"Multiagent Systems Storage Resource Allocation in a Peer-to-Peer Distributed File System"

University of Manitoba TR Number: 03/01
University of Calgary TR Number: 2003-716-19

Sergio Camorlinga [1] and Ken Barker [1 and 2]

[1] Department of Computer Science, University of Manitoba,
Winnipeg, Manitoba R3T 2N2 Canada
sergio_camorlinga@sbrc.ca

[2] Advanced Database Systems and Applications (ADSA) Laboratory,
Department of Computer Science,
University of Calgary, Calgary, Alberta T2N 1N4, Canada
barker@cpsc.ucalgary.ca

January 2003

"Multiagent Systems Storage Resource Allocation in a Peer-to-Peer Distributed File System"

Abstract

The objective of this research project is to understand and develop Multiagent Systems (MAS) storage resource allocation algorithms and methods where a peer-to-peer (P2P) system of computer resources is seen as a community of peers with different capabilities that collaborate to enhance the global performance in terms of storage resource balancing. The research focuses on P2P Distributed File Systems (DFS) storage resource allocation algorithms. Complex adaptive systems are evaluated and used to produce emergent global behaviours that can solve the storage resource allocation problem in a distributed system of peers. Squirrel behaviours provide a metaphor to develop algorithms and methods to allocate resources. A complete simulation software tool was developed to implement different behaviours that are analyzed on different scenarios and experiments. Experimental results support the initial hypothesis that hoarding mechanisms found on squirrels behaviours allocate efficiently resources on a distributed system of consumers and providers of storage resources.

1. Introduction

A Peer-To-Peer (P2P) system is characterized as a distributed system where the entities at the network end are called peers, and a peer can share and exchange resources with other peers in the system. Recent 2nd generation P2P look-up systems (e.g. Chord [1], Content Addressable Networks [2], Tapestry [3], Pastry [4]) have primarily focused on the use of a hash function to produce a key that will uniquely and globally identify a file within a system. Each peer in the P2P system is responsible for a range of the hash function's keys; consequently this peer is responsible for allocating storage resources to store the file, whose hash function value map to the keys managed by the peer. This approach assigns storage across the P2P system and assumes an even allocation of keys by the hash function between the peers. This assumption introduces severe drawbacks in terms of resource allocation (latency, hot spots), inefficient use of peer capabilities, and loss of spatial locality and application dependencies. These drawbacks limit the efficiency of a P2P computing environment because the storage allocation in the distributed file system (DFS) is not balanced.

Multiagent Systems (MAS) provide an interesting model of computing formed by a community of entities, independent of each other, working with no specific leadership, but with a global purpose that translates into corporate result(s). This brings a change in the way we see things, domain community goals are not results of special individuals but the comprehensive, emerging global results of the sum of each individual actions working almost independently.

Due to the MAS computing model's similarity to the P2P computing model in terms of the distributed environment of both, MAS could be applied to a P2P DFS computing environment thereby providing more efficient use and allocation of storage resources to overcome some of the existing limitations of 2nd generation P2P systems. This research develops MAS algorithms and methods for a community of peers working independently to provide resource allocation with an emphasis on storage resources. The basic idea is

there is a community of independent peers that are initially interested in storing their own data, but consequently the community as a whole will balance storage allocation globally thereby making better use of resources in the P2P environment.

Section 2 provides a framework by describing related work based on modeling natural and economic behaviours that facilitate the design of MAS algorithms for resource allocation. The section then describes how these have been applied to P2P models. Section 3 describes squirrels' behaviour that is the metaphor for this research and provides an overview of storage resource allocation in a P2P DFS. Section 4 details the squirrel MAS environment and explains how to obtain resource allocation balance. Section 5 presents the squirrel MAS algorithms developed and illustrates their implementation. Section 6 contributes by describing the squirrel MAS class architecture used to implement the squirrel MAS simulator. Section 7 provides a brief introduction to developed software simulator and details the experiments carried out to validate the approach and present the results obtained. Finally, Section 8 concludes and provides some insights for future research directions.

2. Related Work

Researchers have used natural and economic models to design MAS algorithms to solve different kinds of resource allocation or optimization problems. A common natural model used is that of ant behaviours [16][17]. Although each ant undertakes very simple activities, the global outcome of the collective accomplishes goals that are very difficult without the ants behaving as a team. Ant behaviours are an example of what is known as *complex adaptive systems* (CAS) [6]. These systems have a large number of members with simple functions and limited communication among them. They exploit their synergy to solve complex problems and accomplish global behaviours. The *swarm intelligence* [5] that emerges from the global behaviours of simple members boasts autonomy and self-sufficiency, which allows them to adapt quickly to changing environmental conditions. Several other insects and animal societies have been studied [7] where emergent team or group behaviours exist. The interesting aspect of CAS is that they present team or group models that solve complex distributed problems, which otherwise will be difficult with deterministic centralized or distributed techniques.

CAS is applied successfully to different problem domains. For example, White [8] applies the technique in the telecommunications routing domain using ant behaviours models. His work makes use of genetic algorithms to produce adaptive behaviours. Bonabeau *et al.* [5] describes how ant behaviours solve the classical traveling sales problem effectively though not necessarily optimally. This highlights an important aspect about the paradigm: CAS does not guarantee the best possible outcome because of the randomness and unpredictable emergent behaviour,. Clearly, this means that it is not suitable for all problem domains but its strength lies in its ability to produce "acceptable" (though possibly suboptimal) solutions that are acceptable under certain circumstances. Dorigo *et al.* [9] apply ant behaviours to the traveling salesman problem and to other combinatorial optimization problems. Dorigo's work uses positive feedback to optimize the search. Deadman *et al.* [10] base their work on the CAS metaphor to simulate

common pool resource management problems and to explain the consequences of actions taken by individuals using these common pools.

Another CAS metaphor used in resource allocation and optimization problems is an economic model. Gibney *et al.* [11] propose a framework of self-interested agents, with limited communication and context information, which use a market economy model to exchange information and allow cooperation between "buyers" and "sellers". Kuwabara *et al.* [12] enhance the market economy model by proposing agent learning to improve decision process between buyers and sellers, which consequently reduce oscillations in resource allocation. Chavez *et al.* [13] propose a similar model but apply it to the allocation of processor time resources using learning abilities to adapt more appropriately to load fluctuations and message delays.

P2P distributed computing based on mobile agents is an area of active research [14][15][18]. This work uses agents to design and implement different P2P services and functions that are distributed across the P2P environment. The *Anthill* project [6][19][20][21] applied the CAS framework to design, implement, and evaluate P2P applications. Anthill's metaphor of ants foraging is applied to CPU load balancing [21].

This work use the CAS framework to develop MAS algorithms for storage resource allocation on a P2P DFS. Contrary to previous work ours uses a squirrels' hoarding behaviours as a CAS metaphor. We provide a highly flexible architecture that is applicable to other resource allocation problems. The following sections describe our squirrel MAS design framework, its implementation, and test results.

3. Background

Many ecological and environmental protection groups have studied the hundreds of squirrel species found worldwide [22][23][24]. Squirrels eat various nuts, acorns, and other small foodstuffs[1] by caching acorns in small hoards over a large geographic area. When food becomes scarce they return to the caches. Their failure to recover a great percentage of the horded acorns facilitates the growth of new trees. Squirrels, in conjunction with Jays [22], are responsible for the vast oak extensions through North America after the glacial age 10,000 years ago. This is an example of an individual's simple behaviour leading to the reforestation of vast geographical extensions. From another perspective, squirrels are allocating resources (land space) to storage demands (acorns) in such a way that resources are balanced. The global or corporate consequence is a populated forest of oak trees across a wide geographic area.

