

ONLINE JUDGE FOR AI GAMES

*A Thesis submitted to
Indian Institute of Technology Mandi
for the award of the degree*

of

B.Tech

By

Abhimanyu Kumar

under the guidance of

Prof. Sukumar Bhattacharya

SCHOOL OF COMPUTING AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MANDI

JUNE 2013

CERTIFICATE OF APPROVAL

19/06/2013

Certified that the thesis entitled ONLINE JUDGE FOR AI GAMES submitted by ABHIMANYU KUMAR to Indian Institute of Technology Mandi, for the award of the degree of B. Tech has been accepted after examination held today.

Signatures

CERTIFICATE

This is to certify that the thesis entitled **Online Judge For AI Games**, submitted by **Abhimanyu Kumar** to Indian Institute of Technology Mandi, is a record of bona fide work under my (our) supervision and is worthy of consideration for the award of the degree of B. Tech of the Institute.

Dr. Sukumar Bhattacharya
Head of school
Visiting Associate Professor
School of Computing and Electrical Engineering

Date:

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is my own except for quotations and summaries which have duly been acknowledged. The thesis has not been accepted for any degree until now and is not currently submitted for the award of other degree.

Signature:

Name: **Abhimanyu Kumar**

Enrollment NO: B09001

Date: 19th June 2013

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my project guide **Dr. Sukumar Bhattacharya** who approved our idea and gave me the golden opportunity to do this project, which helped me in doing a lot of research and I came to know about many different things.

Secondly I would like to thank my team member **Samrat** and **Jitesh**, without whom it seems impossible to complete this project. They helped in developing many module of this project which was required.

I would also like to thank all the members of project evaluation team who consistently provided us the way to crack the task. It really helped us in going extra mile in search for a tiny improvement in our project.

Abhimanyu Kumar
B09001
Computer Science & Engineering
Batch: 2009-13
IIT Mandi

CONTENTS

Title Page	i
Certificate of Approval	ii
Certificate By a supervisor	iii
Students Declaration	iv
Acknowledgements	v
Contents	vi
Abstract	1
1. Introduction	1
2. Prior Work	1
2.1 Desdemona	1
2.2 Google AI challenge	2
3. System Specifications	2
4. System Architecture	3
4.1 Manager	4
4.2 Refree	5
4.3 Scheduler	5
4.3.1 Round robin Scheduler	6
4.3.2 Group Scheduling algorithms	12
4.3.3 Priority Based Scheduling Algorithm	21
4.4 Sandbox	27
5. System Requirements	30
5.1 Hardware Requirements	30
5.2 Software Requirements	30
6. References	30

Online Judge for AI Games

Abstract

In this paper we will explain the different aspects involved in developing game engine able to execute and rank Submitted bot according to the true skill value. I will start with the introduction of our project and then subsequent subsection I will explain the overall structure of game engine. I will explain scheduling algorithm in details and the time frame in achieving the efficient Scheduling algorithm. Game engine consists of many module. I will explain all the module in details.

1. Introduction

The aim of this project was to create an infrastructure for hosting artificial intelligence gaming competition. The system should accept bot codes from participants, schedule matches between bots of different participants and produce a ranking amongst the bots based upon their performance in the matches. While achieving these functionalities, focus was kept on making the system general and flexible so that it can host competitions for more than one game. Another important functionality which we tried to achieve is to provide tools to the users to submit specifications of their own games and host competitions for those games. Along with providing the above functionality we have tried on the functionalities of scheduler to make it fast. For obtaining the above bottom line we have gone through many scheduling process. Also idea was to run the users program in safe mode to avoid malicious attack on the system. For achieving this task we have created sandbox where users code will be compiled and executed. To ease the submission of bots by user we have created environment to accept bot in any language. Interaction with the code is being done by *stdin* and *stdout*. So user can submit bot in any languages.

2. Prior work

AI is a very popular field. Courses related to AI are widely taught in universities where students submit. Systems like the one we created in this project do exist in universities where they are mostly used for academic purposes. Also competitions like Google AI Challenge are conducted on similar systems. We will consider following two systems evaluate the prior work done in this area -

2.1 Desdemona

This system developed by Mr Arun Chaganty and is used at Indian Institute of Technology Madras mainly for the course on AI. The system accepts bots from students, holds matches between them and given out ranking based on which the students are assigned grades. Though the system is useful and works well, it has two main disadvantages:

- It accepts bots written in C++ programming language only.
- It supports only one game – Othello (or Reversi).

We have managed to overcome both of these disadvantages by providing support for multiple languages as well as multiple games.

2.2 Google AI Challenge

This is competition conducted every year by Google with each year a new game used for competition. This system supports multiple languages but is able to handle only a single game. Though the architecture of the system is such that it is extensible to other games, it has been observed that doing so does require a considerable amount of work. Comparatively, the system we designed is much flexible since it is possible to switch among different games by merely changing a few variable in a couple of configuration files.

3. System Specification

This section explains the functionalities that is being provided by the system. Following is the system specification offered by system:

Following is a list of all the functionalities offered by the system:

- a) The system allows participants to create an account after which they can submit bots of a competition. A participant can submit multiple bots.
- b) A participant can view step by step progress of the matches played by his/her bot.
- c) Website provides complete specification of the game that is being played and of how the bot should interact with the game engine in order to play the game.
- d) Starter packages are provided for each game to help the participants in getting started with writing bots.
- e) Participants can see performance of their bots relative to that of other bots through the 'Leaderboard' provided on the website.
- f) For ease of creating bots and their debugging, a tester package is provided to the participants through which they can run their bots on their own computers before submitting it for competition.

Apart from these functionalities, the system has following main features:

- a) It is capable of accepting bots in almost all common programming languages.
- b) It is currently supports a single game at time, but it is possible to easily switch between different games. Moreover, a little more work will enable the system to support multiple games at a time, i.e. competitions of different games can run in parallel at same time.

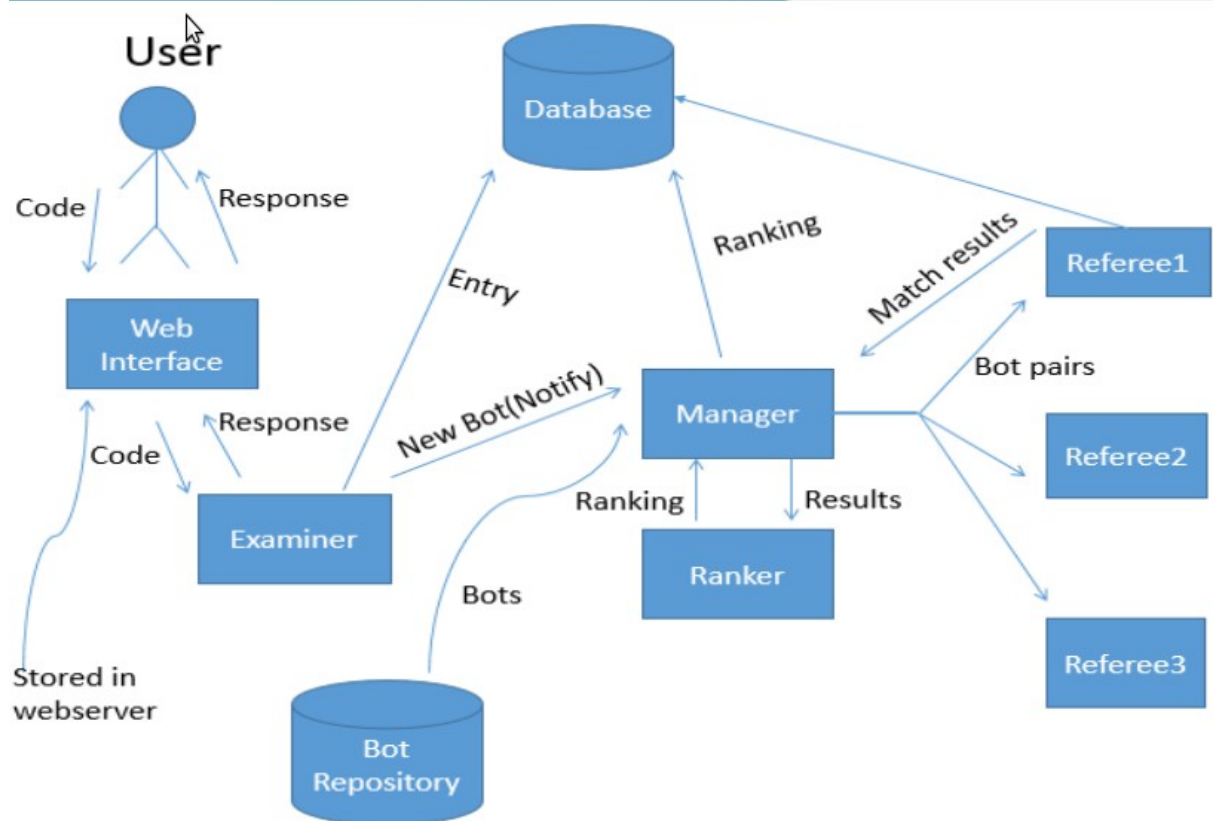
- c) Games with any number of players can be played on the system.
- d) The is possible to easily select from among different scheduling algorithms to pair bots for matches. New scheduling algorithms can also be added.
- e) To support large number of participants, the system implements a client server architecture in which tasks of running matches between bots are distributed to different workstations. Thus the performance limitation due to hardware are overcome.
- f) The bot code is run in a secure sandbox to avoid damage due to malicious code
- g) System also provides a web interface where apart from basic functionalities like submitting bots and viewing bot rankings, participants can also view a graphical display of the match that their bots have played.
- h) The system provides a utility which accepts game description in XML format and generates code for supporting that game on the system.

