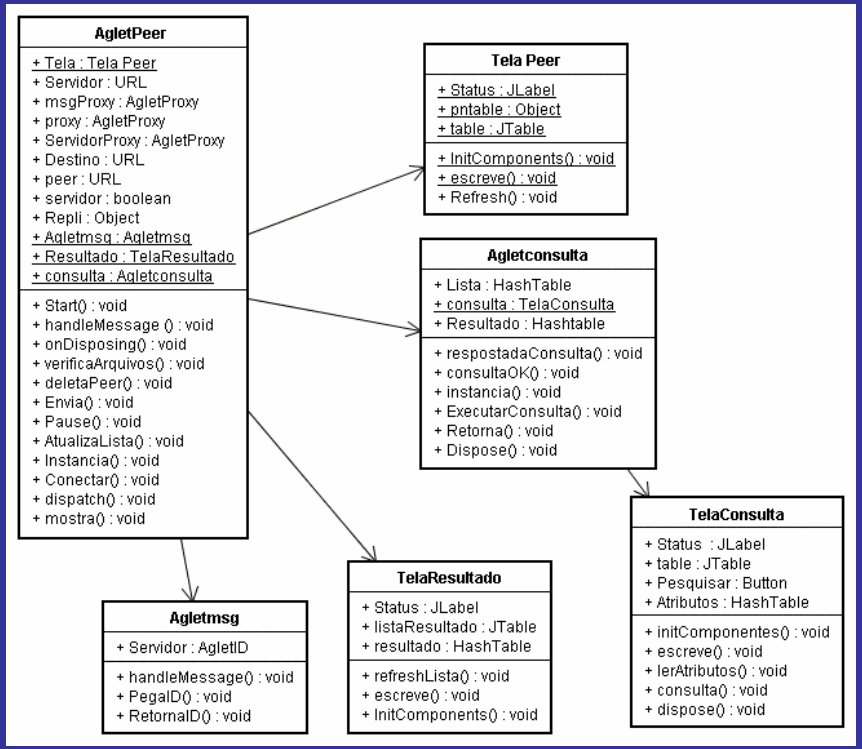
Especificação

- Aplicativo Peer



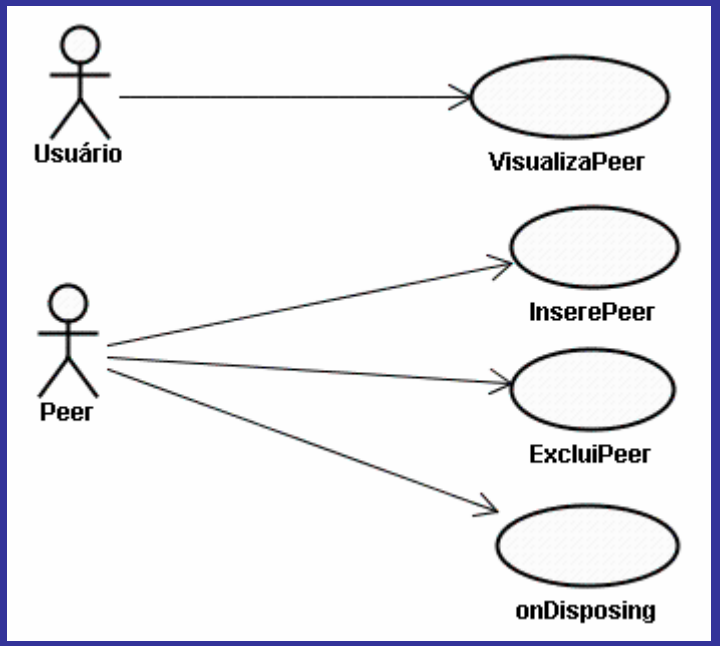
Especificação

- Diagrama de classe



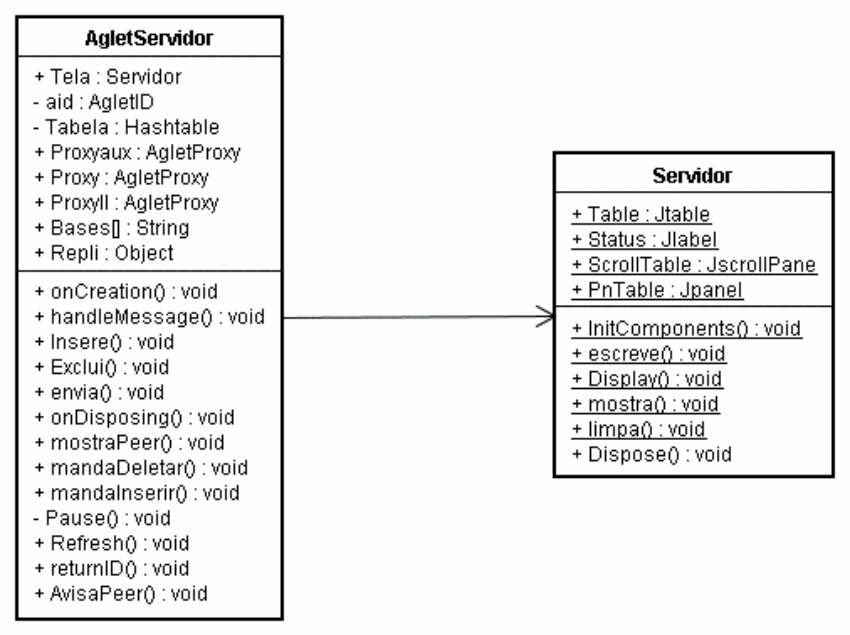
Especificação

- Aplicativo Servidor de Nomes



Especificação

- Diagrama de Classe



Técnicas e Ferramentas utilizadas

- O método de ciclo de vida escolhido para o desenvolvimento do protótipo apresentado neste trabalho é o iterativo e incremental, por se tratar de um método onde todo o trabalho é dividido em partes menores que tem como resultado um incremento ao trabalho final. Este método oferece maior segurança no atendimento dos requisitos e maior flexibilidade durante todo o processo de desenvolvimento.

Técnicas e Ferramentas utilizadas

- A especificação do problema apresenta-se através de um grupo de diagramas da linguagem *Unified Modeling Language* (UML), sendo estes: diagrama de classes, diagrama de seqüência e diagrama de caso de uso. Utilizou-se a ferramenta *Jude Community* para a modelagem da especificação. Uma ferramenta que teve resultado satisfatório às necessidades das especificações necessárias ao desenvolvimento dos diagramas utilizados no trabalho.

Especificação

- Servidor - TelaServidor
- AgletPeer - TelaPeer
- AgletMsg
- Lista de peers
- Insere e Exclui peer
- TelaPesquisa
- AgletPesquisa