The interesting squirrel behaviour is the hoarding, rather than the foraging, as it leads to a global outcome of which the individual squirrels cannot possibly be aware. We are interested in the way squirrels spread out acorns, nuts and other small food pieces in an area to obtain a resource allocation balance. We are exploring how to exploit these

[1] We speak of these foodstuffs collectively using the exemplar "acorns" as we are interested in the consequence of the hording activities.

techniques to design and implement a MAS storage resource allocation suitable for a P2P DFS. The squirrel hoarding behaviours are summarized as:

- Random gathering and burying of small acorn amounts in an area geographically close to its nest,
- They may investigate various random locations ("sniffing" several places) before deciding where to put the acorn,
- A squirrel can decide not hoard its food if there are other squirrels around, and
- Squirrels sometimes work in small teams with others they "know" but avoid places inhabited by unknown squirrels.

We draw an interesting analogy with a DFS that provides file management services to its applications. One of these services is the allocation of storage space when saving a file for an application. Storage allocation occurs in data blocks or complete chunk for a whole file. Storage allocation occurs over the distributed storage resources available. In P2P DFS computing environment, each peer in the P2P system contributes storage resources. It is assumed that each peer has similar (though not equal) capabilities, that is each peer potentially stores (requests data storage resource) or allocates space (provides data storage resource) according to peer capabilities.

Our basic hypothesis is that MAS algorithms based on squirrel behaviours as a CAS metaphor, provides well-balanced resource allocation across a distributed system of resource providers. A P2P DFS storage resource allocation is the problem domain that substantiates the hypothesis.

4. Squirrel MAS Environment

The "squirrels" system consists of a P2P environment of locations. Each location has one or more caches where squirrels hoard acorns. Squirrels live in these locations in small groups. When they have acorns, they go through the locations, "sniffing" to find a cache suitable for the acorn following one of the behaviours described in Section 3. Figure 1 illustrates such an environment.

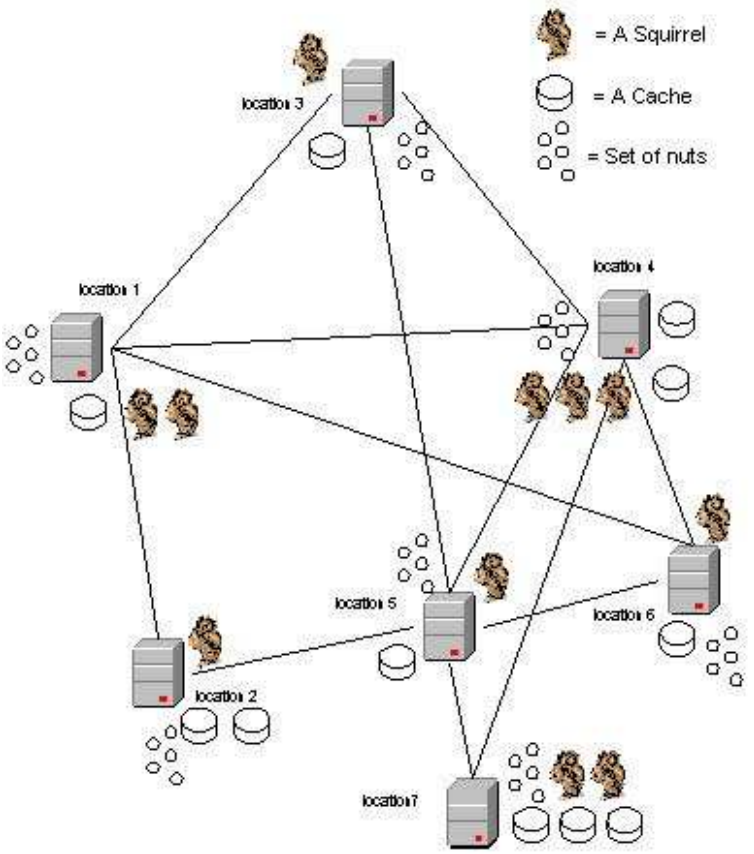

Figure 1: Squirrel MAS Environment

Each location (peer) has different capabilities such as the number of caches and acorns to hoard in the system. If there are no more acorns to store in a location, the squirrels living there sleep until there are more to hoard. We assume that each cache stores the same amount of acorns and each location may have different storage capabilities because each has a different number of caches. Each bcation requests different amount of storage resources because they have different number of acorns to allocate.

In other words, each location is a producer and consumer of storage resources. Translating the squirrel system architecture into a P2P DFS, each location becomes a peer within the P2P environment. Each acorn is a data block or file chunk that must be stored within the DFS. Each cache is a group of data blocks or chucks available that can be allocated. The goal is efficient allocation of storage resources by balancing resource allocation. For the purpose of this research we claim that the allocation algorithm that evenly distributes the resources over the DFS is better than one that does not. Clearly, other metrics must be considered but we leave this discussion in the realm of future research.

Variance is a measure of the spread of a distribution and is the averaged squared deviation of each distribution element from its mean:

$$\sigma^2 = \sum (x-\mu)^2 / N \quad \text{where } \mu = \text{mean}, \ N = \text{number of elements}$$

and the mean is the sum of all element values divided by the number of elements

$$\mu = \sum x / N$$

If we let x be the number of acorns allocated in a location's cache, the variance will indicate how well the system is allocating acorns among location caches, i.e. if deviation $\sigma \to 0$, then we have a better storage resource allocation because each site has a more equal assignment of acorns allocated per cache. Thus, we can have locations with different numbers of caches provided a cache is the common unit of resource. Further, if we balance caches, then a well-balance system is achieved across locations according to their capabilities (for example, processor capabilities can be considered in the metrics).

5. Squirrel MAS Algorithms

This section describes the three algorithms that implement squirrel behaviours described above and implemented in this research.

For all algorithms the following definitions apply

acorn = data block or chunk to allocate
n = number of peers in distributed system
loc_j = a location within the distributed system that has 'x' acorns to allocate, 'p' caches to store acorns and 'm' squirrels living there, $j = 1, \ldots, n$
s_{kj} = squirrel k at location j, $k = 1, \ldots, m$ and $m >= 0$
c_{ij} = cache i at location j, $i = 1, .., p$ and $p >= 0$ that stores 'y' acorns, $y>0$

Gathering and Random Burying Algorithm

The algorithm has the squirrels gather acorns from its home location and randomly search for another location (that could be its own) to cache the acorn. If the selected location is full (i.e. no more space in any of the selected location's caches), then the squirrel will pause and then jump to another location to test for available space. This cycle repeats until a suitable place is found or die looking for one.

Formally this algorithm is:

For each squirrel $s_{k,j}$
  If home loc_j has acorns to bury
    $current = j$
    Select a path randomly to go to another $loc_{current+1}$
    Move squirrel $s_{k,j}$ to $loc_{current+1}$
    If any slot available in cache $c_{i,current+1}$ $\forall$ i in $loc_{current+1}$

Put acorn in $c_{i, current+1}$
Return squirrel $s_{k, j}$ to home loc_j
End if
End if
End for

Gathering and Random Sniffing Burying Algorithm

This algorithm is similar to gathering and random burying algorithm in that the squirrel will gather acorns from its home location and randomly look for another location (that could be its own) to cache the acorn. However, the squirrel will sniff several locations before deciding where to place the acorn. The selected location will be the best option for the squirrel based on a specific (set of) metrics. If all sniffed locations are full, (i.e. no more space in any of the sniffed locations' caches), then the squirrel pauses and jumps to another location. This cycle repeats until a suitable cache is found or the squirrel dies. A squirrel will have a maximum number of steps to find a suitable cache to store the current acorn being allocated i.e. to converge to a solution. If this maximum number is reached, then the squirrel will give up and die. This maximum number is setup to avoid a squirrel living forever without finding an appropriate cache to store the acorn. This situation will occur when there is no more storage space available but we still have acorns to allocate.

