Deanship of Graduate Studies
Al-Quds University

Load Balancing for Dynamic Clients, Game Theory Approach

Awni Seif Aldain Jamil Obeid

M.Sc. Thesis

Jerusalem – Palestine

1430 / 2009

Load Balancing for Dynamic Clients, Game Theory Approach

Prepared By:

Awni Seif Aldain Jamil Obeid

B.Sc. : Computer Science - Alquds University - Palestine

Supervisor:

Dr. Rashid Jayousi

A thesis Submitted in Partial Fulfillment of Requirements for the Master Degree of Computer Science from Computer Science Department of Al-Quds University

1430 /2009

Deanship of Graduate Studies
Al-Quds University

Thesis Approval

Load Balancing for Dynamic Clients, Game Theory Approach

Prepared By: Awni Seif Aldain Jamil Obeid

Registration No: 20410831

Supervisor: Dr. Rashid Jayousi

Master thesis submitted and accepted Date: …………………….

The names and signatures of the examining committee members are follows:

1- Head of Committee: Dr. Rashid Jayousi Signature:…………………

2- Internal Examiner: Dr. Amjad Rattrout Signature:…………………

3- External Examiner: Dr. Yousef Abu Zir Signature:…………………..

Jerusalem – Palestine

1430 / 2009

Dedication

I dedicate my thesis to my parents, Seif Aldain and Fayza, to my wife, Fatima, and for my lovely children Aseel, Hadeel, Seif Aldain and Leen for the understanding and encouragement they provided during all these years of study.

Awni Seif Aldain Jamil Obeid

Declaration:

I certify that this thesis submitted for the degree of Master, is the result of my own research, except where otherwise acknowledged, and that this study (or any part of the same) has not been submitted for a higher degree to any other university or institution.

Signed ……………………

Awni Seif Aldain Jamil Obeid

Date: 3/5/2009

Acknowledgements

I would like to express my gratitude to all those who gave me the possibility to complete this thesis, especially Dr. Rashid Jayousi. I want to thank the department of Computer Science in Alquds University. I also thank Prof. Mohammed Dajani who gave us the opportunity to do the thesis in “Game Theory” as he is the coordinator for this subject.

I owe this work to my beloved wife, Fatima, whom without her help and support I would not have been able to finish. I thank her for standing besides me during the most difficult times and for her patience and understanding. I thank my lovely children Aseel, Hadeel, Seif Aldain and Leen for bringing light and liveliness into my life.

Last, but not least, I thank my mother Faiza, and my father Seif Aldain for their continues love, care, encouragement, and sacrifice, and for their strong commitment to give me the best education despite all the hardships.

Abstract

We address the problem of load balancing which is considered as a technique to spread work between two or more computers in order to get optimal response time and resource utilization between servers by using Nash equilibrium which is the central concern of game theory. We use a normal form table to express the payoff for every client, every client has estimate time for execution and every server has speed, from these facts, we innovate a dynamic payoff matrix to evaluate a Nash equilibrium point and then determine which server can serve an appropriate client to achieve best load balancing which is called "*server matching*" *(Each client is matched to exactly one server, but a server can be matched to multiple clients or none.).* We use Netlogo simulation to implement this matching, besides a useful game theory toolset called GAMBIT (Gambit toolset homepage, 2005) to solve the payoff matrix and compute Nash equilibrium point.

This paper was done between the years 2007 – 2009, and to know *what was done in this paper* we can argue that we contribute in accomplishing the load balancing between servers, using new technique depends on game theory perspective, for that reason we can answer the question *why this thesis was done*, because it is very vital in achieving this goal.

We use in our thesis a *simulation methodology* to prove the results that we have obtained from the simulation program which is a Netlogo V4.0.2, and we compare the results with traditionally techniques in load balancing.

Finally, the results show that we improve the performance for the whole system by 4% in achieving load balancing and overweight the possibilities of using this technique in real system around the world.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Over many years, load balancing, is a problem that inspired the researcher. In this thesis, we try to have a new approach using game theory perspective.

This thesis was affected by the continued growth of internet applications, and in the absence of central authority that can control this growth, so users have choice to choose its own server to download their data, that each user wants to minimize its own latency, without concerning the optimal welfare for the whole system, this selfish behavior could lead us to bottleneck in the system.

Game theory is generally considered to have begun with the publication of von Neumann & Morgenstern's, "*The Theory of Games and Economic Behavior*", in 1944, it introduced the idea that conflict could be mathematically analyzed that have a methodology and provided the terminology with which to do it. The development of the "Prisoner's Dilemma" and Nash's papers on the definition and existence of equilibrium laid the foundations for modern noncooperative game theory. At the same time, cooperative game theory reached important on bargaining games and on the core. (Rasmusen, 2005)

We propose to use the game theory perspective in order to achieve load between several servers which is called "*load balancing*" which aims to spread work between two or more computers, in order to get optimal resource utilization, or response time.

In our thesis, we concentrate on load balancing criteria with response time for every client; every client that achieves minimum time on serving his request on any server will require underpinning for our approach in achieving load balancing between servers.

1.1 Motivation for this thesis

Game theory is considered as an interrelated area of researcher in which it involves computer science, also it is interrelated to many useful aspects as political, marketing, auctions, and any other subject that have to do a decision making.

As the user's population accessing internet services grow in size and dispersion, it is necessary to improve performance and scalability by deploying multiple, distributed server sites. Distributing services has the benefit of reducing access latencies, and improving service scalability by distributing the load among several sites. One important issue in such a scenario is how the user chooses the appropriate server.

Similar problem occurs in the context of routing where the user has to select one of a few parallel links. For instance, many enterprise networks are connected to multiple internet service providers (ISPs) for redundant connectivity, and backbones often have multiple parallel trunks. Users are likely to behave "selfishly" in such cases that is each user makes decisions so as to optimize its own performance, without coordination with the other users.

Basically, each user would like to either maximize the resources allocated to it or, alternatively, minimize its cost. Load balancing and other resource allocation problems are prime candidates for such a "selfish" behavior.

A natural framework to analyze this class of problems is that of non-cooperative games, and an appropriate solution concept is that of Nash equilibrium. Users' strategy at Nash equilibrium if no user can access by unilaterally deviating from its own policy. An interesting class of non-cooperative games, which is related to load balancing, is congestion games and its equivalent model of potential games. In a potential game there is a potential function which maps the current state to a real number (in the load balancing scenario a state would include assignment of jobs to machines).

They consider (Kothari, 2005) deviations of a single player (job) and compare the change in the deviating player's utility (load) to the change in the potential function. In an exact potential game the changes are identical. In a weighted potential game the changes are related by a factor that depends only on the player. In an ordinal potential game the changes are in the same direction, while in a generalized potential game an increase in a player's utility implies an increase in the potential function (but not vice versa).

In this thesis, we develop a new approach that depends on the normal form table which is different from the model in (Kothari, 2005) which focuses on calculating the price of anarchy.

Secondly, we build our model depending on MultiAgent system that operates on local knowledge and possessing limited abilities to achieve a desirable result which is similar to model (Kothari, 2005).

Finally, we also use a state-of-the-art game theory tool from University of Minnesota called GAMBIT (Gambit toolset homepage, 2005), it is open source software, which allow us to calculate the Nash equilibrium on any normal from table.

1.2 Game theory as a predictive theory

Game theory can be a good theory of human behavior for two distinct reasons. First, it may be the case that game theory is a good theory of rationality, that agents are rational and that therefore game theory predicts their behavior well. If game theory was correct for this reason, it could reap the additional benefit of great stability. Many social theories are inherently unstable, because agents adjust their behavior in the light of its predictions. If game theory is a good predictive theory because it is a good theory of rationality, this will be because each player expects every other player to follow the theory's prescriptions and have no incentive to deviate from the recommended course of action. Thus, game theory would already take into account that players' knowledge of the theory has a causal effect on the actions it predicts.

Second, and independently of the question of whether game theory is a good theory of rationality, game theory may be a good theory because it offers the relevant tools to systematize and predict interactive behavior successfully. This distinction may make sense when separating our intuitions about how agents behave rationally from a systematic account of our observations of how agents behave.

We begin with some game theory background information in chapter 2, for readers who might not familiar with game theory. Chapter 3 describes our server matching model in achieving load balancing, chapter 4 explains the details implementation, chapter 5 conclusions and future work.

1.3 Objectives

Our goal in this thesis is to find an optimal strategy to achieve load balancing using game theory with dynamic clients, one of the ways proposed by (Kothari, 2005), they introduce this but with static number of clients, and each of clients chooses a server from a permissible set. A server's latency is inversely proportional to its speed, but it grows linearly with the number of clients matched to it. In our model, we use Netlogo simulation to propose a new model with dynamic clients, that our server matching game is a form of the congestion games, that every congestion game has a pure strategy Nash equilibrium. A strategy in our server matching game is the choice of the servers. We focus on the atomic version of the game, where each client chooses exactly one server.

Chapter 2

Background

2.1 History of game theory

One pedestal of game theory is the concept of mixed strategy based on the concept of probability which is therefore essential to reach some interesting results. The beginning of probability calculus is associated with the correspondence of Pierre de Fermat and Blaise Pascal dated in 1654. Hence the true prehistory can start only after this event that happened just 353 years ago and represent the first key milestone in the history of game theory. (Hyksova, 2004)

2.2 Waldegrave with Mixed strategy

James Waldegrave gave the first known mixed strategy solution of a matrix game. It was related to the game, he was looking for a strategy that maximizes the probability of player's win, whichever strategy was chosen by the opponent, that is, exactly in the sense of today minmax principle. He came to the following mixed strategy solution formulated in terms of black and white chips: Tony should choose the strategy" change 7 and lower" with the probability 5/8 and the strategy" hold 7 and higher" with the probability 3/8; Hanna should choose the strategy" change 8 and lower" with the probability 3/8 and the strategy" hold 8 and higher" with the probability 5/8. (Hyksova, 2004)

2.3 Game theory and Mathematics

In the period 1921 – 1928 the concept of mathematics in game theory was existed (Hyksova, 2004), Emile Borel published a series of notes on symmetric two players zero sum games with a finite number n of pure strategy of each player. Borel was the first who attempted to mathematic the game of strategy, he introduced the concept of method of play in the sense of today pure strategy and he was looking for a solution in mixed strategy in the sense of today minmax solution.

The next important milestone is represented by the publication "Theory of games and Economic behavior" in 1944 (Hyksova, 2004), which was the result of a fruitful collaboration of Von Neumann and Oskar Morgenstern. This event is usually considered as the beginning of the existence of game theory as a fully fledged mathematical discipline. Neumann and Morgenstern started with a detailed formulation of economical problem showed the exceptionally broad application possibilities of game theory in economy; then they settled the foundations of an axiomatic utility theory.

In 1954 Lloyd Shapley published a paper which represents one of the earliest explicit applications of game theory to political sciences. They used the Shapley value, one of the solution concepts for cooperative games introduced by Shapley one year earlier to determine the power of the members of the United Nations Security Council.

2.4 Entrance of Game theory into evolutionary Biology

Nowadays game theory is the main tool for the investigation of conflict and cooperation of animals and plants. As for zoological applications, the game theory is used for the analysis, modeling and understanding the fight, cooperation and communication of animals, coexistence of alternative traits, mating systems, conflict between the sexes, offspring sex ratio, distribution of individuals in their habitats, etc. Among botanical applications we can find the questions of seed dispersal, seed germination, root competition, nectar production, flower size, sex allocation, etc. (Hyksova, 2004)

In 1960’s several isolated works using a game-theoretical approach in biology appeared. A historical milestone is represented by the short but ”epoch-making” paper The Logic of Animal Conflict by J. Maynard Smith and G. R. Price. This treatise, published in 1973, stimulated a great deal of successful works and applications of game theory in evolutionary biology; the development of the following decade was summarized in Maynard Smith’s book Evolution and the Theory of Games Not only proved game theory to provide the most satisfying explanation of the theory of evolution and the principles of behavior of animals and plants in mutual interactions, it was just this field which turned out to provide the most promising applications of the theory of games at all. Is this a paradox? How is it possible that the behavior of animals or plants prescribed on the base of game-theoretical models agree with the action observed in the nature? Can a fly or a fig tree, for example, be a rational decision-maker who evaluates all possible outcomes and by the tools of game theory selects his optimal strategy? How is it possible that even the less developed the thinking ability of an organism is, the better game theory tends to work? The explanation is simple: the players of the game are not taken to be the organisms under study, but the genes in which the instinctive behavior of these organisms is coded.

The strategy is then the behavioral phenotype, i.e. the behavior preprogrammed by the genes – the specification of what an individual will do in any situation in which it may find itself; the payoff function is a reproductive fitness, i.e. the measure of the ability of a gene to survive and spread in the genotype of the population in question. The main solution concept of this model is the evolutionary stable strategy which is defined as a strategy such that, if all the members of a population adopt it, no mutant strategy can invade. In certain specific situations, this somewhat vague concept is expressed more precisely. (Hyksova, 2004)

2.5 Normal form and extensive form

In game theory there are two ways to represent a game. The first one is called normal form (Martin J. Osborne, 1994), which is used in simple games and economics, the second one is called the extensive form, which also used widely in economics.

We can define the three parts that every game must have:

1. The players: the players in a game are actual participants who make relevant decisions that jointly determine the outcome of a game.
2. The strategy: every player have a complete description of how a player could play a game, the strategy is comprised of all a player's possible alternative strategies.
3. The Payoffs: A payoff is what the player will get at the end of the game conditions on the actions of all other opponents in the game (the opponents here do not suggest players are necessarily trying to beat each other, rather they are making decisions to maximize their own utility. Since the game may be zero-sum or non-zero-sum, these independent decisions may lead to conflict or coalition). (Romp, 1997)

2.6 Normal form game

A strategic form game (Martin J. Osborne, 1994) shows the payoffs of all combinations of different players' strategies in a matrix. Players are said to be rational when they seek to maximize their payoff. In game theory we are interested in rational players.

Table 2.1 illustrates the strategic form of a classic two-player game called the prisoner's Dilemma. This game is often used to explain basic game theory concept. The motivation story is that two suspects have been arrested by police and are being held in two different cells. Each suspect has a choice of confess (C) or deny (D). They don't know each other's decision when they have to make their own decision. The entries in the matrix are two numbers representing the utility or payoff suspect 1 and suspect 2 respectively. Note that higher numbers are better (more utility). If either of them deny, both of them will be convicted of a minor crime and put in jail for one month (each of them receive a utility of -2, which represented as (-2,-2) in table 2.1. (Rasmusen, 2005)

Normal game of Prisoner Dilemma

Table 2.1: Normal form table for Prisoner Dilemma

		Player 2	
		C	**D**
Player 1	**C**	-8,-8	0,-10
	D	-10,0	-2,-2

If both of them confess, they will both convicted and sentenced to six month in jail (each of them get a utility of -8, which is represented as (-8,-8) in Table 2.1. If one of them confesses, but the other one doesn't, then the one who confesses will be released immediately and will get a utility of 0, while the other will be sentenced to nine months in jail and will get a utility of -10. This corresponds to the other two cells in Table 2.1. For example if player 1 confesses, but player 2 deny, then player 1 will get a utility of 0 and player 2 will get a utility of -10. (Rasmusen, 2005)

In this example, player 1 and player 2 are the two players. They have identical strategy (C, D), which means each has two different choices of actions. The payoffs are the number in the table. The first number in a cell represents player 1's payoff, the second number represents the player 2's payoff. For example if both players confess, the top-left table item tells us both players will get a payoff of -8. (Romp, 1997)

2.7 Extensive form game

Normal form games (Vidal, 2007) do not provide a simple way to analyze the dynamics of strategic interactions, since all players simultaneously make their decisions. Extensive form game (Rasmusen, 2005) provides more information about how the timing actions may affect outcomes. This type of game is represented as a game tree instead of matrix.

The four parts that combine the Extensive form (Martin J. Osborne, 1994):

1. Nodes: This is a position in the game tree where a player must have to make a decision. Each node is labeled with a number so as to identify who is making the decision.

2. Branches: These branches of the game tree represent different alternative choices available to a player.

3. Payoff vector: These represents the payoffs for each player, the payoffs are listed in the order of players at the leaves of the tree.