- TelaResultado

Implementação-instanciação

```
try{
    //destino recebe um undereço ATP
    destino = new URL("atp://alexandre:12000/");
    // objeto peer do tipo URL recebe Endereço ATP onde esta instanciado este objeto PeerAglet
    peer = this.getCodeBase();
}catch (MalformedURLException e){
    //Erro de formação do endereço ATP
    envia("Erro de conexãoPeerAglet/run() "+e.getMessage());
}
try{
    //Instanciação do objeto AgletMsg
    AgletProxy proxy = getAgletContext().createAglet(getCodeBase(),"AgletMsg",null);
    //Variavel do tipo AgletID recebe o identificador do objeto criado
    aid = proxy.getAgletID();
    //objeto msgProxy recebe a referencia ao objeto dispachado para o destino,
    //onde destino é um endereço ATP
    msgProxy = proxy.dispatch(destino);
```

Implementação-instanciação

```
//Envio de mensagem ao objeto AgletMsg comando para o Aglet pegar o ID do servidor
repli = msgProxy.sendMessage(new Message("Hello"));
//variavel recebe a replicação a mensagem enviada, sendo ela o ID do servidor
serveraid = (AgletID)repli;
////com posse do endereço ATP e do ID do servidor pode-se agora pegar o proxy dele
ServidorProxy = getAgletContext().getAgletProxy(destino,serveraid);
//em caso de chegar até aqui é sinal que ocorreu tudo bem para pegar o proxy do servidor
servidor=true;
```