Formally this algorithm is:

For each squirrel $s_{k, j}$
If home loc_j has acorns to bury
$current = j$
For each sniffing cycle
Select a path randomly to go to another $loc_{current+1}$
Move squirrel $s_{k, j}$ to $loc_{current+1}$
Sniff and record $loc_{current+1}$
End for
Select best cache $c_{i, j+1}$ $\forall$ i in sniffed location set
If $\exists$ cache $c_{i, j+1}$ found
Put acorn in cache $c_{i, j+1}$
Return squirrel $s_{k, j}$ to home loc_j
End if
End if
End for

Gathering and Random Sniffing Burying with No Strangers Algorithm

This algorithm is similar to the previous one in that the squirrel will gather acorns from its home location and randomly investigate several locations to find one (that could be its own) to cache the acorn. However, the squirrel will sniff only locations not being visited by other squirrels to determine the best location to place the acorns. Thus, the selected location is the best available that has not been visited by a stranger. If all sniffed locations

are full or visited by strangers, then the squirrel will pause and move to another location. This cycle repeats until a suitable location is found or the squirrel dies.

Formally this algorithm is:

For each squirrel $s_{k,j}$
    If home loc_j has acorns to bury
        $current = j$
        For each sniffing cycle
            Select a path randomly to go to another $loc_{current+1}$
            Move squirrel $s_{k,j}$ to $loc_{current+1}$
            Sniff $loc_{current+1}$
            If $loc_{current+1}$ not previously visited by another squirrel
                Record $loc_{current+1}$
        End for
        Select best cache $c_{i,j+1}$ $\forall$ i in sniffed and not visited location set
        If $\exists$cache $c_{i,j+1}$ found
            Put acorn in cache $c_{i,j+1}$
            Return squirrel $s_{k,j}$ to home loc_j
        End if
    End if
End for

In algorithms, "gathering and random sniffing burying" and "gathering and random sniffing burying with no strangers", the best cache selection process can be based on different parameters. Our work uses existing free space to decide which cache is best to select, i.e. the cache that has more space available among those sniffed by the squirrel will be selected to allocate storage space for the current acorn being saved.

6. Squirrel MAS Class Architecture

The squirrel MAS class architecture is illustrated in Figure 2. There are two groups of classes: The location class, which has cache, acorn and link classes, and the squirrel class. The location class represents a geographic area where a squirrel can bury acorns from its home location in places called caches. A cache can allocate several acorns in it. There is also a link class that records paths between the different locations. One or several squirrels belong to a location, i.e. it is their nest. The squirrels pick up acorns from their home location so they can be buried in another location's cache. When there are several squirrels in a single (or home) location, they work faster to bury the home location's acorns.

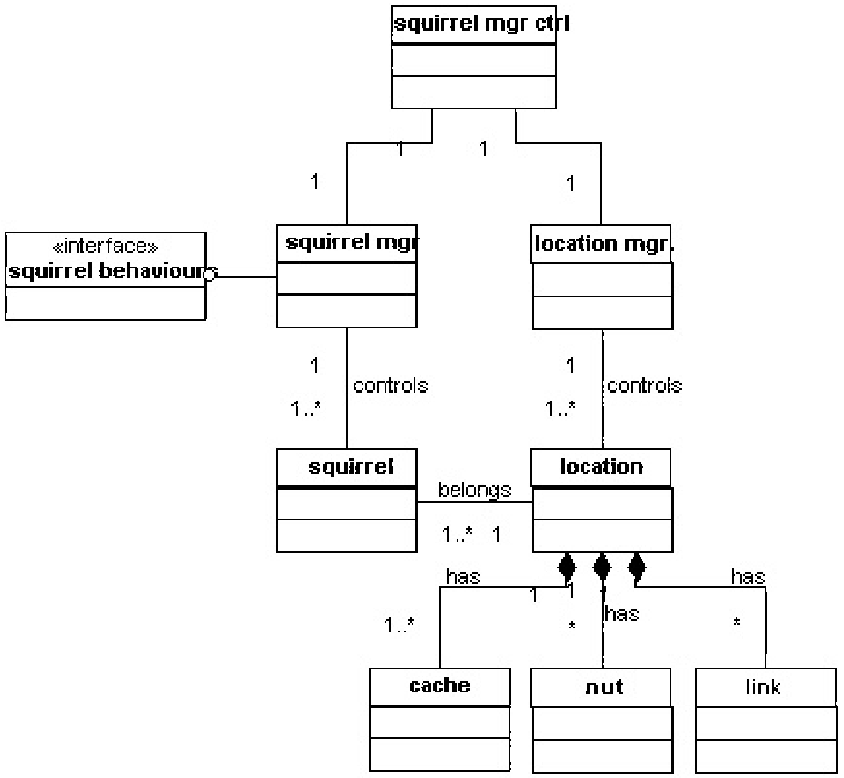

Figure 2: Squirrel MAS Class Architecture

The location manager class controls all locations in the distributed system by tracking their locations and managing them. The squirrel manager manages the squirrels, which are the MAS agents responsible for storing and allocating storage resources to acorns that need to be buried. The squirrel manager implements the interface squirrel behaviours, which define the behaviours supported by this squirrel MAS. These three behaviours are defined in the background section. Finally, the squirrel manager controller controls both managers and provides the GUI to the user making use of the squirrel MAS. It provides all necessary interfaces to setup the squirrel MAS and outputs their results in the working system. Figure 2also indicates the associations among the different classes defined in the squirrel MAS class architecture.

7. Squirrel MAS Experiments

Our simulator is developed using Java. We have undertaken a set of experiments to analyze the effectiveness of various squirrel behaviours on the storage resource allocation of a P2P DFS. The simulator implements the squirrel MAS class architecture described in Section 6. This simulator is briefly described next and then various experiments are reported and discussed.

Squirrel MAS Java Based Software Simulator

The Squirrel MAS Simulator (SMS) allows you to dynamically set the following parameters:

- Number of squirrels per location
- Number of caches per location
- Cache size per location
- Number of locations
- Network topology
- Acorns allocated per location
- Number of times to sniff in sniffing behaviours
- Simulation duration by defining number of steps to execute
- Logging of messages and activity on screen

These parameters provide substantial flexibility in experiment design as the experimenter can define resources' demands (acorn demands), resources' availability (caches, cache size, locations), agents (squirrels per location), resource location (locations network topology), behaviours (sniffing times, behaviour selection) and simulation environment (simulation steps, logging) so experiments can be set up in many interesting ways.

Figure 3 is a SMS screenshot. The software is developed on Linux Red Hat version 7.0 using JBuilder 4.0 IDE and Java SDK 1.3. The main menu bar provides different options as depicted in the simulation menu in Figure 3.

