

Source listing

Mark James Talbot

April 26, 2004

Contents

1	C++ source files	3
1.1	Source File: ./mpi/mpi.cpp	3
1.2	Source File: ./sds/agent.cpp	13
1.3	Source File: ./sds/test.cpp	18
1.4	Source File: ./sds/adu.cpp	19
1.5	Source File: ./sds/sdtext.cpp	23
1.6	Source File: ./lists/prostrlist.cpp	28
1.7	Source File: ./lists/stringlist.cpp	36
1.8	Source File: ./lists/linkedlist.cpp	41
1.9	Source File: ./nodes/node.cpp	49
1.10	Source File: ./rater/rater.cpp	52
1.11	Source File: ./rater/raternode.cpp	54
1.12	Source File: ./httpserv/httpget.cpp	62
1.13	Source File: ./httpserv/httpgetnode.cpp	64
2	C++ header files	81
2.1	Source File: ./mpi/mpi.h	81
2.2	Source File: ./sds/adu.h	84
2.3	Source File: ./sds/agent.h	86
2.4	Source File: ./sds/sdtext.h	88
2.5	Source File: ./lists/prostrlist.h	90
2.6	Source File: ./lists/stringlist.h	93
2.7	Source File: ./lists/linkedlist.h	95
2.8	Source File: ./nodes/node.h	98
2.9	Source File: ./rater/raternode.h	100
2.10	Source File: ./httpserv/httpgetnode.h	102
3	Java source files	105
3.1	Source File: ./mcp/MicroHTTP.java	105
3.2	Source File: ./mcp/PageRequest.java	106
3.3	Source File: ./mcp/httpession.java	110
3.4	Source File: ./mcp/httpserv1.java	113
3.5	Source File: ./mcp/startcluster.java	115
3.6	Source File: ./mcp/urllist.java	124
3.7	Source File: ./mcp/mpi.java	129
3.8	Source File: ./mcp/rateserv.java	134
3.9	Source File: ./mcp/httpserv.java	138
3.10	Source File: ./mcp/mpiserver.java	143
3.11	Source File: ./mcp/message.java	147

3.12	Source File: ./mcp/linkspec.java	149
3.13	Source File: ./mcp/servnode.java	151
4	HTML source files	159
4.1	Source File: ./PHPinterface/template.html	159
4.2	Source File: ./PHPinterface/index.html	160
4.3	Source File: ./PHPinterface/adduser.html	163
4.4	Source File: ./PHPinterface/login.html	164
5	PHP source files	165
5.1	Source File: ./PHPinterface/redirect.php	165
5.2	Source File: ./PHPinterface/adduser.php	166
5.3	Source File: ./PHPinterface/dologin.php	168
5.4	Source File: ./PHPinterface/console.php	169
5.5	Source File: ./PHPinterface/server.php	170
5.6	Source File: ./PHPinterface/dbconnect.php	171
5.7	Source File: ./PHPinterface/adminconsole.php	172
5.8	Source File: ./PHPinterface/start.php	173
5.9	Source File: ./PHPinterface/init.php	174
5.10	Source File: ./PHPinterface/showresults.php	175

1 C++ source files

1.1 Source File: ./mpi/mpi.cpp

```
001 /*
002 ###Source###
003 *****filename*****
mpi.cpp
004 *****auther*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
25/09/2003 14:06:21
009 *****description*****
010 This file contains the implementation to the message
011 passing system of the project. It contains two classes, one
012 for the client side and one for the server side.
013 #####
014 */
015 // This program is free software; you can redistribute it and/or modify
016 // it under the terms of the GNU General Public License as published by
017 // the Free Software Foundation; either version 2 of the License, or
018 // (at your option) any later version.
019 //
020 // This program is distributed in the hope that it will be useful,
021 // but WITHOUT ANY WARRANTY; without even the implied warranty of
022 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
023 // GNU General Public License for more details.
024 //
025 // You should have received a copy of the GNU General Public License
026 // along with this program; if not, write to the Free Software
027 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
```

```
028
029
030 #include "mpi.h"
031 #include <unistd.h>
032 #include <signal.h>
033 #include <sys/wait.h>
034 #include <sys/types.h>
035
036 // Recives exit codes from child threads
037
038 /*
039 Recives the exit code from child process and kills the server
040 if exit code 10 is received (sent when kill message is received)
041 */
042 void childwait(int sig)
043 {
044     signal(SIGCHLD,childwait);
045     int exitcode;
046     wait(&exitcode);
047     cout << exitcode <<endl;
048     if (exitcode==10)
049         exit(0);
050 }
051
052 ////////////////////////////////////////
053 // Message Class //
054 ////////////////////////////////////////
055
056 // deconstructor
057
058 /*
059 Deletes the memory assigned to name and data
060 */
```

```
061 message::~message()
062 {
063     if (name!=NULL)
064         delete name;
065     if (data!=NULL)
066         delete data;
067 }
068
069 // Returns the name
070
071 /*
072 does what it says on the can
073 */
074 char * message::get_name()
075 {
076     return name;
077 }
078
079 // Returns the data
080
081 /*
082 does what it says on the can
083 */
084 char * message::get_data()
085 {
086     return data;
087 }
088
089 // sets the name
090
091 /*
092 deletes the memory assign to a previous name and
093 then copies the input string into the name
```

```
094 */
095 void message::set_name(char * in)
096 {
097     if (name!=NULL)
098         delete name;
099     name=strdup(in);
100 }
101
102 // sets the data
103
104 /*
105  deletes the memory assign to a previous data and
106  then copies the input string into the data
107 */
108 void message::set_data(char * in)
109 {
110     if (data!=NULL)
111         delete data;
112     data=strdup(in);
113 }
114
115 ////////////////////////////////////////
116 //                                     MpiClient Class //
117 ////////////////////////////////////////
118
119
120 mpiclient::mpiclient(unsigned int port)
121 {
122     open=0;
123     lport=port;
124     recvtxt=strdup("");
125     signal(SIGCHLD,childwait);
126 }
```

```
127
128
129 mpiclient::~mpiclient()
130 {
131     closeup();
132     close(servfd);
133 }
134
135
136 int mpiclient::attach()
137 {
138     if (open==0)
139     {
140         struct sockaddr_in self;
141         servfd = socket(AF_INET, SOCK_STREAM, 0);
142         self.sin_family = AF_INET;
143         self.sin_port = htons(lpport);
144         self.sin_addr.s_addr = INADDR_ANY;
145         bind(servfd, (struct sockaddr*)&self, sizeof(self));
146     }
147     if ((open==0)||!(open==2))
148     {
149         socklen_t addrlen;
150         listen(servfd, 20);
151         struct sockaddr_in client_addr;
152         addrlen=sizeof(client_addr);
153         while (1)
154         {
155             sockfd = accept(servfd, (struct sockaddr*)&client_addr, &addrlen);
156             if (socketfd>0)
157             {
158                 if (fork()==0)
159                 {
```

```
160     open=1;
161     return open;
162 }
163 else
164     close (socketfd);
165 }
166 }
167 return open;
168 }
169
170
171 void mpiclient::closeup()
172 {
173     if (open)
174     {
175         close(socketfd);
176         open=2;
177     }
178     free(recvtext);
179     recvtext=NULL;
180     exit(0);
181 }
182
183
184 int mpi::send(char * data)
185 {
186     if (open)
187     {
188         write(socketfd,data,strlen(data));
189     }
190     return open;
191 }
```

```
192
193
194 int mpi::recv(char * data, unsigned int length)
195 {
196     int reed;
197     reed=0;
198     if (open)
199     {
200         reed=read(socketfd,data,length);
201     }
202     return reed;
203 }
204
205
206 void mpi::send_message(message * data)
207 {
208     if (data!=NULL)
209     {
210         char * temp;
211         send("<?xml version='1.0'?'>\n<message name='");
212         temp=data->get_name();
213         send(temp);
214         send("\','>");
215         temp=data->get_data();
216         send(temp);
217         send("</message>\n");
218     }
219 }
220
221
222 message * mpi::recv_message()
223 {
224     char buf[1025],* temp2, * temp3;
```

```
225 int recved;
226 message * ret;
227 free(recvtext);
228 recvtext=(char *)malloc(sizeof(char));
229 *recvtext=0;
230 while (strstr(recvtext,"</")!=NULL)
231 {
232     recved=recv(buf,1024);
233     buf[recved]=0;
234     recvtext=(char *)realloc(recvtext,sizeof(char)*(strlen(recvtext)+recved+1));
235     strcat(recvtext,buf);
236 }
237 temp2=strstr(recvtext,"<message name=\\',',");
238 if (temp2!=NULL)
239 {
240     ret=new message;
241     temp2+=15;
242     temp3=strstr(temp2,"\\',',");
243     if (temp3!=NULL)
244     {
245         *temp3=0;
246         ret->set_name(temp2);
247         temp3+=2;
248         temp2=strstr(temp3,"</message>");
249         if (temp2!=NULL)
250         {
251             *temp2=0;
252             ret->set_data(temp3);
253             cout << temp3 <<endl;
254             free(recvtext);
255             recvtext=strdup("");
256         }
257     }
258     else
```

```
257     {
258         delete ret;
259         ret=NULL;
260         ret=new message;
261         ret->set_name("wtf?");
262         ret->set_data("");
263         this->send_message(ret);
264         delete ret;
265         ret=NULL;
266         delete recvtext;
267         recvtext=strdup("");
268         std::cout<<"error in message\n";
269     }
270 }
271 else
272 {
273     delete ret;
274     ret=NULL;
275     ret=new message;
276     ret->set_name("wtf?");
277     ret->set_data("");
278     this->send_message(ret);
279     delete ret;
280     ret=NULL;
281     delete recvtext;
282     recvtext=strdup("");
283     std::cout<<"error in message\n";
284 }
285 else
286 {
287     ret=new message;
288     ret->set_name("wtf?");
```

```
288 ret->set_data("");
289 this->send_message(ret);
290 delete ret;
291 ret=NULL;
292 delete rcvtext;
293 rcvtext=strdup("");
294 std::cout<<"error in message\n";
295 }
296 return ret;
297 }
298
299
```

1.2 Source File: ./sds/agent.cpp

```
001 // agent.cpp: implementation of the agent class.
002 //
003 ///////////////////////////////////////////////////////////////////
004
005
006 #include "agent.h"
007
008 ///////////////////////////////////////////////////////////////////
009 // Construction/Destruction
010 ///////////////////////////////////////////////////////////////////
011
012 // sets up a agent
013
014 /*
015 takes the base pointer to the array of adu's that
016 make up the model and one for the search space.
017 A base pointer is also passed for the base of the
018 array of agents. alos passed to this function are
019 the lengths of these three arrays and the postion
020 of the first character of the last word found
021 */
022 agent::agent(adu ** basemodel, adu ** basesearch, agent ** basepos, int modelsize, int searchsize, int agentsize, int pbase)
023 {
024     double randnum;
025     lbasemodel=basemodel;
026     lbase=pbase;
027     lbasesearch=basesearch;
028     lmodelsize=modelsize;
029     lsearchsize=searchsize;
030     base=basepos;
031     size=agentsize;
032     active=0;
```

```

033  randnum=(double)rand()/((double)(RAND_MAX)); //creates a random number between 0 and 1
034  searchpos=(int)(randnum*(lsearchsize-lmodelsize)); //uses the privously created random number and works out a random
starting postion in the search space
035 }
036
037 // returns the activation level of the agent
038 int agent::getactive()
039 {
040     return active;
041 }
042
043 // returns the postion in the search space
044 int agent::getsearch()
045 {
046     return searchpos;
047 }
048
049 // does the test faze of sds
050
051 /*
052  checks a random adu from the model with one at the
053  distance releative to an offset and sees if they are
054  the same. If they are this agent then becomes active
055  and will diffuse its information to the rest of the
056  agents.
057 */
058 int agent::test()
059 {
060     int offset;
061     double randnum;
062     int freespace,distance;
063     char model,search;
064     randnum=(double)rand()/((double)(RAND_MAX)); //generates a random number between 0 and 1

```

```

065 freespace=(lsearchsize-searchpos);//works out the amount of space between the agents postion in the search space and
the end of the search space
066 if (freespace<lmodelsize)
067   offset=(int)(randnum*freespace);//less space than the model so would be incomplete model so pick random postion
within the amount of free space
068 else
069   offset=(int)(randnum*lmodelsize);//plenty of space so pick random number anywhere in the model
070 if (lbasearch[searchpos]->gettag()==0)//continue as long as the location of this agent has not been marked as
found
071 {
072   model=lbasemodel[offset]->get();
073   search=lbasearch[searchpos+offset]->get();
074   if (model==search)//compare the chars from the model and the search space
075     {
076       if (active==0)
077         lbasearch[searchpos]->enter();//if agent only just truned active set the enter command for this adu
078         active=1;
079       }
080     else
081       {
082         if (active==1)
083           lbasearch[searchpos]->exit();
084         active=0;
085       }
086     }
087   if (active==1)
088     lbasearch[searchpos]->exit();
089   active=0;
090   return active;
091 }

```

```
092
093 // data diffusen stage of sds
094
095 /*
096 if the agent is inactive it will randomly pick another
097 agent and if that agent is active this agent will move
098 to that postion. If not it will randomly reposition itself
099 within the search space.
100 */
101 void agent::diffuse()
102 {
103     int offset;
104     double randnum;
105     agent * master;
106     if (!active) //if unactive
107     {
108         randnum=(double)rand()/(double)(RAND_MAX);
109         offset=(int)(randnum*size);//pick a random agent
110         master=base[offset];
111         while (master==this)
112         {
113             randnum=(double)rand()/(double)(RAND_MAX);
114             offset=(int)(randnum*size);
115             master=base[offset];
116         }
117         if (master->getactive())
118             searchpos=master->getsearch();//if agent active move to its location
119         else
120             this->init();//else pick a random location
121     }
122 }
123 // sets agent to a random postion
```

```
124 void agent::init()
125 {
126     double randnum;
127     active=0;
128     randnum=(double)rand()/(double)(RAND_MAX);
129     searchpos=(int)(randnum*(lsearchsize-lmodelsize));
130 }
131
132
```

1.3 Source File: ./sds/test.cpp

```
001 #include "sdstext.h"
002 #include <iostream>
003
004 using namespace std;
005
006 int main()
007 {
008     sdstext test=sdstext("one","one two three one");
009     char * demo = test.start('A');
010     cout << demo << endl;
011     return 0;
012 }
013
014
```

1.4 Source File: ./sds/adu.cpp

```
001 // adu.cpp: implementation of the adu class.
002 //
003 ///////////////////////////////////////////////////////////////////
004
005
006 #include "adu.h"
007
008 ///////////////////////////////////////////////////////////////////
009 // Construction/Destruction
010 ///////////////////////////////////////////////////////////////////
011 adu::adu(char data)
012 {
013     me=data;
014     actives=0;
015     tag=0;
016     lstable=0;
017 }
018 adu::~adu()
019 {
020
021 }
022
023 // sets the strong halting criteria
024
025 /*
026 This function sets the strong halting criteria propertyts of the sds system
027 halt_min is the minimum number of active agents on this adu to be considered to be stable
028 halt_max is the maximum number of active agents on this adu to be considered to be stable
029 halt_time is the number of itterations to remain within these boundrys to be a match
030 */
031 void adu::set_halt(int halt_min,int halt_max,int halt_time)
032 {
```

```
033  lhalt_min=halt_min;
034  lhalt_max=halt_max;
035  lhalt_time=halt_time;
036 }
037
038 // new agent becoming stable at this location
039
040 /*
041  When a agent changes from an inactive to an active state this function
042  is called on the adu the agent is sat on to count the number of active
043  agents on a adu.
044 */
045 int adu::enter()
046 {
047   return actives++;
048 }
049
050 // agent has become unstable at this point
051
052 /*
053  This function is for the opposite condition of an agent going from
054  active to inactive on a adu
055 */
056 int adu::exit()
057 {
058   return actives--;
059 }
060
061 // returns the data contained in the adu
062 char adu::get()
063 {
064   return me;
065 }
```

```
066
067 // returns the number of active agents on this adu
068 int adu::getactives()
069 {
070     return actives;
071 }
072
073 // marks an adu as found
074
075 /*
076  If a adu is considered to be a match this function is called to set
077  a tag bit on. This is then checked when a agent attempts to compair
078  at this location and causes it to fail its test and becaome inactive.
079 */
080 void adu::settag()
081 {
082     tag=1;
083 }
084
085 // This function returns the tag it for marking pattern matches
086 int adu::gettag()
087 {
088     return tag;
089 }
090
091 // returns wether the adu is stable or not
092
093 /* This function calculates the strong halting criteria that has
094 been set on this adu and returns true if it has been met and
095 false if it has not. This is used to work out if this adu is
096 the root of a match
097 */
098 int adu::is_stable()
```

```
099 {
100   if ((actives>lhalt_min)&&(actives<lhalt_max)) // This is true if the number of active agents is within the halting
boundrys
101     lstable++; //Increasing the record of number of interations considered consecutivly stable
102   else
103     lstable=0; //or zeroing it if it has failed the test
104   if (lstable>lhalt_time) // This works out if there is a match or not
105     return 1;
106   else
107     return 0;
108 }
```