Implementação-multicast

```
//por não saber o proxy do servidor envia para todos os aglets executando neste contexto
ReplySet replies = getAgletContext().multicastMessage(message);
//espera pela resposta
while (replies.hasMoreFutureReplies()){
    FutureReply future = replies.getNextFutureReply();
    try{
        //variavel servidor recebe ID do servidor
        servidor = (AgletID) future.getReply();
        //é respondido a mensagem Hello contendo o ID do Servidor
        msg.sendReply(servidor);
    }catch(Exception e){
        System.out.println("AgletMsg não enviou "+e.getMessage());
    }

}
//termina a função deste aglet e é exterminado
dispose();
```

Implementação-inserepeer

```
//função que insere peer na tabela de peers, recebe como parametro os dados numa string só
public void insere(String msg){
    //declaração de variaveis
    String peerid;
    //tratamento da string
    peerid = msg.substring(msg.indexOf(" - ")+3);
    try{
        //Instanciação da variaveis
        URL adress= new URL(msg.substring(0, msg.indexOf(" - ")));
        AgletID Ident = new AgletID(peerid);
        //pega o proxy do peer que esta se conectando
        proxy = getAgletContext().getAgletProxy(adress,Ident);
    }catch (Exception e){
        envia("Erro na instanciação do objeto proxy AgletServidor/insere");
    }
    //insere dados do peer na tabela
    Object val = tabela.put(peerid,proxy);
    //chama a função refresh que dá um refresh no JTextArea da tela do servidor
    refresh();
    //manda para o peer a tabela de proxy dos peers conectados ao servidor
}
```

Implementação-handlemessage

```
    public boolean handleMessage(Message msg){
    //variavel aid pega o ID do aglet peer
    aid = getAgletID();
    if (msg.sameKind("Hello")){//implementar caso de o servidor ser terminado
        Tela.display((String)msg.getArg());
    }else if (msg.sameKind("Pega ID")){
        //se solicitaram o ID dar uma resposta a mensagem com o ID do Servidor
        msg.sendReply(aid);
    }else if (msg.sameKind("Conectar")){
        //se houve um pedido de conexão inserir na lista de peers
        insere((String)msg.getArg());
        msg.sendReply("\n\nok inserido na tabela do servidor\n\n");
    }else if (msg.sameKind("Desconectar")){
        //Peer foi desconectado, excluir da lista de peers
        Exclui((String)msg.getArg());
    }else if (msg.sameKind("Atualizar")){
        //Peer pedepara atualizar listade peers
        mandaInserir((String)msg.getArg());
        msg.sendReply("\n\nok Atualizado\n\n");
    }else{
        return false;
    }
    return true;
}
```

Implementação-ondisposing

```
//quando o servidor for desconectado

public void onDisposing(){
    //Verifica se tem algum peer conectado
    if (tabela.size()>0){
        //pega os elementos da tabela
        for (Enumeration enum = tabela.elements();enum.hasMoreElements();){
            try{
                envia("Enviando mensagem de desligamento para os peers");
                //Pega o proxy na tabela
                proxyII = (AgletProxy)enum.nextElement();
                //Envia mensagem avisando que o servidor esta sendo desconectado
                proxyII.sendMessage(new Message("ServidorTchau"));
            }catch(Exception e){
                envia("Erro no AgletServidor/onDisposing -->"+e.getMessage());
            }
        }
    }
}
```