4. System Architecture

The systems has to perform wide range of tasks that have been described in previous section. To make the system easily manageable, following steps were taken while its designing the architecture:

- a) Since we had to adopt a **client server model** for distributing workload across many workstations, separate modules were created for the client and server. We have named the server as '**manager**' in our system since it forms the main component of the system. The clients on the other hand have been named as '**referee**' since they conduct matches between bots and return their results of the manager.
- b) Since we had to adopt a **client server model** for distributing workload across many workstations, separate modules were created for the client and server. We have named the server as '**manager**' in our system since it forms the main component of the system. The clients on the other hand have been named as '**referee**' since they conduct matches between bots and return their results of the manager.
- c) Separate module called '**games**' was created to storing game definitions which in turn contained a separate submodule for each game.
- d) Module named '**scheduling_algo**' was created to keep all the scheduling algorithms separate.
- e) Automation module contains code for automatically generate the code based on user specifications.

So we will get the following architectural diagram:



The working of the system can be summarized as below :

- A participant (or user) submits his bot through the web interface. The bot is stored in the bot repository. Database entries are created for the newly submitted bots.
- Manager, according to the selected scheduling algorithm, creates groups of bots between whom matches are to be played. We call this process as processes of creating '**match-ups**'.
- Referees continuously asks the manager for tasks. Manager assigns them match-ups to run match.
- Referees download the bots from bot repository, conducts matches between them and returns the result to the manager.
- Depending upon the results, the manager updates the ranks of the bots.

A detailed description of each of the components in the architecture diagram is given below.

4.1 Manager

Basically functionality is to manage the scheduling of matches. It provides Refree with the Status of the matches and the operation to be performed.

4.2 Referee

The task of a referee is to run matches for the match-ups assigned to it by the manager.

4.3 Scheduler

Scheduling is important module in our project. Scheduling algorithm decides the match to be happened and the choosing the opponent of bots. Decision of the bots goes through the many stages. The stages include the checking of the code at the time of the submission and the approval or validity of code. Validity of the code or the correctness of the code depends upon the many factor. The following factor is plays important role in deciding the correctness of the code.

- First of all the extension of the file is checked against the language which user has chosen to submit the code. Extension of the file is checked in dictionary and key is checked against value corresponding to that key and if it is matched then submitted code is considered to pass the first step correctly. Otherwise an error is popped out and users have to submit the code again with the required changed made in the code.
- Second step in the approval of the code is to check the any syntactical error is occurring in the user submitted bot. It is checked using the installed compiler in the system. If the compiler gives any error then it is read using the standard input and by parsing the error report valid reports is popped to the user. Users bot is first tested in the constraint environment, also called the jail in the UNIX operating system. Here memory is allocated for every bot and if the memory used by any bot goes beyond the allocated memory then error is popped to the user. Along with having constraint on accessing the memory, there are also constraint on the time. For running the each bot time is limited and if the run time goes beyond the time allocated for each program then it shows the indication of the error in the program. And user is given statistics of the error and required changes to be made in the program. After rectifying the error user can submit the bot again any number of times.
- Third step in verifying the correctness of the program is checking the input structure and output of the program is same as the format of the game input output defined. For each game input and output format is defined and for each submission of the bot this format is checked against the game for which the bot is being submitted. This is being checked by giving input to the program which is already defined and output of the program is matched against the output of the expected output. If the format does not match then error with the same message is popped to the user. After rectifying the bot user can submit the bot any number of times.

Checking the correctness of the program is the first step in processing the code. After the code is verified as correct and malicious free bot is stored in the database corresponding to the user submitting the bot and corresponding to the game code is being submitted. Then the given bot is assigned the status. This includes the verification status, compilation status, and scheduling status. Manager module is responsible for sending the given status value to the Referee and Referee will decide what to do next. Referee will call the scheduling algorithm if the match has not been scheduled and the implemented scheduling will schedule the match and store the result in the database. After the match has been scheduled leader board will get updated and this is done by the statistics module of the Referee.

The above information corresponds to the things being done before the scheduling algorithm actually getting used. In this section we are going to explain the different version of the scheduling algorithm we tried and pros and cons of the scheduling algorithm. We will explain in details the run time associated with the different-different algorithm and how it is advantageous over previously chosen algorithm.

We have implemented following algorithm during our development of our project.

1. Round robin algorithm
2. Group scheduling algorithm
3. priority based scheduling algorithm

Initially we had decided to use the round robin algorithm. During the first phase of our project we had been using round robin algorithm as our match scheduling algorithm. In the subsequent subsection we will be explaining the round robin algorithm and the reason behind switching to the new scheduling algorithm.

4.3.1 Round Robin Algorithm

In our first version of the project we implemented round robin scheduling algorithm. Later on due to time inefficiency of round robin algorithm we have changed our scheduling algorithm and implemented Group scheduling algorithm in second version of our project.

How it works?

Round robin scheduling as the name of the algorithm itself depicts the introduction of the algorithm. Match in this algorithm is scheduled in such a way that every bot has to play match with rest of the bots. In short following operation is carried out in implementation of this algorithm:

- We fetch all the bots available in our database into a list.

- Pick the one bot at a time from the list and make it play with all the bots except itself.
- After each match true skill of each bot is updated.

Pros

1. Easy to understand.
2. Easy to implement.
3. Linear time memory is required for storage of the Bots.

Cons

1. Very time inefficient

Example:

For illustration of this algorithm we are giving one example with operation required for round robin algorithm done. Suppose we have ten bots in our database.

STEP – 1

In step – 1 we will fetch all the bots from the database and put into the list. This works same as explained in the “How it work” Section of this algorithm.

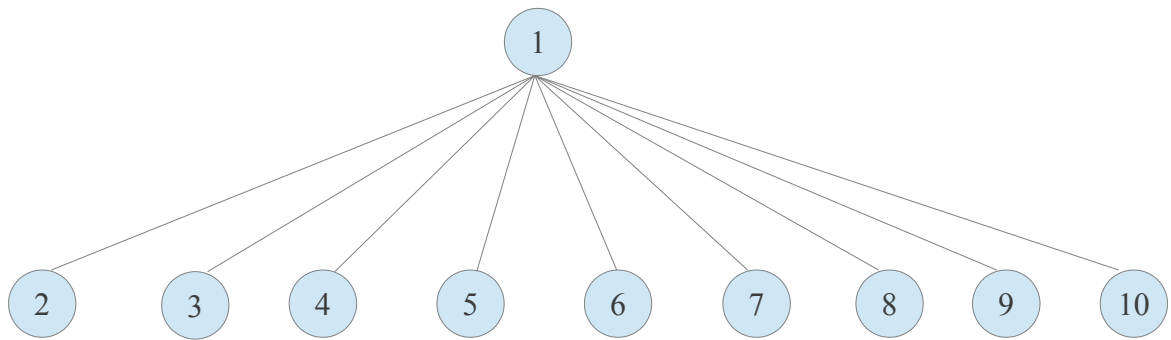


We get list of the bots. There are ten bots in a list. So now we will perform the second step of the algorithm on this list and update the true skill value of each bot after completion of each match.