1.5 Source File: ./sds/sdstext.cpp

```
001 #include "sdstext.h"
002
003 // Constructor
004
005 /*
006 sets up the system with a string for the model and one for
007 the search space
008 */
009 sdstext::sdstext(char * model, char * search)
010 {
011     lmodel=new char[strlen(model)+1];
012     lsearch=new char[strlen(search)+1];
013     strcpy(lmodel,model);
014     strcpy(lsearch,search);
015     base=0;
016 }
017
018 // Constructor
019
020 /*
021 sets up the search system with blank strings
022 */
023 sdstext::sdstext()
024 {
025     lmodel=new char[1];
026     lsearch=new char[1];
027     strcpy(lmodel,"");
028     strcpy(lsearch,"");
029     base=0;
030 }
031
032 // Destructor
```

```

033
034 /*
035 unallocates the memory assigned to the strings used to store
036 the model and search space
037 */
038 sdstext::~sdstext()
039 {
040     delete lmodel;
041     delete lsearch;
042 }
043
044 // Cahnges the text stored in the model and search space strings
045 void sdstext::set_text(char * model, char * search)
046 {
047     delete lmodel;
048     delete lsearch;
049     lmodel=new char[strlen(model)+1];
050     lsearch=new char[strlen(search)+1];
051     strcpy(lmodel,model);
052     strcpy(lsearch,search);
053 }
054
055 // crates and initiales the agent array
056
057 /*
058 takes a pointer to the base of the model and search space adu array,
059 thier lengths, a pointer to a integer for the length of the agent
060 array and the postion of the first char of the last word in the model
061 found.
062 */
063 agent ** sdstext::create_agent_array(adu ** basemodel,adu ** basearch,int modelsize,int searchsize,int *
agentsize,int pbase)
064 {

```

```
065 agent ** agentbase;
066 int c;
067 *agentsize=(int)(modelsize)*30;//works out the number of agents to setup
068 agentbase=(class agent **)calloc(*agentsize,sizeof(class agent *));//allocates enough memory to hold the agent
array.
069 for (c=0;c<*agentsize;c++)
070 agentbase[c]=new agent(basemodel,basearch,agentbase,modelsize,searchsize,*agentsize,pbase);//crates a agent
071 return agentbase;
072 }
073
074 // converts a string to an adu representaion
075
076 /*
077 takes a string and a pointer to an int to store the length
078 of the created array. Creates a adu class for each char in
079 ther passed string.
080 */
081 adu ** sdstext::create_adu_array(char * string,int * length)
082 {
083 adu ** base;
084 int c;
085 *length=strlen(string);//works out size of adu array from size of string
086 base=(class adu **)calloc(*length,sizeof(class adu *));//allocates memory to hold the adu array
087 for (c=0;c<*length;c++)
088 base[c]=new adu(string[c]);//creates an adu class for char at postion c in the string
089 return base;
090 }
091
092 // runs a search
093
094 /*
095 takes a integer boolean that causes it to ether find all incarantions
096 of the first word in the model or to just do the one closest to the
```

```

097 base pointer for the last word and returns the rateing for everything
098 after this point in the recersive chain.
099 */
100 char* sdstext::start(char replacechar)
101 {
102     adu ** modelbase;
103     adu ** searchbase;
104     agent ** agentbase;
105     sdstext * recurse;
106     char * word;
107     char * rest;
108     int modelsize, searchsize, agentsize, c, d, rating, wordlength, thischain;
109     double nextword;
110     srand(time(NULL));
111     modelbase=create_adu_array(lmodel,&modelsize);//converts the word to be looked for into a adu chain
112     searchbase=create_adu_array(lsearch,&searchsize);//converts the search space into an array of adu's
113     agentbase=create_agent_array(modelbase,searchbase,modelsize,searchsize,&agentsize,&agentsize,base);//creates and intialises the
agents
114     for (c=0;c<searchsize;c++)
115     {
116         searchbase[c]->set_halt(modelsize*2,agentsize,10);//sets up the halting critera for this search.
117     }
118     d=0;
119     rating=-1;
120     while (d<1000)//limits the system to 1000 iterations to find a target
121     {
122         d++;
123         for (c=0;c<agentsize;c++)
124             agentbase[c]->test();//performs the test faze of sds
125         for (c=0;c<agentsize;c++)
126             agentbase[c]->diffuse();//performs the diffuse faze of sds
127         for (c=0;c<searchsize;c++)
128             if (searchbase[c]->is_stable())//the search word has been found at point c

```

```
129 {
130     cout <<"stable word found"<<endl;
131     d=0;
132     searchbase[c]->settag();
133 }
134 }
135 char * rett=new char[searchsize+1];
136 strcpy(rett,lsearch);
137 for (c=0;c<modelsize;c++)
138     delete(modelbase[c]);
139 for (c=0;c<searchsize;c++)
140 {
141     if (searchbase[c]->gettag()==1)
142     {
143         rett[c]=replacechar;
144     }
145     delete(searchbase[c]);
146 }
147 for (c=0;c<agentsize;c++)
148     delete(agentbase[c]);
149 free(modelbase);
150 free(searchbase);
151 free(agentbase);
152 return rett;
153 }
154
155
156
```

1.6 Source File: ./lists/prostrlist.cpp

```
001 /*
002 ###Source###
003 *****filename*****
prostrlist.cpp
004 *****author*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
Thu Nov 13 23:05:39 GMT 2003
009 *****description*****
010 based on the string list but allowing fo promotions bassed
011 on stringed tags of orignation
012 #####
013 */
014 // This program is free software; you can redistribute it and/or modify
015 // it under the terms of the GNU General Public License as published by
016 // the Free Software Foundation; either version 2 of the License, or
017 // (at your option) any later version.
018 //
019 // This program is distributed in the hope that it will be useful,
020 // but WITHOUT ANY WARRANTY; without even the implied warranty of
021 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
022 // GNU General Public License for more details.
023 //
024 // You should have received a copy of the GNU General Public License
025 // along with this program; if not, write to the Free Software
026 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
027
028 #include "prostrlist.h"
029
030
```

```
031 ////////////////////////////////////////////////////////////////////
032 //                urlnode //
033 ////////////////////////////////////////////////////////////////////
034
035 // Constructor
036
037 /*
038  sets up a new page linkage entry
039 */
040 urlnode::urlnode()
041 {
042     from=new stringlist();
043     rate=1;
044     name=NULL;
045 }
046
047 // Destructor
048
049 /*
050  deletes the memory assigned to the class
051 */
052 urlnode::~urlnode()
053 {
054     delete from;
055     delete name;
056 }
057 }
058
059 // sets the url of the page
060
061 /*
062  allocates memory on the heap for the page name
063 */
```

```
064 void urlnode::set_page(char * page)
065 {
066     if (name!=NULL)
067         delete name;
068     name=new char[strlen(page)+1];
069     strcpy(name,page);
070 }
071
072 //////////////////////////////////////
073 //          prostrlist          //
074 //////////////////////////////////////
075
076 //  adds a lineage between a parrent and a child page
077
078 /*
079  looks to see if the page is already in the list
080  and adds the parrent page to it else if it dosnt
081  exist it is added to the list and then the parrent
082  is linked to it.
083 */
084 void prostrlist::addlink(char * page,char * from)
085 {
086     node * temp;
087     urlnode * temp3;
088     temp=getlink(page);
089     if (temp!=NULL)
090     {
091         temp3=(urlnode *)temp->data;
092         if (temp3->from->search(from)==0)
093         {
094             temp3->from->add(from,0);
095         }
096     }
}
```

```
097 else
097 {
098     temp3=new urlnode();
099     temp3->set_page(page);
100     temp3->from->add(from,0);
101     linkedlist::add((void *)temp3,0);
102     total++;
103 }
104 }
105
106 //
107
108 /*
109
110 */
111 int prostrlist::findlink(char * page)
112 {
113     if (getlink(page)!=NULL)
114         return 1;
115     else
116         return 0;
117
118 // picks a random element
119
120 /*
121     picks a element at random from the list
122     but a element has a probablty proportianl
123     to its rating. This is the function that
124     gives the system its soft focused property.
125 */
126 int prostrlist::get_random()
127 {
```

```
128 double pos;
129 node * temp;
130 temp=head;
131 urlnode * temp2;
132 int c=1;
133 pos=(rand()/RAND_MAX)*total;
134 while (temp!=NULL)
135 {
136     temp2=(urlnode *)temp->data;
137     pos=pos-temp2->get_rating();
138     if (pos<0)
139         return c;
140     temp=temp->next;
141     c++;
142 }
143 return c;
144 }
145
146 // gets the name of a page from the list
147
148 /*
149 gets an element from the list and returns its name
150 */
151 char * prostrlist::get(int pos)
152 {
153     urlnode * temp=(urlnode *)linkedlist::get(pos);
154     if (temp!=NULL)
155         return temp->get_page();
156     else
157         return "";
158 }
159 // promotes some pages
```

```
160
161 /*
162  adds a rating to a parrent page to all the
163  pages that that page links to hence making
164  them more or less likely to be picked at
165  random.
166 */
167 void prostrlist::promote(char * from,int rating)
168 {
169  node * temp;
170  temp=head;
171  urlnode * temp2;
172  while (temp!=NULL)
173  {
174    temp2=(urlnode *)temp->data;
175    if (temp2->from->search(from)==1)
176    {
177      total=total-temp2->get_rating();
178      temp2->set_rating(rating);
179      total=total+temp2->get_rating();
180    }
181    temp=temp->next;
182  }
183 }
184
185 // gets a node
186
187 /*
188  returns the node that has the url of page
189 */
190 node * prostrlist::getlink(char * page)
191 {
192  node * temp;
```

```
193 temp=head;
194 char * temp2;
195 while (temp!=NULL)
196 {
197     temp2=((urlnode *)temp->data)->get_page();
198     if (strcmp(page,temp2)==0)
199     {
200         return temp;
201     }
202     temp=temp->next;
203 }
204 return NULL;
205 }
206
207 // deletes an item from the list
208
209 /*
210  unallocates the memory assigned to a link and to its
211  parrent link structure.
212 */
213 int prostrlist::remove(int pos)
214 {
215     urlnode * temp;
216     temp=(urlnode *)this->linkedlist::get(pos);
217     if (temp!=NULL)
218     {
219         delete(temp);
220         return this->linkedlist::remove(pos);
221     }
222     else
223     return 0;
224 }
```

```
225 // Destructor
226
227 /*
228  collapses the list yusing the new delete code.
229 */
230 prostrlist::~prostrlist()
231 {
232     while(remove(1)!=0);
233 }
234
235
```

1.7 Source File: ./lists/stringlist.cpp

```
001 /*
002 ###Source###
003 *****filename*****
stringlist.c++
004 *****author*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
23/09/2003 19:45:41
009 *****description*****
010 This file contains the implimentation of a linked list of
011 strings. It is inherited from the generic linked list in
012 linkedlist.h and linkedlist.c++
013 #####
014 */
015 // This program is free software; you can redistribute it and/or modify
016 // it under the terms of the GNU General Public License as published by
017 // the Free Software Foundation; either version 2 of the License, or
018 // (at your option) any later version.
019 //
020 // This program is distributed in the hope that it will be useful,
021 // but WITHOUT ANY WARRANTY; without even the implied warranty of
022 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
023 // GNU General Public License for more details.
024 //
025 // You should have received a copy of the GNU General Public License
026 // along with this program; if not, write to the Free Software
027 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
028
029
030 #include "stringlist.h"
```

```
031
032 // Destructor
033
034 /*
035 reimplitnts the base classes destructor so that
036 the new remove code is used that deletes the
037 space assigned to the string.
038 */
039 stringlist::~stringlist()
040 {
041     while(remove(1)!=0);
042 }
043
044 // adds a string to the list
045
046 /*
047 assigns space on the heap for the string passed
048 and then adds that string to the base classes
049 pointer list.
050 */
051 unsigned int stringlist::add(char * data, int end)
052 {
053     char * temp;
054     temp=new char[strlen(data)+1];//allocates memory on the heap for the string
055     strcpy(temp,data);//copies the passed string onto the heap
056     return this->linkedlist::add((void *)temp,end);//adds the pointer to the new string into the pointer list
057 }
058
059 // inserts a string into the list
060
061 /*
062 assigns space on the heap for the string passed
063 and then inserts that string to the base classes
```

```
064 pointer list at the correct point.
065 */
066 unsigned int stringlist::insert(char * data, unsigned int pos)
067 {
068     char * temp;
069     temp=new char[strlen(data)+1];//allocates memory on the heap for the string
070     strcpy(temp,data);//copies the passed string onto the heap
071     return this->linkedlist::insert((void *)temp,pos);//inserts the pointer to the new string into the pointer list at
the correct point
072 }
073
074 // Returns the string at a given point
075
076 /*
077 uses the base classes get function to retrieve the pointer
078 of the list for a string and then casts it to a char pointer
079 */
080 char * stringlist::get(unsigned int pos)
081 {
082     return (char *) (linkedlist::get(pos));
083 }
084
085 //
086
087 /*
088
089 */
090 unsigned int stringlist::remove(unsigned int pos)
091 {
092     char * temp;
093     unsigned int ret;
094     temp=get(pos);
095     if (temp!=NULL)
```

```

096 {
097     delete(temp);
098     ret=this->linkedlist::remove(pos);
099 }
100 else
101     ret=0;
102 return ret;
103 }
104 // converts the list into a delimited string
105
106 /*
107 converts the list into a string using the passed
108 string to separate each entry
109 */
110 char * stringlist::flatern(char * seperator)
111 {
112     node * temp;
113     int llength=0;
114     char * ret;
115     temp=head;
116     while (temp!=NULL) // works out the length of the string needed to store the list
117     {
118         llength+=strlen((char*)temp->data);
119         temp=temp->next;
120     }
121     ret=new char[llength+1+(length*strlen(seperator))]; // allocates the memory needed for the string
122     ret[0]=0;
123     temp=head;
124     while (temp!=NULL)//loops through the list
125     {
126         strcat(ret,(char*)temp->data); //concatenates the string at this point of the list to the string
127         strcat(ret,seperator); // adds the seperator

```

```
128     temp=temp->next;//moves onto the next point on the list
129 }
130 return ret;
131 }
132
133 // finds the index of a string on the list
134
135 /*
136  takes a string and loops through the list until it finds the
137  first entry that is the same and returns 1 or returns 0;
138 */
139 int stringlist::search(char * model)
140 {
141     node * temp;
142     temp=head;
143     char * temp2;
144     while (temp!=NULL) //loops through the list
145     {
146         if (temp->data!=NULL)
147         {
148             temp2=((char *)temp->data);
149             if (strcmp(model,temp2)==0) //if they model and the string at this point in the list are the same then return 1
150             {
151                 return 1;
152             }
153             delete temp2;
154         }
155         temp=temp->next;//move onto next entry on the list
156     }
157     return 0;
158 }
159
160
```

1.8 Source File: ./lists/linkedlist.cpp

```
001 /*
002 ###Source###
003 *****filename*****
LinkedList.cpp
004 *****auther*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
15/09/2003 14:12:37
009 *****description*****
010 Implementation of a generic linkedlist
011 #####
012 */
013 // This program is free software; you can redistribute it and/or modify
014 // it under the terms of the GNU General Public License as published by
015 // the Free Software Foundation; either version 2 of the License, or
016 // (at your option) any later version.
017 //
018 // This program is distributed in the hope that it will be useful,
019 // but WITHOUT ANY WARRANTY; without even the implied warranty of
020 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
021 // GNU General Public License for more details.
022 //
023 // You should have received a copy of the GNU General Public License
024 // along with this program; if not, write to the Free Software
025 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
026
027
028 #include "linkedlist.h"
029
030
```

```
031 //Implimentation of the c string function strdup that is not present in the c++ version
032
033 /*
034 Takes a string and returns a copy of it placed upon the heap in new memory.
035 */
036 char * strdup(char * data)
037 {
038     char * temp;
039     temp=new char[strlen(data)+1];
040     if (temp!=NULL)
041         strcpy(temp,data);
042     return temp;
043 }
044
045
046 // Constructor of linkedlist
047
048 /*
049 creates a instataion of a void pointer linked list
050 sets the list top be empty;
051 */
052 linkedlist::linkedlist()
053 {
054     head=NULL;
055     tail=NULL;
056     length=0;
057 }
058
059 // Destructor of linkedlist
060
061 /*
062 Colapses the list freeing all the assigned pointers and memory.
063 */
```

```

064 linkedlist::~linkedlist()
065 {
066     while(remove(1)!=0);
067 }
068
069 // adds element to the list
070
071 /*
072 Adds the void pointer data to the list at the end set by end.
073 if end is true it is placed after the tail of the list, or it
074 is placed at the head of the list. The function returns the
075 place that the element was added on the list.
076 */
077 unsigned int linkedlist::add(void * data, int end)
078 {
079     node * temp;
080     unsigned int ret;
081     temp=new node;
082     ret=0;
083     if (temp!=NULL)
084     {
085         temp->data=data;
086         if (head==NULL) //inserts as the head and tail if the list was privosly empty
087         {
088             temp->next=NULL;
089             temp->prev=NULL;
090             length=1;
091             head=temp;
092             tail=temp;
093             ret=1;
094         }
095         else if (end) //adds element to the end of the list
096         {