4. Information set: An information set is a collection of decision nodes for a single player, but which are indistinguishable for the player who is making the decision. Since any two nodes in the same information set are indistinguishable, they must have exactly the same number of branches.

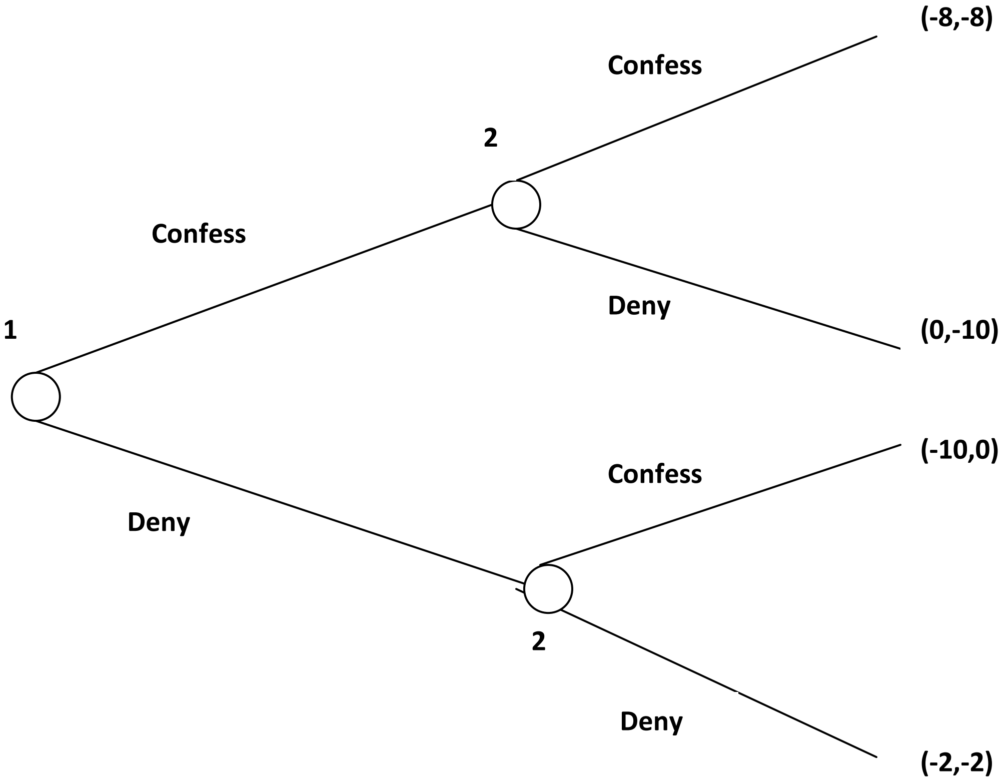


Figure 2.1: Extensive form of Prisoner's Dilemma **(Martin J. Osborne, 1994)**

Each circle represents a node in the game tree (Romp, 1997); the label on the node represents which player is going to make decision. The branches coming out of a node represent the actions available to the player at that point in the game. Player 1 can either confess or deny. At the end of these two branches there is a node representing player 2's decision. Player 2 also has two choices: Confess or deny, these are represented by the branches stemming out of player 2's node. In this case, player 2 has to make his decision without knowing player 1's action. This means when player 2 needs to make his decisions, he only knows he is at one node in this information set, but he is not sure which node he is at. Finally, at the end of each branch, is the payoff vector, with the payoffs of each player listed in order. For example, the top payoff vector is (-8,-8). This means if player 1 and player 2 both decide to confess, the payoff of player 1 is -8, the payoff of player2 is -8.

2.8 Pure strategy

A pure strategy (Martin J. Osborne, 1994) in game theory is a policy that states the player's decision at any decision node in game tree.

2.9 Mixed strategy

A mixed strategy (Rasmusen, 2005) allows the player to select from a set of actions randomly selecting one of choices. The choices are weighted by pre assigned probabilities. It is a fundamental concept in game theory, whereas, in certain situations your best strategy is to behave unpredictably. In fact, a pure strategy is just a special case of mixed strategy with only one action in the decision set. A mixed strategy is beneficial when given your opponent's action.

In mixed strategy of players 1 and 2 are the vectors of probabilities p, q for which the following conditions hold:

$$P=(p_1,p_2,\ldots p_m);\quad p_i \geq 0,\ p_1 + p_2 +\ldots.+p_m =1, \qquad (2.1)$$

$$Q=(q_1,q_2,\ldots q_n);\quad q_i \geq 0,\ q_1 + q_2 +\ldots.+q_n =1. \qquad (2.2)$$

So a mixed strategy is therefore again a certain strategy that can be characterized the following way:

"Use the strategy s_1 S with the probability $p_{1,}$

...,

Use the strategy s_m S with the probability p_m."

Definition 1: the expected payoffs are defined by the relations:

Player 1 $\quad \prod_1(p,q)=\sum \sum p_i q_j a_{ij} \qquad (2.3)$

Player 2: $\quad \prod_2(p,q)=\sum \sum p_i q_j b_{ij} \qquad (2.4)$

A mixed strategy s^* $=(p_1,..p_m)$ is the best reply to t^* if and only if each of pure strategies that occur in s^* with positive probabilities is the best reply to t^*. (Morris, 1994)

Example: Consider the following payoff:

Table 2.2: Normal form table **(Morris, 1994)**

Player 2

	Strategy	$\mathbf{T_1}$	$\mathbf{T_2}$
Player 1	$\mathbf{S_1}$	**(4,-4)**	**(-1,-1)**
	$\mathbf{S_2}$	**(0,1)**	**(1,0)**

Expected values for particular players are the following:

$$\prod_1(p,q) = 4pq-p(1-q) + 0 + (1-p)(1-q)$$

$$= p\ (6q-2) -q + 1$$

$$\prod_2(p,q) = -4pq-p(1-q) + (1-p)q + 0$$

$$= q\ (-4p +1) -p$$

Now we will search best replies of the player of the player 1 to various choices of probabilities q:

If $0 \leq q < \frac{1}{3}$, then for a fixed value of q, $\prod_1(p,q)$ is a linear function with the negative slope, which is therefore decreasing. Maximum of this function occurs for the least possible value of p. i.e. for p=0; in this case it is: $R_1\ (q) =0$.

If $q=\frac{1}{3}$, then $\prod_1(p,\frac{1}{3})=\frac{2}{3}$ is a constant function for which each value is maximal and minimal – player 1 is therefore indifferent between both strategies, $R_1(\frac{1}{3}) =(0,1)$.

If $\frac{1}{3} < q \leq 1$, then for a fixed value of q, $\prod_1(p,q)$ is a liner function with the positive slope, which is therefore increasing. Maximum occurs for the greatest possible value of p, i.e. for p=1; in this case it is $R_1\ (q) =1$.

On the whole

$$R_1\,(q) = \begin{cases} 0 & for \quad 0 \le q < \frac{1}{3} \\ (0,1) & for \quad q = \frac{1}{3} \\ 1 & for \quad \frac{1}{3} < q \le 1 \end{cases} \qquad (2.5)$$

Similarly for player 2:

$$R_2(p) = \begin{cases} 1 & for \quad 0 \le p \le \frac{1}{4} \\ (0,1) & for \quad p = \frac{1}{4} \\ 0 & for \quad \frac{1}{4} \le p \le 1 \end{cases} \qquad (2.6)$$

Equilibrium point is therefore $((\frac{1}{4},\frac{3}{4}),(\frac{1}{3},\frac{2}{3}))$

Provided the players follow these strategies, the expected payoff to the first player will be ⅔ and to second one -1/4. (Morris, 1994)

2.10 Elimination of dominated strategies

In some cases it is possible to eliminate obviously bad, so called dominated strategies:

Definition 2: The strategy s_l S of the player 1 is called dominating another strategy s_i S if, for each strategy t T of the player 2 we have:

$$U_1\,(s_l, t) \ge u_1\,(s_i, t) \qquad (2.7)$$

If there remains the only element in the bimatrix after an iterated elimination of dominated strategies, it is the desired equilibrium point. If there remain more elements, we have at least a simpler bimatrix.

We see the following example:

Table 2.3: Table before elimination of dominated element

Player 2

	Strategy	**T_1**	**T_2**	**T_3**
	S_1	**(1,0)**	**(1,3)**	**(3,0)**
Player 1	**S_2**	**(0,2)**	**(0,1)**	**(3,0)**
	S_3	**(0,2)**	**(2,4)**	**(5,3)**

Example: Consider the following payoff

The strategy s_2 of the first player is dominated by the strategy s_3, because he receives more when he chooses s_3 than when he chooses s_2, whatever strategy is chosen by the second player. Similarly the strategy t_3 of the second player is dominated by the strategy t_2. Since the rational player 1 will not choose the dominated strategy s_2 and the rational player 2 will not choose the dominated strategy t_3, the decision is reduced in this way:

The reduced Table

Table 2.4: The reduce table after elimination of dominated element

Player 2

	Strategy	**T_1**	**T_2**
Player 1	**S_1**	**(1,0)**	**(1,3)**
	S_3	**(0,2)**	**(2,4)**

Strategy t_1 dominated by the strategy t_2, the second player therefore chooses t_2. The first player now decides between the values in the second column of the bimatrix, and since $1 < 2$, he chooses s_3. Hence an equilibrium point of the game is (s_3, t_2) think over the fact that in the original matrix, one sided deviation from the equilibrium strategy does not bring an improvement to the "deviant".

2.11 Games with complete and incomplete information

Perfect information (Haurie, 2000): at each move in the game, the player with the move knows the full history of the play of the game.

Imperfect information (Haurie, 2000): at some move the player with move does not know the history of the game.

2.12 Nash equilibrium

In games, a pair of strategies (a^*, b^*) is defined to be a Nash equilibrium (Rasmusen, 2005) if a^* is player A's best strategy when player B plays b^*, and b^* is player B's best strategy when player A plays a^*. For example, in a two person strategic interaction, a Nash equilibrium combination of strategic is such that each agent's best reply to the other agent's best reply to it. We will say that each strategy in the combination is a Nash equilibrium component strategy. Solving a game is just the process of finding the Nash equilibrium of this game.

2.12.1 Shortcomings of the Nash equilibrium concept (Multiple equlibria)

It is possible that a bimatrix game (Haurie, 2000) can have several equilibria in pure strategies. There may be also additional equilibria in mixed strategies as well. The multiple values of Nash equilibria for bimatrix games are a serious theoretical and practical problem.

The following table shows the two equilibria in pure strategies.

Table 2.5: This bimatrix game has two equilibria

Player 2

		A	B
Player 1	A	2 2	0 0
	B	0 0	2 2

It is difficult to decide how this game should be played if player 1 choose (A,A) and player2 choose (B,B) independently of one another. Each player has two pure strategies A and B. The numbers in the table denote the utility of player 1 and player 2 respectively. There are two pure strategy Nash equilibria at (A,A) and (B,B). There is also one mixed strategy Nash equilibrium where both player randomize with a 1/2 chance of A and a 1/2 chance of B.

What would we expect to happen in this game? Both players prefer either of the two pure strategy Nash equilibria to the mixed strategy Nash equilibrium, since the expected utility to each player at the pure equilibrium is 2, while the expected utility at the mixed equilibrium is only 1. But in the absence of any coordinating device, it is not obvious how the two players can guess which equilibrium to go to. This might suggest that they will play the mixed equilibrium. But at the mixed equilibrium, each player is indifferent, so while equilibrium requires that they give each strategy exactly the same probability, there is no strong reason for them to do so. Moreover, if player 1, say, believes that player 2 is even slightly more likely to play A than B, then player 1 will want to play A with probability one. From an intuitive point of view, the stability of this mixed strategy equilibrium seems questionable.

In contrast, it seems easier for play to remain at one of the pure equilibria, because here each player strictly prefers to play his part of the Nash equilibrium profile as long as he believes there is a high probability that his opponent is playing according to that equilibrium. Intuitively, this type of equilibrium seems more robust.

This coordination game example, although simple, illustrates the two main questions that the theory of learning in games has tried to address, namely: When and why should we expect play to correspond to Nash equilibrium? And, if there are several Nash equilibria, which ones should we expect to occur?

Moreover, these questions are closely linked: Absent an explanation of how the players coordinate their expectations on the same Nash equilibrium, we are faced with the possibility that player 1 expects the equilibrium (A,A) and so plays A, while player 2 expects (B,B) and plays B, with the result the non-equilibrium outcome (A.B). Briefly, the idea of learning-based explanations of equilibrium is that the fact that the players share a common history of observations can provide a way for them to coordinate their expectations on one of the two pure-strategy equilibria. Typical learning models predict that this coordination will eventually occur, with the determination of which of the two equilibria arise left either to (unexplained) initial conditions or to random chance.

However, for the history to serve this coordinating role, the sequence of actions played must eventually become constant or at least readily predictable by the players, and there is no presumption that this is always the case. Perhaps rather than going to Nash equilibrium, the result of learning is that the strategies played, wander around aimlessly, or perhaps play lies in some set of alternative larger than the set of Nash equilibria.

2.12.2 Algorithms for the Computation of Nash Equilibria in Bimatrix Games

Linear programming (Haurie, 2000) is closely associated with the characterization and computation of saddle points in matrix games. For bimatrix games, one has to rely on algorithms solving either quadratic programming or complementarily problems. There are also a few algorithms which permit us to find equilibrium of simple bimatrix games.

2.13 Shapley Value

The Shapley value (Gul, 1989)represents each player's bargaining power in terms of a percentage of the total value created. Bargaining power varies with value contributed. Persons who contribute more receive a higher percentage of the benefits.

Unlike non-cooperative game theory, cooperative game theory does not specify through the game a minute description of the strategic environment, including where to move, and a set of procedures in each step, and the consequences of return to play for rather than what might be detrimental to this set of data to the coalitional form. Cooperation should be a game theorist accurately forecast the reward of all the opportunities available to the alliance, which moved one of the real number: gone are the actions and movements of individual payoffs. Prime advantage of this approach, at least in the multiple player environments, is the practical usefulness. A real-life situation more easily fit in the form of a coalitional game, in which the structure is more than that of non-cooperative game, whether in the form of a normal or large-scale.

Before the advent of the Shapley value, one solution to the concept of cooperative game theory: the von Neumann Morgenstern solution (Gul, 1989). The core would not be defined until around the same time as the Shapley value. As set-valued solutions suggesting "reasonable" allocations of the resources of the grand coalition, both the von Neumann–Morgenstern solution and the core are based on the coalitional form game. However, no single-point solution concept existed as of yet to associate a single payoff vector to a coalitional form game. In fact, in the form of coalitional game these days, even in a little black box of information on the establishment of one point and it seems that the solution can not be defended by. It was in spite of these sharp limitations that Shapley came up with the solution. Obviously, the use of an approach based Shapley not only a great solution for the definition of an attractive and intuitive, but also for the unique characteristics of a set of reasonable axioms.

2.14 Using computer algebra to find Nash equilibrium

A central concern of game theory is the computing of Nash equilibrium. These are characterized by systems of polynomial equations and inequalities.

Game theory has been used to model conflict and cooperation between rational agents. So game theory is a mathematical model of this interaction.

The main computer package for studying game theory today is Gambit which is, developed by McKelvey, McLennan, and Turocy, is currently the standard software package for computing Nash equilibrium. Most of the code focuses on solving two person games because the two persons' situation is already quite rich and interesting. Furthermore, the last version of this package gives precise solutions for more than two person games. (Datta, 2003)

2.14.1 GAMBIT

GAMBIT (Gambit toolset homepage, 2005) is a library of game theory software toolsets for the construction and analysis of finite extensive and normal form games. Its core functionality was from 1994 to 1996. The project was supported by a NSF award to Caltech and the University of Minnesota. This software has been updated several times since. The latest version Gambit 0.2007.01.30 was released on January 30, 2007. Gambit is designed to work on both Microsoft windows (95/98/NT/XP) and UNIX (Linux, Solaris and others) platforms.

Gambit is comprised of 3 parts:

1- A GUI interface that can be used to construct and solve a normal form or extensive form game.