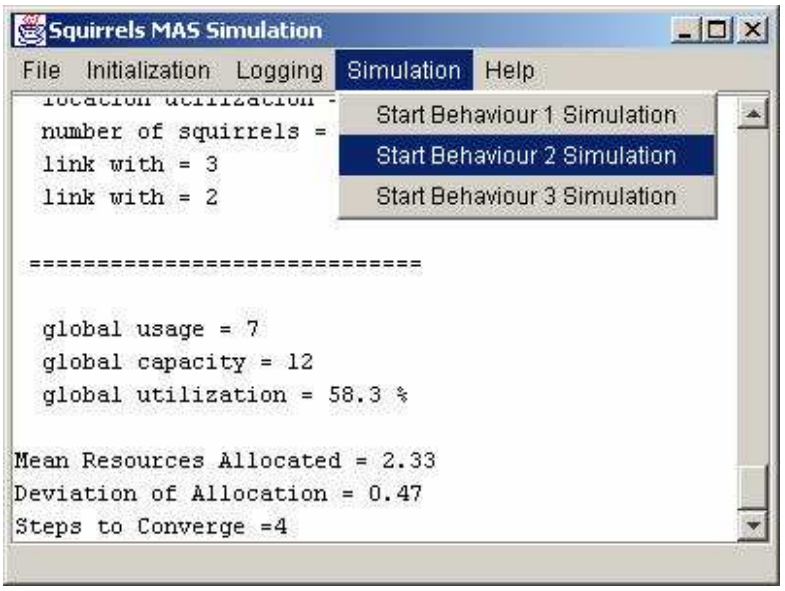

Figure 3: SMS Main Menu Bar

Experiments Executed and Results

All experiments analyze the three behaviours described above. Both sniffing burying algorithms were run with one, two and three sniffing steps each, while random burying only requires one-step at a time. The resource variance in cache allocation is used as an indicator of how well the P2P DFS resources are balanced. Lower variance indicates better balance as explained earlier.

The following notations are used to present the results:

- beh 1 = Gathering and random burying algorithm behaviour
- beh 2 – x = Gathering and random sniffing burying algorithm behaviour with 'x' number of sniffing steps, x = 1..3
- beh 3 – x = Gathering and random sniffing burying with no strangers algorithm behaviour with 'x' number of sniffing steps, x = 1..3
- deviation = square root of variance in resources allocated (acorns) to locations' caches
- % acorns/total cache capacity = percentage of resources to be allocated in relation to the total capacity of all locations' caches.

A fixed topology of ten nodes is used in all experiments and a fixed capacity of 20 acorns per cache is assumed. There is only 1 cache per location. The resource allocation balance and converging time with different resource demands in the squirrels MAS behaviours was the main focus of these experiments. The resource allocation balance will be obtained by calculating the deviation of acorns allocated across the set of location' caches available in the distributed system. The less deviation, the better resource allocation balance obtained. The number of steps the squirrels take to allocate the acorns required to be stored will obtain the converging time. The less steps taken, the faster the whole system is at obtaining the storage balance allocation.

These experiments verify the initial hypothesis, which states that MAS algorithms, based on squirrels behaviours as a CAS metaphor, provide the basis for a well-balanced storage resource allocation across a distributed system.

Experiment 1:

Figure 4 and Figure 5 illustrate experiment 1 results. There is one squirrel per location seeking a cache to place acorns. The demand for resource allocation is evenly distributed across locations. Beh 2 – 3 and beh 2 – 2 offer the best allocation of resources while beh 3 (all three) is between the others. Further, beh 2 – 3 and beh 2 – 2 converge to complete resource allocation in fewer steps than the others (see Figure 5).

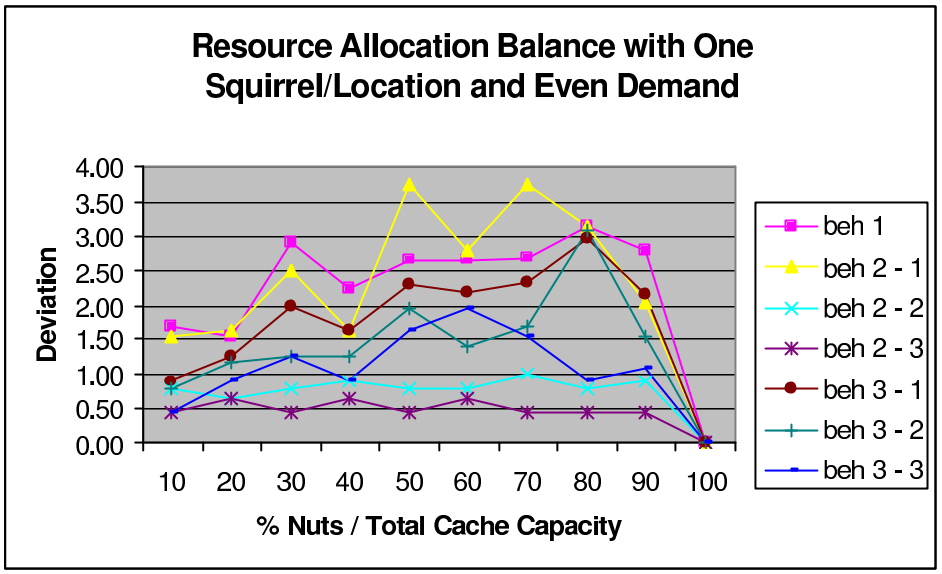

Figure 4: Experiment 1 Deviation Results

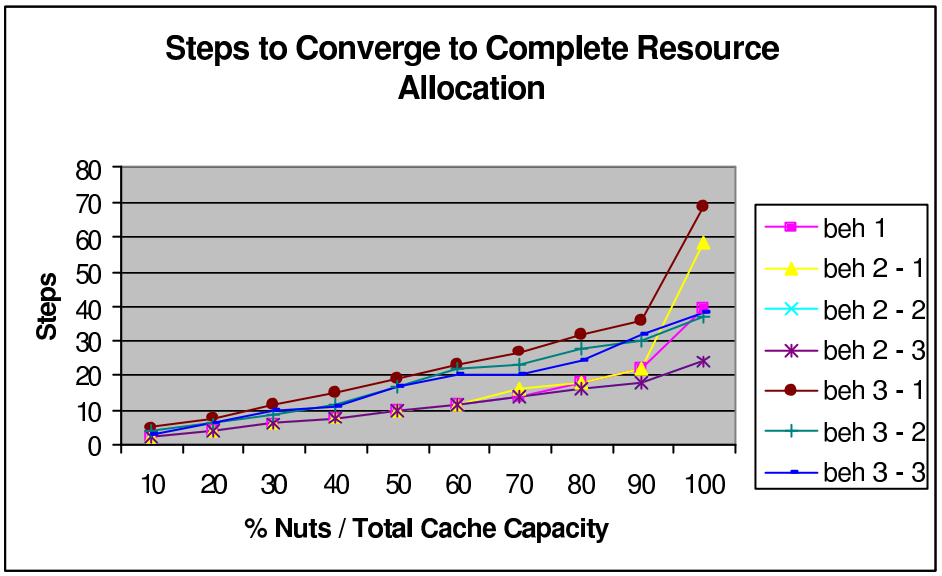

Figure 5: Experiment 1 Steps to Converge Results

Experiment 2 and 3:

In these experiments, all conditions are fixed similar to experiment 1 but the number of squirrels available to allocate resources per location is varied. Figure 6 and Figure 7; show results when there are two squirrels per location, while Figure 8 and Figure 9 report the results when there are 4 squirrels per location. Better resource allocation was obtained

when there were 4 squirrels for all behaviours, suggesting an emergent global result of better resource allocation when more entities are working. Further, there is improvement in the number of steps required to converge when more squirrels hoard acorns. Beh 2 – 2 and beh 2 – 3 followed by beh 3 – x continue to be the best performers in resource allocations, but beh 3 – x takes longer to converge to a final allocation, primarily when the proportion of acorns to total cache capacity approaches 100%. This is explained because when you have more acorns to allocate, the squirrels in beh 3 – x will need more steps to find a suitable cache for placement because they avoid another squirrel at the same cache.