```

```

097 temp->next=NULL;
098 temp->prev=tail;
099 tail->next=temp;
100 length++;
101 tail=temp;
102 ret=length;
103 }
104 else // adds element to the head of the list
105 {
106     temp->next=head;
107     temp->prev=NULL;
108     head->prev=temp;
109     length++;
110     head=temp;
111     ret=1;
112 }
113 }
114 return ret;
115 }
116
117 // Inserts an element at position pos
118
119 /*
120 Inserts the void pointer data before the element pos if it exists hence taking that position
121 else it retruns with 0 indicating a error. The function returns the position that it has added
122 the new elemen to.
123 */
124 unsigned int linkedlist::insert(void * data, unsigned int pos)
125 {
126     node * temp, * temp2;
127     int ret;
128     temp=getnode(pos); //return the element to place data before
129     ret=0;

```

```

130 if (temp!=NULL) //test if element exists
131 {
132     if (temp==head) //use the add fuction if this is the head of the list
133     {
134         ret=add(data,0);
135         if (ret)
136             length++;
137     }
138     else if (temp==tail) //use the add fuction if this is the tail of the list
139     {
140         ret=add(data,1);
141         if (ret)
142             length++;
143     }
144     else //inserts the elemenet into the middle of the list
145     {
146         temp2=new node;
147         if (temp2!=NULL)
148         {
149             temp2->data=data;
150             temp2->next=temp; //next 4 lines sets up the correct order of the list
151             temp2->prev=temp->prev;
152             temp->prev->next=temp2;
153             temp->prev=temp2;
154             length++;
155             ret=pos;
156         }
157         else
158             ret=0;
159     }
160     return ret;
161 }

```

```

162
163 // Returns the whole data structure for a node in the list
164
165 /*
166 Returns the pointer to the node at position pos or NULL if that doesnt exist.
167 */
168 node * linkedlist::getnode(unsigned int pos)
169 {
170     node * temp;
171     unsigned int c;
172     temp=NULL;
173     c=1;
174     if ((pos>0)&&(pos<=length))
175     {
176         if (pos<(length / 2))
177         {
178             temp=head;
179             while (c!=pos)
180             {
181                 c++;
182                 temp=temp->next;
183             }
184         }
185         else
186         {
187             c=length;
188             temp=tail;
189             while (c!=pos)
190             {
191                 c--;
192                 temp=temp->prev;
193             }

```

```

194 }
195 return temp;
196 }
197
198 // Returns the pointer to the data placed on the list at a specific position
199
200 /*
201 uses get node to find the correct element and returns that elements data
202 or NULL if the element doesnt exist.
203 */
204 void * linkedlist::get(unsigned int pos)
205 {
206     node * temp;
207     void * ret;
208     temp=NULL;
209     ret=NULL;
210     temp=getnode(pos);
211     if (temp!=NULL)
212         ret=temp->data;
213     return ret;
214 }
215
216 // Deletes a node
217
218 /*
219 removes node pos and redirects the list around it.
220 does not delete the data stored at the pointer as this data is not contained on this list
221 */
222 unsigned int linkedlist::remove(unsigned int pos)
223 {
224     node * temp;
225     int ret;
226     temp=getnode(pos);

```

```
227 ret=0;
228 if (temp != NULL)
229 {
230     if (temp == head) //deletes the head and move sthe head pointer to the next element
231     {
232         head=head->next;
233         if (head!=NULL)
234             head->prev=NULL;
235         delete temp;
236     }
237 }
238 else if (temp == tail) //deletes the tail and moves the tail pointer to the previous element
239 {
240     tail=tail->prev;
241     if (tail!=NULL);
242     tail->next=NULL;
243     delete temp;
244 }
245 else //deletes an element form the middle of a list.
246 {
247     temp->prev->next=temp->next;
248     temp->next->prev=temp->prev;
249     delete temp;
250 }
251 length--;
252 ret=pos;
253 }
254 return ret;
255 }
256
257
```

1.9 Source File: ./nodes/node.cpp

```
001 /*
002 ###Source###
003 *****filename*****
node.c++
004 *****author*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
06/10/2003 16:39:50
009 *****description*****
010 This file contains the generic implementation of a node.
011 This class is then inherited to define the specific
012 behaviour of a node;
013 #####
014 */
015 // This program is free software; you can redistribute it and/or modify
016 // it under the terms of the GNU General Public License as published by
017 // the Free Software Foundation; either version 2 of the License, or
018 // (at your option) any later version.
019 //
020 // This program is distributed in the hope that it will be useful,
021 // but WITHOUT ANY WARRANTY; without even the implied warranty of
022 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
023 // GNU General Public License for more details.
024 //
025 // You should have received a copy of the GNU General Public License
026 // along with this program; if not, write to the Free Software
027 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
028 #include "node.h"
029
030
```

```
031 //@@brief Initialises a node
032
033 /*
034 takes the port number to listen on for connections
035 and then opens the mpi on that port
036 */
037 pnode::pnode(unsigned int port)
038 {
039     comms=new mpiclient(port);
040     comms->attach(); //starts forking proceses
041     std::cout<<"attached to mcp\n";
042 }
043
044 //@@brief Initialises a node
045
046 /*
047 sets the default port to listen on for connections on 3124
048 and then starts listening on that port
049 */
050 pnode::pnode()
051 {
052     comms=new mpiclient(3124);
053     comms->attach();
054     std::cout<<"attached to mcp\n";
055 }
056
057 //@@brief Starts the message loop
058 /*
059 passes received messages to the
060 message handler and then trasmits back the response
061 */
062 int pnode::start()
063 {
```

```
064 message * temp, * temp2;
065 int killher=0;
066 temp=NULL;
067 int c=0;
068 while (temp==NULL) //loops round untill a message is found
069     temp=comms->recv_message();
070 cout << temp->get_name()<<endl;
071 if (strstr(temp->get_name(),"kill")) //stops the node when a kill message is received
072     {
073         cout << "killing"<<endl;
074         killher=1;
075     }
076 temp2=this->process_message(temp); //sends the message to the handler
077 comms->send_message(temp2); //transmits the response to the mcp
078 comms->closeup();
079 delete temp;
080 delete temp2;
081 temp=NULL;
082 temp2=NULL;
083 if (killher==1)
084     return 10;
085 else
086     return 0;
087 }
088
```

1.10 Source File: ./rater/rater.cpp

```
001 /*
002 ###Source###
003 *****filename*****
rater.cpp
004 *****author*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
Mon Nov 17 16:41:28 GMT 2003
009 *****description*****
010 rapper for raternode class
011 #####
012 */
013 // This program is free software; you can redistribute it and/or modify
014 // it under the terms of the GNU General Public License as published by
015 // the Free Software Foundation; either version 2 of the License, or
016 // (at your option) any later version.
017 //
018 // This program is distributed in the hope that it will be useful,
019 // but WITHOUT ANY WARRANTY; without even the implied warranty of
020 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
021 // GNU General Public License for more details.
022 //
023 // You should have received a copy of the GNU General Public License
024 // along with this program; if not, write to the Free Software
025 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
026 #include "raternode.h"
027
028 // main entry point for the rater node
029
030 int main(int argc, char ** argv)
```

```
031 {
032     pagerater * thisnode;
033     int ret;
034     if (argc>1)
035         thisnode=new pagerater(atoi(argv[1])); //sets up the rater up with a custom port
036     else
037         thisnode=new pagerater(); //sets up the rater with the default port
038     ret=thisnode->start(); //starts the node
039     delete thisnode;
040     return ret;
041 }
042
```

1.11 Source File: ./rater/raternode.cpp

```
001 /*
002 ###Source###
003 *****filename*****
raternode.cpp
004 *****auther*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
Tue Oct 28 23:06:26 GMT 2003
009 *****description*****
010 This file describes the page rating algorithms.
011 #####
012 */
013 // This program is free software; you can redistribute it and/or modify
014 // it under the terms of the GNU General Public License as published by
015 // the Free Software Foundation; either version 2 of the License, or
016 // (at your option) any later version.
017 //
018 // This program is distributed in the hope that it will be useful,
019 // but WITHOUT ANY WARRANTY; without even the implied warranty of
020 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
021 // GNU General Public License for more details.
022 //
023 // You should have received a copy of the GNU General Public License
024 // along with this program; if not, write to the Free Software
025 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
026 #include "raternode.h"
027
028 // Constructor
029
030 /*
```

```
031 Starts the serve on port port
032 */
033 pagerater::pagerater(unsigned int port):pnode(port)
034 {
035
036 }
037
038 // Constructor
039
040 /*
041 starts the server on port 3124
042 */
043 pagerater::pagerater():pnode(3124)
044 {
045
046 }
047
048 // Processes Received Messages from the mcp
049
050 /*
051 recives messages from the mcp and then decides what to do about
052 them. If a rate message is received the system starts up the sds
053 search system and searches a page for the search string, rates
054 the page and then send s back the result. If a kill message is
055 received the systems shutdowns .
056 */
057 message * pagerater::process_message(message * data)
058 {
059 message * temp;
060 char * name , * ldata, * temppos,* temppos2,* word,* rat,* words, * from, *tempdata, *sstrng ,* tagtext;
061 int oldpos,pos,distance,numofwords,initword,realrating,c,c2,c1,dist;
062 double rating;
063 sdstext * sds;
```

```

064 temp=new message;
065 name=data->get_name();
066 ldata=data->get_data();
067 cout <<"new message\n";
068 if (strcmp(name,"rate")==0)
069 {
070     cout<<"new page to rate\n";
071     tempdata=new char[strlen(ldata)+1];
072     strcpy(tempdata,ldata);
073     temppos=strstr(tempdata,"[end]");//finds the first [end] tag that denotes the end of the url of the page
074     if (temppos!=NULL)
075     {
076         *temppos=0;
077         from=new char[strlen(tempdata)+1];
078         strcpy(from,tempdata);
079         temppos2=strstr(temppos+5,"[end]");//finds the next [end] tag that denotes the end of the model
080         tagtext=strstr(temppos2+5,"[end]");//finds the next [end] tag that denotes the end of the search space
081         if (temppos2!=NULL)
082         {
083             *temppos2=0;
084             sstring=new char[strlen(temppos+5)+1];
085             strcpy(sstring,temppos+5);//creates a copy of the search model
086             *tagtext=0;
087             tagtext=tagtext+5;
088             words=new char[strlen(temppos2+5)+2];
089             strcpy(words,temppos2+5);//creates a copy of the search space
090             strcat(words,";");
091             rating=0;
092             oldpos=0;
093             numofwords=0;
094             word=strtok(sstring," "); // finds the first space
095             char * subword1,* subword2;
096             char achar;

```

```

097 while (word!=NULL)//runs through all the search words
098 {
099     numofwords++;
100     subword1=strchr(word, '_');
101     subword2=word;
102     if (subword1==NULL)
103     {
104         cout << word<<": "<<(char)('A'-1+numofwords)<<endl;
105         sds=new sdstext(word,words);// creates a new sds text seach on the new search model and search space
106         words=sds->start('A'+numofwords-1) ;// starts the text search
107         delete sds;
108     }
109     while (subword1!=NULL)//if the string is a collection search as if they were the same word
110     {
111         achar=*subword1;
112         *subword1=NULL;
113         cout << subword2<<": "<<(char)('A'-1+numofwords)<<endl;
114         sds=new sdstext(subword2,words);// creates a new sds text seach on the new search model and search space
115         words=sds->start('A'+numofwords-1) ;// starts the text search
116         delete sds;
117         subword2=subword1+1;
118         *subword1=achar;
119         subword1=strchr(subword2, '_');
120         if ((subword1==NULL)&&((*subword2)!=NULL))//if no more _ in text place end of word as end of string
121         {
122             subword1=&subword2[strlen(subword2)];
123         }
124     }
125 }
126 word=strtok(NULL, " ");
127 }
128 cout << words<<endl;
129 rating=0;

```

```

130 char * newchars=new char[strlen(words)+1];//the next section of code removes all the words and just leaves a
131 single character per word, either a space if unrecognized or the letter representing the word
132 for (c=0;c<strlen(words);c++)
133     newchars[c]=0;
134 c2=0;
135 char * comp="QWERTYUIOPLKJHGFDSAZXCVBNM ";
136 for (c=0;c<strlen(words);c++)
137 {
138     if (strchr(comp,words[c])!=NULL)//only add char if a tag letter or space
139     {
140         newchars[c2]=words[c];
141         c2++;
142     }
143     strcpy(words,newchars);
144     delete newchars;
145     newchars=new char[strlen(words)+1];
146     for (c=0;c<strlen(words);c++)
147         newchars[c]=0;
148     newchars[0]=words[0];
149     c1=1;
150     for (c=1;c<strlen(words);c++)
151     {
152         if (((words[c]==' ')&&(words[c-1]!=' '))||(words[c]!=' '))//filters out spaces after the tag letter
153         {
154             newchars[c1]=words[c];
155             c1++;
156         }
157     }
158     cout << newchars << endl;
159     char next1,next5;
160     char * next2, * next3, *first;
161     int numofinc=0;

```

```

162 double wordgap;
163 double penalty;
164 char * letters=new char[numofwords+1];
165 letters[numofwords]=NULL;
166 c=0;
167 for (next1='A';next1<=('A'+numofwords-1);next1++)//creates a string containing all the letters used as tags
168 {
169     letters[c]=next1;
170     c++;
171 }
172 next2=strpbrk(newchars,letters);//finds the first tagged word
173 next1='A'-1;
174 first=next2;//sets it as the first word of a instance
175 if (next2!=NULL)
176 {
177     c=1;
178     c1=0;
179     while (next2!=NULL)
180     {
181
182         if (*next2>=(next1+1))//if the char is after the current one in the search model keep going
183         {
184             c1++;
185             cout << *next2 << (char)(next1+1) << endl;
186             next1=*next2;
187             next3=next2;
188             next2=strpbrk(next2+1,letters);
189         }
190         else//else work out the addition to the rating and then start again in the loop
191         {
192             dist=next3-first+1;
193             penalty=((double)(c1*c1))/((double)(numofwords*dist));//penlty function for the differnece between this
instantaion and the original model

```

```

194     cout <<penalty<<": "<<dist<<endl;
195     rating=rating+penalty;//adds it to the rating
196     first=next2;
197     next1='A'-1;
198     c++;
199     c1=0;
200     }
201     }
202     dist=next3-first+1;
203     penalty=((double)(c1*c1))/((double)(numofwords*dist));//performs the rating calcs for the last instance
204     cout << penalty<<": "<<dist<<endl;
205     rating=rating+penalty;
206     rating=1/(abs(c-rating)+1);//inverses the difference between this and perfect so that 1 is a perfect match
and anything else is much lower
207     }
208     else
209         rating=0;
210     delete newchars;
211     cout << rating<<endl;
212     realrating=(int)(rating * 100);//calculates the rating as a percentage
213     temp->set_name("rating");//sets up the new message to sent back
214     rat=new char[strlen(from)+50+strlen(tagtext)];
215     sprintf(rat, "%s[end]%d[end]%s", from, realrating, tagtext);//formats the new message data string
216     printf("%s\n", rat);
217     temp->set_data(rat);
218     delete rat;
219     delete words;
220     delete sstring;
221     }
222     delete from;
223     }
224     delete tempdata;
}

```

```
225 else if (strcmp(name, "kill")==0)
226 {
227     temp->set_name("bye");
228     temp->set_data("");
229 }
230 else
231 {
232     temp->set_name("wtf?");
233     temp->set_data("");
234     return temp;
235 }
236
237
```