STEP – 2

From the previous step we got the list containing the ten bots. Now from the algorithm we have to pick each bot from the list and schedule the match with all the other bots in the list. We have ten bots in the list, so we will pick up ten bots one by one and make it play with rest of the bot.

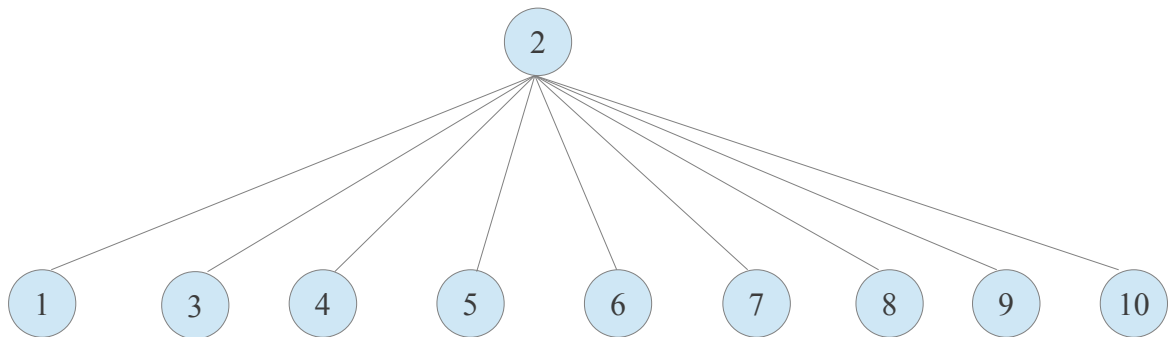
- In this step we will pick the first bot of the list and play it against all the remaining bots. We will get configuration like explained in the following figure.



So here Bot number 1 will play with all the other bots in the list.

Total number of match in this group = 9

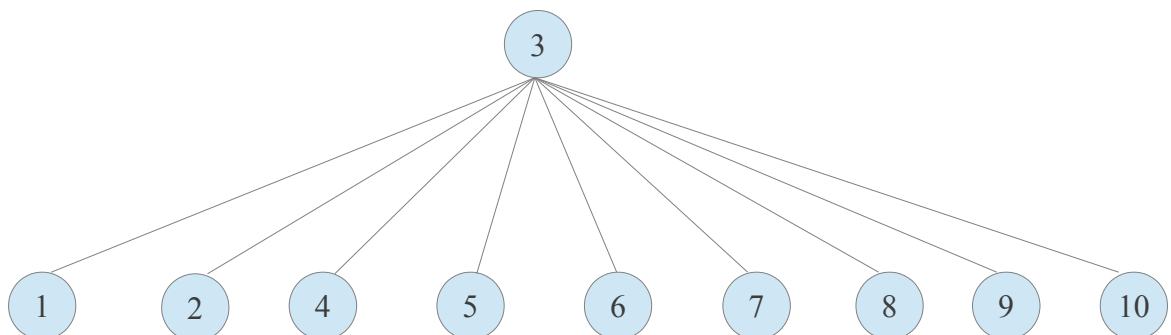
- In this step we will pick second bot and play it against all other bot. So we will get the configuration like:



So here Bot number 2 will play with all the other bots in the list.

Total number of match in this group = 9

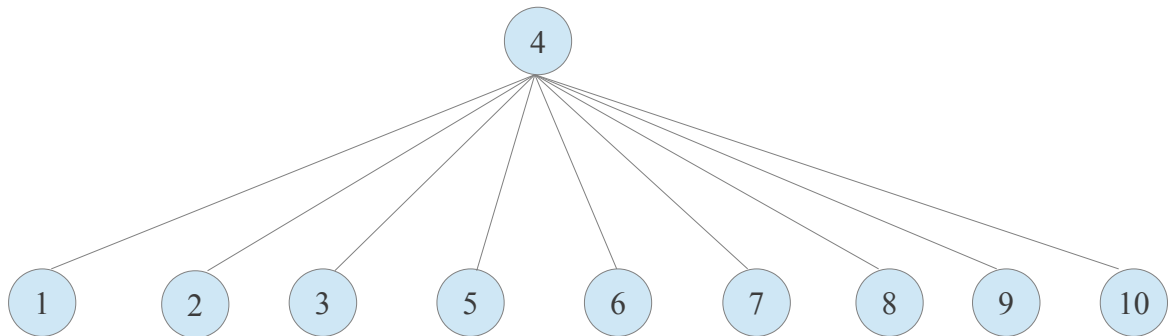
- In this step we will pick Third bot and play it against all other bot. So we will get the configuration like:



So here Bot number 3 will play with all the other bots in the list.

Total number of match in this group = 9

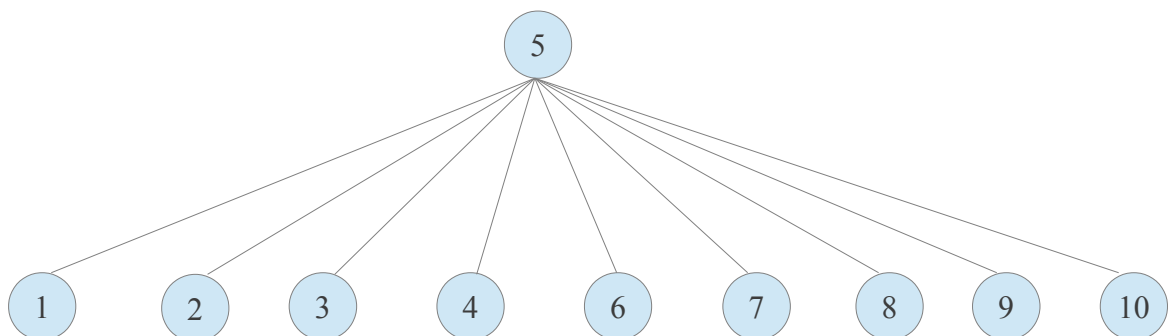
- In this step we will pick Fourth bot and play it against all other bot. So we will get the configuration like:



So here Bot number 4 will play with all the other bots in the list.

Total number of match in this group = 9

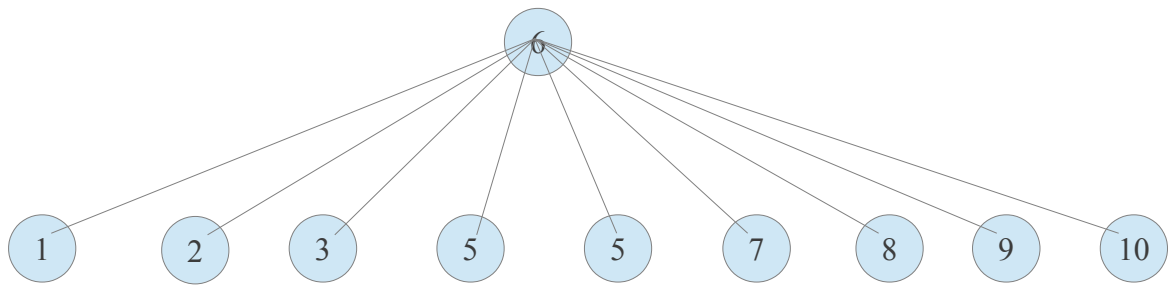
- In this step we will pick Fifth bot and play it against all other bot. So we will get the configuration like:



So here Bot number 5 will play with all the other bots in the list.