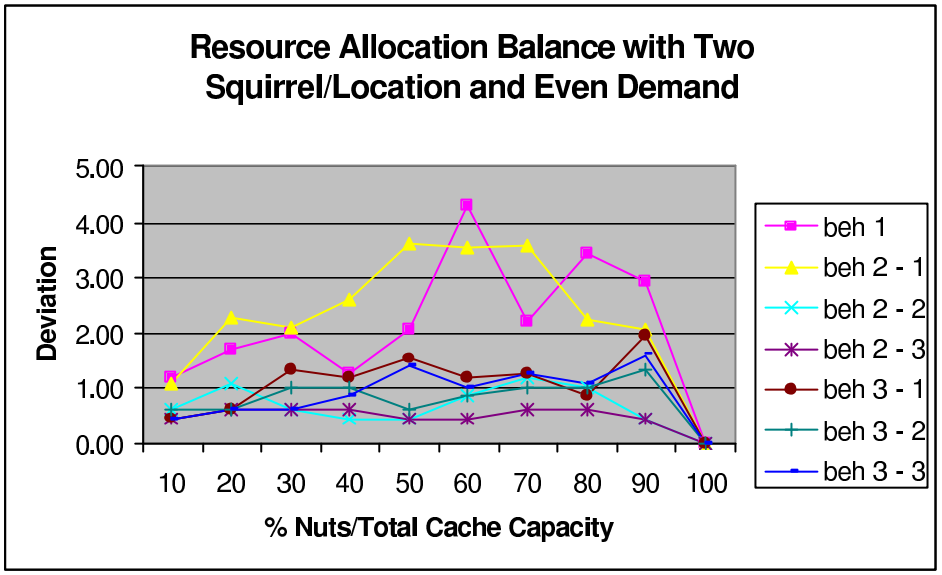

Figure 6: Experiment 2 Deviation Results

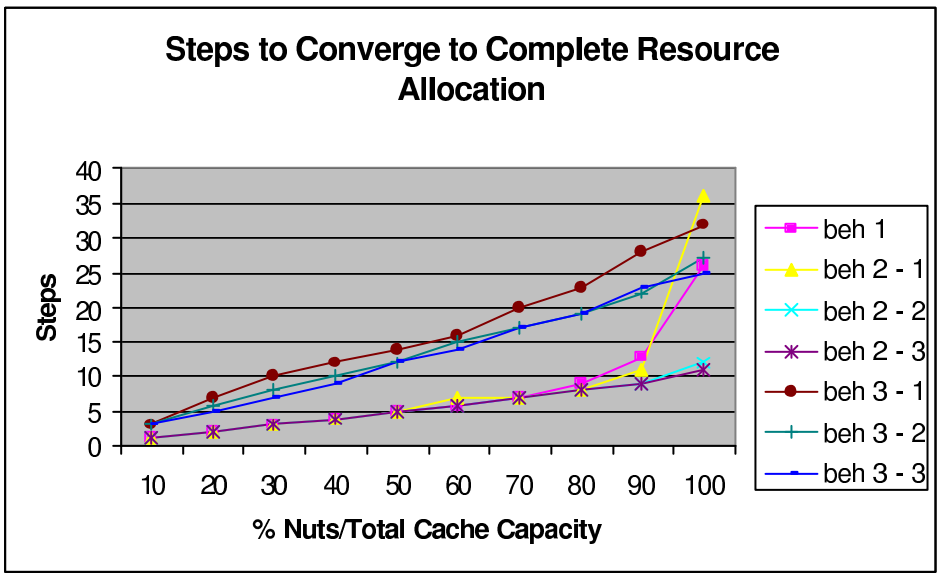

Figure 7: Experiment 2 Steps to Converge Results

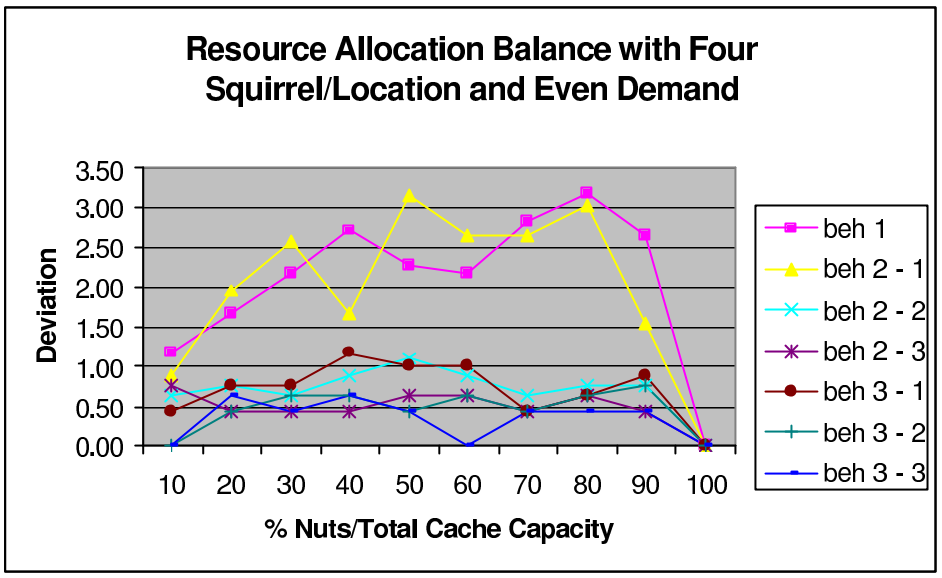

Figure 8: Experiment 3 Deviation Results

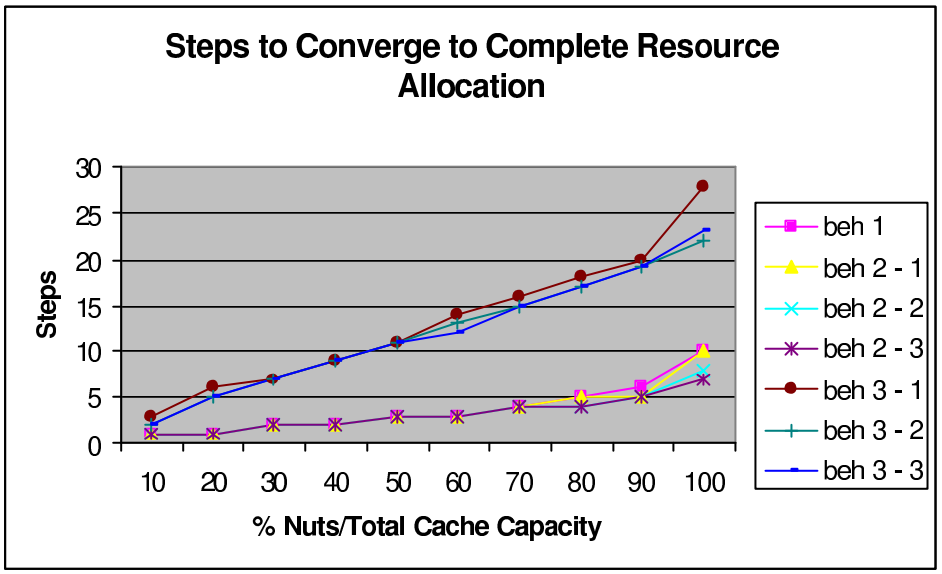

Figure 9: Experiment 3 Steps to Converge Results

Experiment 4:

In this experiment the demand is distributed unevenly across locations. We test two scenarios when we have a different number of squirrels per location: one scenario is with 50% locations demanding and another with 20% locations demanding. The percentage of "location demanding" means the proportion of the total number of locations that require

storing data on the whole system. For example, it is possible that only 20% of the locations produce the entire demand to store acorns on the P2P DFS but all locations' caches are storage repositories available. The results are shown on Figure 10to Figure 15.