2- A Gambit command line language, this is a script language, somewhat like Lisp. It has a set of built-in functions that can be used to write small programs to construct and analyze games.

3- A library of C++ source code for representing games. This library can be incorporated in other applications to facilitate the analysis of games.

2.15 Game theory and Computer science

The influence of computer science in game theory has perhaps been most strongly felt through complexity theory. They consider a game-theoretic problem that originated in the computer science literature, but should be of interest to the game theory community:

2.15.1 The Price of Anarchy

In a computer system, there are situations where they may have a choice between invoking a centralized solution to a problem or a decentralized solution. By “centralized” here, they mean that each agent in the system is told exactly what to do and must do so; in the decentralized solution, each agent tries to optimize his own selfish interests. Of course, centralization comes at a cost. For one thing, there is a problem of enforcement. For another, centralized solutions tend to be more vulnerable to failure. On the other hand, a centralized solution may be more socially beneficial. How much more beneficial can it be? (Papadimitriou E. K., 1999) They formalized this question by comparing the ratio of the social welfare of the centralized solution to that of the social welfare of the Nash equilibrium with the worst social welfare (assuming that the social welfare function is always positive). They called this ratio the *price of anarchy* (Halpern, 2007), and proved a number of results regarding the price of anarchy for a scheduling problem on parallel machines. Since the original paper, the price of anarchy has been studied in many settings, including traffic routing, facility location games (e.g., where is the best place to put a factory), and spectrum sharing (how should channels in a Wi-Fi network be assigned).

2.15.2 Game Theory and Distributed Computing

Distributed computing and game theory are attentive in much the same problems: this system that many agents have different aims with uncertainty environment. In practice, however, there has been a significant difference in emphasis in the two areas. In distributed computing, the focus has been on problems such as fault tolerance, asynchrony, scalability, and proving correctness of algorithms; in game theory, the focus has been on strategic concerns. (Halpern, 2007)

2.15.3 Implementing Mediators

The question of whether there is a problem in MultiAgent system that can not be resolved with a trusted mediator can be solved only through agents in the system, without the mediator, has attracted a great deal of attention in both computer science (particularly in the cryptography community) and game theory. In cryptography, the focus on the problem has been on *secure multiparty computation*. (Halpern, 2007)

2.15.4 Interactive epistemology:

There has been a great deal of activity in trying to understand the role of knowledge in games, and providing epistemic analyses of solution concepts in computer science, there has been a parallel literature applying epistemic logic to reason about distributed computation. One focus of this work has been on characterizing the level of knowledge needed to solve certain problems. For example, to achieve Byzantine agreement common knowledge among the no faulty agents of an initial value is necessary and sufficient. More generally, in a precise sense, common knowledge is necessary and sufficient for coordination. Another focus has been on defining logics that capture the reasoning of resource-bounded agents. This work has ranged from logics for reasoning about awareness, a topic that has been explored in both computer science and game theory. (Halpern, 2007)

2.15.5 Network growth:

If we view networks as being built by selfish players (who decide whether or not to build links), what will the resulting network look like? How does the growth of the network affect its functionality? For example, how easily will influence spread through the network? How easy is it to route traffic? (Halpern, 2007)

2.15.6 Learning in games:

There has been a great deal of work in both computer science and game theory on learning to play well in different settings for an overview of the work in game theory). One line of research in computer science involves learning to play optimally in a reinforcement learning setting, where an agent interacts with an unknown (but fixed) environment. The agent then faces a fundamental tradeoff between *exploration* and *exploitation*. The question is how long it takes to learn to play well (i.e., to get a reward within some fixed e of optimal; for the current state of the art. A related question is efficiently finding a strategy minimizes *regret*— that is, finding a strategy that is guaranteed to do not much worse than the best strategy would have done in hindsight (that, even knows what the opponent would have done). (Halpern, 2007)

2.16 Related Work

Game theory provides a good starting point for computer scientists in their endeavor to understand selfish rational behavior in complex networks with many agents (clients). Such scenario are readily modeled using techniques from game theory, where players with potentially conflicting goals participate in a common setting with well prescribe interaction.

A.Kothari, S. Suri, And Y. Zhou in paper (Kothari, 2005) has focusing there work on balancing the load across servers, but they assume that there are static clients set. They considered a set *U* on n client and a set *V* of m servers, there is a bipartite graph G between *U* and and a server v_j is permissible for client u_i only if (u_i, v_j) is an edge in G.

This problem for assigning clients (jobs) to servers (machines) back to the early days of computing or distributed scheduling, and there is an enormous literature on it. A small sample of these results includes the following (Westbrook, 1998), and (Shmoys, 1995)investigate the online assignment of unit length jobs which is measured according to maximum number of jobs assigns to any server. (Alon, 1997) And (Lenstra, 1990) consider offline assignment of unit length jobs; (Avidor A., 2001), and (Coffman, 1976) consider the greedy assignment of weighted jobs under this measure, where the client-server graph is complete bipartite.

They consider a set of *U* of *n* clients and a set *V* of *m* servers, then a server matching is a mapping $M\colon U \rightarrow V$ that assigns each client to a server. The number of clients assigned to server *v* in a (many to one) matching *M* is denoted by $d_M(v)$, the load of v in M. When the matching M is clear from the context, we will simply use the shortened notation d(v). Each client matched to *v* in the matching *M* experiences a latency $\lambda_V(d_M(v))$. The cost of a matching M is the total latency of all the clients.

Koutsoupias and Papadimitriou in paper (Papadimitriou E. K., 1999) considered a transportation problem where n independent agents wish to rout their traffic through a network of m parallel edges. They studied the price of anarchy on unsplittable flow games. In the general version of this game we have n agents; agent i wants to transfer an amount w_i of flow between a source s_i and a destination t_i and chooses, as his strategy, a path from s_i to t_i to transfer his flow. Unsplittable means agents are not allowed breaking their flow and transferring it partially across more than one path. As a result, each edge e carries some amount of traffic, say l_e, going through it, and there would be a latency caused by this edge which is a function of l_e and is denoted by $f_e(l_e)$. Assume agent i is using path P_i to transfer his flow. The latency corresponding to him would be the sum of latencies over all edges in P_i, i.e.

$$w_i \sum_{e \;\, P_i} f_e(l_i). \tag{2.8}$$

The aim of each agent is to minimize his associated latency whereas the social objective is to minimize either the total latency or the maximum latency over all edges in the network.

That is, within constant factors, the worst case network is the simplest one (the parallel links network). This implies that, for this family of networks, the network structure does not affect the quality of the outcome of the congestion games played on the network in an essential way.

Panagopoulou and Spirakis (Spirakis, 2005) consider selfish routing in single commodity networks, where selfish users select paths to route their loads (represented by arbitrary integer weights). They consider identical delay functions for the links of the network. That work focuses also on an algorithm suggested in (Fotakis, 2005) ; this is a potential based algorithm for finding pure Nash equilibrium in such works. the analysis of this algorithm (Fotakis, 2005) has given an upper bound on its running time, which is polynomial in n (the number of users) and the sum W of their weights. This bound can be exponential in n when some weights are superpolynomial. Therefore, the algorithm is only known to be pseudo polynomial. The work of Panagopoulou and Spirakis (Spirakis, 2005) provides strong experimental evidence that this algorithm actually converges to a pure Nash equilibria in polynomial time in n (and, therefore, independent of the weights values).

In (Vocking, 2005), Fischer and Vocking reexamined the question of worst case Nash Equilibria for the selfish routing game associated with the KP model (Papadimitriou E. K., 1999), where n weighted jobs are allocated to m identical machines. Recall that Gairing et al (Gairing, 2005), had conjectured that the fully mixed Nash equilibrium is the worst Nash equilibrium for this game (with respect to the expected maximum load over all machines). The known algorithms for approximating the price of anarchy relied on proven case of that conjecture.

In (Vocking, 2005) the authors interesting present a counter example to the conjecture shows that fully mixed Nash equilibria cannot be generally used to approximate the price of anarchy within reasonable factors. In addition, they present an algorithm that constructs the so called concentrated Nash equilibria, which approximate the worst-case Nash equilibrium within constant factors.

Although the work of Fischer and Vocking (Vocking, 2005) has disproved the fully mixed Nash equilibrium conjecture for the case of weighted users and identical links, the possibility that the conjecture holds for the case of identical users and arbitrary links is still open.

Chapter 3

Game theory model for load balancing

3.1 Load Balancing

load balancing (www.wikipedia.org, 2008) is a technique (usually performed by load balancers) to spread work between two or more computers, network links, CPUs, hard drives, or other resources, in order to get optimal resource utilization, throughput, or response time. Using multiple components with load balancing, instead of a single component, may increase reliability through redundancy.

3.2 Game theory as basic rule for load balancing

Game theory is the branch of decision theory with interdependent decisions, and it is considered as the study of MultiAgent decision problem, from these facts, we can use the game theory to play the rule for realization the load balancing between servers. From the axiom of Nash equilibrium that we can get the equilibrium between any two players as the well-known “Prisoners Dilemma” game shown in Table 3.1. This game involves two players, whose names simply A and B. In this basic game, each player must simultaneously choose one of the two possible actions: cooperation or aggression. The payoff for each player depends on both of their actions, as shown in Table 3.1. For each pair of actions, Table 3.1 lists two numbers, the first being A’s payoff and the second being B’s payoff.

Table 3.1: A game with pervasive incentives for aggression (the Prisoners’ Dilemma).

	B cooperative	B aggressive
A cooperative	0,0	-8,1*
A aggressive	*1,-8	*-3,-3*

The asterisks (*) here indicate the best payoff that each player could get in response to each possible action of the other player (Myerson, 2006).

This game has an achievement to equilibrium between these players, so it could have best results in achievement the load balancing between servers through the MultiAgent system.

3.3 MultiAgent system

The goal of MultiAgent systems' research is to find methods that allow us to build complex systems composed of autonomous agents while operating on local knowledge and possessing only limited abilities, are nonetheless capable of enacting the desired global behaviors. We want to know how to take a description of what a system of agents should do and break it down into individual agent behaviors. At its most ambitious, MultiAgent systems aim to reverse-engineering emergent phenomena as typified by ant colonies, the economy, and the immune system. MultiAgent systems approach the problem using the well proven tools from game theory, Economics, and Biology. It supplements these with ideas and algorithms from artificial intelligence research, namely planning, reasoning methods, search methods, and machine learning.

These disparate influences lead to the development of many different approaches, some of which end up incompatible with each other. That is, it is sometimes not clear if two researchers are studying variations of the same problem or completely different problems. Still, the model that has thus far gained most attention, probably due to its flexibility as well as its well established roots in game theory and artificial intelligence, is that of modeling agents as utility maximizers who inhabit some kind of Markov decision process.

3.3.1 Utility

We generally assume that an agent's preferences are captured by a utility function (Vidal, 2007). This function provides a map from the states of the world or outcome of game to a real number. The bigger the number is, the more the agent likes that particular state. Specifically, given that S is the set of states in the world the agent can perceive then agent i's utility function is of the form

$$U: S \rightarrow R \tag{3.1}$$

Notice that the states are defined as those states of the world that the agent can perceive. For example, if a robot has only one sensor that feeds him a binary input, say 1 if it is bright and 0 if its dark, then that robot has a utility function defined over only two states regardless of how complicated the real world might be. In practice, agents have sophisticated inputs and it is impractical to define a different output for each input. Thus, must agents also end up mapping their raw inputs to a smaller set of world states. Creating this mapping function can be challenges as it requires a deep understanding of the problem setting.

We can use utility functions to describe the behavior of almost every agent. Utility functions are also useful for capturing the various tradeoffs that an agent must make, along with the value or expected value of its actions. For example, we can say that a robot receives a certain payment for delivering a package but also incurs a cost in terms of the electricity used as well as the opportunity cost which he could have been delivering other packages. If they translate all these payments and costs into utility numbers then they can easily study the tradeoffs among them.

Once they have defined a utility function for all the agents then all they have to do is to take actions which maximize their utility. As in Economics, they use word selfish to refer to a rational agent that wants to maximize its utility. Notice that this use is slightly different from the everyday usage of the word which often implies a desire to harm others, a true selfish agent simply cares exclusively about its utility. The use of selfish agents does not preclude the implementation of cooperative MultiAgent systems. They can view a cooperative MultiAgent system as one where the agents’ utility functions have been defined in such a way so that the agents seem to cooperate. For example, if an agent receives a higher utility for helping other agents then the resulting behavior will seem cooperative to an observer even though the agent is acting selfishly. Since they are concerned with building agents, they needn’t be distracted by the question of whether or not humans are really selfish or not.

3.3.2 Utility is Not Money

Note that while utility represents an agent's preferences which are not necessarily equated with money. In fact, the utility of money has been found to be roughly logarithmic. For example, say Bill has $100 million while Tim has $0 in the bank. They both contemplate the possibility of winning one million dollars. Clearly, that extra million will make a significant difference in Tim's lifestyle while Bill's lifestyle will remain largely unchanged, thus Tim's utility for the same million dollars is much larger than Bill's. There is experimental evidence that shows this holds for most people. Very roughly, people's utility for smaller amounts of money is linear but for larger amounts it becomes logarithmic. Note that here we are considering the marginal utility or money, that is, it is the utility for the next million dollars. We assume that both Bill and Tim have the same utility for their first million dollars. (Vidal, 2007)

3.3.3 Expected Utility

Once we have utility functions we must then determine how the agents will use them. They assume each agent has some sensors which it can use to determine the state of the world and take action. These actions can lead to new states of the world. For example, an agent senses its location and decides to move forward one foot. Of course, these sensors and effectors might not operate perfectly: the agent might not move exactly one foot or its sensors might be noisy

3.3.4 Markov Decision Processes

So far we have considered only a fixed state of the world. But, the reality in most cases is that agents inhabit an environment whose state changes either because of the agent's action or due to some external event. They can think of the agent sensing the state of the world then taking an action which leads to a new state. They also make the further simplifying assumption that the choice of the new state therefore depends only on the agent's current state and the agent's action. This idea is formally captured by a Markov decision process or MDP.

3.3.5 MultiAgent Markov Decision Processes

The MDP model represents the problems of only one agent, not of a MultiAgent system. There have been several ways of transforming an MDP into a MultiAgent MDP. The easiest way is simply to place all the other agents' effects into the transition function. That is, assume they don't really exist as entities and are merely part of the environment. This technique can work for simple cases where the agents are not changing their behavior since the transition function in an MDP must be fixed. Unfortunately, agents that change their policies over time, either because of their own learning or because of input from the users, are very common.

3.3.6 MultiAgent system and game theory model for load balancing

In our model we have agents , every agent has local knowledge and possessing limited abilities, and because the most commonly used problem representation in game theory is the payoff matrix which shows the utility the agents will receive given their actions, in this game, we assume that the players have common knowledge of the utilities that all players can receive, and we can also assume that the players take their actions simultaneously, so every agent (client) from his payoff can determine a suitable strategy (server) to select, and then to achieve the desired load balancing that we model using Netlogo simulation program.

Chapter 4
The Model

4.1 Our Model

In our model we have dynamic number of servers and dynamic number of clients, we innovate a dynamic payoff matrix to evaluate the Nash equilibrium point, we will use the Netlogo simulation to implement this situation. Because of this dynamic clients that some clients arrive and other departure, for that we can implement this as queue, also we must take this behavior into consideration to apply the suitable distribution.

Every server has different speed, in our model each client needs request from server but we fulfill the game theory fundamentals, so every client chooses a server not depending on the speed only, but every client must not play unilateral.

Our model use the M/D/c queuing system in which the arrival has a Poisson distribution and the time spent on servers is deterministic and the number of servers can be two or four or six.

In my hypothesis we assume the following variables.

s_1[1] : *speed for server 1*

s_2: *speed for server 2*

t_1: *estimate time for client 1 to execute instructions.*

t_2: *estimate time for client 2 to execute instructions.*

We innovated a dynamic normal form matrix (payoff matrix) based on the facts that we can acquire from both clients and servers. After we compute the payoff matrix the GAMBIT toolset was used to attain the Nash equilibrium point.