Total number of match in this group = 9

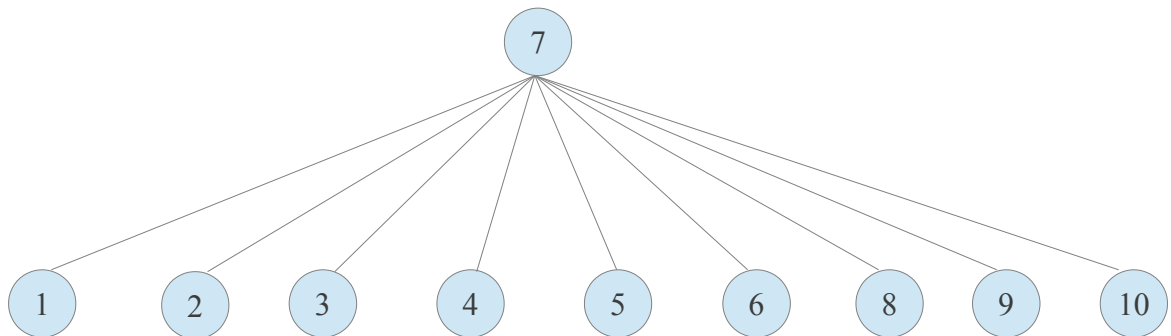
- In this step we will pick Sixth bot and play it against all other bot. So we will get the configuration like:



So here Bot number 6 will play with all the other bots in the list.

Total number of match in this group = 9

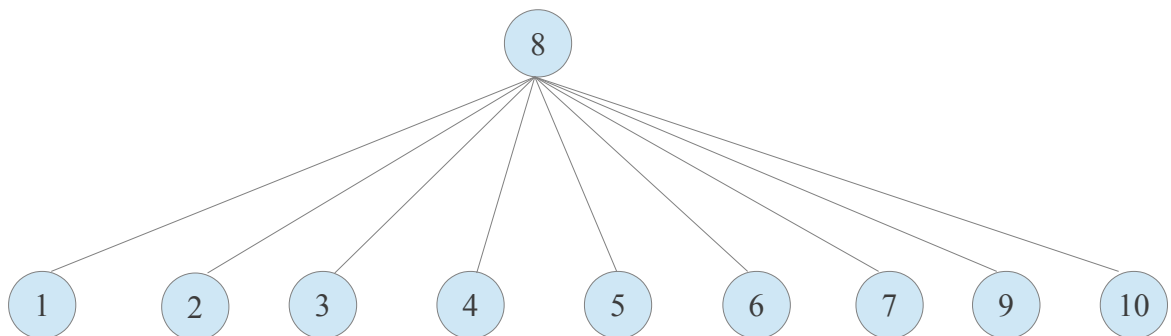
- In this step we will pick Seventh bot and play it against all other bot. So we will get the configuration like:



So here Bot number 7 will play with all the other bots in the list.

Total number of match in this group = 9

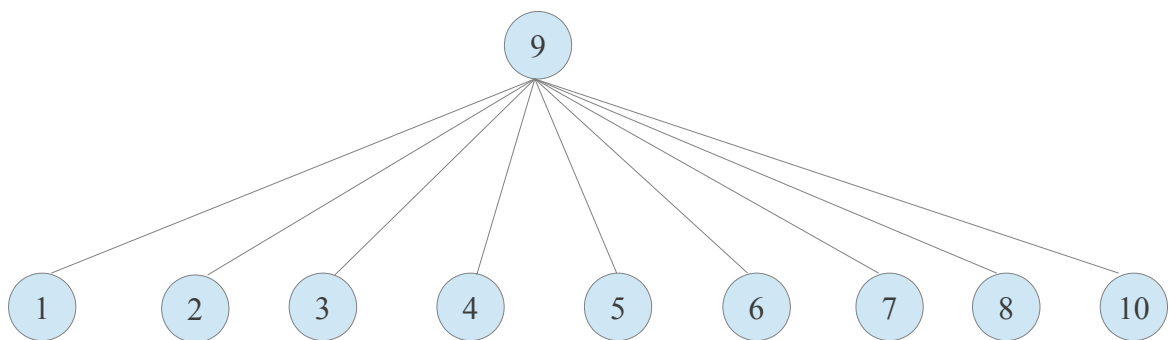
- In this step we will pick Eighth bot and play it against all other bot. So we will get the configuration like:



So here Bot number 8 will play with all the other bots in the list.

Total number of match in this group = 9

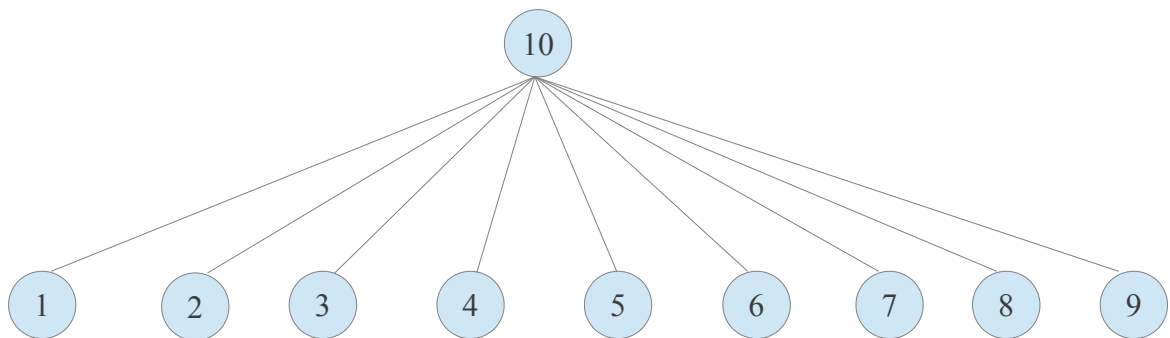
- In this step we will pick Ninth bot and play it against all other bot. So we will get the configuration like:



So here Bot number 9 will play with all the other bots in the list.

Total number of match in this group = 9

- In this step we will pick tenth bot and play it against all other bot. So we will get the configuration like:



So here Bot number 10 will play with all the other bots in the list.

Total number of match in this group = 9

Results

Here we have 10 bots and each bot will play with every other bot. Like in this case bot number 1 will play with 2,3,4,5,6,7,8,9,10.

So the match list will become
[(1,2),(1,3),(1,4) (1,10),
(2,1),(2,3),(2,4) (2,10),
(3,1),(3,2),(3,4) (3,10),
(4,1),(4,2),(4,3) (4,10),
(5,1),(5,2),(5,3) (5,10),
(6,1),(6,2),(6,3) (6,10),
(7,1),(7,2),(7,3) (7,10),
(8,1),(8,2),(8,3) (8,10),
(9,1),(9,2),(9,3) (9,10),
(10,1),(10,2),(10,3) (10,9)]

So in this case total number of matches to be played is $9 \times 10 = 90$.

We found this algorithm not good for our case. So we have changed our scheduling algorithm. In subsequent subsection we will be explaining the group scheduling algorithm which we are using in our project and its advantages over the previously implemented scheduling algorithm.

4.3.2 Group Scheduling Algorithm

Earlier we were using round robin algorithm for scheduling the matches between Bots and we found round robin algorithm very inefficient. In round robin algorithm each bot has to play match with every other Bot. It is very time inefficient. For large number of bot it is simply going to take time in order of square of the number of bots. To fulfill our need of schedule the match in less time we came up with this new scheduling algorithm called group scheduling algorithm.

How it Works?

In this algorithm we will group the bot in which each group contain some specific number of bot to be decided by us and based on this matches will be played in each group in round robin fashion as explained in the previous section. Based on the result some of the top bots will be selected from each of the group and then matches will be scheduled by making the group as usual as explained in previous paragraph. In short we perform the following step in the group scheduling algorithm:

- Fetch all the bot from the database, which is stored after correctness of the program has been satisfied and we are sure of having protection from malicious code.
- Decide on the number of bot that can be in a group based on the number of bots currently available in the database.
- Divide all the bots into number of group.

- Schedule the matches in each group in round robin fashion.
- Decide on the number of the bots to be selected from each group that will participate in scheduling in the next iteration.
- Flatten the selected bots from each group into a list.
- Again divide the bots into group based on the number of bots that can be in a group based on the value we came up with in second stage.
- Repeat the procedure until we number of bots became less than number of bots that can be in a group.
- When we find out the number of bots which is less than number of bots that can be in a group, we can play the match in round robin fashion to decide on the top bots.
- Terminate the scheduler.

Above points describe the step wise operation for the group scheduling algorithm in details. In next section we are going to explain the pros and cons of this scheduling algorithm over other known scheduling algorithm.