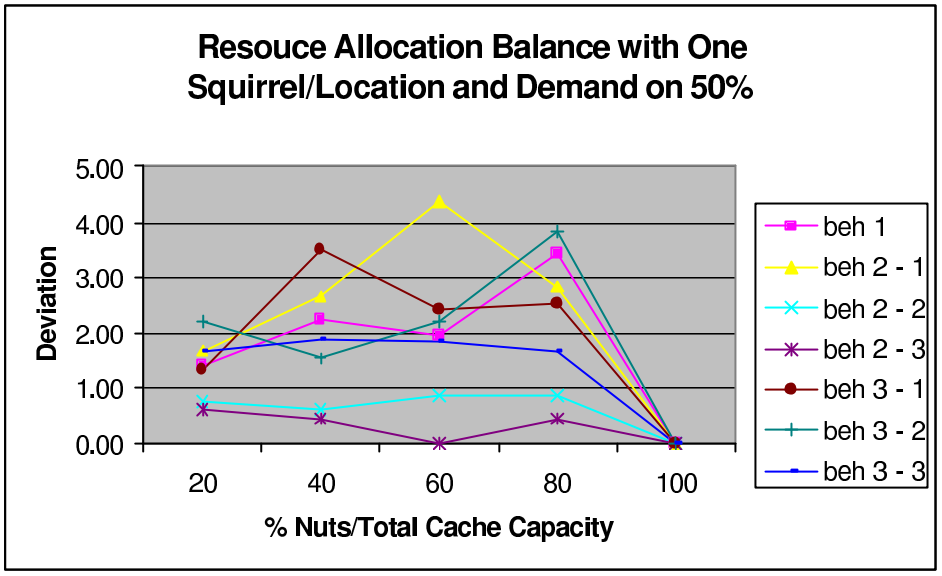

Figure 10: Experiment 4 Deviation Results on 50% Demand and 1 Squirrel/Location

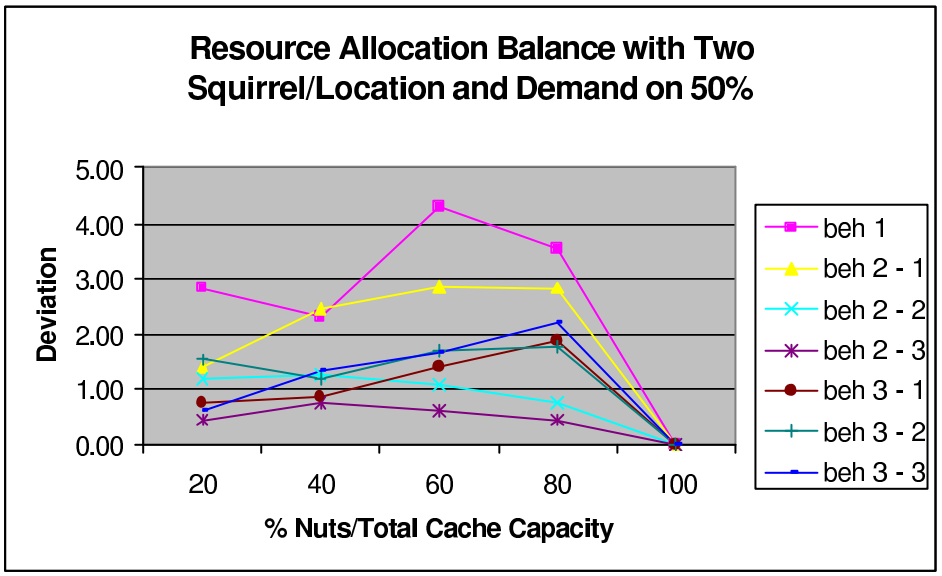

Figure 11: Experiment 4 Deviation Results on 50% Demand and 2 Squirrel/Location

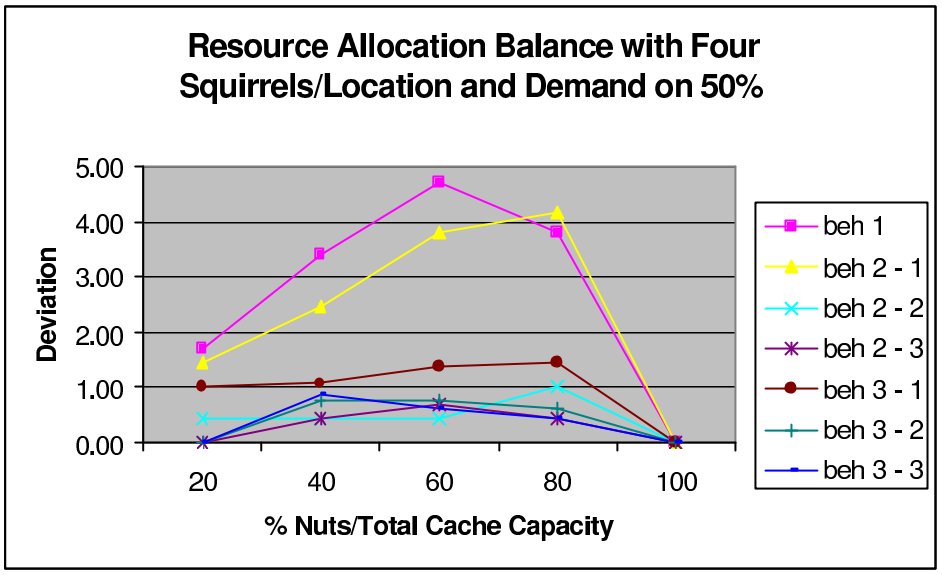

Figure 12: Experiment 4 Deviation Results on 50% Demand and 4 Squirrel/Location

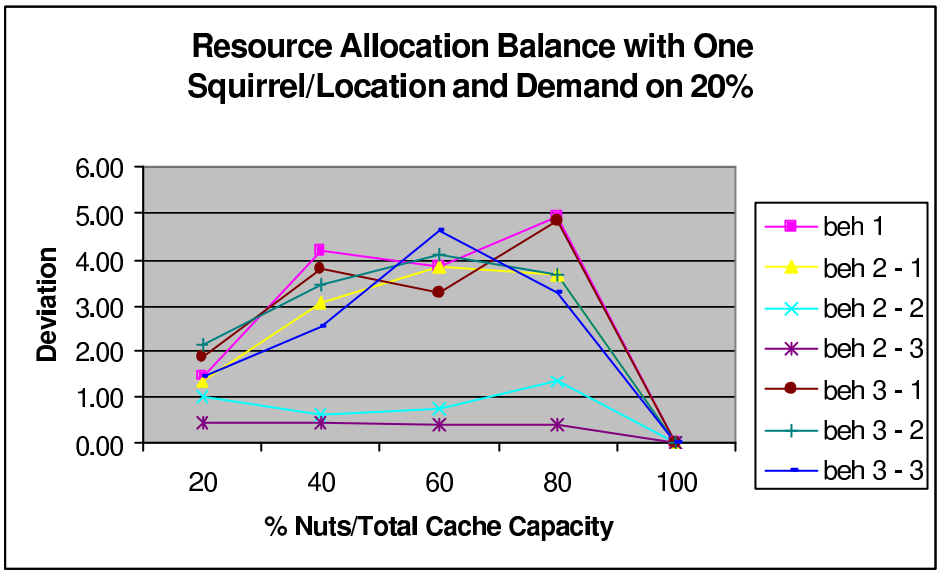

Figure 13: Experiment 4 Deviation Results on 20% Demand and 1 Squirrel/Location

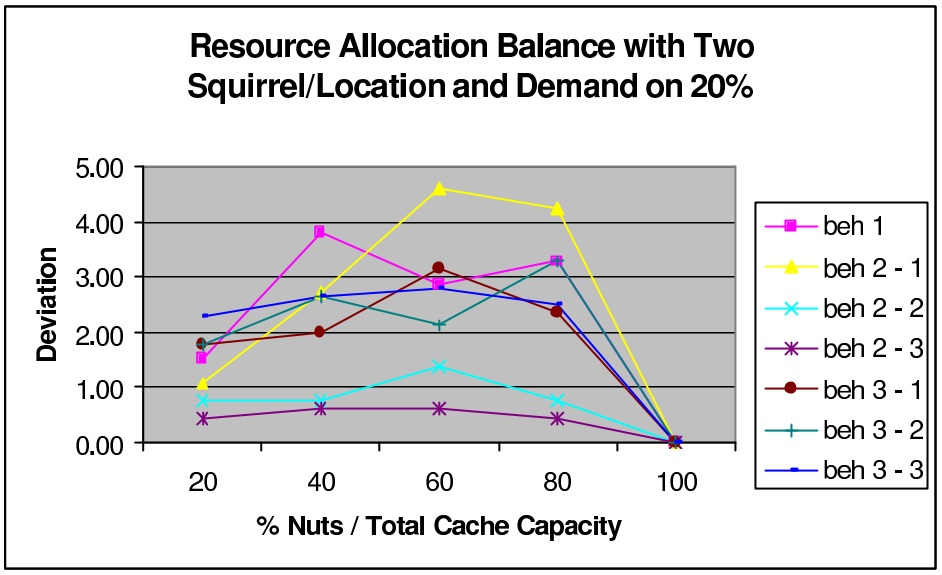

Figure 14: Experiment 4 Deviation Results on 20% Demand and 2 Squirrels/Location

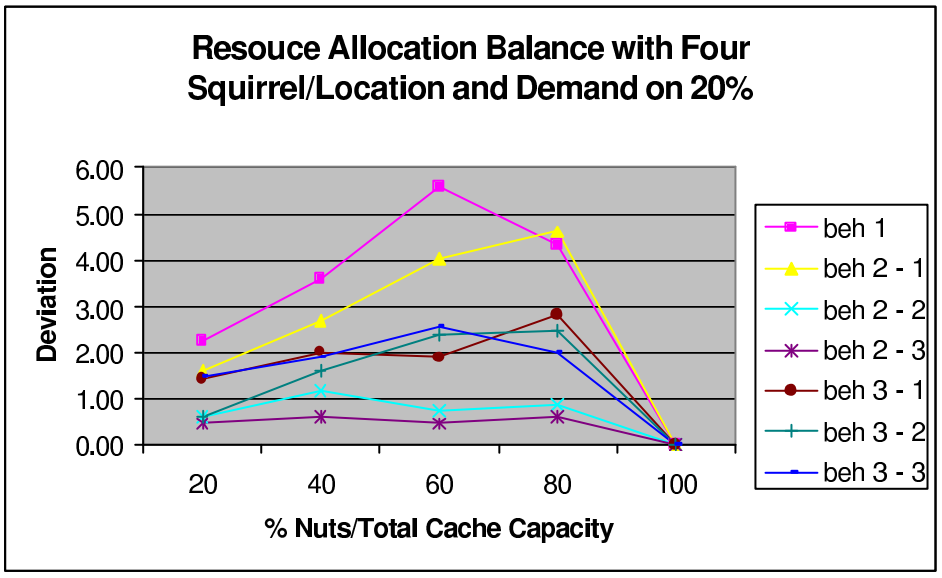

Figure 15: Experiment 4 Deviation Results on 20% Demand and 4 S quirrels/Location

In all experiment 4 scenarios, beh – 2 with 2 or 3 sniffing steps produce a very good resource allocation generally with a deviation of less than 1 unit. This demonstrates that even with a very unbalance demand of resources to allocate n the P2P DFS, the gathering and random sniffing burying algorithm with more than 2 sniffing steps is a good CAS to solve the complex problem of resource allocation.

Experiment 5:

We also ran different scenarios to analyze the steps required to achieve convergence when we have a different number of squirrels per location. Figure 16 and Figure 17 demonstrate the improvement in the gathering and random sniffing burying algorithm when we have a 50% and 20% demand. A substantial reduction in the number of steps is observed in both cases, that together with a well resource allocation balance obtained in previous experiments, proves the squirrel MAS to be good algorithms for resource allocation on a P2P DFS as a smaller number of steps is required as the number of squirrels (agents) increases.