[1] In our thesis we assume that every execution time that need 1 second will spend this execution over

So the payoff for the two clients will be as following:

Table 4.1: The payoff for two clients by meaning of variables

		Client 2	
		Server 1	Server2
Client 1	Server 1	$\frac{s_1}{t_1}$ $\frac{s_1}{t_2}$	$\frac{s_1}{t_1}\times a$ $\frac{s_2}{t_2}\times a$
	Server 2	$\frac{s_2}{t_1}\times b$ $\frac{s_1}{t_2}\times b$	$\frac{s_2}{t_1}$ $\frac{s_2}{t_2}$

Where

$$a = If\ (t_1 > t_2)\ and\ (s_1 > s_2)\ then\ a = \frac{t_1}{t_2} \quad else \quad a = \frac{t_2}{t_1} \tag{4.1}$$

$$b = If\ (t_1 > t_2)\ and\ (s_1 < s_2)\ then\ b = \frac{t_1}{t_2} \quad else \quad b = \frac{t_2}{t_1} \tag{4.2}$$

For ($s_1 > s_2$) and ($t_1 > t_2$) the payoff matrix will be as following:

Table 4.2: The payoff for two clients by meaning of variables

		Client 2	
		Server 1	Server2
Client 1	Server 1	$\frac{s_1}{t_1}$ $\frac{s_1}{t_2}$	$\frac{s_1}{t_2}$ $\frac{s_2 \times t_1}{t_2^{\ 2}}$
	Server 2	$\frac{s_2 \times t_2}{t_1^{\ 2}}$ $\frac{s_1}{t_1}$	$\frac{s_2}{t_1}$ $\frac{s_2}{t_2}$

For ($s_1 > s_2$) and ($t_1 < t_2$) the payoff matrix will be as following:

Client 2

Table 4.3: The payoff for two clients by meaning of variables for $S_1 > S_2$

		Server 1	Server2
Client 1	Server 1	$\frac{s_1}{t_1}$ $\frac{s_1}{t_2}$	$\frac{s_1 \times t_2}{t_1^{2}}$ $\frac{s_2}{t_1}$
	Server 2	$\frac{s_2 \times t_2}{t_1^{2}}$ $\frac{s_1}{t_1}$	$\frac{s_2}{t_1}$ $\frac{s_2}{t_2}$

For ($s_1 < s_2$) and ($t_1 > t_2$) the payoff matrix will be as following:

Table 4.4: The payoff for two clients by meaning of variables for $S_1 < S_2$

Client 2

		Server 1	Server2
Client 1	Server 1	$\frac{s_1}{t_1}$ $\frac{s_1}{t_2}$	$\frac{s_1 \times t_2}{t_1^{2}}$ $\frac{s_2}{t_1}$
	Server 2	$\frac{s_2}{t_2}$ $\frac{s_1 \times t_1}{t_2^{2}}$	$\frac{s_2}{t_1}$ $\frac{s_2}{t_2}$

For ($s_1 < s_2$) and ($t_1 < t_2$) the payoff matrix will be as following:

Table 4.5: The payoff for two clients by meaning of variables for $S_1 < S_2$

Client 2

		Server 1	Server2

Client 1	Server 1	$\frac{s_1}{t_1}$ $\frac{s_1}{t_2}$	$\frac{s_1 \times t_2}{t_1^2}$ $\frac{s_2}{t_1}$
	Server 2	$\frac{s_2 \times t_2}{t_1^2}$ $\frac{s_1}{t_1}$	$\frac{s_2}{t_1}$ $\frac{s_2}{t_2}$

We will now use several estimated times for the clients and several speeds for the servers and in different order to explore the efficiency for the new method in load balancing:

4.1.1 Objective for the following example

In the following example we have four variables:

t_1 : estimation time for client 1

t_2 : estimation time for client 2

s_1 : speed for server 1

s_2 : speed for server 2

In the following example we have sixteen clients; every two clients have estimation time to serve their requests on several servers that have different speeds. This example will illustrate the difference in time for the two models (Nash model and sequential model).

Example:

Table 4.6: This table compare in time between Sequential and Nash for several clients

No.	t_1	t_2	s_1	s_2	Nash point	t_1 on server	t_2 on server	T on Nash	T on seq
1	30	17	3	2	(18,21)	Server 1	Server 2	10	10
2	30	17	2	3	(18,21)	Server 2	Server 1	10	15
3	30	20	3	1	(10,15)	Server 1	Server 1	10	20
4	15	12	2.5	3	(25,26)	Server 2	Server 2	5	6

5	6	6	2	3	(50,50)	Server 2	Server 2	2	3
6	12	15	3	2.5	(26,25)	Server 2	Server 1	5	6
7	17	30	3	2	(21,18)	Server 2	Server 1	10	15
8	25	30	2	3	(12,10)	Server 2	Server 2	10	12.5

In No. 1 the payoff matrix will be as following:

Table 4.7: The Nash equilibrium point in upper right corner

		Client 2	
		Server 1	Server2
Client 1	Server 1	10 18	**18** **21**[2]
	Server 2	4 10	7 12

In No.2 the payoff matrix will be as following:

Table 4.8: The Nash equilibrium point in lower left corner

		Client 2	
		Server 1	Server2
Client 1	Server 1	7 12	4 10
	Server 2	**18** **21**	10 18

In No.3 the payoff matrix will be as following:

Table 4.9: The Nash equilibrium point in upper left corner

[2] The Nash equilibrium point is in bold font.

		Client 2			
		Server 1		Server2	
Client 1	Server 1	**10**	**15**	15	8
	Server 2	2	10	3	5

In No.4 the payoff matrix will be as following:

Table 4.10: The Nash equilibrium point in lower left corner

		Client 2			
		Server 1		Server2	
Client 1	Server 1	17	21	13	20
	Server 2	**25**	**26**	20	25

In No.5 the payoff matrix will be as following:

Table 4.11: The Nash equilibrium point in lower right corner

		Client 2			
		Server 1		Server2	
Client 1	Server 1	33	33	33	50
	Server 2	50	33	**50**	**50**

In No.6 the payoff matrix will be as following:

Table 4.12: The Nash equilibrium point in lower left corner

		Client 2			
		Server 1		Server2	
Client 1	Server 1	25	20	31	21

Server 2	**26** **25**	21 17

In No.7 the payoff matrix will be as following:

Table 4.13: The Nash equilibrium point in lower left corner

		Client 2			
		Server 1		Server2	
Client 1	Server 1	18	10	31	12
	Server 2	**21**	**18**	12	7

In No.8 the payoff matrix will be as following:

Table 4.14: The Nash equilibrium point in lower left corner

		Client 2			
		Server 1		Server2	
Client 1	Server 1	8	7	10	12
	Server 2	14	8	**12**	**10**

Finally, we implement this situation by using Netlogo V.4.0.2, so we assume that we have two servers and dynamic number of clients that need requests form servers, and to explore the difference between the Nash solution and sequential solution to obtain the optimal load balancing, we use two models, the first for the Nash equilibrium and the second for the sequential one.

4.1.1.1 Discussion for the previous example

From the previous example, we can draw a conclusion that from the eight runs we have one run only gives the same execution time, three runs the Nash model have a surpass by 10%, one run the Nash model surpasses by about 25%, two runs the Nash model surpass by about 50%, and finally one run the Nash model surpasses by about 100%.

Chapter 5

Implementing the Model

5.1 Implementing the model

The main purpose of our model is to serve the clients that arrive in a minimum time, we assume that every client arrives and holds his estimation time, in our model we deal with every two clients to determine which client to choose suitable server in order to achieve load balancing for the whole clients.

So our goal is to construct a model to tell client to match the best server in order to achieve load balancing using Nash equilibrium which is the central concern of game theory, to enhance the system overall and to optimize the time needed to complete his request.

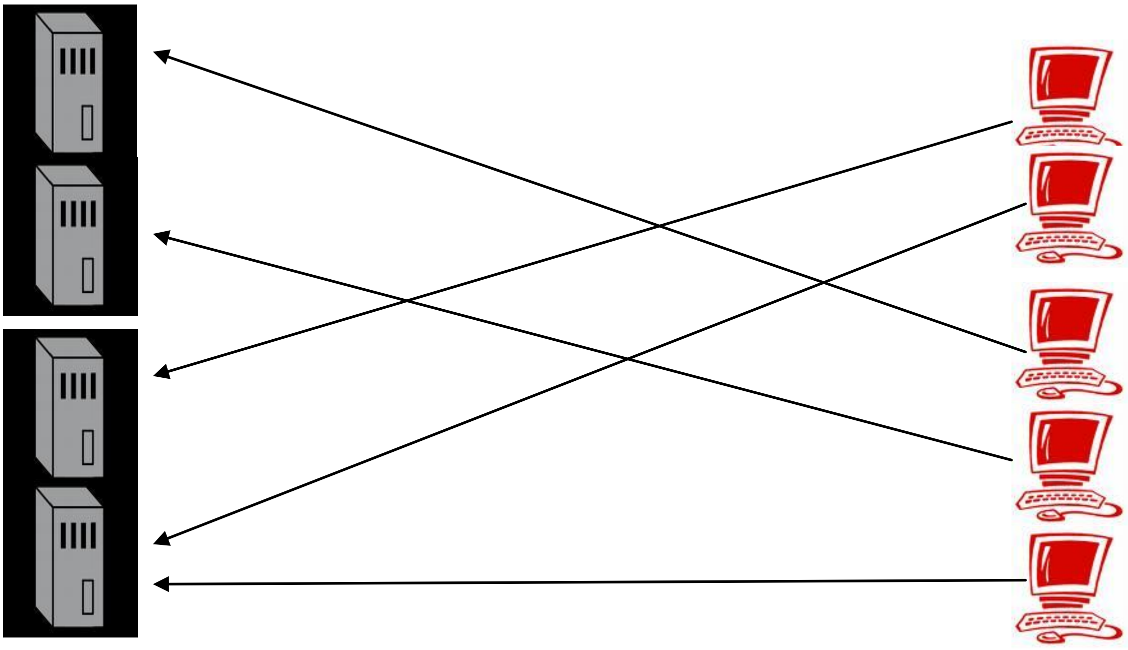


Figure 5.1: The environment for the model of load balancing, at left side the servers and at the right side the clients

The environment setup in our model which we have from two to six servers, each one has a different speed, and we have a dynamic number of clients from two to thirty, each one has an estimation time, the experiments here reported have been programmed in Netlogo 4.0.2 (Figure 5.1).

The figure A.1 in the appendix show the interface for our simulation model

The typical simulation would have a number of clients and servers, where each client has a request, we express this request by a red line from client to specify server. Then it would run until the whole clients finish their requests.

The results we present in this thesis are mostly taken from typical runs, so the monitor labeled with "*Total Time*" denoted to the total time consumed by the all clients. Although we run 20 experiments and averaged the result, we obtain a desired outcome for the all experiments.

In figure A.1 we have the following buttons:

1- Switch with label *Run-Nash* if it is *on* it will run the experiment in Nash; otherwise it will run the experiment in sequential balancing.

2- We have two sliders for the number of servers and clients.

3- We have one monitor to show for us the total time spent on servers.

We do twenty random experiments;

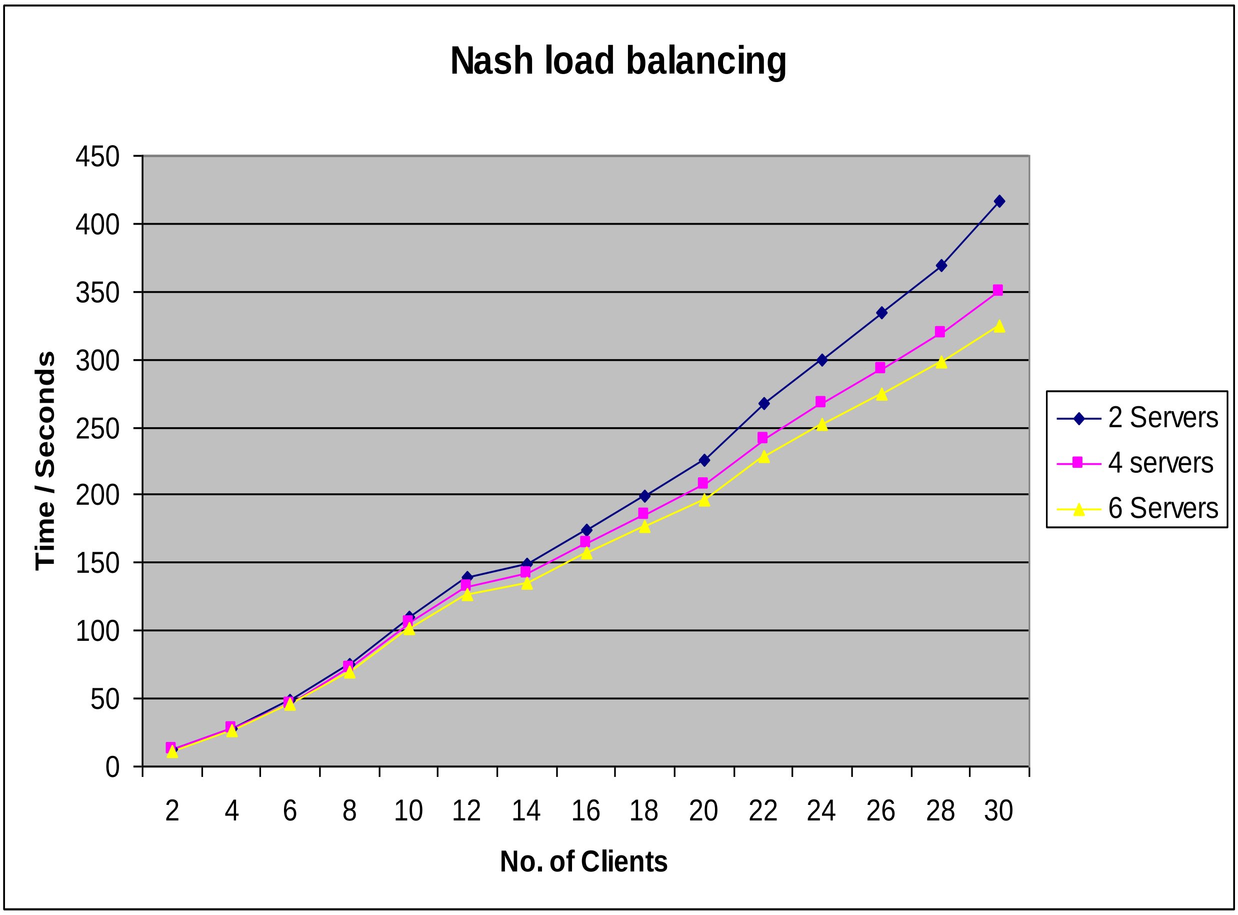


Figure 5.2: This graph represents the number of clients versus time under different number of servers.