Pros

1. More time efficient than round robin algorithm.
2. Easy to implement.

Cons

1. Not fair share scheduling. It does not give fair chance to each bot for getting selected for the next round. In this case selected bot from one group can loose the game against bot from another group which has bot been selected for the next round.
2. We have to maintain the state of the previous as well as currently partitioned group, so it take more memory compare to the round robin algorithm, where Only one state at a time is maintained in the memory.

Example:

We are going to illustrate the group scheduling algorithm using a simple example step-wise. For simplicity number of bots assumed in this example is ten. We will fetch the program corresponding to each bot from the database. So we have ten bots stored in a list. Scheduler will perform

scheduling action on ten bots using the group scheduling algorithm.

Ten bots in a list will look like as:

STEP - 1



As explained above in the first step we have decide upon the number of bots going to participate or selected the bots from database and found the number of bots to be ten.

STEP – 2

Step 2 is to select the number of bots that can be in a group. As we found number of bots to be ten.

To illustrate the example by assuming the number of bots that can be in a group is 3.

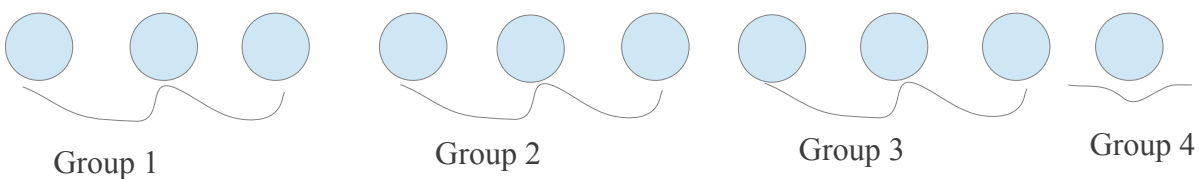
So total number of group will become

$$\text{roof}(10/3) = 4$$

So we have 4 group. First, second and third group contains three bots while fourth group contains only one bots.

STEP – 3

In the previous step we found the number of bots that can be in a group. In this step we will divide the 10 bots in group, each group can contain maximum number of bots equal to 3.

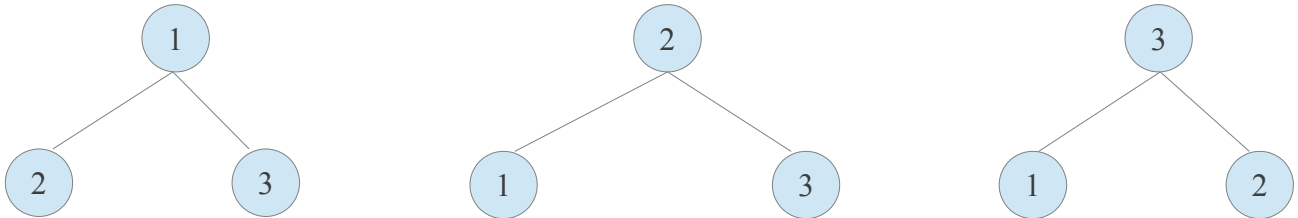


Now we have divided 10 bots into four groups, each group containing maximum of 3 bots. We will schedule the match between bots of each group in a round robin fashion which is explained in the next section.

STEP – 4

Now that each group contains maximum of three bots. all the bots in one group will play with all the other bots from the same group. This is called group wise round robin scheduling. This is carried out in same way as explained in the previous section.

Following figure shows the ways in which group wise round robin matches is carried out:



Above diagram shows the matches in each group is carried out in round robin fashion. So for the case when there are three bots in a group, total number of matches carried out is 6. in general total number of matches for the n bots will be $n*(n-1)$.

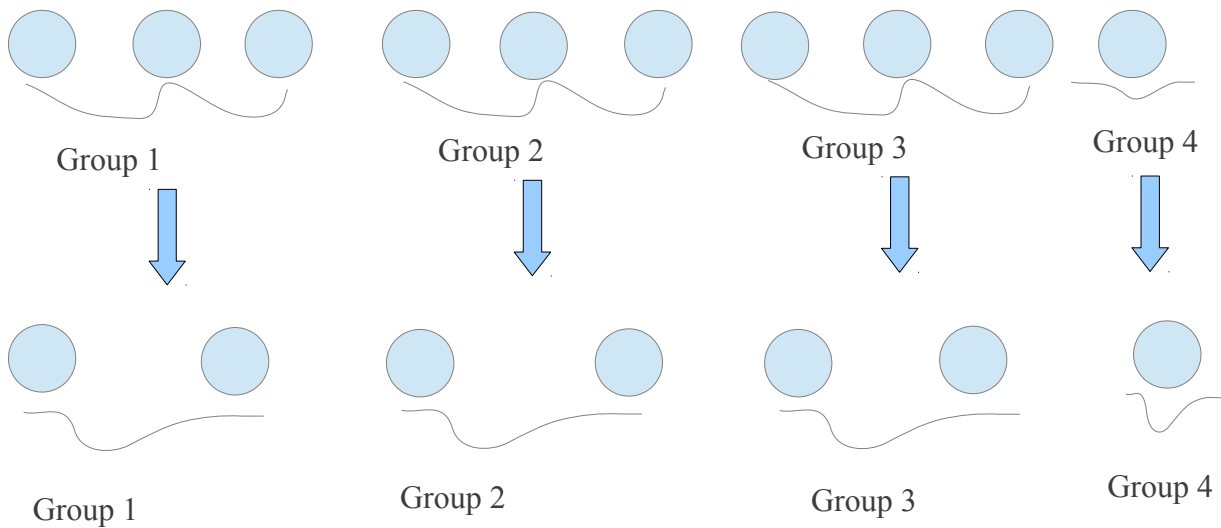
STEP – 5

In the previous step matches in each group has been carried out in the round robin fashion. Now for proceeding to the next step we need to select number of the bots from each group that will be advances to the next step. As in our case there are three bots in each group, so we can select two top bots from each group for demonstration purposes.

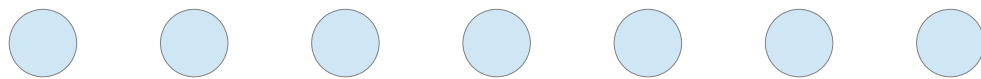
So Total number of bots that will be advances to the next level in each group = 2

STEP – 6

In this step we will select the top 2 bots from each group and then flatten the list to repeat the second step till we get the total number of bots less than maximum number of bots that can be selected.



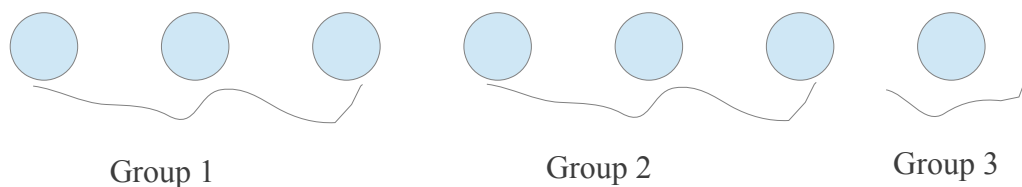
We have selected two bots from each group and now we have total number of seven bots. We will flatten all the bots in a list.



Now we see that we have the arrangement similar to the step -1. in step – 1 we had 10 bots in a list. While here we have seven bots in a list. So from onwards we can repeat the step-2 to get the desired result.

REPEAT STEP – 3

Similar to step-3 we will divide all the bots in a list in a group with total number of bots that can be in a group as decided.