Técnicas e Ferramentas

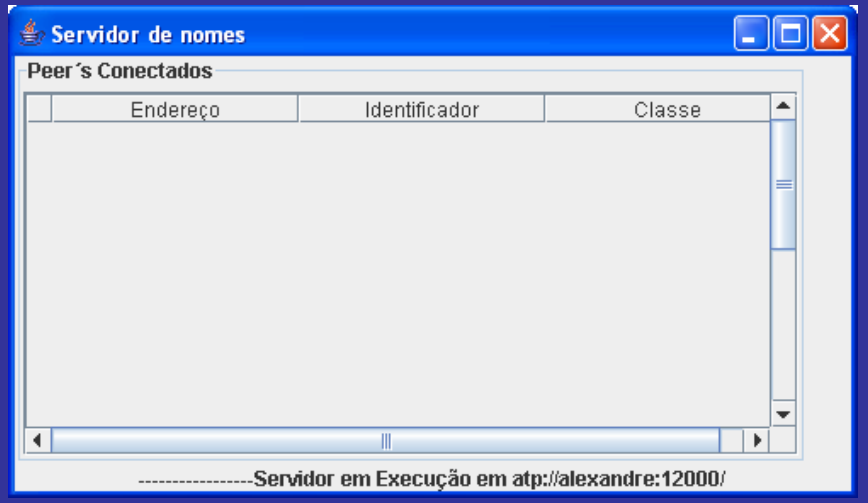
- A linguagem de programação escolhida para o desenvolvimento do trabalho apresentado foi a linguagem Java.
- Verificou-se a necessidade de utilizar um ambiente que possibilite o emprego do conceito de agentes móveis, para isto foi utilizada o *Aglets Software development Kit* ou simplesmente ASDK.

Técnicas e Ferramentas

- Os *Aglets* são executados em um servidor de agentes denominado *Aglet Server*. O *Aglet Server* fornece aos usuários um aplicativo gráfico, denominado *Tahiti*.
- Quanto ao ambiente de desenvolvimento, foi escolhido o *Eclipse* 3.0 para a plataforma *Windows*.

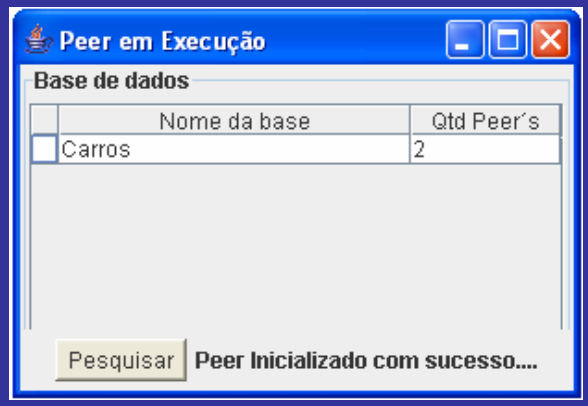
Operacionalidade

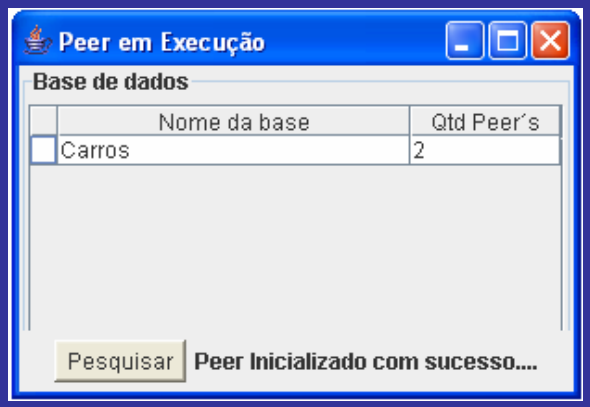
- Em um computador conectado a uma rede, executa-se através do Tahiti o protótipo do Servidor de Nomes.