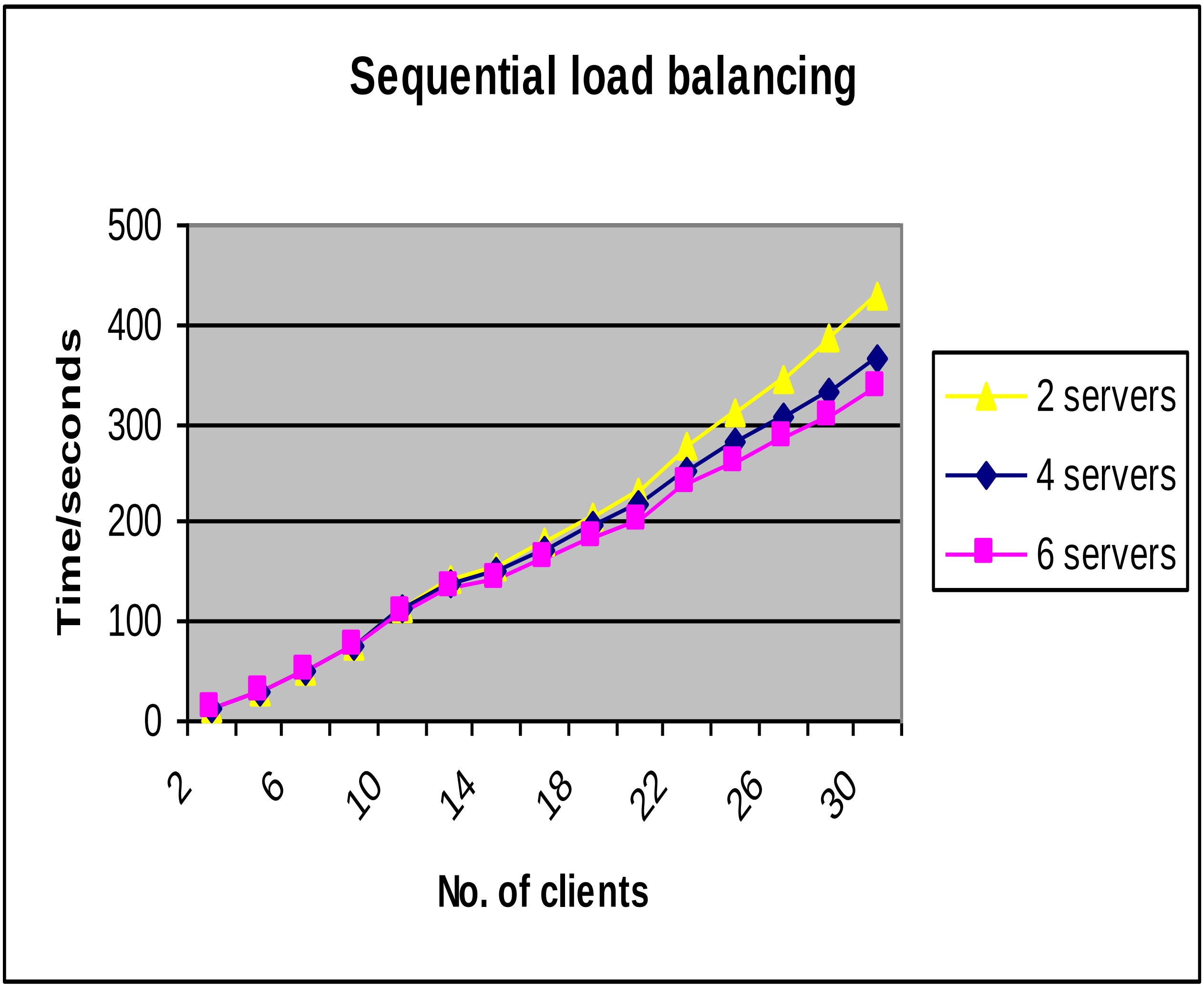


Figure 5.3: Sequential load balancing.

From Figure 5.2 and Figure 5.3 we note the following:

When the number of clients less than 10 clients, the difference in time less than 5 seconds, and when the number of clients is more than 15 clients the difference will be more than 10 seconds between Nash load balancing and sequential load balancing.

Second we note that when we immigrate from two servers to four servers the difference will be more than 60 seconds, when we immigrate to 6 servers the difference will be less than 30 seconds.

Finally, our plot was linear multiplying with a constant number, so we can predict what will be the time at any circumstances.

Objective for Experiment1

In this experiment we have six clients and two servers have different speed, we assume that every two clients arrive at the same time, we illustrate in this experiment how the clients will behave and to decide which strategy (server) to select through strategic form table (normal form table) depending on Nash equilibrium point.

Experiment 1

The following table shows the six clients, and two servers (speed 3 GHz and the second one 2.5 GHz respectively), the six clients each of one has the following estimation time respectively,

Table 5.1: We have 6 clients and six different estimation times in sequential order.

Client Number	**Estimation Time in Seconds**
1	15
2	18
3	20
4	22
5	25
6	30

In our model using Netlogo we have two methods to make a comparison between them, the first one sequential load balancing that every client is served by the first server and the second client is served by the second one respectively and so on for the remaining clients. The second method was the Nash load balancing which is the core of our thesis that depends on normal form table.

If the *switch* Run-Nash is *off* and we push the *Setup* button then we push the *Start* button, after finished we can see the result on the *Monitor Total time* 49.91 seconds Figure A.2.

If we run the same experiment but we turn the *Switch* Run-Nash *on* we can observe the *Monitor Total Time* will be 47.72 seconds, Figure A.4.

So in our Nash model we can save 2.19 seconds.

To trace how clients choose a suitable server as following:

Client 1 has estimation time 15 seconds and the second *client 2* has 18 seconds, if we use the normal form table we have the following values:

Table 5.2: The Nash equilibrium point in lower left corner

20 17	24 17
20 **20**	17 14

The second column and the first row have Nash equilibrium which is the *client 1* chooses *server 2* and *client 2* chooses *server 1*.

Next we have *client 3* 20 seconds and *client 4* 22 seconds which give us the following values:

Table 5.3: The Nash equilibrium point in upper right corner

15 **13**	16 12
13 15	12 11

The Nash exists in column one and row one that's mean that *client 1* chooses *server 1* and *client 2* chooses *server 1* also.

Finally we have *client 5* 25 seconds and *client 6* 30 seconds which give us the following values:

Table 5.4: The Nash equilibrium point in lower left corner

11	9	13	10
12	**11**	10	8

The Nash exists in column one and row two that's mean that *client 1* chooses *server 2* and *client 2* chooses *server 1*.

We must take in consideration that the servers have some degradation while the clients serve by these servers.

Discussion for Experiment 1:

From these inputs we can conclude a normal form table, and from this table we can orient each client to serve his request by a suitable server, finally, when we use the Nash model the time used by clients was 47.72 seconds, where the time used by clients in sequential model was 49.91, the difference was 2.19 seconds.

Objective for Experiment 2

In the following experiment we use six clients, and two servers, in this experiment the estimation time for the clients could be the same, we assume every two clients arrive at the same time.

Experiment 2:

In the second experiment we have six clients and two servers the first server speed 3 GHz and the second one 2.3 GHz, the six clients each of one has the following estimation time respectively,

Table 5.5: We have 6 clients and six estimation time, some estimation time was equal

Client Number	**Estimation Time in Seconds**
1	15
2	12
3	10
4	12
5	15
6	12

If the *switch* Run-Nash is *off* and we push the *Setup* button then we push the *Start* button after finished we can see the result on the *Monitor Total time* 16.39 seconds Figure A.3 .

If we run the same experiment but we turn the *Switch* Run-Nash *on* we can observe the *Monitor Total Time* will be 14.9 seconds Figure A.6.

So in our Nash model we can save 1.49 seconds.

To trace how clients choose a suitable server as following:

Client 1 has estimation time 15 seconds and the second *client 2* has 12 seconds, if we use the normal form table we have the following values:

Table 5.6: The Nash equilibrium point in upper left corner

20	**25**	25	24
12	20	15	19

The first column and the first row have Nash equilibrium which is the *client 1* choose *server 1* and *client 2* choose *server 1* also, in this case we note that the two clients choose the same server to get best utilization and less time for the two clients.

Next we have *client 3* 10 seconds and *client 4* 12 seconds which give us the following values:

Table 5.7: The Nash equilibrium point in lower left corner

28	23	34	23
28	**28**	23	19

The Nash exists in column one and row two that's mean that *client 1* choose *server 2* and *client 2* choose *server 1*.

Finally we have *client 5* 15 seconds and *client 6* 12 seconds which give us the following values:

Table 5.8: The Nash equilibrium point in upper right corner

18	23	**23**	**23**
12	18	15	18

The Nash exists in column two and row one that's mean that *client 1* choose *server 1* and *client 2* choose *server 2*.

Discussion for Experiment 2:

In this experiment we use also six clients but some clients have the same estimation time, it doesn’t exceed 16 seconds, and two servers have different speeds, from the results we obtain we can observe that the Nash model has less time for serving the clients with opposite to the sequential model, Nash model 14.9 seconds and the sequential model 16.39 seconds, the difference was 1.49 seconds in this experiment.

Objective for Experiment 3

In the last experiment we have eight clients and two servers, the first server has less speed than the other one, and we illustrate in this experiment the performance of the Nash model over the sequential model.

Table 5.9: Eight clients every one has an estimation time

Client Number	**Estimation Time in Seconds**
1	45
2	30
3	18
4	14
5	13
6	15
7	8
8	8

Experiment 3:

In the third experiment we have eight clients and two servers the first server speed 2.7 GHz and the second one 3 GHz respectively, the eight clients each of one has the following estimation time respectively,

If the *switch* Run-Nash is *off* and we push the *Setup* button then we push the *Start* button after finished we can see the result on the *Monitor Total time* 32.12 seconds Figure A.5.

If we run the same experiment but we turn the *Switch* Run-Nash *on* we can observe the *Monitor Total Time* will be 29.37 seconds Figure A.7

So in our Nash model we can save 2.75 seconds.

To trace how clients choose a suitable server as following:

Client 1 has estimation time 45 seconds and the second *client 2* has 30 seconds, if we use the normal form table we have the following values:

Table 5.10: The Nash equilibrium point in lower left corner

6	9	4	7
10	**14**	7	10

The first column and the second row have Nash equilibrium which is the *client 1* choose *server 2* and *client 2* choose *server 1*

Next we have *client 3* 18 seconds and *client 4* 14 seconds which give us the following values:

Table 5.11: The Nash equilibrium point in lower left corner

14	19	11	16
21	**24**	16	21

The Nash exists in column one and row two that's mean that *client 1* choose *server 2* and *client 2* choose *server 1*.

Next we have *client 5* 13 seconds and *client 6* 15 seconds which give us the following values:

Table 5.12: The Nash equilibrium point in upper right corner

19	17	**22**	**22**
25	19	22	19

The Nash exists in column two and row one that's mean that *client 1* choose *server 1* and *client 2* choose *server 2*.

Finally we have *client 7* 8 seconds and *client 8* 8 seconds which give us the following values:

Table 5.13: The Nash equilibrium point in lower right corner

30	30	30	34
34	30	**34**	**34**

The Nash exists in column two and row two that's mean that *client 1* choose *server 2* and *client 2* choose *server 2 also*.

Objectives for Experiment 3:

In the last experiment we use eight clients two of them have the same estimation time, and we have two servers, the first one has less speed contrary to the first two experiments, from the results we obtain we can observe that our Nash model also has less time for serving the clients with opposite to the sequential model, the time we save by Nash model was 2.75 seconds for the whole experiment.

Finally, we can summarize these results in the following table, that show the time was saved (in seconds) in the last row by the second method (Nash Equilibrium):

Table 5.14: The summary for the three experiments

Method	**Experiment 1**	**Experiment 2**	**Experiment 3**
Sequential Balancing	21.82	16.39	32.12
Nash equilibrium	18.83	14.9	29.37
Time saved /in seconds	***2.99***	***1.49***	***2.75***

5.2 Discuss the results

From the result we obtain we notice that the total time for 2 servers 30 clients 416.34 seconds and for 4 servers 30 clients 350.25 seconds, the difference will be *66.09* seconds, and the total time for 6 servers 30 clients 324.58 seconds, the difference will be *25.67* seconds, so we can see the difference when we immigrate from 2 to 4 servers is greater than when we immigrate from 4 to 6 servers, that Nash load balancing cannot work.

Chapter 6

Conclusion

6.1 Conclusion and Future Work

From game theory fundamentals (chapter 2), we computed the Nash equilibrium which we discussed in (chapter 2.12.2) to have a rational choice (chapter 1.2) for every client, that clients are rational and therefore game theory predicts their behavior well. If game theory is correct for this reason, it could reap the additional benefit of great stability and achieve load balancing between servers. This equilibrium that makes the game to have rational strategy in any decision, it also accomplishes the balancing of load (chapter 3, 3.1) between servers through a new technique based on dynamic strategic normal form (chapter 4, the model) that guide the players to choose suitable strategies.

In this thesis we develop a dynamic normal form table (*As we introduce in Chapter 4*) to explore the best server matching for the clients. Due to the complexity of solving the Nash equilibrium, we use the GAMBIT toolset to derive the results. We ran some simulations to test the performance of our system (*Chapter 5*). The experimental results show that the new technique for server matching gives us preferable results from the previous one "sequential load balancing"(*Chapter 4*). In our simulations we use the Netlogo program to model our system. The techniques we explore are quite flexible, and relaxations of the assumptions in the model lead to interesting future research topics.

We have demonstrated that our mathematical model (*Chapter 4, Our model section*) and our simulation agrees with each other by showing that the Nash equilibrium that exists in a normal form table gives us best results and this is achieved by the simulation(*Chapter 5, Implementing the model section*).

Our main results contribute an important idea to make the normal form table more dynamic to give us a best server matching for every two clients.

We compare between two techniques (*Chapter 5*), the first one sequential load balancing and between the new models Nash load balancing, we show the improvement of the performance for the new model. For 2 servers the performance is 3.19% according to table A.4, for 4 servers the performance is 4.59%, and for 6 servers the performance is 3.98%. Finally, our model is linear as shown in figure.5.2.

6.2 Future Work

As future work, our current model does not consider a non atomic[3] version of data, so it could be possible to work in this direction. Other future work may include the pure Nash equilibrium for the dynamic situation that was implemented through Netlogo simulation. Finally, we can extend this model as a MultiAgent system to have dynamic number of clients and servers and not to be restricted by the time requested by the clients.

[3] Non atomic version that client can choose more than one server to access one request.

References

Alon, N. ,. (1997). "Approximation schemes for scheduling". *Proc. 8th ACM–SIAM Sympos. on Discrete Algorithms, ACM Press, New York* .

Avidor A., A. A. (2001). "Ancient and new algorithms for load balancing in the Lp norm". *Algorithmica 29(3)* .

Coffman, R. A. (1976). "Record allocation for minimizing expected retrieval costs on drum-like storage devices". *J. ACM 23(1)* .

Datta, R. S. (2003). "Using computer algebra to find Nash equilibria". *University of California* .

Fotakis, D. ,. (2005). "Selfish Unplittable flows". *Theoretical computer Science* .

Gairing, M. ,. (2005). "Extreme Nash equilibria". *In proceeding of the 8th Italian conference of theoretical computer science* .

Gambit toolset homepage. (2005, 9 1). Retrieved from Gambit toolset homepage: http://www.hss.caltech.edu/gambit

Gul, F. (1989). "Bargaining foundations of shapley value". *Econometrica* , 81-95.

Halpern, J. (2007). *Computer science and game theory: a brief survey*. University of Cornell.

Haurie, A. ,. (2000). *"An introduction to dynamic games"*. Italy.

Hyksova, M. (2004). Several Milestones in the history of game theory. *Oxford University* , 49-56.

Kothari, A. ,. (2005). "Congestion games, load balancing, and price of anarchy". *University of California* .

Lenstra, J. K. (1990). "Approximation algorithms for scheduling unrelated parallel machines". *Math. Programm. 46(3)* .

Martin J. Osborne, A. R. (1994). *"A course in game theory"*. London England: MIT press.

Mavronicolas, M. (n.d.). *Marios Mavronicolas Homepage*. Retrieved from Marios Mavronicolas Homepage: http://www.cs.ucy.ac.cy/~mavronic/

Morris, P. (1994). *"Introduction to game theory"*. New York: Springer Verlag.

Myerson, R. B. (2006). "Force and restraint in strategic deterrence: a game theorist's perspective". *Chicago Humanities Festival* .

Papadimitriou, C. (2001). "Algorithms, games, and the Internet". *In proc. 33rd Sympos. On Theory of computing* .

Papadimitriou, E. K. (1999). "Worst case equilibria". *University of California* .

Rasmusen, E. (2005). *"Games and Information, Fourth Edition, an Introduction to Game Theory".* USA: Free Press.

Romp, G. (1997). *"Game theory : introduction and applications".* USA: Oxford University.

Rosernthal, R. W. (1973). "A class of games possessing pure-strategy Nash equilibria". *International Journal of Game Theory* .

Shmoys, D. B. (1995). "Scheduling parallel machines on-line". *SIAM J. Comput.*

Spirakis, P. P. (2005). "Efficient convergence to pure Nash equilibria in weighted network congestion games". *In proceeding of the 4th International workshop on efficient and Experimental algorithms* .

Vidal, J. m. (2007). *"Fundamentals of Multiagent systems with Netlogo examples".*

Vocking, S. F. (2005). "On the structure and complexity of worst case Equiliria". *In proceedings of the 1st workshop on Internet and network economics, volume 3828 of lecture notes in computer science.*

Westbrook, S. P. (1998). "On-line load balancing and network flow". *Algorithmica 21(3)* .