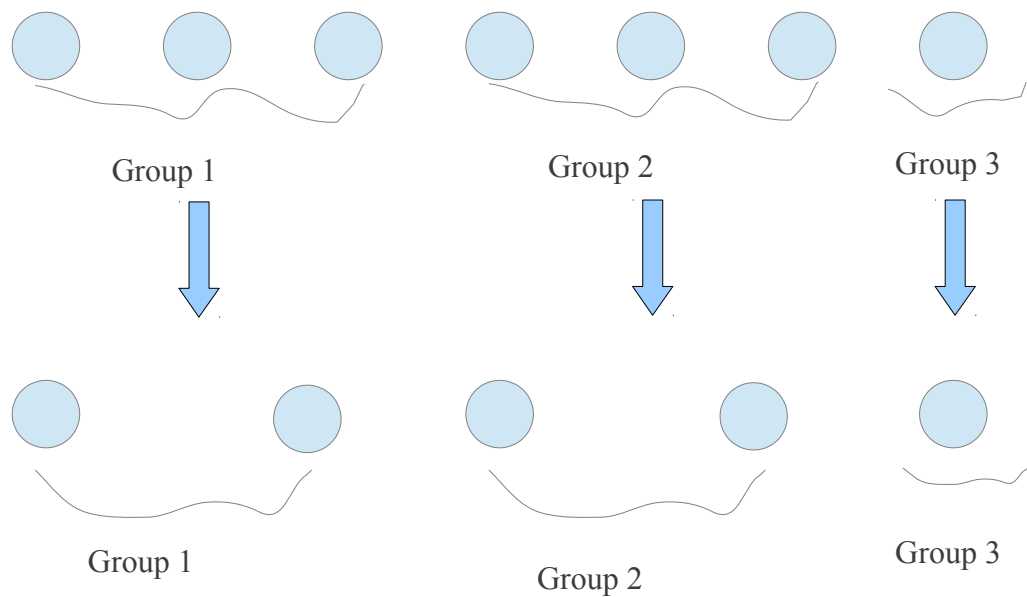
We have divided the 7 bots in a group of 3 bots each group as decided in step – 2. So we got the total number of groups = 3.

REPEAT STEP – 4

In this step again we will carry the operation as done in step -4. We will schedule the match in a group in round robin fashion.

REPEAT STEP – 6

In this step we will carry out the same operation done in the step – 6.



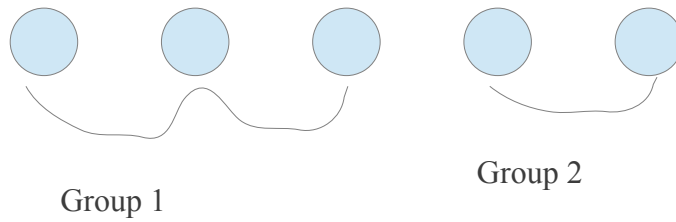
So we have selected 2 bots from each group and now we have total of five bots. We will keep this five bots in one list.



So we have five bots. We still have total number of bots greater than number of bots required for one group, so we will again split it into group.

REPEAT STEP – 3

In this step we will divide the total number of bots into group of 3.



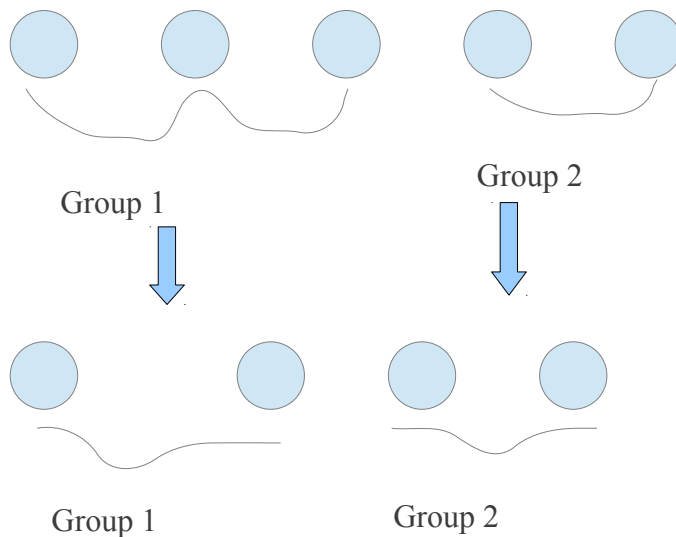
After splitting the total bots into group we obtained two groups containing 3 and 2 bots respectively.

REPEAT STEP – 4

In this step again we will schedule the matches in each group in round robin fashion as explained in the step – 4.

REPEAT STEP – 6

In this step we will select the top 2 bots from each group and flatten the all bots into one list.



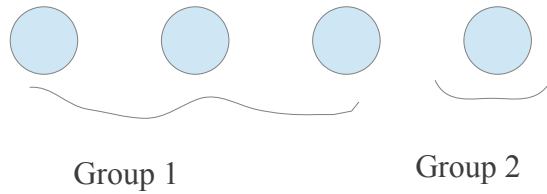
So after selecting the two bots from each group, we got the total number of 4 bots which will be putted into one list.



We have total number of bots currently in the list equal to four, which is still more than the total number of bots in a group. So we again repeat step – 3 and carry out the procedure.

REPEAT STEP – 3

We will split the currently available bots in a list into group of 3 bots per group. So we will obtain the following configuration.



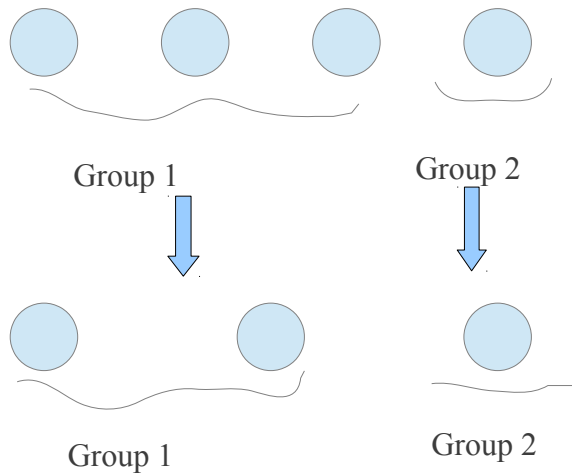
So we have divided all the bots into groups of 3 bots and 1 bots respectively.

REPEAT STEP – 4

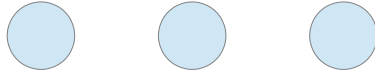
In this step again we will schedule the matches in each group in round robin fashion as explained in the step – 4.

REPEAT STEP – 6

In this step we will select top 2 bots from each group in case group contains more than three bots.



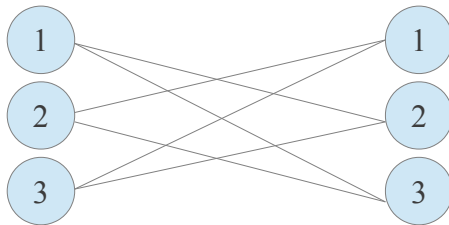
After selecting top 2 bots from each group we got total number of bots equal to 3. We will put this 3 bots into one list.



Now we got the total number of bots equal to 3 which is equal to total number of bots that can be in a group. So Now we don't need to split the list any more. We will apply round robin algorithm on these three bots and update the result. Which will be final score of all the bots.

FINAL STEP

Finally we have three bots in list. So we will apply round robin scheduling on list to obtained the result.



Round robin scheduling on the list will work same as depicted in above figure. Based on the result of the game true skill value of each bot will get updated. Bots will be ranked based on the true skill value of each bot. Bot with highest true skill value will be ranked best and on decreasing order of the true skill value ranking will also be in decreasing order.

Leader board will look like:

Leader Board:

Rank	Bot number
1	Bot 2
2	Bot 8
3	Bot 3
.	
.	
.	
.	
10	Bot 9

Result

Total number of matches that was played in this scheduling =

$$\begin{aligned} & 3*3*2 = 18 \\ + & 2*3*2 = 30 \\ + & 1*3*2 = 36 \\ + & 2 = 38 \\ + & 1*3*2 = 44 \\ + & 1*3*2 = 50 \end{aligned}$$

So this scheduling algorithm gave us total number of matches to be played In case of 10 number of bots is 50. While in case of round robin scheduling algorithm we are getting total number of matches to be played in case of 10 bots is 90. So this scheduling algorithm is more efficient than round robin scheduling algorithm.

4.3.3 Priority Based Scheduling Algorithm

We have also looked up for the priority based scheduling algorithm for applying in our problem. However we didn't find it efficiently and later on decided not to use it at all. In the subsequent subsection we will be explaining the working procedure of the priority based scheduling algorithm. We will also be explaining the disadvantages of the priority based scheduling algorithm over the algorithm we have explained above. Priority based scheduling algorithm is implemented in many real world problem. For example it is implemented in many network devices like router and other devices. Seeing the implementation of this algorithm in many area we thought that it will be good for serving our purposes.