1.12 Source File: ./httperv/httpget.cpp

```
001 /*
002 ###Source###
003 *****filename*****
httpget.cpp
004 *****auther*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
10/10/2003 13:47:52
009 *****description*****
010 main loop for http get node
011 #####
012 */
013 // This program is free software; you can redistribute it and/or modify
014 // it under the terms of the GNU General Public License as published by
015 // the Free Software Foundation; either version 2 of the License, or
016 // (at your option) any later version.
017 //
018 // This program is distributed in the hope that it will be useful,
019 // but WITHOUT ANY WARRANTY; without even the implied warranty of
020 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
021 // GNU General Public License for more details.
022 //
023 // You should have received a copy of the GNU General Public License
024 // along with this program; if not, write to the Free Software
025 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
026
027
028 #include <iostream>
029 #include <stdlib.h>
030 #include "httpgetnode.h"
```

```
031
032 int main(int argc, char ** argv)
033 {
034     httpgetnode * thisnode;
035     int ret;
036     if (argc==2)
037         thisnode=new httpgetnode(atoi(argv[1])); // starts the system with the passed port
038     if (argc==5)
039     {
040         thisnode=new httpgetnode(atoi(argv[1])); // starts the system with the passed port
041         thisnode->set_proxy(argv[2]); // and also sets up the proxy
042         thisnode->set_port(atoi(argv[3]));
043         thisnode->set_authstring(argv[4]);
044     }
045     else
046         thisnode=new httpgetnode(); //starts with the default port
047     ret = thisnode->start(); //initialise the system
048     delete thisnode;
049     cout << ret << endl;
050     return ret;
051 }
052
```

1.13 Source File: ./httperv/httpgetnode.cpp

```
001 /*
002 ###Source###
003 *****filename*****
httpgetnode.c++
004 *****auther*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
10/10/2003 12:24:19
009 *****description*****
010 implimentation of http get node class
011 #####
012 */
013 // This program is free software; you can redistribute it and/or modify
014 // it under the terms of the GNU General Public License as published by
015 // the Free Software Foundation; either version 2 of the License, or
016 // (at your option) any later version.
017 //
018 // This program is distributed in the hope that it will be useful,
019 // but WITHOUT ANY WARRANTY; without even the implied warranty of
020 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
021 // GNU General Public License for more details.
022 //
023 // You should have received a copy of the GNU General Public License
024 // along with this program; if not, write to the Free Software
025 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
026
027
028 #include <iostream>
029 #include <stdlib.h>
030 #include "httpgetnode.h"
```

```
031 #define debug
032
033 using namespace std;
034
035 httpgetnode::httpgetnode(unsigned int port):pnode(port)
036 {
037     authstring=NULL;
038     proxy=NULL;
039     wport=80;
040 }
041
042 httpgetnode::httpgetnode():pnode(3124)
043 {
044     authstring=NULL;
045     proxy=NULL;
046     wport=80;
047 }
048
049 httpgetnode::~httpgetnode()
050 {
051 }
052
053 // Processes messages arriving from the MCP
054
055 /*
056  if the received message is of type get a page is fetched and processed,
057  else if it is a kill message the system shuts down else a error message
058  is sent back
059 */
060 message * httpgetnode::process_message(message * data)
061 {
062     message * temp;
063     char * name, * ldata;
```

```
064 char * page;
065 temp=NULL;
066 name=data->get_name();
067 ldata=data->get_data();
068 cout <<name<<endl;
069 if (strcmp(name,"get")==0)
070 {
071     cout<<"received get message"<<endl;
072     page=get_page(ldata); // fetch and process the url within the message
073     if (strcmp(page,"")!=0) // if page is collected return the page data
074     {
075         temp=new message;
076         temp->set_name("page");
077         temp->set_data(page);
078     }
079     else // else return a error
080     {
081         temp=new message;
082         temp->set_name("error");
083         temp->set_data("badurl");
084     }
085     delete page;
086 }
087 else if (strcmp(name,"kill")==0)
088 {
089     temp=new message;
090     temp->set_name("bye");
091     temp->set_data("");
092 }
093 else
094 {
095     temp=new message;
096     temp->set_name("wtf?");
```

```

097     temp->set_data("");
098 }
099 return temp;
100 }
101
102 // Connects to the server to download a page
103
104 /*
105  if the proxy details are set then the proxy is attached to and the
106  correct headers are sent to the proxy, else a direct connection to
107  the web server is made and the request is sent directly.
108 */
109 int httpgetnode::connect(char * url)
110 {
111     char * domain, * page, * dir, * ip;
112     int sockfd;
113     if (strstr(url,"http://")!=NULL)
114     {
115         if ((domain=get_domain(url))==NULL)
116             return -1;
117         if ((page=get_file(url))==NULL)
118             return -1;
119         if ((dir=get_dir(url))==NULL)
120             return -1;
121         if ((proxy!=NULL)&&(authstring!=NULL))
122         {
123             cout<<"doing proxy"<<endl;
124             if ((ip=get_ip(proxy))==NULL) // sets the ip to attach to as the proxy
125                 return -1;
126         }
127     }
128     else
129     {
130         if ((ip=get_ip(domain))==NULL) //sets the ip to attach to as the web server

```

```

129     return -1;
130 }
131 struct sockaddr_in sockadd; //structure to hold the socket connection information
132 sockfd=socket(AF_INET, SOCK_STREAM, 0);
133 sockadd.sin_family=AF_INET;
134 sockadd.sin_addr.s_addr=inet_addr(ip);
135 sockadd.sin_port=htons(wport);
136 if (connect(sockfd,(struct sockaddr *)&sockadd,sizeof(sockadd))==0) //attemp to connect to the server
137 {
138     char * request;
139     if ((proxy!=NULL)&&(authstring!=NULL)) // if we are connecting to the proxy use these headers
140     {
141         cout<<"doing proxy"<<endl;
142     }
143     request=new char[strlen(url)+strlen("GET HTTP/1.1\nProxy-Authorization: Basic\nHost: \nUser-Agent:
Sdswebsearch\nConnection: close\n\n")+strlen(domain)+strlen(authstring)+1];
144     sprintf(request,"GET %s HTTP/1.1\nHost: %s\nProxy-Authorization: Basic %s\nUser-Agent:
Sdswebsearch\nConnection: close\n\n",url,domain,authstring);
145 }
146 else //else if we are connecting directly use thes headers
147 {
148     cout << "conect directly"<<endl;
149     request=new char[strlen(page)+strlen(dir)+strlen("GET HTTP/1.1\nHost: \nUser-Agent: Sdswebsearch\nConnection:
close\n\n")+strlen(domain)+1];
150     sprintf(request,"GET %s%s HTTP/1.1\nHost: %s\nUser-Agent: Sdswebsearch\nConnection:
close\n\n",dir,page,domain);
151 }
152 /*
153
154     Header Explanation
155
156 Proxy-Authorization:
157 This sets the authorization scheme for the connection. Most authenticated proxies use the basic scheme which uses

```

```
base 64 encoded
158 string to send the user name and password. This is sent following the Basic string
159 Host:
160 This is a requirement for HTTP/1.1 and contains the host domain name for the server
161 User-Agent:
162 This is an arbitrary text string which denotes the name of the client attaching.
163 Connection:
164 This sets whether a connection dies straight after communication, this is the case with the close statement, or
continues to stay open
165 waiting for another request statement.
166 */
167 write(socketfd,request,strlen(request)); //sends request to server
168 cout << request << endl;
169 delete request;
170 return socketfd;
171 }
172 }
173 return -1;
174 }
175
176 char * httpgetnode::get_page(char * url)
177 {
178     char * domain, * page, * dir, * ret, * buffer, * temp;
179     int socketfd,bytesin;
180     buffer=new char[1025];
181     ret=(char *)malloc(sizeof(char));
182     ret[0]=0;
183     bytesin=100;
184     if (strstr(url,"http://")!=NULL) //checks to see if valid url
185     {
186         if ((domain=get_domain(url))==NULL)
187             return ret;
188         if ((page=get_file(url))==NULL)
```

```

189     return ret;
190     if ((dir=get_dir(url))==NULL)
191         return ret;
192     if ((domain!=NULL)&&(page!=NULL))
193     {
194         if ((socketfd=connect(url))>=0) //connects to server
195         {
196             while (bytesin>0)
197             {
198                 bytesin=read(socketfd,buffer,1024); // reads 1k from server
199                 buffer[bytesin]=0;
200                 ret=(char *)realloc((void *)ret,sizeof(char)*(strlen(ret)+strlen(buffer)+1)); //makes ret long enough to
stroce downloaded page
201                 cout <<strlen(ret)<<endl;
202                 if (ret!=NULL)
203                 {
204                     strcat(ret,buffer);
205                     if ((strstr(ret,"text/html")==NULL)&&(strlen(ret)>1024)) //drops out is html mime type not found within
first 1k
206                         bytesin=0;
207                 }
208                 else
209                 {
210                     free(ret); //drops out if an out of memory error occurs
211                     return "";
212                 }
213                 if (strstr(ret,"text/html")!=NULL) //received html
214                 {
215                     temp=process_html(ret,domain,dir); // processes the downloaded page and returns the data section of the
return message
216                     free(ret);
217                     ret=new char[strlen(temp)+strlen(url)+2];

```

```

218 strcpy(ret,url);
219 strcat(ret,"\n");
220 strcat(ret,temp);
221 cout<<ret<<endl;
222 delete temp;
223 }
224 else //returns blank if incorrect mime type downloaded. This can sometimes happen if the server is brocken and
is sending the wrong mime type for files
225 {
226     delete ret;
227     ret=strdup("");
228 }
229     delete buffer;
230 }
231 }
232 delete domain;
233 delete page;
234 delete dir;
235 }
236 return ret;
237 }
238
239 // Returns the domain of a url
240
241 /*
242 Takes a string containing a fully qualified URL and returns only the domain i.e.
243 the section after the protocal identifier and before the beging of the directory structure
244 */
245
246 char * httpgetnode::get_domain(char * url)
247 {
248     char * temp, * temp2, * temp3,* ret;
249     temp=strdup(url);

```

```
250 if (temp!=NULL)
251 {
252     temp2=temp+7;
253     temp3=strchr(temp2, '/');
254     if (temp3!=NULL)
255     {
256         *temp3=0;
257     }
258     ret=strdup(temp2);
259     delete temp;
260 }
261 else
262     ret=NULL;
263 return ret;
264 }
265 // looks up a domain and retruns the ip
266
267 /*
268  takes a string containing a domain and returns the ipv4 dotted quad ip
269  address for that machine
270 */
271 char * httpgetnode::get_ip(char * domain)
272 {
273     struct hostent *hostinfo;
274     char * ret;
275     hostinfo=gethostbyname(domain); //lookup the domain using the systems defined in /etc/nsswitch.conf, normally
    /etc/hosts then dns server
276     if (!hostinfo)
277         return NULL; //if host does not exist returns null
278     ret=inet_ntoa(*(struct in_addr *)*hostinfo->h_addr_list); //else returns the dotted quad ipv4 ip address
279     return ret;
280 }
```

```
281
282 // extracts the directory path
283
284 /*
285 Takes a URL and returns the path of the file pointed to by the URL
286 */
287 char * httpgetnode::get_dir(char * url)
288 {
289     char * temp, * temp2, * temp3,* ret, * temp4;
290     temp=strdup(url);
291     if (temp!=NULL)
292     {
293         temp2=temp+7;
294         temp3=strchr(temp2,'/');
295         temp4=strrchr(temp2,'/');
296         if (temp3!=temp4)
297         {
298             if (temp4!=NULL)
299                 *temp4=0;
300             if (temp3!=NULL)
301             {
302                 ret=strdup(temp3);
303             }
304             else
305                 ret=strdup("");
306         }
307         else
308             ret=strdup("");
309         delete temp;
310     }
311     else
312         ret=NULL;
313     return ret;

```

```

311 }
312
313 // filters unwanted chars
314
315 /*
316 Takes a string to be filtered and returns the filtered string.
317 This happens in two parts. A filter for unwanted characters is run using
318 the set {Q,W,E,R,T,Y,U,I,O,P,A,S,D,F,G,H,J,K,L,Z,X,C,V,B,N,M,1,2,3,
319 4,5,6,7,8,9,0,q,w,e,r,t,y,u,i,o,p,a,s,d,f,g,h,j,k,l,z,x,c,v,b,n,m} replacing
320 all characters not found in the set with a space. After this all multiple
321 spaces are converted to single spaces.
322 */
323 char * httpgetnode::remove_whitespace(char * page)
324 {
325     char * ret, tests[2];
326     unsigned int c,c2;
327     char wanted[]="QWERTYUIOPASDFGHJKLZXCVBNM1234567890qwertyuiopasdfghjklzxcvbnm";
328     c2=0;
329     ret=new char[strlen(page)+1];
330     ret[0]=0;
331     for (c=0;c<(strlen(page));c++)
332     {
333         tests[0]=page[c];
334         tests[1]=page[c+1];
335         if (strcmp(wanted,tests[0])==NULL)
336             tests[0]=' ';
337         if (strcmp(wanted,tests[1])==NULL)
338             tests[1]=' ';
339         if ((tests[0]!=' ')||(tests[1]!=' '))
340             {
341                 ret[c2]=tests[0];
342                 c2++;
343             }

```

```
344 }
345 ret[c2]=0;
346 return ret;
347 }
348
349 // Extracts file name from url
350
351 /*
352 returns the string left after the last "/"
353 in the passed string.
354 */
355 char * httpgetnode::get_file(char * url)
356 {
357     char * temp, * temp2, * temp3,* ret;
358     temp=strdup(url);
359     if (temp!=NULL)
360     {
361         temp2=temp+7;
362         temp3=strchr(temp2,'/');
363         if (temp3!=NULL)
364         {
365             ret=strdup(temp3);
366         }
367         else
368             ret=strdup("/");
369         delete temp;
370     }
371     else
372         ret=NULL;
373     return ret;
374 }
375
376 // extracts wanted information from the html file
```

```

375
376 /*
377 takes the page, the domain it came from and the directory and returns
378 the data section of the return message to the MCP. This forms two stages,
379 the first being extraction of all the urls contained in the page and then
380 stripping away all the useless data.
381 */
382 char * httpgetnode::process_html(char * page,char * domain,char * dir)
383 {
384     char * temp, * temp2, * temp3, * temp4, * temp5, * temp6, * temp7 , * temp8, * tagless, * ret, * newurl,
        *title,*title1,*title2;
385     int whites;
386     stringlist * urls=new stringlist; // sets up the list to hold the urls in during processing
387     temp=strdup(page); // duplicates the page so that it can be altered without corrupting the original
388     if (temp==NULL) // if the is a memory problem cleans up and drops out
389     {
390         delete urls;
391         return "";
392     }
393     lower(temp); // converts page to lower case to make the processing easier
394     title=new char[1];
395     title[0]=0; //sets the title to blank incase there is no title
396     title1=strstr(temp,"<title>"); //finds the beging tag of the title
397     if (title1!=NULL) // if a title exists then process it
398     {
399         title2=strstr(temp,"</title>");//finds the ending tag of the title
400         if (title2!=NULL)
401         {
402             title1=title1+7;
403             *title2=0;
404             delete title;
405             title=remove_whitespace(title1); //stores the title minus any unwanted characters in title
406             *title2='<';

```

```

407     }
408 }
409 temp2=strstr(temp,"<body"); //finds the beginning of the body of the html document
410 if (temp2!=NULL)
411 {
412     while ((temp3=strchr(temp2,'<'))!=NULL) //finds the beging of a tag
413     {
414         temp4=strchr(temp2,'>'); //finds the end of the tag
415         *temp4=0;
416         whites=strlen(temp3);//length of the tag
417         temp5=strdup(temp3);//copies the tag so that it can be fiddled with
418         if (temp5==NULL)
419         {
420             delete urls;
421             delete temp;
422             return "";
423         }
424         if ((temp6=strstr(temp5,"href"))!=NULL) //if the tag contains a href link extract the link
425         {
426             temp7=strchr(temp6,'\'''); //find the beginning of the quoted link
427             if (temp7!=NULL)
428             {
429                 *temp7=' ';
430                 temp8=strchr(temp6,'\'''); //find the end of the quoted link
431                 if (temp8!=NULL)
432                 {
433                     *temp8=0;
434                     newurl=temp7;
435                     urls->add(newurl,0); // add the link to the url list
436                     cout << newurl <<endl;
437                 }
438             }
439         }

```

```

440 else if ((temp6=strstr(temp5,"src"))!=NULL) //if the tag contains a src link extract the link
441 {
442     temp7=strchr(temp6,'\'); //find the beginning of the quoted link
443     if (temp7!=NULL)
444     {
445         *temp7=' ';
446         temp8=strchr(temp6,'\'); //find the end of the quoted link
447         if (temp8!=NULL)
448         {
449             *temp8=0;
450             newurl=temp7;
451             urls->add(newurl,0); // add the link to the url list
452             cout << newurl <<endl;
453         }
454     }
455 }
456 cout <<temp3<<endl;
457 delete temp5;
458 memset(temp3,'',(size_t)whites+1); //blank out the tag
459 }
460 temp7=urls->flatern("\n"); // converts the list of urls into a newline delimited string
461 temp5=remove_whitespace(temp2); // filters unwanted chars out of the body of the page
462 if ((temp5==NULL)||(temp7==NULL)) //if a memory error occurs drop out
463 {
464     delete urls;
465     delete temp;
466     delete temp5;
467     delete temp7;
468     delete tagless;
469     return "";
470 }
471 tagless=temp5;
472 ret=new char[strlen(title)+strlen(temp7)+strlen(tagless)+14]; // create a largenough string to hold all the