Appendixes

Twenty random experiments; we have the following results (for 2 servers):

Table A. 1: Twenty random experiments done for two servers.

	2 Clients		**4 Clients**		**6 Clients**		**8 Clients**		**10 Clients**		**12 Clients**	
	Nash	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**
1	12.67	12.67	27.8	28.73	47.39	48.5	73.92	76.25	108.1	109.96	136.27	138.52
2	12.67	12.67	27.8	28.74	47.39	48.59	73.98	76.32	108.22	111.3	136.78	139.29
3	12.67	12.67	27.8	28.79	47.64	48.89	74.05	76.35	108.51	111.69	137.77	139.81
4	12.67	12.67	27.8	28.79	47.65	48.89	74.48	76.4	109.76	112.3	137.88	139.85
5	12.67	12.67	27.8	28.87	47.7	48.96	74.5	76.41	109.78	112.32	137.9	139.9
6	12.67	12.67	28.68	29.01	47.86	48.97	75.01	76.42	109.95	112.84	137.96	140.71
7	12.67	12.67	28.68	29.01	47.9	49.01	75.09	76.5	110.19	112.87	138.01	140.78
8	12.67	12.67	28.68	29.04	47.99	49.1	75.13	76.52	110.35	112.98	138.04	140.99
9	12.67	12.67	28.79	29.12	48.19	49.17	75.18	77.14	110.42	112.98	138.09	141.02
10	12.67	12.67	28.79	29.12	48.54	49.33	75.25	77.19	110.44	113.05	138.85	141.03
11	12.67	13	28.79	29.15	48.65	49.4	75.25	77.35	110.53	113.08	139.04	141.28
12	12.67	13	28.79	29.31	48.71	49.62	75.32	77.72	110.62	113.39	139.18	141.52
13	12.67	13	28.79	29.4	48.71	49.8	75.4	77.97	110.71	113.5	139.24	141.55
14	12.67	13	28.79	29.46	48.72	49.88	75.45	78.07	111.06	113.89	139.44	142.3
15	12.67	13	28.79	29.54	48.72	49.94	75.6	78.11	111.12	113.96	139.58	142.72
16	12.67	13	28.79	29.59	48.77	50.08	75.91	78.21	111.64	114.02	139.7	142.8
17	12.67	13	28.95	29.59	48.83	50.19	76.1	78.26	111.7	114.35	139.77	143.58
18	12.67	13	28.95	29.62	48.86	50.23	76.27	78.63	111.88	115.49	140.51	144.08
19	12.67	13	28.95	29.62	49	50.26	76.3	78.66	112.55	115.62	140.57	144.27
20	12.67	13	28.95	29.62	49.17	50.31	76.71	78.76	113.1	115.85	141.4	145.79
Av	**12.67**	**12.84**	**28.56**	**29.21**	**48.32**	**49.46**	**75.25**	**77.36**	**110.53**	**113.27**	**138.80**	**141.59**

Table A.1

	14 Clients		**16 Clients**		**18 Clients**		**20 Clients**		**22 Clients**		**24 Clients**	
	Nash	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**
1	145.26	153.67	170.39	174.84	194.83	201.86	219.5	225.38	261.9	266.22	291.72	299.71
2	146.96	153.08	171.05	175.08	195.43	201.91	220.57	226.2	263.08	268.55	292.73	303.04
3	147.39	154.37	172.17	176.94	196.1	202.82	221.08	226.78	263.48	269.37	293.11	305.67
4	147.72	153.13	172.48	176.97	197.67	203.02	222.48	227.13	264.08	269.54	294.71	305.85
5	147.94	149.39	172.69	177.52	197.78	203.7	222.9	228.13	265.45	270.21	295.52	305.89
6	148.12	151.49	173.07	178.05	197.9	203.92	222.94	228.59	265.47	271.64	296	306.63
7	148.44	153.57	173.22	178.17	197.95	204.39	224.13	229.38	265.67	272.53	296.98	307.48
8	148.59	154.85	173.49	179.66	198.32	205.07	224.13	231.37	266.04	275.33	297.45	307.64
9	148.73	152.08	174.17	180.15	198.49	205.56	224.29	231.97	266.39	276.47	298.15	308.35
10	149.18	158.73	174.35	180.59	198.67	205.63	224.87	232.15	267.34	276.99	298.5	308.39
11	149.7	155.04	174.81	182.18	199.48	205.77	225.04	232.76	267.34	277.7	298.72	309.31
12	149.93	153.99	174.89	182.64	200.7	206.04	227.25	232.96	267.4	278.02	299.53	310.13
13	149.95	151.76	175.04	183.31	200.77	206.55	227.3	233.06	268.29	280.23	299.81	310.66
14	149.95	158.23	175.4	183.83	200.92	207.17	228.18	234.42	268.97	281.35	301.53	311.03
15	150.38	152.2	175.54	184.02	201.3	208.33	229.04	235.13	270.3	282.09	302.49	311.44
16	150.67	152.86	175.89	184.09	201.51	208.84	229.05	235.91	270.31	283.44	302.93	314.21
17	150.78	150.11	175.9	185.16	202.93	212.03	229.57	236.32	271.77	285.15	303.33	314.4
18	151.04	156.98	177.39	186.76	203.04	212.46	229.86	237.19	272.15	286.37	304.39	315.64
19	151.45	148.97	178.95	188.11	203.8	213.54	230.02	238.75	272.25	286.91	304.77	320.76
20	151.9	156.2	180.3	190.78	203.91	214.16	232.21	240.82	272.63	287.91	304.98	321.65
Av	**149.2**	**153.54**	**174.56**	**181.44**	**199.58**	**206.64**	**225.72**	**232.22**	**267.52**	**277.30**	**298.87**	**309.89**

Table A.1

	26 Clients		**28 Clients**		**30 Clients**	
	Nash	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**
1	324.78	335.07	354.81	372.7	404.53	434.21
2	327.92	336.88	358.21	372.9	406.16	438.81
3	328.1	338.71	360.33	374.46	407.03	434.53
4	330.85	338.83	364.6	374.86	407.99	425.22
5	331.32	340.83	364.67	380.82	411.34	429.36
6	332.7	341.79	367.46	381.5	412.02	415.14
7	333.02	341.85	368.09	381.86	414.1	428.61
8	334.17	342.38	368.39	383.85	415.6	420.74
9	334.17	343.27	368.53	384.48	415.72	444.81
10	335.12	343.5	369.97	384.9	415.73	426.42
11	335.18	344.8	370.11	385.64	415.85	430.44
12	335.3	345.87	371.73	388.22	417.73	411.23
13	335.9	347.49	371.9	388.26	418.01	410.27
14	336.41	348.68	372.48	389.3	420.7	447.55
15	339.38	349.29	372.58	389.59	420.97	446.21
16	340.25	349.52	375.76	389.87	421.12	442.66
17	340.9	350.61	377.1	394.74	422.23	416.65
18	341.25	350.86	379.19	395.16	422.88	426.68
19	341.99	355.42	379.9	395.77	423.35	419.84
20	342.22	356.14	380.75	397.75	433.76	433.96
Av	**335.1**	**345.09**	**369.83**	**385.33**	**416.34**	**429.17**

And we do twenty random experiments; we have the following results (for 4 servers):

Table A. 2: Twenty random experiments done for four servers.

	2 Clients		**4 Clients**		**6 Clients**		**8 Clients**		**10 Clients**		**12 Clients**	
	Nash	**Seq.**	**Nash**	**Seq.**	**Nash**	**Seq.**	**Nash**	**Seq.**	**Nash**	**Seq.**	**Nash**	**Seq.**
1	12.02	13.66	27.24	29.52	46.31	49.58	72.9	77.21	106	113.09	131.56	137.81
2	12.02	13.19	27.24	29.28	46.5	50.03	72	76.02	105.66	113.77	131.06	141.29
3	12.02	13.19	27.27	30.3	46.64	52.7	72.83	78.2	106.34	111.28	132.11	139.21
4	12.02	13.14	27.25	30.12	46.47	49.85	73.71	77.24	105.47	116.4	132.08	136.23
5	12.02	14.7	27.24	29.66	46.9	48.17	71.97	76.95	105.44	114.94	131.5	140.08
6	12.02	14.05	27.39	29.62	46.29	50.68	73.07	77.14	104.67	112.81	131.46	138.5
7	12.02	14.05	27.25	29.46	46.35	49.53	73.03	78.94	106.6	111.72	133.26	136.95
8	12.02	14.05	27.21	30.01	46.42	50	72.66	77.08	106.96	113.23	131.69	139.73
9	12.02	13.66	27.27	30.21	46.52	49.41	72.55	76.77	105.93	107.03	133.19	137.73
10	12.02	14.05	27.31	29.75	46.75	49.53	73.58	76.81	105.53	112.63	131.87	140.52
11	12.02	12.14	27.25	29.66	46.97	48.19	72.43	75.72	104.89	112.14	130.92	139.91
12	12.02	14.7	27.27	29.28	46.4	51.15	73.57	77.98	104.85	114.56	132.07	138.84
13	12.02	12.14	27.24	29.42	46.32	51.37	72.76	76.54	104.46	114.35	131.34	136.69
14	12.02	13.66	27.24	29.75	46.43	49.45	73.05	75.26	104.08	111.15	131.84	141.33
15	12.02	13.19	27.39	29.52	46.63	50.36	72.24	76.39	104.76	115.62	131.48	137.63
16	12.02	12.67	27.31	29.85	46.44	48.33	72.97	76.01	105.4	116.18	132.45	136.55
17	12.02	13.19	27.21	29.66	46.37	48.66	73.35	76.86	105.86	112.68	133.11	139.8
18	12.02	14.52	27.27	29.96	46.38	47.75	72.83	76.94	105.19	112.37	131.03	138.46
19	12.02	12.14	27.21	29.32	47.07	49.9	74.14	76.73	105.92	112.64	130.18	141.31
20	12.02	12.67	27.27	29.28	46.53	50.29	72.51	75.66	105.93	112.84	130.67	139.71
Av	**12.02**	**13.44**	**27.27**	**29.68**	**46.53**	**49.75**	**72.91**	**76.82**	**105.50**	**113.07**	**131.74**	**138.91**

Table A.2

	14 Clients		**16 Clients**		**18 Clients**		**20 Clients**		**22 Clients**		**24 Clients**	
	Nash	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**
1	142.64	146.13	165.85	175.77	184.9	197.23	205.77	217.54	243.99	251.31	267.6	280.02
2	141.61	148.89	164.71	176.35	186.88	195.37	206.13	216.18	239.75	248.45	265.8	276.6
3	141.34	147.42	165.59	167.4	186.27	194.95	207.28	217.17	242.41	248.58	269.89	277.89
4	143.22	150.06	162.94	174.82	189.42	191.25	206.05	217.17	242.68	256.68	270.22	284.12
5	141.96	150.79	164.86	172.5	187.43	195.85	211.76	217.11	241.83	257.1	266.6	283.73
6	144.02	148.51	165.55	169.27	185.03	198.28	207.37	218.26	241.79	255.77	270.12	271.96
7	140.83	146.83	166.82	173.23	184.12	198.45	209.22	214.64	241.22	253.51	269.13	275.83
8	140.67	151.46	164.9	175.28	185.59	198.73	207.73	218.62	239.84	257.36	264.68	283.42
9	140.24	149.3	165.16	173.77	186.3	195.46	204.55	214.48	239.77	252.67	268.33	277.17
10	143.33	151.5	162.02	172.84	184.09	199.03	205.86	215.87	241.7	254.36	267.8	279.83
11	143.72	151.79	164.13	175.03	184.97	191.25	207.1	218.86	244.53	251.17	266.6	276.98
12	141.77	148.28	164.13	166.76	185.47	197.26	207.32	217.15	242.12	252.11	270.64	281.32
13	141.28	147.32	164.32	171.26	185.6	197.63	208.18	221.17	240	250.19	267.87	281.96
14	141.02	150.73	162.85	175.33	187.15	195.88	203.5	212.91	238.92	254.15	270.18	284.76
15	141.79	150.05	165.31	170.46	187.6	190.86	208.38	217.8	243.76	255.76	266.91	279.11
16	141.35	148.11	164.47	175.49	186.13	196.8	204.19	215.35	241.99	256.89	267.09	277.95
17	140.63	149.79	164.42	172.16	183.47	194.96	205.44	218.54	239.65	252.17	266.72	281.03
18	140.89	148.6	165.23	174.88	184.56	195.6	211.88	214.08	243.53	243.37	270.19	276.98
19	140.05	148.45	164.51	169.79	186.37	191.96	206.59	214.44	243.55	252.16	267.48	282.82
20	141.41	152.32	162.44	171.41	186.39	191.27	206.07	217.54	240.82	250.24	267.55	281.35
Av	**141.69**	**149.32**	**164.51**	**172.69**	**185.89**	**195.40**	**207.02**	**216.74**	**241.69**	**252.70**	**268.07**	**279.74**

Table A.2

	26 Clients		**28 Clients**		**30 Clients**	
	Nash	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**
1	293.22	308.52	317.67	335.2	345.46	371.95
2	291.82	301.49	315.89	335.51	349.4	362.52
3	295	300.82	315.72	336.61	351.49	359.58
4	296.95	302.36	317.22	334.22	346.81	359.82
5	293.14	299.62	315.08	324.75	350.35	358.67
6	294.49	310.39	324.68	338.91	349.34	370.62
7	291.31	313.39	320.14	325.9	347.56	363.52
8	294.49	314.55	323.53	326.35	356.56	371.27
9	293.16	306.92	316.23	332.96	354.54	362.06
10	292.25	301.98	320.36	331.38	349.57	361.42
11	293.95	303.62	319.88	332.41	350.24	372.92
12	294.85	301.49	321.64	330.97	347.47	359.74
13	291.38	302.24	319.67	337.9	355.77	364.06
14	295.39	300.17	319.16	334.32	347.87	370.68
15	289.15	309.27	320.32	334.03	351.75	369.34
16	291.38	306.57	321.39	326.44	353.93	365.03
17	290.41	302.25	320.33	331.08	347.84	370.32
18	292.75	309.08	322.71	326.47	351.68	365.5
19	296.96	303.54	316.74	328.74	346.37	362.68
20	290.57	314.46	313.17	328.47	350.96	362.12
Av	**293.13**	**305.64**	**319.08**	**331.63**	**350.25**	**365.19**

And we do twenty random experiments; we have the following results (for 6 servers):

Table A. 3: Twenty random experiments done for six servers

	2 Clients		**4 Clients**		**6 Clients**		**8 Clients**		**10 Clients**		**12 Clients**	
	Nash	**Seq.**	**Nash**	**Seq.**	**Nash**	**Seq.**	**Nash**	**Seq.**	**Nash**	**Seq.**	**Nash**	**Seq.**
1	11.84	12.22	26.89	29.55	45.4	48.44	70.41	77.27	101.81	108.78	126.17	135.79
2	11.84	13.56	26.79	28.58	45.45	49.1	70	72.92	102.58	110	127.26	132.92
3	11.84	12.9	26.94	29.13	45.34	47.81	70.31	77.42	102.19	107.82	126.1	133.66
4	11.84	13.7	26.89	28.47	45.16	48.89	69.93	75.85	102.35	106.78	126.65	134.67
5	11.84	13	26.79	30.38	45.2	48.44	70.21	73.66	102.45	107.47	126.61	132.71
6	11.84	13.14	26.83	29.4	45.24	49.33	69.98	72.73	101.84	109.1	125.94	131.94
7	11.84	13.66	26.79	29.75	45.2	48.89	69.82	74.92	101.77	108.3	126.06	134.37
8	11.84	13.56	26.94	30.38	45.46	48.19	70.58	76.55	101.59	105.69	126.21	136.05
9	11.84	13.96	26.79	29.52	45.14	48	69.86	72.83	101.37	108.37	126.98	133.28
10	11.84	12.32	26.79	29.7	45.46	48.72	69.84	76.78	101.53	106.71	125.97	134.12
11	11.84	13.17	26.94	28.82	45.14	48.33	69.56	72.81	101.76	110.57	126.3	132.22
12	11.84	12.41	26.85	28.58	45.16	49.15	69.66	74.95	101.74	110.24	126.46	131.43
13	11.84	11.84	26.79	29.11	45.28	49.17	69.93	76.3	102.79	109.38	126.86	132.59
14	11.84	13.66	26.94	29.78	45.22	47.95	69.86	73.7	101.13	108.32	127.56	133.58
15	11.84	13.93	26.79	27.51	45.22	48.36	70.11	75.58	102.34	109.57	126.49	132.04
16	11.84	14.52	26.94	30.26	45.41	48.38	70.02	75	102.01	109.63	125.66	134.89
17	11.84	14.05	26.85	29.57	45.44	49.07	70	73.67	102.58	111.07	126.33	131.96
18	11.84	12.32	26.85	30.54	45.41	48.21	69.69	75.39	101.2	109.09	126.89	133.8
19	11.84	12.41	26.89	29.87	45.4	48.82	69.78	74.46	101.05	109.84	126.03	132.29
20	11.84	12.9	26.83	28.54	45.22	48.8	69.78	73.94	102.23	107.37	125.88	136.35
Av	**11.84**	**13.16**	**26.86**	**29.37**	**45.30**	**48.60**	**69.97**	**74.84**	**101.92**	**108.71**	**126.42**	**133.53**