Operacionalidade

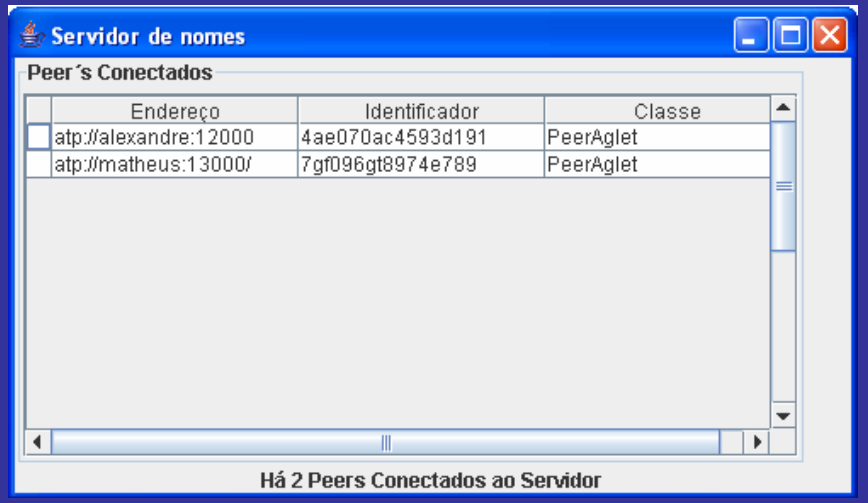
- Em outros computadores, executam-se o protótipo peer, os quais se conectam com o servidor de nomes e disponibilizam suas bases de dados xml para serem pesquisadas.





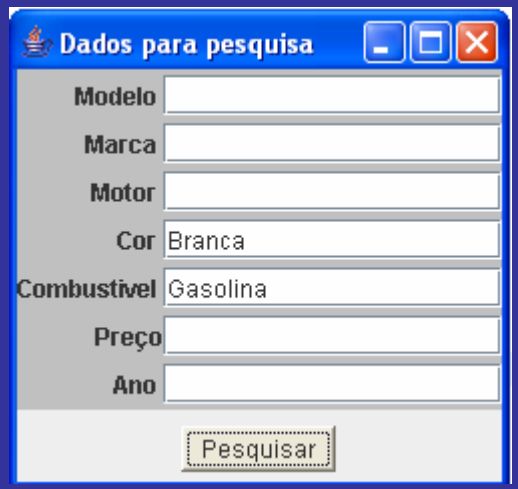
Operacionalidade

- O servidor de nomes após a conexão de cada peer, atualiza sua lista de peers da rede e envia para os peers também terem sua lista e assim não precisar a cada pesquisa do peer, consultar o servidor de nomes.



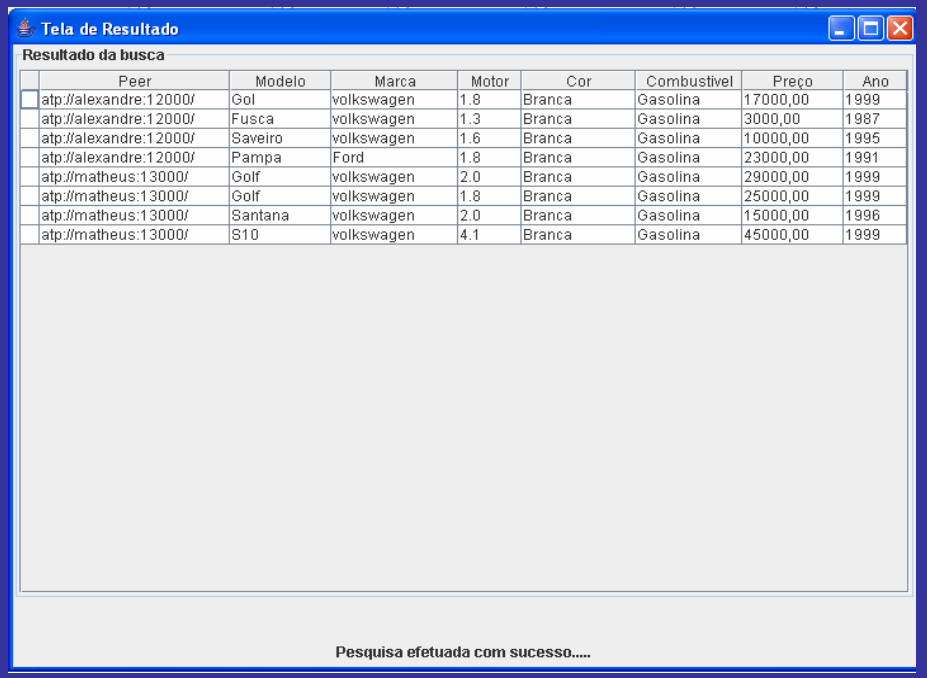
Operacionalidade

- Em posse da lista, os peers podem fazer pesquisas nas bases xml disponibilizadas pelos peers. envia então aglets para pesquisar nas bases xml da rede.