```

```
outputed data
473  if (ret==NULL)
474  {
475      delete urls;
476      delete temp;
477      return "";
478  }
479  strcpy(ret,temp7);//place the url list at the head
480  strcat(ret,"[end]\n");//then the end of section deleminator
481  strcat(ret,tagless);//then add the filtered page
482      strcat(ret,"[end]\n");//then the end of section deleminator
483      strcat(ret,title);//then the title of the page
484  delete temp7;
485  delete temp5;
486  }
487  else
488      ret=strdup("[end]\n");
489      delete title;
489  delete temp;
490  delete urls;
491  return ret;
492 }
493
494 // lower cases a string
495
496 /*
497  takes a pointer to the string to lowercase
498  and converts all the characters in that string
499  into there lowercase version
500 */
501 void lower(char * data)
502 {
503     char uplet;
```

```
504 char * repl;
505 for (uplet='A'; uplet<='Z'; uplet++)
506 {
507     while ((repl=strchr(data, uplet)) != NULL)
508     {
509         *repl=*repl+32;
510     }
511 }
512 }
513
514
```

2 C++ header files

2.1 Source File: ./mpi/mpi.h

```
001 /*
002 ###Header###
003 *****filename*****
mpi.h
004 *****auther*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
25/09/2003 13:58:43
009 *****description*****
010 This file contains the interface to the message passing
011 system of the project. It contains two classes, one for the
012 client side and one for the server side.
013 #####
014 */
015 // This program is free software; you can redistribute it and/or modify
016 // it under the terms of the GNU General Public License as published by
017 // the Free Software Foundation; either version 2 of the License, or
018 // (at your option) any later version.
019 //
020 // This program is distributed in the hope that it will be useful,
021 // but WITHOUT ANY WARRANTY; without even the implied warranty of
022 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
023 // GNU General Public License for more details.
024 //
025 // You should have received a copy of the GNU General Public License
026 // along with this program; if not, write to the Free Software
027 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
```

```
028
029
030 #include <iostream>
031 #include <stdlib.h>
032 #include <sys/socket.h>
033 #include <netinet/in.h>
034 #include <arpa/inet.h>
035 #include <unistd.h>
036 #include <cstring>
037 #include "../lists/linkedlist.h"
038 #ifndef mpih
039 #define mpih
040
041 using namespace std;
042
043 class message
044 {
045 public:
046     message(){name=NULL;data=NULL;}
047     ~message();
048     void set_name(char * in);
049     void set_data(char * in);
050     char * get_name();
051     char * get_data();
052 private:
053     char * name;
054     char * data;
055 };
056 class mpi
057 {
058 public:
059     virtual unsigned int get_port()=0;
060     void send_message(message * data);
```

```
061 message * recv_message();
062 virtual void closeup()=0;
063 protected:
064 int send(char * data);
065 int recv(char * data, unsigned int length);
066 unsigned int lport;
067 int sockfd;
068 int open;
069 char * recvtext;
070 };
071
072 class mpiclient : public mpi
073 {
074 public:
075 mpiclient(unsigned int port);
076 mpiclient(){open=0;recvtext=strdup("");}
077 virtual ~mpiclient();
078 void set_port(unsigned int port){lport=port;}
079 unsigned int get_port(){return lport;}
080 int attach();
081 void closeup();
082 protected:
083 unsigned int lport;
084 int sockfd;
085 };
086
087 #endif
088
089
```

2.2 Source File: ./sds/adu.h

```
001 // adu.h: interface for the adu class.
002 //
003 ///////////////////////////////////////////////////////////////////
004
005 #ifndef aduh
006 #define aduh
007
008 class adu
009 {
010 public:
011     adu(char data);
012     char get();
013     void set_halt(int halt_min,int halt_max,int halt_time);
014     int enter();
015     int exit();
016     int getactives();
017     void settag();
018     int gettag();
019     int is_stable();
020     virtual ~adu();
021 private:
022     int actives;
023     char me;
024     int tag;
025     int lstable;
026     int lhalt_min;
027     int lhalt_max;
028     int lhalt_time;
029 };
030 #endif
031
032
```


2.3 Source File: ./sds/agent.h

```
001 // agent.h: interface for the agent class.
002 //
003 ///////////////////////////////////////////////////////////////////
004 #include "adu.h"
005 #include <stdlib.h>
006 #include <cmath>
007
008 #ifndef agenth
009 #define agenth
010 const int scaler=30;
011 class agent
012 {
013 public:
014     agent(adu ** basemodel, adu ** basesearch, agent ** basepos, int modelsize, int searchsize, int agentsize, int pbase);
015     int getactive();
016     int test();
017     void diffuse();
018     int getsearch();
019     void init();
020 private:
021     int active;
022     adu ** lbasemodel;
023     adu ** lbasesearch;
024     int lmodelsize;
025     int lsearchsize;
026     int searchpos;
027     agent ** base;
028     int size, lbase;
029 };
030 #endif
031
032
```


2.4 Source File: ./sds/sdstext.h

```
001 #include "adu.h"
002 #include "agent.h"
003 #include <cstring>
004 #include <time.h>
005 #include <iostream>
006
007 #ifndef sdstexth
008 #define sdstexth
009
010 using namespace std;
011
012 class sdstext
013 {
014 public:
015     sdstext(char * model, char * search);
016     sdstext();
017     ~sdstext();
018     void set_text(char * model, char * search);
019     char * get_model(){return lmodel;}
020     char * get_search(){return lsearch;}
021     void set_base(int pbase){base=pbase;}
022     int get_base(){return base;}
023     char * start(char replacechar);
024 private:
025     agent ** create_agent_array(adu ** basemodel,adu ** basemodel,adu ** basemodel,int searchsize,int * agentsize,int
pbase);
026     adu ** create_adu_array(char * string,int * length);
027     char * lmodel, * lsearch;
028     int base;
029 };
030
031 #endif
```


2.5 Source File: ./lists/prostrlist.h

```
001 /*
002 ###Header###
003 *****filename*****
prostrlist.h
004 *****author*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
Thu Nov 13 23:05:39 GMT 2003
009 *****description*****
010 based on the string list but allowing fo promotions bassed
011 on stringed tags of orignation
012 #####
013 */
014 // This program is free software; you can redistribute it and/or modify
015 // it under the terms of the GNU General Public License as published by
016 // the Free Software Foundation; either version 2 of the License, or
017 // (at your option) any later version.
018 //
019 // This program is distributed in the hope that it will be useful,
020 // but WITHOUT ANY WARRANTY; without even the implied warranty of
021 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
022 // GNU General Public License for more details.
023 //
024 // You should have received a copy of the GNU General Public License
025 // along with this program; if not, write to the Free Software
026 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
027
028 #include <iostream>
029 #include <stdlib.h>
030 #include "linkedlist.h"
```

```
031 #include "stringlist.h"
032
033 class urlnode
034 {
035 public:
036 urlnode();
037 ~urlnode();
038 void set_page(char * page);
039 char * get_page(){return name;}
040 void set_rating(int rating){rate=(rate+rating)/2;}
041 int get_rating(){return rate;}
042 stringlist * from;
043 private:
044 char * name;
045 int rate;
046 };
047
048 class prostrlist : public linkedlist
049 {
050 public:
051 prostrlist(){total=0;}
052 ~prostrlist();
053 void addlink(char * page, char * from);
054 int findlink(char * page);
055 void promote(char * from, int rating);
056 int get_random();
057 int remove(int pos);
058 char * get(int pos);
059 private:
060 node * getlink(char * page);
061 int total;
062 };
063
```


2.6 Source File: ./lists/stringlist.h

```
001 /*
002 ###Header###
003 *****filename*****
stringlist.h
004 *****author*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
23/09/2003 19:44:02
009 *****description*****
010 This is the class header of a linked list of strings. It is
011 intheadited from the generic linked list in linkedlist.h .
012 #####
013 */
014 // This program is free software; you can redistribute it and/or modify
015 // it under the terms of the GNU General Public License as published by
016 // the Free Software Foundation; either version 2 of the License, or
017 // (at your option) any later version.
018 //
019 // This program is distributed in the hope that it will be useful,
020 // but WITHOUT ANY WARRANTY; without even the implied warranty of
021 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
022 // GNU General Public License for more details.
023 //
024 // You should have received a copy of the GNU General Public License
025 // along with this program; if not, write to the Free Software
026 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
027
028
029 #include <iostream>
030 #include <stdlib.h>
```

```
031 #include <cstring>
032 #include "linkedlist.h"
033
034 #ifndef stringlisth
035 #define stringlisth
036
037 class stringlist : public linkedlist
038 {
039 public:
040     ~stringlist();
041     unsigned int add(char * data, int end);
042     unsigned int insert(char * data, unsigned int pos);
043     char * get(unsigned int pos);
044     unsigned int remove(unsigned int pos);
045     char * flatern(char * seperator);
046     int search(char * model);
047 };
048
049 #endif
050
051
```

2.7 Source File: ./lists/linkedlist.h

```
001 /*
002 ###Header###
003 *****filename*****
linkedlist.h
004 *****author*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
15/09/2003 13:56:34
009 *****description*****
010 this file describes the class members and properties of a
011 generic linked list
012 #####
013 */
014 // This program is free software; you can redistribute it and/or modify
015 // it under the terms of the GNU General Public License as published by
016 // the Free Software Foundation; either version 2 of the License, or
017 // (at your option) any later version.
018 //
019 // This program is distributed in the hope that it will be useful,
020 // but WITHOUT ANY WARRANTY; without even the implied warranty of
021 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
022 // GNU General Public License for more details.
023 //
024 // You should have received a copy of the GNU General Public License
025 // along with this program; if not, write to the Free Software
026 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
027
028
029 #include <stdlib.h>
030 #include <cstring>
```

```
031
032 #ifndef linkedlisth
033 #define linkedlisth
034
035 char * strdup(char *);
036
037 class node
038 {
039 public:
040     void * data;
041     node * next;
042     node * prev;
043 };
044
045 class linkedlist
046 {
047 public:
048     linkedlist();
049     ~linkedlist();
050     unsigned int add(void * data, int end);
051     unsigned int insert(void * data, unsigned int pos);
052     void * get(unsigned int pos);
053     unsigned int get_length(){return length;}
054     unsigned int remove(unsigned int pos);
055 protected:
056     node * head;
057     node * tail;
058     unsigned int length;
059     node * getnode(unsigned int pos);
060
061 };
062
063 #endif
```


2.8 Source File: ./nodes/node.h

```
001 /*
002 ###Header###
003 *****filename*****
node.h
004 *****author*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
06/10/2003 16:41:23
009 *****description*****
010 This file contains the definition of a basic node which is
011 then inherited to make the specific working code of each
012 node;
013 #####
014 */
015 // This program is free software; you can redistribute it and/or modify
016 // it under the terms of the GNU General Public License as published by
017 // the Free Software Foundation; either version 2 of the License, or
018 // (at your option) any later version.
019 //
020 // This program is distributed in the hope that it will be useful,
021 // but WITHOUT ANY WARRANTY; without even the implied warranty of
022 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
023 // GNU General Public License for more details.
024 //
025 // You should have received a copy of the GNU General Public License
026 // along with this program; if not, write to the Free Software
027 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
028
029
030 #include <iostream>
```

```
031 #include <stdlib.h>
032 #include <pthread.h>
033 #include "../mpi/mpi.h"
034 #include "../lists/linkedlist.h"
035
036 #ifndef pnodeh
037 #define pnodeh
038
039 // Virtual ansetore class for all nodes
040 class pnode
041 {
042 public:
043     pnode(unsigned int port);
044     pnode();
045     int start();
046 protected:
047     virtual message * process_message(message * data)=0; //virtual function that is passed all received messages and
returns the reply to be sent
048     mpiclient * comms;
049 };
050
051 #endif
052
053
```

2.9 Source File: ./rater/raternode.h

```
001 /*
002 ###Header###
003 *****filename*****
raternode.h
004 *****author*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
Tue Oct 28 23:01:28 GMT 2003
009 *****description*****
010 This file contains the definition of the sds node
011 #####
012 */
013 // This program is free software; you can redistribute it and/or modify
014 // it under the terms of the GNU General Public License as published by
015 // the Free Software Foundation; either version 2 of the License, or
016 // (at your option) any later version.
017 //
018 // This program is distributed in the hope that it will be useful,
019 // but WITHOUT ANY WARRANTY; without even the implied warranty of
020 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
021 // GNU General Public License for more details.
022 //
023 // You should have received a copy of the GNU General Public License
024 // along with this program; if not, write to the Free Software
025 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
026
027
028 #include <iostream>
029 #include <stdlib.h>
030 #include <stdio>
```

```
031 #include <cstring>
032 #include "../nodes/node.h"
033 #include "../lists/stringlist.h"
034 #include "../sds/sdstext.h"
035
036 #ifndef raternodeh
037 #define raternodeh
038
039 using namespace std;
040
041 class pagerater : public pnode
042 {
043 public:
044     pagerater(unsigned int port);
045     pagerater();
046 private:
047     message * process_message(message * data);
048     stringlist * querywords;
049 };
050
051 #endif
052
053
054
```

2.10 Source File: ./httperv/httpgetnode.h

```
001 /*
002 ###Header###
003 *****filename*****
httpgetnod.h
004 *****auther*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
10/10/2003 12:18:19
009 *****description*****
010 definition of the http get node class
011 #####
012 */
013 // This program is free software; you can redistribute it and/or modify
014 // it under the terms of the GNU General Public License as published by
015 // the Free Software Foundation; either version 2 of the License, or
016 // (at your option) any later version.
017 //
018 // This program is distributed in the hope that it will be useful,
019 // but WITHOUT ANY WARRANTY; without even the implied warranty of
020 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
021 // GNU General Public License for more details.
022 //
023 // You should have received a copy of the GNU General Public License
024 // along with this program; if not, write to the Free Software
025 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
026
027
028 #include <iostream>
029 #include <stdlib.h>
030 #include <cstring>
```

```
031 #include <sys/socket.h>
032 #include <stdio.h>
033 #include <errno.h>
034 #include <netdb.h>
035
036 #include "../nodes/node.h"
037 #include "../lists/stringlist.h"
038
039 using namespace std;
040
041 #ifndef httpgetnodeh
042 #define httpgetnodeh
043
044 void lower(char * data);
045 class httpgetnode : public pnode
046 {
047 public:
048 httpgetnode(unsigned int port);
049 httpgetnode();
050 virtual ~httpgetnode();
051 void set_proxy(char * proxyname){if (proxy!=NULL) delete proxy;proxy=new
char[strlen(proxyname)];strcpy(proxy,proxyname);}
052 char * get_proxy(){return proxy;}
053 void set_authstring(char * auth){if (authstring!=NULL) delete authstring;authstring=new
char[strlen(auth)];strcpy(authstring,auth);}
054 char * get_authstring(){return authstring;}
055 void set_port(int aport){wport=aport;}
056 int get_port(){return wport;}
057 private:
058 message * process_message(message * data);
059 char * get_page(char * url);
060 char * get_domain(char * url);
061 char * get_ip(char * domain);
```

```
062 char * get_dir(char * url);
063 char * process_html(char * page,char * domain,char * dir);
064 char * get_file(char * url);
065 char * remove_whitespace(char * page);
066 int connect(char * url);
067 char * proxy;
068 char * authstring;
069 int wport;
070 };
071
072 #endif
073
074
```

3 Java source files

3.1 Source File: ./mcp/MicroHTTP.java

```
001
002 //Starting point for the java server
003
004
005 public class MicroHTTP
006 {
007
008     // Creates a new instance of MicroHTTP
009     public MicroHTTP()
010     {
011     }
012
013     //starting point for this application
014     public static void main(String[] args)
015     {
016         httpserv_1 server=new httpserv_1();//creates the ellement that waits for http requests and then fires up the
server
017         while (true)
018             server.listen();//starts the server listening for new connections
019     }
020
021 }
022
023
```

3.2 Source File: ./mcp/PageRequest.java

```
001 import java.io.*;
002 import java.util.*;
003
004 //run each time a rquest comes through for a new server to be started up
005 public class PageRequest
006 {
007     public PageRequest()
008     {
009         request_headers=new String("");
010         request_header_table=new Hashtable();
011     }
012     public void set_request_headers(String headers)
013     {
014         System.out.println(headers);
015         request_headers=headers;
016     }
017
018     // extracts the headers from the received http request
019
020     /*
021     splits the headers received into a hash table
022     */
023     public void extract_headers()
024     {
025         BufferedReader lines=new BufferedReader(new StringReader(request_headers));//sets up a buffered string reader
026         class that allows the string to be read a line at a time in the same way http headers are sent
027         StringTokenizer strtok;
028         try
029         {
030             while (lines.ready())//while there are still more lines to be reader from the buffered string reader class
031             {
032                 String header_name=new String("");