How it works ?

In this scheduling algorithm every bot is assigned equal priority at the beginning. As the match proceed bot which is loosing their matches increases their priority. Scheduler checks for the winner of the match and based on the result, priority of the bot is increased. Now when again scheduler comes in active mode then two highest priority bot is selected and match between them is scheduled. Again the result is updated. As any bot wins then their priority is decreased. We have limitation factor which bounds or imposes restriction on the number of matches a particular bot can play. This is because when this condition will not be imposed then one bot which keeps loosing will be able to play forever because as bot looses its priority keeps increasing and it will be keep playing forever. So the imposition of the constraints on the number of matches each bot can play will help us from overcome this problem.

Pros

1. More efficient than round robin scheduling algorithm.
2. Fair share scheduler as it gives each bot a fair amount of match to play. Each bot even after losing gets chances to play more matches so that it can prove itself. If in some matches one bot is not able to win even after having encoded with an enriched strategy then this bot can use this strategy in the next matches and can take advantage of their strategy against the new bots.
3. On an average this scheduling algorithm schedules less number of matches compared to the round robin algorithm.

Cons

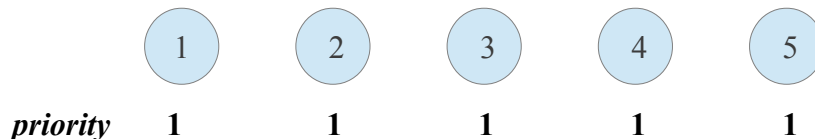
1. In the worst case the number of matches that will be played is equal to the number of matches played in case of the round robin algorithm.
2. Difficult to implement.

Example:

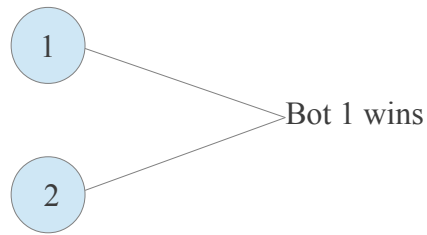
Suppose we have initially five bots in our database. As we know any bot is putted in the database after the verification of the bot against the correctness of the program. If the bot is found not follow the correctness rule then it is discarded and not putted in the database. So we assume that those five bots putted in the database are correct and not contains the malicious code that can harm our system. So we have five bots and scheduler is given task to perform the scheduling task on these five bots.



So we have five bots numbered 1 to 5. Initially each bot is initialized with the priority of 1.



As all the bots having the same priority initially so we will break this condition by arbitrarily choosing the two bots. Suppose we have chosen two bots with bot number 1 and 2.



Let in match between Bot 1 and Bot 2. Bot 1 won the match. So as per our algorithm priority of the bot will change.

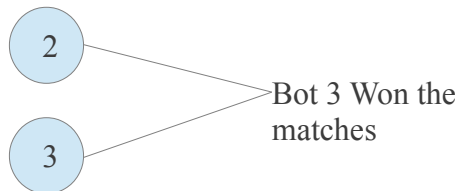
So the new priority table will be as Shown below:

Bot number	Priority	Number of Matches
1	0	1
2	2	1
3	1	0
4	1	0
5	1	0

As Bot 1 won the match and bot 2 loses the match so the priority of the bot 2 will increase while priority of the bot 1 will decrease. We are here also assuming that priority can be negative. So each time Bot wins its priority is increased by 1 and whenever any bot loses the match its priority is increased by 1.

So next time to schedule the match we will consider the Bot 2 as one playing bot and opponent will be the Bot among Bot 3, Bot 4 and Bot 5. Let's impose restriction that one bot can not play more than 3 matches. So we will also keep track of the number of matches each bot has played till now. And whenever bot has highest priority and it has already been played three matches then this bot will not be considered even after having the highest priority.

Now let next time we have selected Bot 2 and Bot 3 as pair going to play the match. And assume that again Bot 2 loses the match. So The configuration will become as shown pictorially below.

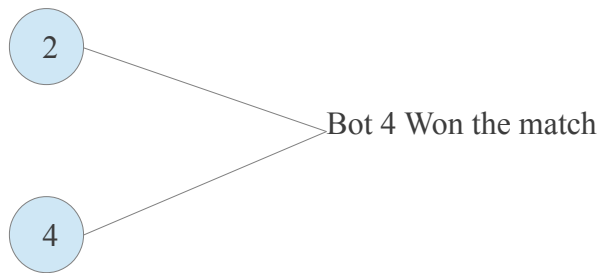


Priority table will be like as:

Bot number	Priority	Number of Matches
1	0	1
2	3	2
3	0	1
4	1	0
5	1	0

As Bot 2 again loosed the game so priority of the Bot 2 will get increase by 1 and probability of the Bot 3 will get decreased by 1. And also we update the number of matches row in priority table. Number of matches for the Bot 2 will become 3 and For bot 3 will get updated to 1. Now we can see that priority of the Bot 2 is highest, so in the next match Bot 2 will be part of one side and other Bot will be from Bot 4 and Bot 5.

Let in the next match Bot 2 and Bot 4 makes a pair for playing the match. So Now let bot 2 again looses the match as depicted in the below diagram.

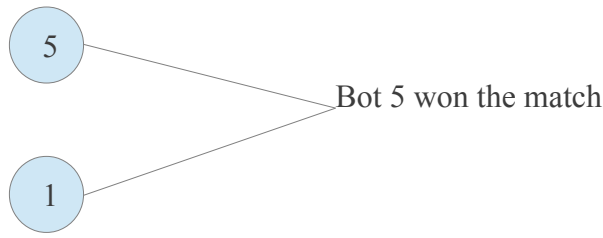


So Now let the match between Bot 2 and Bot 4 goes in hand of the Bot 4 i.e. Bot 4 won the match. So again we update the priority table as Bot 2 having priority increased by 1 and priority of the Bot 4 Decreased by 1. Number of matches column of the priority table also gets updated with Number of matches for Bot 2 and Bot 4 increased by 1. So the final priority table will look like as shown below:

Bot number	Priority	Number of Matches
1	0	1
2	4	3
3	0	1
4	0	1
5	1	0






So as we can see from the above table priority of the Bot 2 is having the highest priority. So according to priority Bot 2 will be participating in the next matches but it has exceeded the restriction on the number of matches each bot can play. So Bot 2 will not be participating in the next matches as Number of matches for Bot 2 will become 4 which will exceed our restriction for playing each bot maximum number of matches equal to 3. So Bot 2 will not participate in the next matches. So besides Bot 2, Bot 5 is having maximum priority. So in the next match Bot 5 will participate and against it among the Bot 1, Bot 3, Bot 4 will participate as Bot 1, Bot 3 and Bot 4 is having the same priority. Bot 2 will not participate as it has played the maximum number of matches restriction imposed on each bot.

Let in the next match Bot 5 and Bot 1 is playing against each other. And Bot 5 won the match.

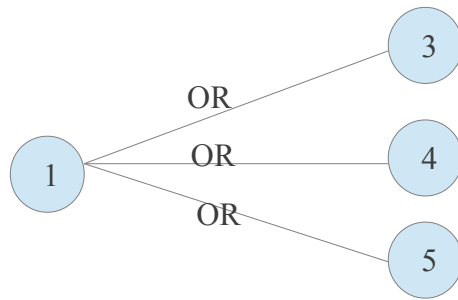


As bot 5 won the match so the priority table will get updated with the priority of Bot 5 decreased by one and priority of the Bot 1 increased by one as it loses the match. Number of Matches column of the priority table will also get increased by one for both the Bot (Bot 5 and Bot 1).

Configuration of priority table after this match will be like as shown below:

Bot number	Priority	Number of Matches
	1	2
	4	3
	0	1
	0	1
	0	1

So we can see that priority of Bot 1 become 1 and priority of Bot 5 become zero. Also the number of matches for Bot 1 become 2 and Number of matches for Bot 5 become 1. So all the Bot except bot 4 has played less number of matches than maximum number of matches each Bot can play. So next time Bot 1 will play the match as it is having the highest priority except Bot 2 against the Bot among Bot 3, Bot 4 and Bot 5. So the possible next matches will be selected as depicted in the figure.