Operacionalidade

- Ao terminar o processamento, os aglet responsáveis pela pesquisa enviam mensagens com o resultado para o peer que solicitou a pesquisa e este então mostrar os resultados ao usuário.



Resultados

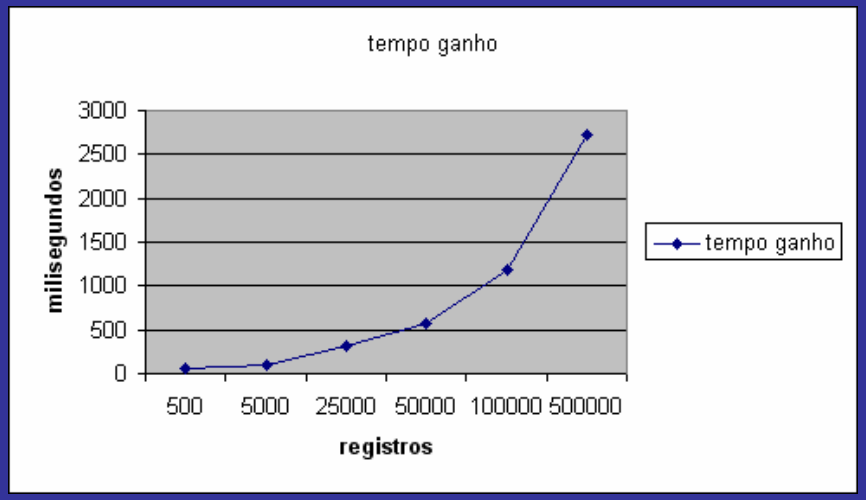
- De acordo com os teste realizados para o estudo de caso apresentado, obteve-se bons resultados em todos os aspectos do protótipo, como no caso da utilização da API do java para leitura dos arquivos XML, pois verificou-se uma boa performance nos resultados dos testes.

Resultados

- Gráfico comparativo entre o tempo gasto por registro em ambos os casos da pesquisa, onde a linha que contém os nodos quadrados referem-se ao tempo gasto por registros executados por um computador e a linha com os nodos triangulares representam o tempo gasto por registro na pesquisa efetuada utilizando os protótipos.

Resultados

- Gráfico que mostra a diferença entre o tempo total gasto na leitura de cada arquivo XML, para os testes realizados em um computador e nos testes realizados utilizando dois computadores em uma rede utilizando os protótipos desenvolvidos neste trabalho.



Conclusão

- Foram implementados dois protótipos: Um que realiza a tarefa de servidor de nomes gerenciando e distribuindo a lista de *peers* da rede para todo computador que executando o protótipo do aplicativo *peer*, forem adicionados à lista do servidor de nomes; Outro que realiza a função de *peer*, conectando-se ao servidor de nomes, disponibilizando seu poder de processamento, trocando mensagens com outros *peers* e executando processamento em arquivos XML.

Conclusão

- Ao término do desenvolvimento do trabalho pode-se concluir que os resultados foram alcançados em relação aos objetivos previamente formulados.

Extensões

- Adaptação do protótipo atual para a leitura de diversas bases de dados XML, de forma a moldar a consulta de acordo com a base escolhida para ser pesquisada.
- desenvolvimento de um servidor de *aglets*, para não utilizar o aplicativo *Tahiti*, diminuindo assim a quantidade de Softwares, aplicativos, *apis* e afins instalados para a utilização do protótipo.
- desenvolver uma forma de balanceamento das cargas, levando em consideração o poder computacional de cada computador da rede.