```

```

031 String header_data=new String("");
032 strtok=new StringTokenizer(lines.readLine());//place the line into a stringtokenizer to covert it into its
constituent parts
033 header_name=strtok.nextToken();//the name of the header
034 while (strtok.hasMoreTokens())//next two lines extracts the data contanted in that header
035 header_data=header_data.concat(strtok.nextToken()).concat(" ");
036 request_header_table.put(header_name,header_data);//place the data in the header hash table using the name of
the header as the key
037 System.out.println(header_name);
038 System.out.println(header_data);
039 }
040 }
041 catch (Exception e)
042 {
043 }
044 }
045
046 // returns the header
047
048 /*
049 returns the entry in the header hash table pointed
050 to by name.
051 */
052 public String get_header(String name)
053 {
054 return (String)request_header_table.get(name);
055 }
056
057 // gets te name of the page requested
058 public String get_page()
059 {
060 String request_data=this.get_header("GET");
061 StringTokenizer strtok=new StringTokenizer(request_data);

```

```

062 String requested_page=strtok.nextToken();
063 strtok=new StringTokenizer(requested_page);
064 String page_name=strtok.nextToken("?");//extracted the name of the page before the peramiters bassed by the get
    cgi interface
065     return page_name;
066 }
067
068 // Returns a hashtable of the elements passed in the cgi string
069
070 /*
071  creates a hashtable containing all the element value pairs
072  passed in the get statement used to start a new server up.
073 */
074 public Hashtable get_cgi()
075 {
076     String request_data=this.get_header("GET");//retrive the get header
077     System.out.println(request_data);
078     StringTokenizer strtok=new StringTokenizer(request_data);//tokenizr for the requested data
079     String requested_page=strtok.nextToken();//extracted the page and dump the http protocol level id
080     strtok=new StringTokenizer(requested_page,"?&=");//tokenizer the page bassed upon the chars used as seperaters in
    urls
081     strtok.nextToken();
082     Hashtable ret=new Hashtable();
083     String param,value;
084     while (strtok.hasMoreTokens())//loop while ther are more element value pairs
085     {
086         param=strtok.nextToken();
087         value=strtok.nextToken();
088         value=value.replace('+',' ');//convert the pluses within the url back to spaces
089         System.out.println(param);
090         System.out.println(value);
091         ret.put(param,value);
092     }

```

```
093     return ret;
094 }
095 private String request_headers;
096 private Hashtable request_header_table;
097 }
098
099
```

3.3 Source File: ./mcp/httpsession.java

```
001 /*
002  * httpsession.java
003  *
004  * Created on 11 March 2004, 13:40
005
006
007 import java.net.*;
008 import java.io.*;
009 import java.util.*;
010
011 class httpsession extends Thread
012 {
013
014     // Constructor
015
016     /*
017     creates a new http session with thge passed socket
018     */
019     public httpsession(Socket newsock)
020     {
021         sock=newsock;
022     }
023
024     // starts the communication with the client
025
026     /*
027     recives the request from the client and then starts the server
028     up with the settings transmitted within the http request
029     */
030     public void run()
031     {
```

```

032 try
033 {
034     PageRequest newpage=new PageRequest();
035     InputStream indata=sock.getInputStream();
036     OutputStream outdata=sock.getOutputStream();
037     String request_header=new String("");
038     String temp;
039     String return_header=new String("HTTP/1.1 200 OK\nServer: MicroHTTP\nContent-Type: text/html\nConnection:
close\n\n");//returned headers
040     String return_data=new String("<html><body><div>This is the java MicroHTTP server v0.1<br />Unknowen System
Asked For</div></html>");//returned if there is and error in the request
041     String return_data2=new String("<html><body><div>This is the java MicroHTTP server v0.1<br />Started
SDSwebsearch</div></html>");//returned on successfull completion of request
042     BufferedReader buff=new BufferedReader(new InputStreamReader(indata));//Buffered reader for the input stream to
allow for per line file reading
043     while (buff.ready())//whilst there is data comming
044     {
045         temp=buff.readLine();//reads in a line
046         if (temp.equals(""))//if the line is the break line after the headers are transmitted
047         {
048             break;
049         }
050         else
051         {
052             request_header=request_header.concat(temp).concat("\n");//place headers into string
053         }
054     }
055     newpage.set_request_headers(request_header);
056     newpage.extract_headers();//extract the headers from the transmitted string
057     String page_name=newpage.getPage();//name of the page to be run
058     Hashtable perams=newpage.getCgi();//peramiters sent in the request
059     System.out.println(page_name);
060     if (page_name.equals("/sds"))//if page is sds

```

```
059     {
060         startcluster clust=new
startcluster("sdswebsearch.conf", (String)perams.get("query"), (String)perams.get("user")); //start up the system with the
sent perams
061         outdata.write(return_header.getBytes()); //and return the success page
062         outdata.write(return_data2.getBytes());
063     }
064     else
064     {
065         outdata.write(return_header.getBytes()); //or return the failure page on incorrect request
066         outdata.write(return_data.getBytes());
067     }
068     indata.close();
069     outdata.close();
070     sock.close();
071 }
072 catch (Exception e)
073 {
074
075 }
076 }
077 private Socket sock;
078 }
079
080
```

3.4 Source File: `./mcp/httpserv1.java`

```
001
002
003 import java.net.*;
004
005 public class httpserv_1
006 {
007
008     /** Creates a new instance of httpserv */
009     public httpserv_1()
010     {
011         try
012         {
013             sock=new ServerSocket(2080);//creates a new socket listening on port 2080
014         } catch (Exception e)
015         {
016
017         }
018     }
019     public void listen()
020     {
021         try
022         {
023             httpsession newsess;
024             newsess=new httpsession(sock.accept());//runs a new http session when a client connects
025             newsess.start();
026         } catch (Exception e)
027         {
028
029         }
030     }
}
```

```
031 private ServerSocket sock;  
032 }  
033  
034
```

3.5 Source File: ./mcp/startcluster.java

```
001 /*
002 ###Source###
003 *****filename*****
startcluster.java
004 *****auther*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
Fri Nov 14 12:58:36 GMT 2003
009 *****description*****
010 Loads and attaches a cluster out of the conf file
011 #####
012 */
013 // This program is free software; you can redistribute it and/or modify
014 // it under the terms of the GNU General Public License as published by
015 // the Free Software Foundation; either version 2 of the License, or
016 // (at your option) any later version.
017 //
018 // This program is distributed in the hope that it will be useful,
019 // but WITHOUT ANY WARRANTY; without even the implied warranty of
020 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
021 // GNU General Public License for more details.
022 //
023 // You should have received a copy of the GNU General Public License
024 // along with this program; if not, write to the Free Software
025 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
026
027 import java.util.*;
028 import java.net.*;
029 import java.io.*;
030 import java.sql.*;
```

```
031
032 class startcluster
033 {
034
035     //
036
037     /*
038
039     */
040     public startcluster(String conffile,String query,String user)
041     {
042         String nextpage;
043         Integer port;
044         int numof,c,d;
045         servnode tempnode;
046         System.out.println(conffile);
047         System.out.println(query);
048         System.out.println(user);
049         urller=new urllist();
050         thequery=new String(query);
051         theuser=new String(user);
052         this.loadcluster(conffile);
053         this.initlist(https);
054         this.initlist(raters);
055         this.startlist(raters);
056         this.startlist(https);
057         this.get_startingpages();
058         for (c=0;c<https.size();c++)
059             {
060                 nextpage=new String("");
061                 if (urller.is_more())
062                     break;
063                 nextpage=urller.get_next();
```

```

064 ((servnode)https.get(c)).addressmessage(new message("get",nextpage));
065 }
066 while (true)
067 {
068 }
069 }
070
071 //
072
073 /*
074
075 */
076 private startcluster()
077 {
078 }
079
080 //
081
082 /*
083
084 */
085 private void get_startingpages()
086 {
087     try
088     {
089         String sql=new String("SELECT URL FROM DefaultPages,Site Where SID=id UNION SELECT URL FROM
StartingPages,Site Where SID=id AND UID=");
090         sql=sql.concat("theuser");
091         sql=sql.concat(";");
092         Statement stmt=con.createStatement();
093         ResultSet res = stmt.executeQuery(sql);
094         while (res.next())
{

```

```

095         urler.add_page(res.getString(new String("URL")), "null");
096     System.out.println(res.getString(new String("URL")));
097     }
098 }
099 catch (Exception e)
100 {
101     System.err.println("Error on starting sites");
102     System.err.println(e);
103     System.exit(1);
104 }
105 }
106
107 //
108
109 /*
110
111 */
112     protected void initlist(LinkedList toinit)
113     {
114         int c,d,numof;
115         servnode tempnode;
116         numof=toinit.size();
117         for (c=0;c<numof;c++)
118         {
119             tempnode=(servnode)toinit.get(c);
120             for (d=0;d<raters.size();d++)
121                 tempnode.addrateserv((rateserv)raters.get(d));
122             for (d=0;d<https.size();d++)
123                 tempnode.addhttpserv((httpserv)https.get(d));
124                 tempnode.addurlistserv(urller);
125                 tempnode.set_query(thequery);
126                 tempnode.set_connection(con);
127                 tempnode.set_user(theuser);

```

```

128 }
129 }
130
131 //
132
133 /*
134
135 */
136 protected void startList(LinkedList tostart)
137 {
138
139     int c,d,numof;
140     servnode tempnode;
141     numof=tostart.size();
142     for (c=0;c<numof;c++)
143     {
144         tempnode=(servnode)tostart.get(c);
145         tempnode.start();
146     }
147 }
148
149 // loads the config file
150
151 /*
152 creates all the mcp threads for the separate cluster nodes and
153 connects to the database.
154 */
155 protected void loadcluster(String filename)
156 {
157     String http=new String();
158     String data=new String();
159     String rater=new String();
160     String temp;

```

```

161 StringTokenizer work;
162 String server;
163 Integer port;
164 int numof,c,d;
165 servnode tempnode;
166 message tempmes;
167     try
168     {
169         BufferedReader prefs=new BufferedReader(new FileReader(filename));
170         c=0;
171         temp=prefs.readLine();
172         while (c==0)//loops through all the server lines in the config file
173         {
174             if (temp.startsWith(new String("httpnode")))
175             {
176                 http=http.concat(" ").concat(temp.substring(9));//if it was a httpnode line add to the string of httpserv
177             }
178             else if (temp.startsWith(new String("ratenode")))
179             {
180                 rater=rater.concat(" ").concat(temp.substring(9));//if it was a ratenode line add to the string of ratenode
181             }
182             else if (temp.startsWith(new String("database")))
183             {
184                 data=temp.substring(9);//if a database line set throws database config string
185             }
186         }
187     } catch (FileNotFoundException e)
188     {
189         http=new String("localhost 3124");
190     }
191 }

```

```

192         catch (Exception e)
193         {
194         }
195     work=new StringTokenizer(http);//tokenize the http server string into address and ports
196     numof=work.countTokens()/2;//the number of servers is half the number of tokens as the is a token for address
    followed by a token for port
197     https=new LinkedList();
198     for (c=0;c<numof;c++)
199     {
200         server=work.nextToken();
201         port=new Integer(work.nextToken());
202         https.add(new Httpserv(server,port.intValue()));//create a httpserv class with the specified address and port
203         ((servnode)https.getLast()).attach();//attach the class to the node
204     }
205     work=new StringTokenizer(rater);
206     numof=work.countTokens()/2;
207     raters=new LinkedList();
208     for (c=0;c<numof;c++)
209     {
210         server=work.nextToken();
211         port=new Integer(work.nextToken());
212         raters.add(new rateserv(server,port.intValue()));//create a rateserv class with the settings extracted from the
    config file
213         ((servnode)raters.getLast()).attach();//attach to the latter node
214     }
215     try
216     {
217         Class.forName("com.mysql.jdbc.Driver").newInstance(); //loads the database driver
218         con = DriverManager.getConnection(data);//connects to the database stored in the config file
219     }
220     catch (Exception e)
221     {
222         System.err.println("error on connect to database");

```

```

223 System.err.println(e);
224 System.exit(0);
225 }
226 }
227
228 // Sends a message to a group
229
230 /*
231  sends the message tempmes to all of the nodes in the LinkedList
232  sebdto
233 */
234 protected void sendtoall(LinkedList sendto,message tempmes)
235 {
236     int numof,c;
237     numof=sendto.size();
238     servnode tempnode;
239     for (c=0;c<numof;c++)//loop through all entrys in the list
240     {
241         tempnode=(servnode)sendto.get(c);
242         tempnode.addressmessage(tempmes,1);//send the message to the node
243     }
244 }
245
246 // destructor
247
248 /*
249  sends a kill message to all the nodes
250 */
251 protected void finalize()
252 {
253     System.out.println("finalizing");
254     message tempmes=new message();
255     tempmes.set_name("kill");

```

```
256     tempmes.set_data("");
257         sendtoall(https,tempmes);
258     sendtoall(raters,tempmes);
259 }
260 protected LinkedList https;
261 protected LinkedList raters;
262     protected urllist urllier;
263     protected String thequery;
264     protected String theuser;
265     protected Connection con;
266 }
267
268
```

3.6 Source File: ./mcp/urllist.java

```
001 /*
002 ###Source###
003 *****filename*****
mpi.java
004 *****auther*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
Wed Oct 29 10:48:01 GMT 2003
009 *****description*****
010 This file describes the java version of the c++ mpi classes
011 #####
012 */
013 // This program is free software; you can redistribute it and/or modify
014 // it under the terms of the GNU General Public License as published by
015 // the Free Software Foundation; either version 2 of the License, or
016 // (at your option) any later version.
017 //
018 // This program is distributed in the hope that it will be useful,
019 // but WITHOUT ANY WARRANTY; without even the implied warranty of
020 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
021 // GNU General Public License for more details.
022 //
023 // You should have received a copy of the GNU General Public License
024 // along with this program; if not, write to the Free Software
025 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
026
027 import java.util.*;
028 import java.net.*;
029
030
```

```
031 class urllist
032 {
033     private HashSet addresses;
034     private HashSet tocrawl;
035
036     // Constructor
037
038     /*
039     sets up the list
040     */
041     public urllist()
042     {
043         addresses=new HashSet();
044         tocrawl=new HashSet();
045     }
046
047     // adds a page
048
049     /*
050     adds a page to the list if it is not already present else it
051     adds the refreing page to the already present entry for this
052     page.
053     */
054     synchronized public void add_page(String page,String from)
055     {
056         if (addresses.add(page))
057         {
058             linkspec temp=new linkspec(page);
059             tocrawl.add(temp);//if a new message add it to the tocrawl hash set
060             temp.add_link(from);
061         }
062     else
063     {
```

```
063 Iterator it=tocrawl.iterator();
064 linkspec temp;
065 while (it.hasNext())
066 {
067     temp=(linkspec)it.next();
068     if (temp.get_page()==page)//finds the entry that is of the page
069     {
070         temp.add_link(from);//and adds the link to that entry
071         break;
072     }
073 }
074 }
075 }
076
077 // updates the rating
078
079 /*
080  finds all the entries with this page as a referer and updates the
081  rating with this new rating.
082 */
083 synchronized public void change_rating(String page,int rating)
084 {
085     Iterator it=tocrawl.iterator();
086     linkspec temp;
087     while (it.hasNext())//loops through the list
088     {
089         temp=(linkspec)it.next();
090         temp.update_rating(page,rating);//checks if the page is a referer to this page and updates its rating accordingly
091     }
092 }
093
094 // returns false if there are more entries to crawl
095 synchronized public boolean is_more()
```

```

096 {
097     return tocrawl.isEmpty();
098 }
099
100 // gets next page to fetch.
101
102 /*
103  randomly picks a page to fetch. This randomness is biased by the
104  rating for each page. A page becomes more likely the higher its rating
105  */
106 synchronized public String get_next()
107 {
108     int total=0;
109     int temprating;
110     Iterator it=tocrawl.iterator();
111     linkspec temp=new linkspec("");
112     while (it.hasNext())//finds the sum of all the rating values
113     {
114         temp=(linkspec)it.next();
115         temprating=temp.get_rating();
116         if (temprating>1)
117             temprating=temprating+tocrawl.size();
118         total=total+temprating;
119     }
120     if (total==0)
121         return new String("");
122     int rand=(new Random()).nextInt(total);//picks a random number between 0 and the sum of ratings
123     total=0;
124     it=tocrawl.iterator();
125     while (it.hasNext())//the next section of code steps through the list, summing the amount of rating, until the
    random number is smaller than the rating, at which point it stops as this is the selected link
126     {
127         temp=(linkspec)it.next();

```

```
128     temprating=temp.get_rating();
129     if (temprating>1)
130         temprating=temprating+tocrawl.size();
131     total=total+temprating;
132     if (total>=rand)
133     {
134         it.remove();
135         return temp.get_page();
136     }
137 }
138 it.remove();
139 return temp.get_page();
140 }
141 }}
```