Table A.3

	14 Clients		**16 Clients**		**18 Clients**		**20 Clients**		**22 Clients**		**24 Clients**	
	Nash	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**
1	135.21	141.3	157.99	167.25	175	187.09	196.57	206.89	227.21	239.3	254.96	265.48
2	135.68	144.26	156.73	163.78	176.27	183.42	193.14	200.82	227.58	238.64	255.29	262.66
3	134.47	142.55	155.5	158.91	174.76	185.74	198.45	209.32	229.35	236.14	251.92	258.16
4	135.88	138.31	155.07	163.49	176.21	187.53	195.19	200.28	228.6	237.15	251.87	261.09
5	135.86	145.96	156.05	163.95	177.21	187.22	195.25	204.17	227.64	232.54	250.83	258.52
6	134.95	141.95	157.11	166.21	176.44	183.19	195.88	201.26	230.22	240.84	252.57	258.5
7	135.28	140.24	157.61	165.95	178.4	180.79	197.34	203.47	225.27	241.06	252.76	258.88
8	134.41	144.87	157.1	162.16	175.04	186.68	197.63	204.71	227.81	236.44	252.78	261.49
9	135.83	141.88	156.2	161.05	175.86	184.46	195.8	202.02	227.79	237.67	255.01	256.56
10	135.42	144.44	158.32	161.7	175.9	187.08	196.62	200.67	229.22	244.48	250.23	261.15
11	135.58	142.87	156.7	157.88	175.33	182.97	194.93	201.58	227.61	237.37	250.1	259.99
12	135.6	141.61	156.63	162.69	176.56	181.94	194.93	198.17	229.99	238.91	252.29	261.95
13	135.6	140.73	157.16	166.6	176.58	183.6	195.94	206.9	227.28	238	251.3	257.29
14	136.19	141.29	158.75	169.2	176.32	184.55	197.24	204.35	229.97	237.8	254.88	257.64
15	135.6	140.96	157.04	162.97	177.56	184.79	196.05	203.45	227.49	236.15	253.54	260.45
16	136.92	140.4	158.18	162.2	176.26	186.64	196.39	203.16	225.96	238.47	253.12	261.83
17	134.96	143.95	157.8	162.26	176.41	184.63	196.27	202.37	225.9	234.64	249.84	260.23
18	135.63	141.83	156.12	162.6	176.02	182.03	196.12	201.87	230.74	236.66	251.73	259.69
19	136.01	142.47	156.99	163.26	176.55	186.15	196.71	201.96	228.16	234.63	251.07	260.46
20	134.11	140.24	156.32	165.01	176.48	183.93	194.92	198.52	227.07	235.68	252.09	260.89
Av	**135.46**	**142.11**	**156.97**	**163.46**	**176.26**	**184.72**	**196.07**	**202.80**	**228.04**	**237.63**	**252.41**	**260.15**

Table A.3

	26 Clients		**28 Clients**		**30 Clients**	
	Nash	**Sequential**	**Nash**	**Sequential**	**Nash**	**Sequential**
1	276.2	287.27	300.33	312.32	323.15	339.95
2	272.47	279.81	295.18	305	325.4	334.55
3	274.72	281.69	297.36	305.74	325.27	332.34
4	273.59	281.83	296.12	305.36	328.56	333.63
5	276.29	280.55	297.2	308.03	324.35	334.44
6	276.93	282.1	297.32	305.64	323.58	335.88
7	275.02	284.69	298.06	311.79	324.31	327.79
8	274.18	281.56	298.5	307.39	322.04	334.98
9	272.82	289.27	294.81	309.48	323.25	333.35
10	275.21	282.89	295.7	305.54	327.82	335.08
11	275.96	286.44	297.47	308.32	326.06	334.45
12	272.3	281.72	297.48	303.48	323.83	335.51
13	277.56	279.42	299.54	305.87	324.66	331.95
14	273.88	286.17	298.28	304.37	321.96	336.62
15	271.78	292.59	299.73	310.58	323.99	338.62
16	275.16	280.5	294.91	303.84	326.33	337.25
17	276.53	290.39	298.65	310.35	325.13	334.58
18	271.68	286.51	298.43	306.25	322.3	330.65
19	273.1	283.98	297.14	303.88	327.39	333.22
20	272.13	282.12	299.14	298.55	322.21	337.41
Av	**274.38**	**284.08**	**297.57**	**306.59**	**324.58**	**334.61**

The average for the two strategies

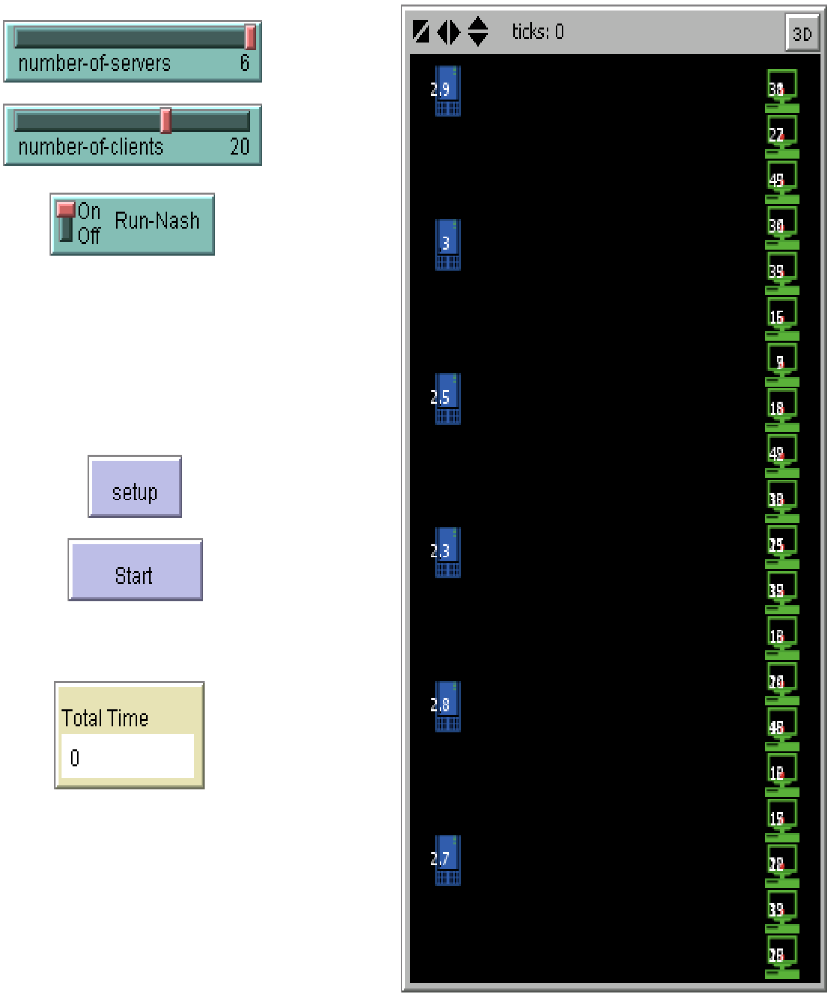
Table A. 4: This table is average of averages variables for Nash and Sequential.

Servers / Strategy	Av (Av(Nash))	Av(Av(Sequential))	R=$\frac{\sigma n}{\sigma s} \times 100\%$
2	190.052	196.32	96.81%
4	171.15	179.38	95.41%
6	161.6	168.28	96.02%

The standard deviation for the two strategies

Table A. 5: This table is standard deviation of averages variables for Nash and Sequential.

Servers / Strategy	StDev (Av(Nash))	StDev(Av(Sequential))	R=$\frac{\sigma n}{\sigma s} \times 100\%$
2	126.89	131.5	96.49%
4	107.8	111.8	96.42%
6	99.94	102.39	97.61%



number-of-servers 6

number-of-clients 20

On Off Run-Nash

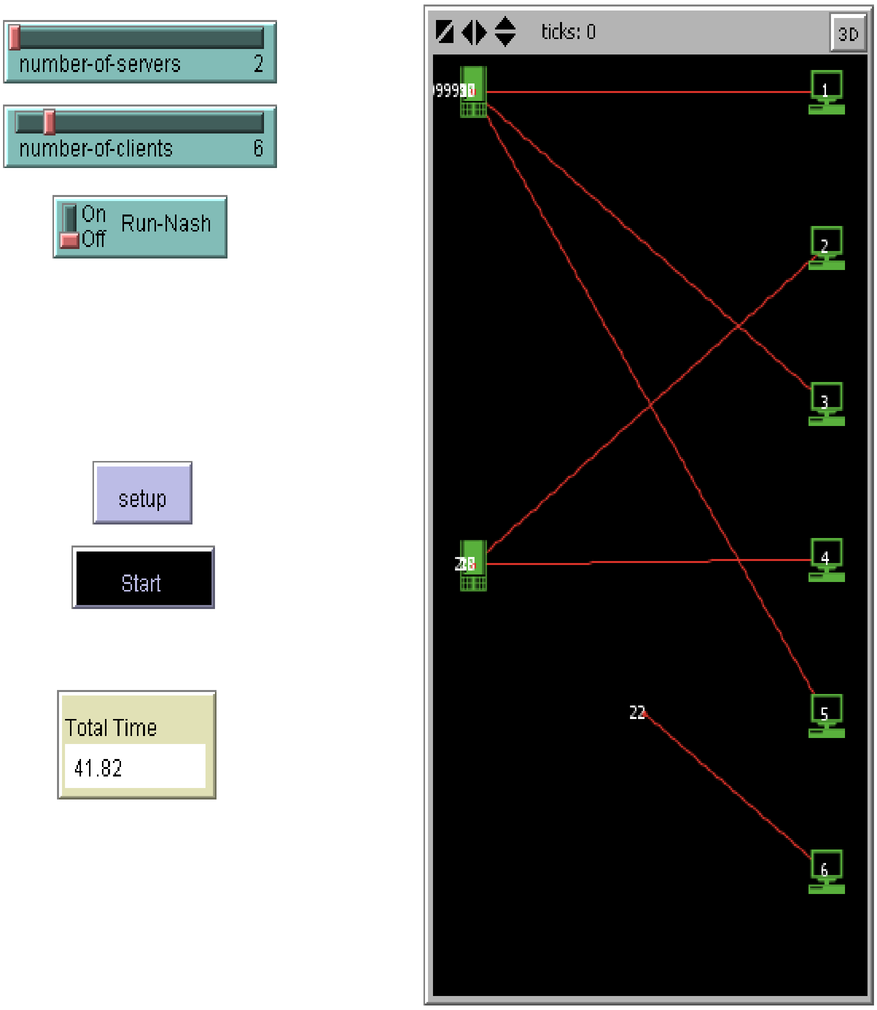
setup

Start

Total Time 0

ticks: 0

Figure A. 1: The environment for the simulation model.



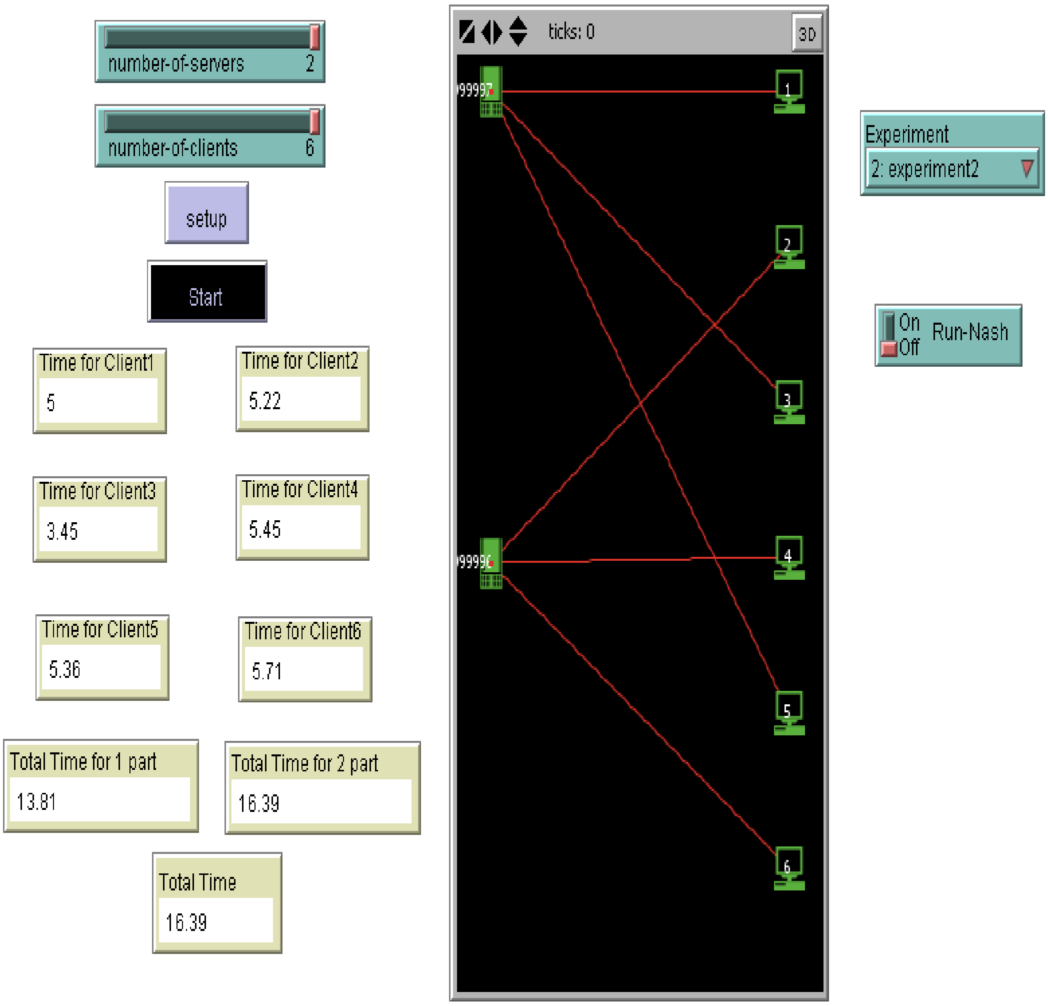
number-of-servers 2

number-of-clients 6

On Off Run-Nash

Total Time 41.82

Figure A. 2: The sequential load balancing for the first experiment.