Above diagram depicts the possible pair of matches that can be scheduled after the previous step has been executed and the updated priority table shows us the way to schedule the next matches based on the priority and Number of matches each Bot has played.

4.4 Sandbox

Security is one of the important concerns in our project. We don't want user to exploit our system by performing malicious code. This can happens in many ways. User with bad intention may write some system command that can be harmful for system. It may also happens that user wrote some code that requires large amount of memory. We want to save our system from such kind of effects. Sandbox is a term used in computer science to avoid harmful effects that can be imposed on the system. It is widely used in computer security as security is one of the most important concern these days. It is a security mechanism in computer security. Sandbox is sometimes also called live directory or working directory or test servers or development server. It is kind of providing the user virtual environment for testing and running their code.

Following are the reason behind implementing sandbox:

1. User can write code consists of malicious system command.
2. Program can acquire memory more than the desirable.
3. Users program can access network for some bad intention.
4. Keyboard input output access.

Following is the advantages of using sandbox.

1. Provide a tightly controlled set of resources for guest programs to run in.
2. Tightly controlled set of resources includes scratch space on the disk or memory.
3. Prohibition of network access.
4. Taking input from keyboard is heavily restricted.

The system can compile and execute codes and test them with the per-constructed data. Submitted

code may be run with restrictions including time limit, memory limit, security restriction and so on. It prevent incoming data from affecting a live system unless/until defined requirements or criteria have been met. It provides a virtual environment to test code and secure server from being get impacted by the malicious code.

We have implemented the sandbox part for the linux/unix operating system. The same idea can be extended for the windows operating system. It is example of the Jail sandbox system.

Here are the implementation step of the sandbox in Ubuntu we have did:

- A new Directory is created under the root of the real file system, where all the submitted code is going to execute.
- To prevent the malicious attack on the real file system, newly created directory is changed to root to create the illusion of real file system.
- Following is the Unix command to change the directory to root and treat it as a jail for the bot.
 - Sudo mkdir security
 - cd security
 - mkdir jail
 - cd jail
 - cd ../../
 - sudo chroot/security/jail

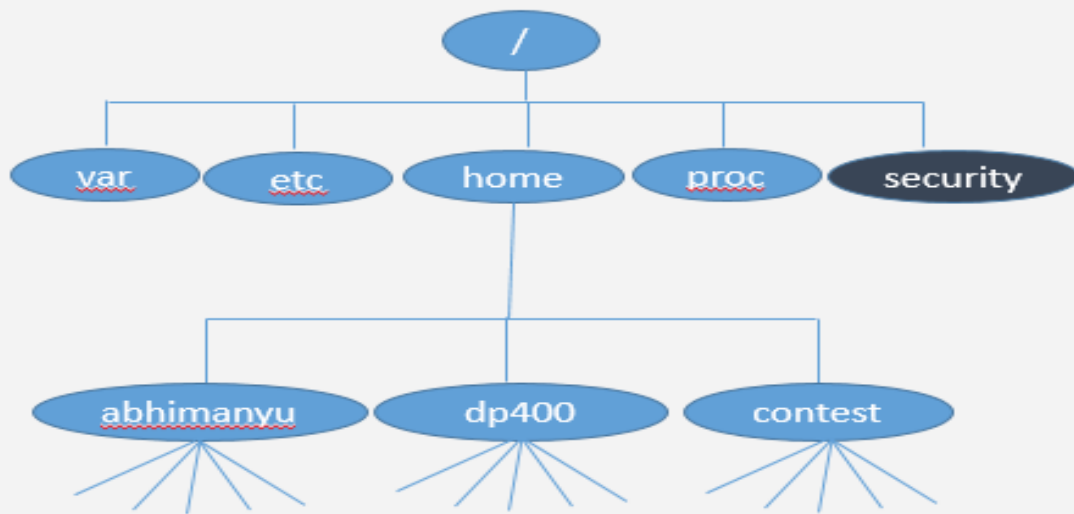
Explanation

Here we are creating a folder name security. Mkdir command of Unix eases the problem of creating a directory. We are then creating jail folder inside the security folder. Cd command in Unix helps us in switching from one directory to another directory.

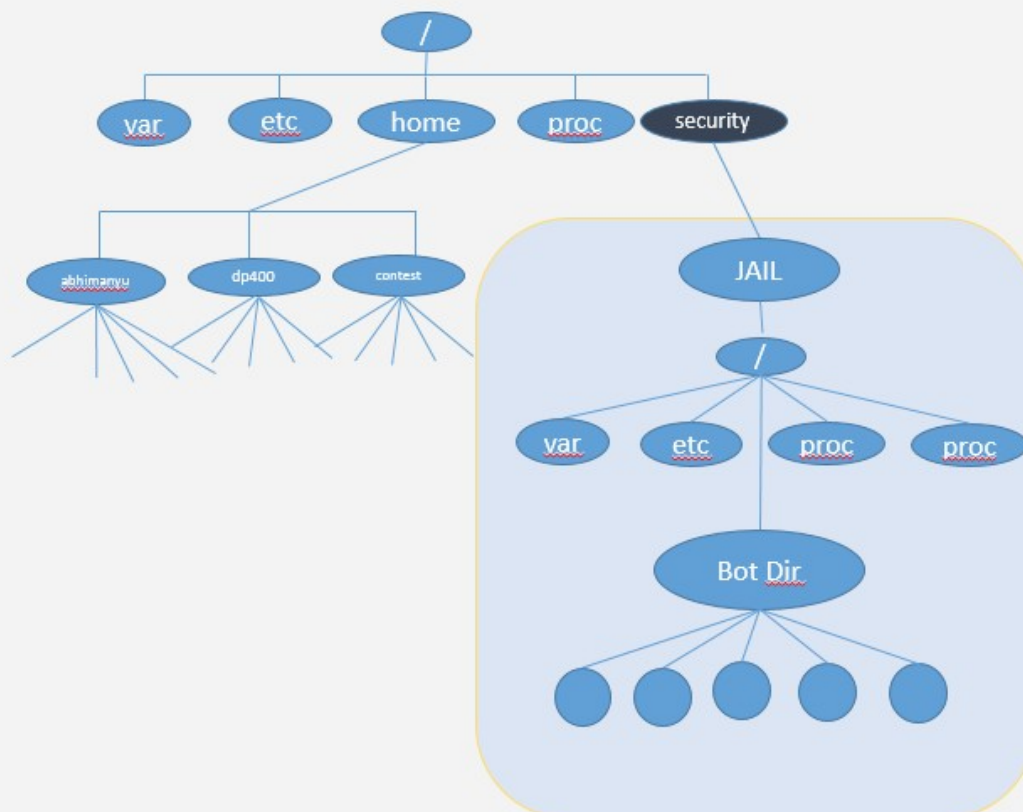
A chroot on Unix operating system is an operation that changes the apparent root directory for the current running process and its children. Chroot command will make our jail directory as root for the current running process. So when any process will run in this environment it can't change the files outside the designated directory tree.

In our case each Bots is compiled and run in the secure environment. Whenever new Bot is submitted then new process is created and newly created process is executed in virtual environment where root directory is jail. So Malicious attack will not make impact on the system.

Real File system on Ubuntu



Directory Structure After creating illusion



5. System Requirements

5.1 Hardware requirements

- Processor – 1 GHz
- Memory (RAM) – 1 GB
- Network Interface Card or Wireless adapter
- Disk to store the code of each bot.

5.2 Software Requirements

- Operating System – Any Linux based OS
- MySQL database [only for manager]
- Python (Version 2.7.3 or greater)
- Apache Server (version 2.2.22) with PHP (version 5.3.10) [only or manager]
- phpMyAdmin [only for manager, optional]

6. References

1. Priority Scheduling

Url – <http://siber.cankaya.edu.tr/OperatingSystems/ceng328/node124.html>, Accessed-15/06/2013

2. Sandbox (computer security) | Wikipedia

Url – [http://en.wikipedia.org/wiki/Sandbox_\(computer_security\)](http://en.wikipedia.org/wiki/Sandbox_(computer_security)), Accessed-17/06/2013

3. Page numbers

Url – https://help.libreoffice.org/Writer/Page_Numbers, Accessed-18/06/2013