3.7 Source File: ./mcp/mpi.java

```
001 /*
002 ###Source###
003 *****filename*****
004 mpi.java
005 *****auther*****
006 Mark James Talbot
007 *****e-mail*****
008 siu00mjt@rdg.ac.uk
009 *****date*****
010 Wed Oct 29 10:48:01 GMT 2003
011 *****description*****
012 This file describes the java version of the c++ mpi classes
013 #####
014 */
015 // This program is free software; you can redistribute it and/or modify
016 // it under the terms of the GNU General Public License as published by
017 // the Free Software Foundation; either version 2 of the License, or
018 // (at your option) any later version.
019 //
020 // This program is distributed in the hope that it will be useful,
021 // but WITHOUT ANY WARRANTY; without even the implied warranty of
022 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
023 // GNU General Public License for more details.
024 //
025 // You should have received a copy of the GNU General Public License
026 // along with this program; if not, write to the Free Software
027 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
028
029 import java.net.*;
030 import java.io.*;
031
032 class mpi
```

```

033 {
034 // returns the port being used to conect to the recipient
035 public int get_port()
036 {
037     return lport;
038 }
039
040 // sends a message
041
042 /*
043     encodes a message class as xml and then transmits it to the
044     recipient via the stream already set up.
045 */
046 public void send_message(message data)
047 {
048     String front=new String("<?xml version='1.0'?'>\n<message name=''\"); //beging of the message
049     String middle=new String("\",'>"); //midlel of the message
050     String back=new String("</message>"); //end of the message
051     if (open==1)//if connection open
052     {
053         try
054         {
055             send.write(front.getBytes()); //sending the beging
056             send.write(data.get_name().getBytes()); //sending the name of the message
057             send.write(middle.getBytes()); //sending the midle
058             send.write(data.get_data().getBytes()); //sending the data of the message
059             send.write(back.getBytes()); //sending the end of the message
060             send.flush();
061         }
062         catch (Exception err)
063         {
064         }
065     }

```

```

066 }
067
068 // receives a message from the transmitter
069
070 /*
071  listens for a message on the socket and when it arrives it processes
072  it using the tailored xml decoder and places the information within a
073  message class to be used by the mcp.
074 */
075 public message recv_message()
076 {
077     String back=new String("</message>");
078     String middle=new String("\''>");
079     String front=new String("<message name=\''");
080     message ret=new message();
081     ret.set_name(new String("wtf?"));
082     ret.set_data(new String(""));
083     String in;
084     if (open==1)
085     {
086         try
087         {
088             while (recvtext.indexOf(back)==-1)//keep receiving the message untill the end part of the message is received
089             {
090                 in=recv.readLine();
091                 recvtext=recvtext.concat(in);
092                 recvtext=recvtext.concat("\n");
093             }
094             if (recvtext.indexOf(front)!=-1)
095             {
096                 int pos,len;
097                 pos=recvtext.indexOf(front)+front.length();//postion of the begining of the name section of the message
098                 len=recvtext.indexOf(middle);//postion of the midle section of the name

```

```

099 String temp=recvtext.substring(pos,len);//the string of the name
100 ret.set_name(temp);//places name into message class
101 pos=recvtext.indexOf(middle)+middle.length();//postion of the beging of the data section
102 len=recvtext.indexOf(back);//postion of the end of the data section
103 temp=recvtext.substring(pos,len);//extracts the data from the recvied text
104 ret.set_data(temp);//places data into message class
105     }
106 String temp=recvtext.substring(recvtext.indexOf(back)+back.length());//leaves the persistant received sting
with only new data
107     recvtext=temp;
108     System.out.println(ret.get_name());
109     }
110     catch (Exception err)
111     {
112     }
113     }
114     return ret;
115     }
116
117 // closes the stream being used for comunication
118 public int closeup()
119 {
120     if (open==1)
121     {
122         try
123         {
124             serversock.close();
125             open=0;
126         }
127         catch (Exception e)
128         {
129         }
130     }

```

```
131     return open;
132 }
133 protected int lport;
134 protected Socket serversock;
135 protected int open;
136 protected String recvtext;
137 protected OutputStream send;
138 protected BufferedReader recv;
139 }
140
141
```

3.8 Source File: ./mcp/rateserv.java

```
001 /*
002 ###Source###
003 *****filename*****
urlistserv.java
004 *****auther*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
Sun Nov 02 23:34:57 GMT 2003
009 *****description*****
010 This file contains the definition of the server side of the
011 url list client.
012 #####
013 */
014 // This program is free software; you can redistribute it and/or modify
015 // it under the terms of the GNU General Public License as published by
016 // the Free Software Foundation; either version 2 of the License, or
017 // (at your option) any later version.
018 //
019 // This program is distributed in the hope that it will be useful,
020 // but WITHOUT ANY WARRANTY; without even the implied warranty of
021 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
022 // GNU General Public License for more details.
023 //
024 // You should have received a copy of the GNU General Public License
025 // along with this program; if not, write to the Free Software
026 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
027
028 import java.net.*;
029 import java.util.*;
030 import java.io.*;
```

```
031 import java.sql.*;
032
033 //implements the server node for the ratter
034
035 class rateserv extends servnode
036 {
037
038
039     // Constructor
040
041     /*
042     sets up via the super class a node with a specified address and port.
043     */
044     public rateserv(String address,int port)
045     {
046         super(address,port);
047     }
048
049     // Constructor
050
051     /*
052     sets up the communication with the default address and port.
053     */
054     public rateserv()
055     {
056     }
057
058     // processes the next message in the queue
059
060     /*
061     sends the next message in the queue and then decides what to do
062     with the return message. If the return message is a rating message
063     the rating and page details are placed in the databas so that the
```

```

064 web page interface can pick up any new ratings.
065 */
066 public void processnextmessage(message send)
067 {
068     serv.send_message(send); //sends message to rater
069     message ret=serv.recv_message();
070     if (ret.get_name().equals(new String("rating")))
071     {
072         System.out.println(ret.get_data());
073         int pos1=ret.get_data().indexOf("[end]");
074         int pos2=ret.get_data().lastIndexOf("[end]");
075         String from=new String(ret.get_data().substring(0,pos1)); //extracts the url of this page
076         Integer rating=new Integer(ret.get_data().substring(pos1+5,pos2)); //extracts the rating of this page
077         String linktext=new String(ret.get_data().substring(pos2+5)); //extracts the title of this page
078         urller.change_rating(from,rating.intValue()); //update the pages that this page links to with
this pages rating
079     try
080     {
081         Statement insert=con.createStatement();
String state=new String("INSERT INTO Results (User,Url,Rating,Query,Linktext) VALUES (','"); //Begining of the
sql statement to place the new rating into the database
082
state=state.concat(user).concat("\','\','").concat(from).concat("\','\','").concat(rating.toString()).concat("\','\','").concat(query).con
the new data entrys onto the fixed sql statement.
083         insert.executeUpdate(state); //runs the sql statement
084     }
085     catch (Exception e)
086     {
087         System.err.println("error in sql insert");
088         System.err.println(e);
089     }
090 }
091 }

```

092 }
093
094

3.9 Source File: ./mcp/htpserv.java

```
001 /*
002 ###Source###
003 *****filename*****
htpserv.java
004 *****auther*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
Sat Nov 01 13:59:50 GMT 2003
009 *****description*****
010 defines the server side classes for the httpnode
011 #####
012 */
013 // This program is free software; you can redistribute it and/or modify
014 // it under the terms of the GNU General Public License as published by
015 // the Free Software Foundation; either version 2 of the License, or
016 // (at your option) any later version.
017 //
018 // This program is distributed in the hope that it will be useful,
019 // but WITHOUT ANY WARRANTY; without even the implied warranty of
020 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
021 // GNU General Public License for more details.
022 //
023 // You should have received a copy of the GNU General Public License
024 // along with this program; if not, write to the Free Software
025 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
026
027 import java.net.*;
028 import java.io.*;
029 import java.util.*;
030
```

```
031 class httpserv extends servnode
032 {
033     // Constructor
034
035     /*
036     sets up the httpserv with the address and port passed to it.
037     */
038     public httpserv(String address,int port)
039     {
040         super(address,port);
041     }
042
043     // Constructor
044
045     /*
046     sets up the server with the localhost address and with port 3124
047     */
048     public httpserv()
049     {
050     }
051
052     // process the next message in the queue
053
054     /*
055     takes the next message from the queue and then sends it, receives the
056     return message and decides what to do with it. If the message received
057     is a page message it is split into the url list which is added to the
058     local url list and the rest is sent to the page rater.
059     */
060     public void processnextmessage(message send)
061     {
062         serv.send_message(send); //sends next message in queue to the http node
063         message ret=serv.recv_message();//listen and receives the reply
```

```

064 message newmessage;
065 if (ret.get_name().equals(new String("page")))//if a page message then
066 {
067     String temp;
068     URL thispage;
069     BufferedReader buf;
070     try
071     {
072         buf=new BufferedReader(new StringReader(ret.get_data()));
073         thispage=new URL(buf.readLine());//reads the url for this page from the message
074         temp=buf.readLine();
075     }
076     catch (Exception e)
077     {
078         System.err.println("httperv bufferedreader");
079     }
080     System.err.println(e);
081     return;
082 }
083 URL nextlink;
084 newmessage=new message();
085 while ( temp.equals(new String(" [end]"))==false)//loops through all the urls contained in the message
086 {
087     try
088     {
089         nextlink=new URL(thispage,temp);//resolves the url into a fully qualified one
090         urller.add_page(nextlink.toString(),thispage.toString());//and adds it to the local url list
091     }
092     catch (Exception e)
093     {
094         System.err.println("httperv URL ");
095     }
096 }
097 System.err.println(e);
098 }
099 try
100 {
101     System.err.println(e);
102 }
103 catch (Exception e)
104 {
105     System.err.println(e);
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }

```

```

094 {
095     temp=buf.readLine();
096 }
097 catch (Exception e)
098 {
099     System.err.println("httpserv readline");
100     System.err.println(e);
101     return;
102 }
103 }
104 try
104 {
105     if (raters.size()>0)//creates the message to be sent to a rate containing this page
106     {
107         newmessage=new message();
108         newmessage.set_name("rate");
109         newmessage.set_data(thispage.toString().concat(new String("[end]")).concat(query).concat(new
String("[end]")).concat(buf.readLine()).concat(buf.readLine()));//formats the data so that it contains the url for this
page,the query string,the page content,the title of the page.
110         this.get_quickest(raters).address(newmessage);//send the message to the lowest loaded rater on the system
111     }
112 }
113 catch (Exception e)
114 {
115     System.err.println("httpserv rater");
116     System.err.println(e);
117 }
118     newmessage=new message();//sends a message containing the next url to grab to the http node
119     newmessage.set_name("get");
120     newmessage.set_data(urller.get_next());
121     this.get_quickest(httpgetters).address(newmessage);
122 }
123 }

```

```
124 else if (ret.get_name().equals(new String("bye")))
125 {
126     this.stop();
127 }
128 else if (ret.get_name().equals(new String("error")))
129 {
130     newmessage=new message();//gets the next page in the url list if the last one failed
131     newmessage.set_name("get");
132     newmessage.set_data(urller.get_next());
133     this.get_quickest(httpgetters).addressmessage(newmessage);
134 }
135 }
136 }
137
138
```

3.10 Source File: ./mcp/mpiserver.java

```
001 /*
002 ###Source###
003 *****filename*****
mpi.java
004 *****auther*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
Wed Oct 29 10:48:01 GMT 2003
009 *****description*****
010 This file describes the java version of the c++ mpi classes
011 #####
012 */
013 // This program is free software; you can redistribute it and/or modify
014 // it under the terms of the GNU General Public License as published by
015 // the Free Software Foundation; either version 2 of the License, or
016 // (at your option) any later version.
017 //
018 // This program is distributed in the hope that it will be useful,
019 // but WITHOUT ANY WARRANTY; without even the implied warranty of
020 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
021 // GNU General Public License for more details.
022 //
023 // You should have received a copy of the GNU General Public License
024 // along with this program; if not, write to the Free Software
025 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
026
027 import java.net.*;
028 import java.io.*;
029
030 // specific additions for the mpi server
```

```
031
032 class mpiserver extends mpi
033 {
034     // Constructor
035
036     /*
037      sets up the system to connect to a specified address and port.
038     */
039     mpiserver(String address,int port)
040     {
041         laddress=new String(address);
042         lport=port;
043         recvtext=new String("");
044         open=0;
045     }
046
047     // Constructor
048
049     /*
050      connects to the localhost address with the port of 3124
051     */
052     mpiserver()
053     {
054         laddress=new String("localhost");
055         lport=3124;
056         recvtext=new String("");
057         open=0;
058     }
059
060     // sets the client
061
062     /*
063      sets the address and port of the client to attach to

```

```

064 */
065 public void set_client(String address, int port)
066 {
067     laddress=new String(address);
068     lport=port;
069 }
070
071 // Returns the address of the system being connected to
072 public String get_address()
073 {
074     return laddress;
075 }
076
077 // attach to client
078
079 /*
080  if the socket is already closed connects to the client and sets up the buffered reader
081  to read from the socket and the output stream to transmit to the server.
082 */
083 public int attach()
084 {
085     if (open==0)
086     {
087         try
088         {
089             serversock=new Socket(laddress,lport);//opens the socket communication to the client
090             send=serversock.getOutputStream();//sets up the output stream
091             recv=new BufferedReader(new InputStreamReader(serversock.getInputStream()));//sets up the input buffered
092             reader
093             open=1;
094             }
095         catch (Exception err)
096         {

```

```
095     open=0;
096     System.err.println("Failed to connect");
097     }
098     }
099     return open;
100     }
101
102     // Destructor
103
104     /*
105     closes the connection if it is still open
106     */
107     protected void finalize() throws Throwable
108     {
109         if (open==1)
110             serversock.close();
111     }
112     protected String laddress;
113 }
114
115
```

3.11 Source File: ./mcp/message.java

```
001 /*
002  * message.java
003  *
004  * Created on February 18, 2004, 3:43 PM
005
006 /**
007  *
008  * @author siu00mjt
009  */
010
011 //java implimentation of the message passing class
012
013 class message
014 {
015     // Constructor
016
017     /*
018     Sets up a blank message
019     */
020     message()
021     {
022         name=new String("");
023         data=new String("");
024     }
025
026     // Constructor
027
028     /*
029     Sets up a message with the name namein and the data field datain
030     */
031     message(String namein,String datain)
```

```
030 {
031     name=new String(namein);
032     data=new String(datain);
033 }
034
035 // sets the name of the message
036 public void set_name(String in)
037 {
038     name=new String(in);
039 }
040
041 //sets the data of the message
042 public void set_data(String in)
043 {
044     data=new String(in);
045 }
046
047 //retrives the name of the message
048 public String get_name()
049 {
050     return name;
051 }
052
053 //retrives the data of a message
054 public String get_data()
055 {
056     return data;
057 }
058 private String name;
059 protected String data;
060 }
061
062
```

3.12 Source File: ./mcp/linkspec.java

```
001 /*
002  * linkspec.java
003  *
004  * Created on February 18, 2004, 4:54 PM
005  */
006 import java.util.*;
007 // class describes the linkage between a page and its referers for the url list
008 class linkspec
009 {
010     private HashSet links;
011     private String here;
012     private int avgrating;
013
014     // allocates memory for this link
015     public linkspec(String thispage)
016     {
017         here=thispage;
018         avgrating=1;
019         links=new HashSet();
020     }
021
022     // Adds a link into the hashtable
023     public void add_link(String page)
024     {
025         links.add(page);
026     }
027
028     //changes the average rating to reflect the new rating
029     public void update_rating(String page,int rating)
030     {
031         if (links.contains(page))
```

```
032 {
033     avgrating=(avgrating+rating)/2;
034 }
035 }
036
037 // retruns the page name
038 public String get_page()
039 {
040     return here;
041 }
042
043 //returns the average rating
044 public int get_rating()
045 {
046     return avgrating+1;
047 }
048 }
049
050
```