number-of-servers 2

number-of-clients 6

setup

Start

Time for Client1 5

Time for Client2 5.22

Time for Client3 3.45

Time for Client4 5.45

Time for Client5 5.36

Time for Client6 5.71

Total Time for 1 part 13.81

Total Time for 2 part 16.39

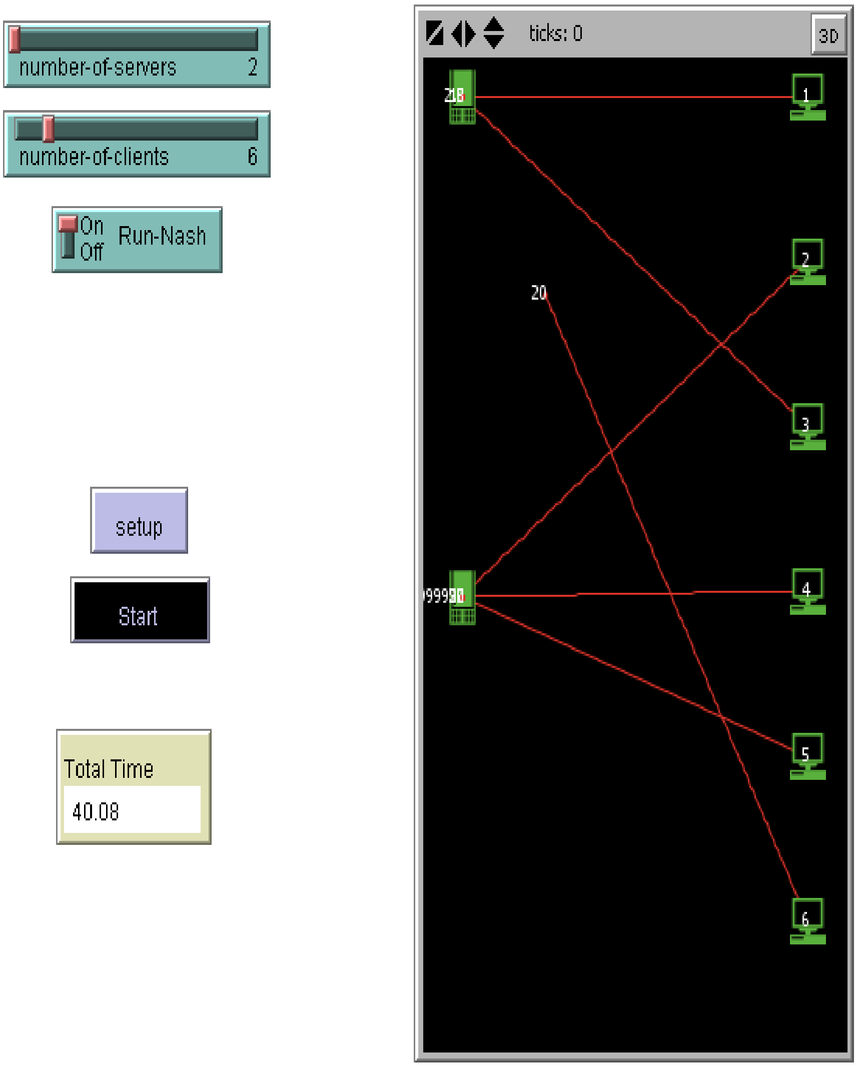
Total Time 16.39

ticks: 0

Experiment 2: experiment2

On Off Run-Nash

Figure A. 3: The sequential load balancing for the second experiment.



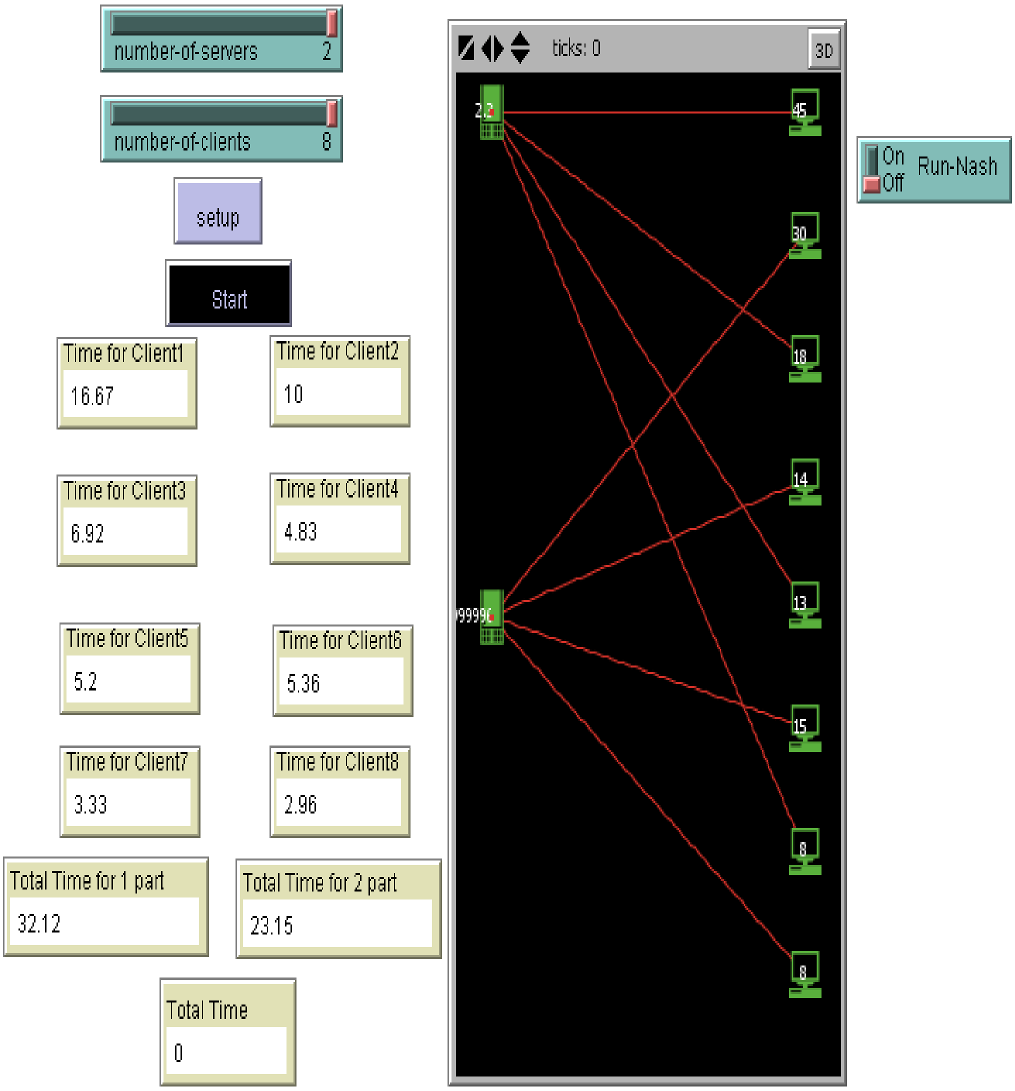
number-of-servers 2

number-of-clients 6

On Off Run-Nash

Total Time 40.08

Figure A. 4: The Nash Equilibrium load balancing for the first Experiment.



number-of-servers 2

number-of-clients 8

Time for Client1
16.67

Time for Client2
10

Time for Client3
6.92

Time for Client4
4.83

Time for Client5
5.2

Time for Client6
5.36

Time for Client7
3.33

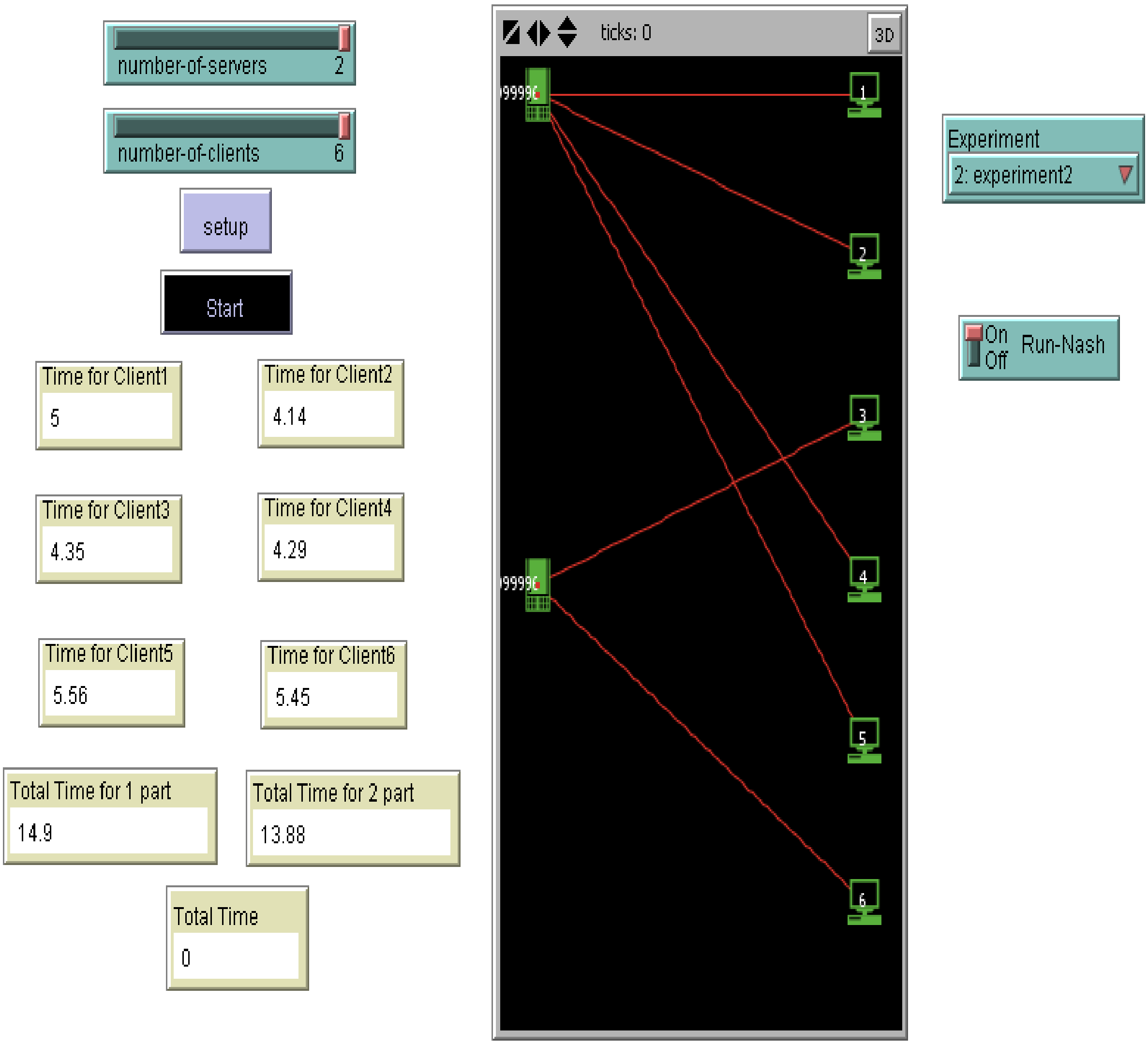
Time for Client8
2.96

Total Time for 1 part
32.12

Total Time for 2 part
23.15

Total Time
0

Figure A. 5: The sequential load balancing for the third experiment.



number-of-servers 2

number-of-clients 6

Time for Client1 5

Time for Client2 4.14

Time for Client3 4.35

Time for Client4 4.29

Time for Client5 5.56

Time for Client6 5.45

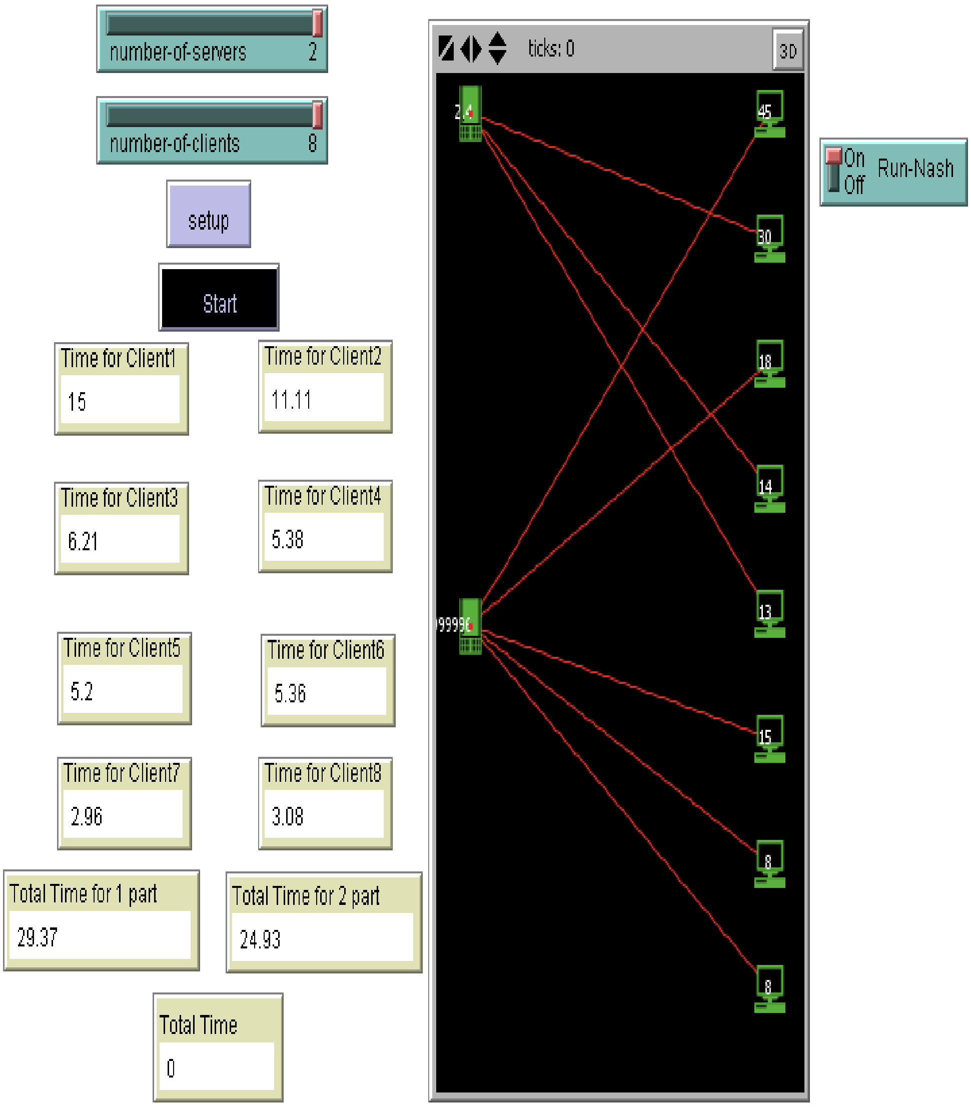
Total Time for 1 part 14.9

Total Time for 2 part 13.88

Total Time 0

Experiment 2: experiment2

Figure A. 6: the Nash equilibrium load balancing for the second experiment.



number-of-servers 2

number-of-clients 8

setup

Start

Time for Client1: 15

Time for Client2: 11.11

Time for Client3: 6.21

Time for Client4: 5.38

Time for Client5: 5.2

Time for Client6: 5.36

Time for Client7: 2.96

Time for Client8: 3.08

Total Time for 1 part: 29.37

Total Time for 2 part: 24.93

Total Time: 0

ticks: 0

On Off Run-Nash

Figure A. 7: The Nash equilibrium load balancing for the third experiment.

موازنة الأحمال لزبائن يتغيرون بشكل مستمر، نهج نظرية اللعبة

إعداد :

عوني سيف الدين جميل عبيد

إشراف :

الدكتور رشيد الجيوسي

ملخص:

في هذه الرسالة قام الباحث بابتكار طريقة جديدة لموازنة الاحمال في الخوادم من خلال استخدام نظرية اللعبة حيث استخدام نظرية موازنة ناش وهي عبارة عن مصفوفة مكونة من عمودين وصفين حيث يستطيع الزبون الاول الاختيار من بين الاستراتيجيات المتوفرة وغالباً ما تكون اثنتين، بحيث يختار هذا الزبون أفضل ما لديه مع النظر والاهتمام لما سيقوم به الزبون الآخر بحيث يكون اختياره هو الآخر هو الأفضل بالنسبة اليه، لقد استخدام الباحث السرعة المتوفرة لكل خادم بالاضافة الى الوقت المتوقع ان يشغله من خلاله كل زبون من خلال الخوادم المختلفة، وقد أظهرت النتائج تحسناً أفضل لأداء الخوادم من خلال هذه التقنية الجديدة.