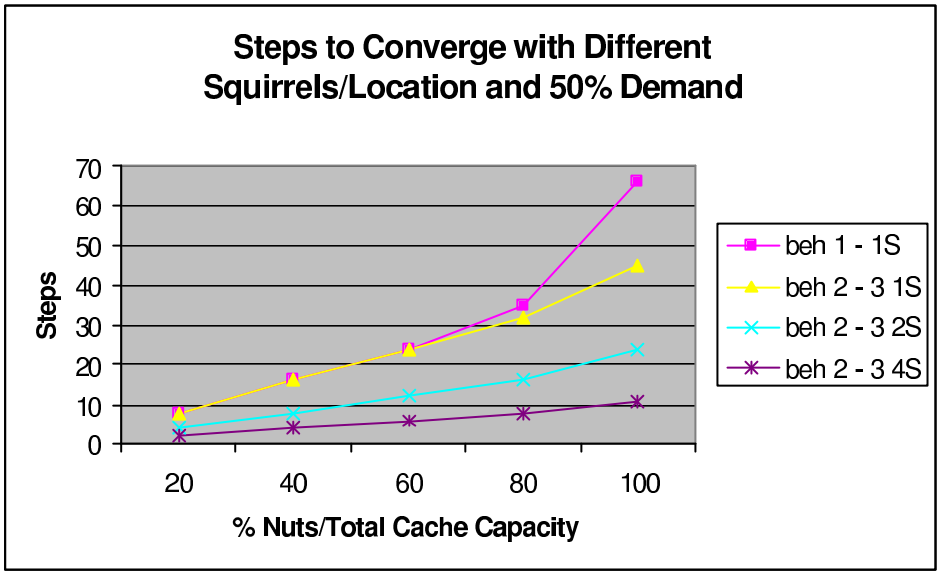

Figure 16: Steps to Converge with 1, 2, and 4 Squirrels/Location and 50% Demand

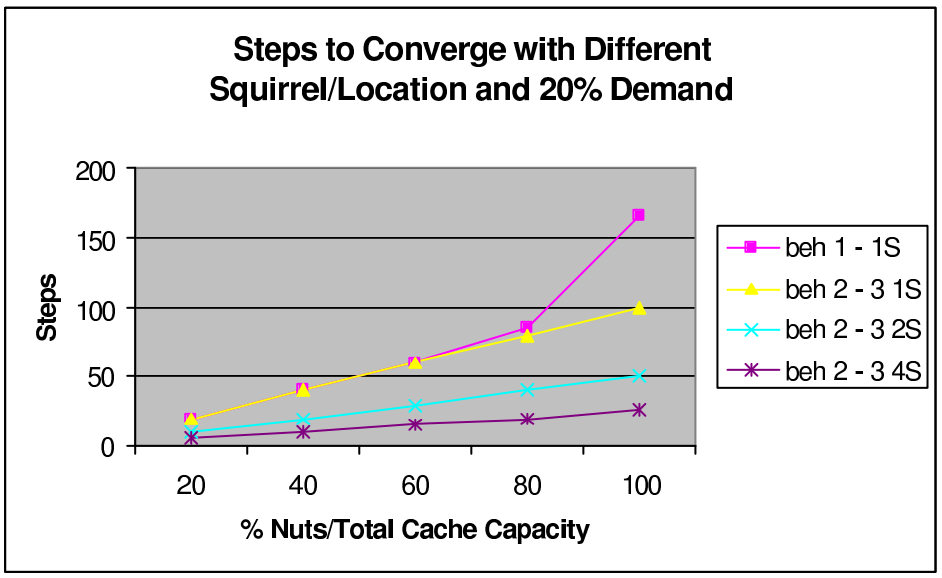

Figure 17: Steps to Converge with 1, 2, and 4 Squirrels/Location and 20% Demand

8. Conclusion

This research analyzes CAS as a model to solve the complex problem of resource allocation in a distributed system. Typical work on resource allocation for distributed systems has followed a traditional centralized or distributed pre-designed technique, which are not flexible or resilient to failures. CAS is inherently flexible and resilient so the approach proves to be high adaptable to changes and resilient to agent or system failures. Consequently, CAS models are suitable for P2P environments where there exists a continuously changing environment with no distributed deterministic algorithm capable of coping with the inherent dynamism.

This research uses CAS ideas to produce emergent global results from the activities of quasi-independent agents. The squirrel's hoarding nature has been studied, exploited, and simulated on a P2P DFS. We base our work on a new CAS metaphor, that of squirrels rather than the previously reported ant metaphor so it produces a truly novel result. Experimental results obtained corroborate our hypothesis that a well balanced storage resource allocation in a DFS across a P2P environment with different demand proportions among the peers can be achieved with a reduction in the number of steps required to converge as the number of agents available increases. This is important because resource allocation balance is maintained even with unbalance resource demands in a reasonable number of steps.

The system is sufficiently flexible that new behaviours can readily be defined and implemented in the model. These can then be incorporated in the simulation tool and tested with only minor changes to analyze its results.

Many opportunities for future work can be identified including:

- Scalability of CAS models on P2P computing models.
- Its application to other areas of resource allocation like operating systems (CPU, I/O devices) and networking (bandwidth, network devices) resources.
- Different CAS behaviours
- Variation of existing parameters with other conditions, for example, having peers with different number of caches, network topologies, etc
- Design of P2P applications (e.g. CPU scheduler, DFS) based on CAS behaviours.

CAS has proven to be a valuable model to be blended together with P2P computing model to produce interesting MAS algorithms that can be leveraged for its application on distributed systems as it has been shown in this work.

9. References

[1] Stoica I, Morris R, Karger D, Kaashoek MF and Balakrishnan H. *"Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications"*, Proceedings of the ACM SIGCOMM 2001 Technical Conference, San Diego, CA, USA, Aug 2001.

[2] Ratnasamy S, Francis P, Handley M, Karp R and Shenker S. *"A Scalable Content Addressable Network"*, Proceedings of the ACM SIGCOMM 2001 Technical Conference, San Diego, CA, USA, Aug 2001.

[3] Zhao B, Kubiatowicz J and Joseph A. *"Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing"*, Technical Report UCB/CSD-01-1141, University of California at Berkeley, Computer Science Dept., 2001.

[4] Druschel P and Rowstron A. *"Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-To-Peer Systems*", Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), November 2001.

[5] Bonabeau E and Theraulaz G. *"Swarm Smarts"*, Scientific American, March 2000

[6] Montresor A, Meling H, Babaoglu O. *"Toward Self-Organizing, Self-Repairing and Resilient Large-Scale Distributed Systems"*, University of Bologna, Italy, Technical Report UBLCS-2002-10, September 2002

[7] Anderson C, Franks N. *"Teams in Animal Societies*", Behavioral Ecology, Volume 12 No. 5: 534-540, September 2000

[8] White T, *"Swarm Intelligence and Problem Solving in Telecommunications"*, Canadian AI Magazine, Spring 1997

[9] Dorigo M, Maniezzo V, Colorni A. *"The Ant System: Optimization by a Colony of Cooperating Agents"*, IEEE Transactions on Systems, Man and Cybernetics, Volume 26, No. 1, 1996

[10] Deadman P, Schlager E, Gimblett R. *"Simulating Common Pool Resource Management Experiments with Adaptive Agents Employing Alternate Communication Routines"*, Journal of Artificial Societies and Social Simulation, Volume 3, No. 2, March 2000

[11] Gibney MA, Jennings NR. *"Dynamic Resource Allocation by Market-Based Routing in Telecommunications Networks*", Proceedings of the 2nd International Workshop on Multiagent Systems and Telecommunications (IATA-98), Paris, France, 1998

[12] Kuwabara K, Ishida T, Nishibe Y, Suda T. *"An Equilibratory Market Based Approach for Distributed Resource Allocation and Its Applications to Communication Network Control"* in Scott H. Clearwater Ed., Market Based Control: A Paradigm for Distributed Resource Allocation, World Scientific, Singapore, 1995.

[13] Chavez A, Moukas A, Maes P. *"Challenger: A Multi-Agent System for Distributed Resource Allocation*", Proceedings of the 1st International Conference on Autonomous Agents, Marina del Rey, CA USA, 1997

[14] Kleijkers S, Wiesman F, Roos N. *"A Mobile Multi-Agent System for Distributed Computing"*, International Conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy, July 2002

[15] Penserini L, Liu L, Mylopoulos J et al. *"Modeling and Evaluating Cooperation Strategies in P2P Systems"*, International Workshop on Agents and P2P Computing (AP2PC 2002), Bologna, Italy, July 2002

[16] Bonabeau E, Dorigo M, Theraulaz G. "Inspiration for Optimization from Social Insect Behaviour", Nature 406, Macmillan Publishers Ltd, 2000

[17] Ward M, *"There's an Ant in my Phone"*, New Scientist, 1998. Available at www.newscientist.com/hottopics/ai/antinmyphone.jsp

[18] Bonsma E, Hoile C. *"A Distributed Implementation of the SWAN Peer-To-Peer Look-Up System Using Mobile Agents"*, International Workshop on Agents and P2P Computing (AP2PC 2002), Bologna, Italy, July 2002

[19] Babaoglu O, Meling H, Montresor A. "Anthill: A Framework for the Development of Agent Based Peer-To-Peer Systems", University of Bologna Technical Report UBLCS-2001-09, September 2002

[20] Montresor A, "Anthill: A Framework for the Design and Analysis of Peer-To-Peer Systems", 6th European Research Seminar on Advances in Distributed Systems (ERSADS 2001), Bertinoro, Italy, May 2001

[21] Montresor A, Meling H, Babaoglu O. "*Messor: Load-Balancing through a Swarm of Autonomous Agents*", Proceedings of the 1st International Workshop on Agents and Peer-To-Peer Computing, Bologna, Italy, July 2002

[22] Shulman A, Nelson E. "Foraging Behavior of Squirrels as a Function of Season Progression From Early Fall Towards Winter", University of the South Ecology, see documents at www.sewanee.edu/biology/journal/2001/squirrels/squirrelbehavior.html

[23] Bureau of Natural Resources, Wildlife Division, "Wildlife in Connecticut: Gray Squirrel", Connecticut Department of Environmental Protection, see documents at http://dep.state.ct.us/burnatr/wildlife/factshts/gsqrl.htm

[24] Skwurls Squirrels Resources, see documents and links at http://skwurls.com