3.13 Source File: ./mcp/servnode.java

```
001 /*
002 ###Source###
003 *****filename*****
servnode.java
004 *****auther*****
005 Mark James Talbot
006 *****e-mail*****
007 siu00mjt@rdg.ac.uk
008 *****date*****
Sat Nov 01 16:05:41 GMT 2003
009 *****description*****
010 describes the basic functions common to all server nodes
011 #####
012 */
013 // This program is free software; you can redistribute it and/or modify
014 // it under the terms of the GNU General Public License as published by
015 // the Free Software Foundation; either version 2 of the License, or
016 // (at your option) any later version.
017 //
018 // This program is distributed in the hope that it will be useful,
019 // but WITHOUT ANY WARRANTY; without even the implied warranty of
020 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
021 // GNU General Public License for more details.
022 //
023 // You should have received a copy of the GNU General Public License
024 // along with this program; if not, write to the Free Software
025 // Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
026
027 import java.net.*;
028 import java.util.*;
029 import java.io.*;
030 import java.sql.*;
```

```
031
032 abstract class servnode extends Thread
033 {
034
035     // Constructor
036
037     /*
038      sets up a server with a specified address and port
039     */
040     public servnode(String address,int port)
041     {
042         serv=new mpiserver(address,port);
043         queue=new LinkedList();
044         raters=new LinkedList();
045         httpgetters=new LinkedList();
046         running=-1;
047         query=new String();
048     }
049
050     // Constructor
051
052     /*
053      sets up a default server
054     */
055     public servnode()
056     {
057         serv=new mpiserver();
058         queue=new LinkedList();
059         raters=new LinkedList();
060         httpgetters=new LinkedList();
061         running=-1;
062         query=new String();
063     }
```

```
064
065 // sets the query string for this search
066 public void set_query(String thequery)
067 {
068     query=new String(thequery);
069 }
070
071 // sets the host details for this server
072 public void set_host(String address,int port)
073 {
074     serv.set_client(address,port);
075 }
076
077 // returns the address of this server
078 public String get_address()
079 {
080     return serv.get_address();
081 }
082
083 // returns the port of this server
084 public int get_port()
085 {
086     return serv.get_port();
087 }
088
089 // returns the length of the queue
090
091 /*
092  returns the number of messages in the queue including any that
093  are being currently being process
094 */
095 public int get_queue_length()
096 {
```

```
097 if (running>0)
098     return queue.size()+running;
099 else
100     return queue.size();
101 }
102 // finds the least loaded node
103
104 /*
105  takes the list of a type of node and returns the one with the
106  smallest queue
107 */
108 protected servnode get_quickest(LinkedList find)
109 {
110     int rand;
111     if (find.size(>1)
112         rand=(new Random()).nextInt(find.size()-1);
113     else
114         rand=0;
115     return (servnode)find.get(rand);
116 }
117 // attaches to the client
118 public void attach()
119 {
120     try
121     {
122         while (serv.attach()==0)
123             sleep(1000);
124     } catch (Exception e)
125     {
126     }
```

```
127 }
128
129 // adds a new http server to the list of http fetch nodes
130 synchronized public void addhttpserv(httpserv newserv)
131 {
132     httpgetters.add(newserv);
133 }
134
135 // sets the list to use to store urls
136 synchronized public void addurllistserv(urllist newserv)
137 {
138     urllist=newserv;
139 }
140
141 // adds a new rater to the list of raters
142 synchronized public void addrateserv(rateserv newserv)
143 {
144     raters.add(newserv);
145 }
146
147 // adds a message to the queue
148
149 /*
150  * if priority is more than 0 then the message is added to the
151  * front of the queue else it is added to the end of the queue
152  */
153 synchronized public void addmessage(message send,int priority)
154 {
155     if (priority==0)
156     {
157         queue.addLast(send);
158     }
159     else
```

```
159 {
160     queue.addFirst(send);
161 }
162 }
163
164 // adds a message to the queue at the end
165 synchronized public void addmessage(message send)
166 {
167     addmessage(send,0);
168 }
169
170 // retrives the next message
171
172 /*
173  retrives the next message in the loop in a threa safe way
174 */
175 synchronized private message get_next()
176 {
177     message temp=(message)queue.getFirst();//get the next message
178     queue.remove(0);//remove the message from the list
179     return temp;
180 }
181
182 // main loop
183
184 /*
185 */
186 protected void checkifsend()
187 {
188     if (queue.size()>0)
189     {
190         message temp=get_next();//pulls next message of the list
191     }
```

```
192 if (((temp.get_name()).trim()).equals(new String("")))//skips if a blank message
193 {
194 }
195 else
196 {
197     running=1;//sets the running flag
198     processnextmessage(temp);//processes this message
199     running=0;//unsets the running flag
200     serv.closeup();//closes the connection
201     try
202     {
203         this.sleep(500); //sleeps for 500 milliseconds to give the connection time to rest
204     }
205     catch (Exception e)
206     {
207     }
208     serv.attach();//reattaches to the sclient
209 }
210 }
211 // sets the class to be used to access the data base
212 public void set_connection(Connection thecon)
213 {
214     con=thecon;
215 }
216 }
217 // sets the pre authenticated user id
218 public void set_user(String theuser)
219 {
220     user=new String(theuser);
221 }
222 }
```

```
223 // starts the thread
224
225 /*
226  starts the loop of sending messages from the queue and processing there returns
227 */
228 public void run()
229 {
230     while (true)
231     {
232         checkifsend();
233     }
234 }
235 abstract void processnextmessage(message send);
236 protected InputStream server_file;
237 protected mpiserver serv;
238 protected LinkedList queue;
239 protected LinkedList raters;
240 protected LinkedList httpgetters;
241     protected LinkedList datas;
242     protected urllist urller;
243 protected int running;
244     protected String query;
245     protected Connection con;
246     protected String user;
247 }
248
249
```

4 HTML source files

4.1 Source File: ./PHPinterface/template.html

```
001 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
002 <html>
003 <head>
004 <link rel="stylesheet" type="text/css" href="defaultskin.css" />
005 <title>
006 </title>
007 </head>
008 <body>
009 </body>
010 </html>
011
012
```

4.2 Source File: ./PHPinterface/index.html

```
001 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
002 <html>
003 <head>
004 <link rel="stylesheet" type="text/css" href="defaultskin.css" />
005 <title>
006 Sdswebsearch.com: Searching the web intelegntly
007 </title>
008 </head>
009 <script language="JavaScript">
010 // selects the frame size
011
012 /*
013 looks at the page being loaded into the frame and decides
014 the size the frame should be to hold that page
015 */
016 function chooseframesize()
017 {
018 framesrc=frames["theframe"].location.href;//gets the url of the fram being loaded
019 framesrc=framesrc.substr(framesrc.lastIndexOf("/")+1);//strips off the server and directry information from the
url leaving just the filename of the page
020 c=framesrc.indexOf("?");//finds the point that a get statement starts in the url
021 if (c>=0)
022 {
023 framesrc=framesrc.substr(0,c);//cuts out any get statement from the filename
024 }
025 else if (framesrc=="init.php")//if the startup script is loaded
026 {
027 framesize(window.innerWidth,0);//dont display anything
028 }
029 else if (framesrc=="showresults.php")//if the results page is loaded
030 {
031 framesize(window.innerWidth,1000);//set the frame to 1000 pixels deep. This is enough to dispaly the results
```

```

set page
032 }
033 else//if any outhter page
034 {
035     frameresize(window.innerWidth,50);//set to 50 pixels which is enough to display one line
036 }
037 }
038 // resizes the frame
039
040 /*
041 sets the frame size to be width pixels wide and height pixels tall
042 */
043 function frameresize(width ,height)
044 {
045     document.getElementById("theframe").width=width;//sets the width
046     document.getElementById("theframe").height=height;//sets the height
047 }
048 </script>
049 <noscript>
050 Please enable java script or use a Javascript enabled browser to have the iframe containing the search engine to
resize
051 <br />
052 </noscript>
053 <body>
054 <div>
055 <div id="header">
056 
057 </div>
058 <br />
059 <div id="aframe">
060 <iframe scrolling="No" id="theframe" name="theframe" onLoad="chooseframesize()" src="start.php">
061 </iframe>
062 </div>

```

```
063     </div>
064     </body>
065 </html>
066
067
```

4.3 Source File: ./PHPinterface/adduser.html

```
001 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
002 <html>
003 <head>
004 <link rel="stylesheet" type="text/css" href="defaultskin.css" />
005 <title>
006 Add new user
007 </title>
008 </head>
009 <body>
010 <form action="adduser.php" method="post" enctype="application/x-www-form-urlencoded">
011 <div>
012 <b>Username:</b>
013 <input type="text" name="user" />
014 <b>Password:</b>
015 <input type="password" name="pass" />
016 <b>Access Level:</b>
017 <select name="type">
018 <option value="user" />User
019 <option value="admin" />Administrator
020 </select>
021 <input value="Go" type="submit" />
022 </div>
023 </form>
024 </body>
025 </html>
026
027
```

4.4 Source File: ./PHPinterface/login.html

```
001 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
002 <html>
003 <head>
004 <link rel="stylesheet" type="text/css" href="defaultskin.css" />
005 <title>
006 Login
007 </title>
008 </head>
009 <body>
010 <form action="dologin.php" method="post" enctype="application/x-www-form-urlencoded">
011 <div>
012 <b>Username:</b>
013 <input type="text" name="user" />
014 <b>Password:</b>
015 <input type="password" name="pass" />
016 <input value="Go" type="submit" />
017 </div>
018 </form>
019 </body>
020 </html>
021
022
```

5 PHP source files

5.1 Source File: ./PHPinterface/redirect.php

```
001 <?php
002 include("dbconnect.php");//loads database
003 $id=$_GET["id");//gets the page id
004 $sql="SELECT Url FROM Results WHERE id=\',$id\','";//reuests the url for the page in the database
005 $sql_result=mysql_query($sql,$con);//runs sql
006 while ($row=mysql_fetch_array($sql_result))//if the url is in the database
007 {
008     $Url=$row["Url"];
009     header("Location: $Url");//send the browser to the correct page
010 }
011 ?>
012
013
```

5.2 Source File: ./PHPinterface/adduser.php

```
001 <?php
002 include("dbconnect.php");//connects to the database
003 $user=$_POST["user"];//gets the username from the post header sent
004 $pass=$_POST["pass"];//gets the password from the post header sent
005 $type=$_POST["type"];//gets the type of user(admin or user)
006 if ($type="admin")//converts the user string to a number denoting type
007     $priv=1;
008 else
009     $priv=0;
010 $sql="INSERT INTO Users (Username,Password,Privileges) VALUES (\'$user\',\'$pass\',\'priv\')";//inserts the
user into the database
011 ?>
012 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
013 <html>
014 <head>
015 <link rel="stylesheet" type="text/css" href="defaultskin.css" />
016 <title>
017 Added User
018 </title>
019 </head>
020 <body>
021 <div>
022 <?php
023     echo "Added $user to the user database";
024 ?>
025 </div>
026 </form>
027 </body>
028 </html>
029
```


5.3 Source File: ./PHPinterface/dologin.php

```
001 <?php
002 include("dbconnect.php");//connects to database
003 $user=$_POST["user"];//gets username
004 $pass=$_POST["pass"];//and password from post header
005 $query="SELECT id FROM Users WHERE username=\'\'".$user."\'\' AND password=\'\'".$pass."\'\'";//sets the sql query to see
    if user exists
006 $sql_result=mysql_query($query,$con);//runs sql
007 $userid=0;
008 if ($row=mysql_fetch_array($sql_result))//if user exists send to the console
009 {
010     $userid=$row["id"];
011     header("Location: console.php?user=$userid");
012 }
013 else//else send them back to the login form
014 {
015     header("Location: login.html");
016 }
017 ?>
018
019
```

5.4 Source File: ./PHPinterface/console.php

```
001 <?php
002 if (isset($_GET["user"]))
003 {
004     include("dbconnect.php");//connect to database
005     $userid=$_GET["user"];//get user name from
006     $sql="SELECT Privileges FROM Users WHERE id=$userid";//sql to find out user rights
007     $sql_result=mysql_query($sql,$con);//run the sql
008     if ($row=mysql_fetch_array($sql_result))
009     {
010         $priv=$row["Privileges"];
011         if ($priv==1)
012             header("Location: adminconsole.php?user=$userid");//if a admin user send to admin console
013         else
014             header("Location: start.php?user=$userid");//else send to the start page
015     }
016     else
017     {
018         header("Location: login.html");//if user not accept send to login screen
019     }
020     else
021     {
022         header ("Location: login.html");//if user not sent go to login page e.g. when somebody manual enters url of page
023     }
024 }
025
```

5.5 Source File: ./PHPinterface/server.php

```
001 <?php
002 $server="grieg";
003 function start($querytorun,%theuser)
004 {
005     $allow_url_fopen=true;//allows php to open urls like files
006     readfile("http://".$server.":2080/sds?query=".$querytorun."&user=".$theuser);//starts the server
007 }
008 ?>
009
010
011
012
```

5.6 Source File: ./PHPinterface/dbconnect.php

```
001 <?php//config file for connecting to the database
002 $con=mysql_connect("streeep", "siu00mjt", "siu00mjt");//connects to the database
003 $db=mysql_select_db("siu00mjt", $con);//selects the database to use
004 ?>
005
006
```

5.7 Source File: ./PHPinterface/adminconsole.php

```
001 <?php
002 $user=$_GET["user"];//gets the user id pass from login
003 ?>
004 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
005 <html>
006 <head>
007 <link rel="stylesheet" type="text/css" href="defaultskin.css" />
008 <title>
009 </title>
010 </head>
011 <body>
012 <div id="menu">
013 <a href="adduser.html">Add a User</a><br />
014 <a href="start.php?user=?php echo $user"; ?>>Start a search</a><br />
015 </div>
016 </body>
017 </html>
018
019
```

5.8 Source File: ./PHPinterface/start.php

```
001 <?php
002 if (isset($_GET["user"]))//if the user is set display the query enry form
003 {
004     echo "<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN\"
    \ 'http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\"><html><head><link rel=
    \ 'stylesheet\"
    href=
    \ 'defaultskin.css\" /><title>Sdswebsearch.com</title>";
005     echo "</head>\n";
006     echo "<body>\n";
007     echo "<form action=
    \ 'init.php\" method=
    \ 'post\" enctype=
    \ 'application/x-www-form-urlencoded\">\n";
008     echo "<div>\n";
009     echo "<b>Query:</b>\n";
010     echo "<input type=
    \ 'text\" name=
    \ 'query\" />\n";
011     echo "<input value=
    \ 'Go\" type=
    \ 'submit\" />\n";
012     echo "<input type=
    \ 'hidden\" name=
    \ 'user\" value=
    \ '$_REQUEST["user"].\" />";
013     echo "</div>\n";
014     echo "</form>\n";
015     echo "</body>\n";
016     echo "</html>";
017 }
018 //or if not send back to login form
019 {
020     header ("Location: login.html");
021 }
022 ?>
023
024
025
```

5.9 Source File: ./PHPinterface/init.php

```
001 <?php
002 include("dbconnect.php");//loads the database
003 $query=$_POST["query"];//the query as a text string with words seperated as spaces unless they are considered the same
    word where they are seperated by _
004 $user=$_POST["user"];//gets the user id
005 $sql="DELETE FROM Results WHERE User=\'$user\' AND Query=\'$query\'"; //removes any outdate query from the
    database
006 $sql_result=mysql_query($sql,$con);//runs sql
007 $passed_query=urlencode($query);//encodes the query in url form
008 include("server.php");//loads config
009 start($passed_query,$user);//starts server
010 header("Location: showresults.php?query=".$passed_query."&user=".$user);//transfers user to results page
011 ?>
012
013
```

5.10 Source File: ./PHPinterface/showresults.php

```
001 <html>
002 <?php
003 include("dbconnect.php");//loads the database
004 $query=$_GET["query"];//gets the query that was passed
005 if (isset($_GET["page"]))
006 {
007     $page=$_GET["page"];//if the page to view has been sent view it
008 }
009 else
010 {
011     $page=0;//else view the first page
012 }
013 $user=$_GET["user"];//get the user that was passed
014 $sql="SELECT Url,Rating,id,Linktext FROM Results WHERE User=\'$user\' AND Query=\'$query\' AND Rating>0 ORDER BY
Rating DESC LIMIT".($page*30).",30";//sql to select the correct page of results for this search and this user
015 $sql_result=mysql_query($sql,$con);//run sql
016 <head>
017 <link rel="stylesheet" type="text/css" href="defaultskin.css" />
018 <meta http-equiv="refresh" content="10" />
019 <title>
020 Page <?php echo $page ?> of results for <?php echo $user ?> on <?php echo $query ?>
021 </title>
022 </head>
023 <body>
024 <div class="tbrow">
025 <div class="tbheadrating">Page Rating</div>
026 <div class="tbheadurl">Page Url</div>
027 </div>
028 <?php
029 while ($row=mysql_fetch_array($sql_result))//for each result
030 {
```

```
031 $Url=$row["Url"]; //get its url
032 $Rating=$row["Rating"]; //rating
033 $id=$row["id"]; //id
034 $linktext=$row["Linktext"]; //and title
035 echo "<div class='tbrow'>\n"; //start a new row in the table
036 echo " <div class='tbcelling'>$Rating</div>\n"; //display the rating
037 echo " <div class='tbcelling'><a href='redirect.php?id=$id',
target=''_top''>". $linktext. "</a></div>\n"; //put a link in the page to redirect the page to the correct url for this
page
038 echo "</div>\n";
039 }
040 ?>
041 <?php
042 echo "<div><hr /><a href='showresults.php?query=". (strpos($query, " ", "+")). "&user=". (strpos($user, " ",
"+")). "&pid=$pid&page=" . ($page+1). "'>Next Page</a></div>" //display the buttons to move forward and back within the pages
043 ?>
044 </body>
045 </html>
046
047
```