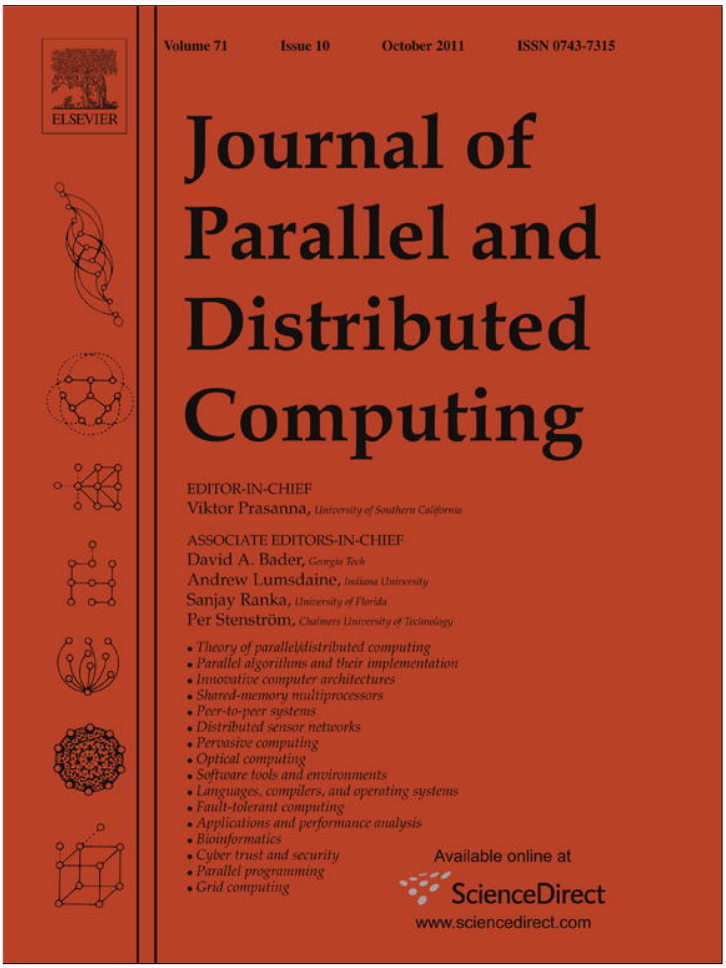


J. Parallel Distrib. Comput. 71 (2011) 1278–1292

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Redundant movements in autonomous mobility: Experimental and theoretical analysis

Natalia Chechina *, Peter King, Phil Trinder

Department of Computer Science, Heriot-Watt University, Edinburgh, EH14 4AS, UK

ARTICLE INFO

Article history:
Received 28 January 2010
Received in revised form
22 June 2011
Accepted 4 July 2011
Available online 13 July 2011

Keywords:
Load balancing
Autonomous mobile program
Mobile agent

ABSTRACT

Distributed load balancers exhibit *thrashing* where tasks are repeatedly moved between locations due to incomplete global load information. This paper shows that systems of autonomous mobile programs (AMPs) exhibit the same behaviour, and identifies two types of redundant movement (greedy effect). AMPs are unusual in that, in place of some external load management system, each AMP periodically recalculates network and program parameters and may independently move to a better execution environment. Load management emerges from the behaviour of collections of AMPs.

The paper explores the extent of greedy effects by simulating collections of AMPs and proposes *negotiating* AMPs (NAMPs) to ameliorate the problem. We present the design of AMPs with a competitive negotiation scheme (cNAMPs), and compare their performance with AMPs by simulation. We establish new properties of balanced networks of AMPs, and use these to provide a theoretical analysis of greedy effects.

1. Introduction

Autonomous mobile programs (AMPs) are mobile agents that improve execution efficiency by managing load; AMPs are aware of their resource needs, sensitive to the execution environment and periodically seek a better location to reduce execution time [11]. To analyse AMP behaviour on local area networks (LANs) we have constructed a simulation model [6]. Comparison of the simulation results and observations of the real system [11] shows that simulated and real AMPs enter the same *stable states* (i.e. where no AMP can reduce its execution time by moving). The differences between simulated and real AMPs are minor and readily explained.

Like other distributed load balancing systems [16,30,33], both real and simulated collections of AMPs exhibit greedy effects. These greedy effects are a phenomenon that results in redundant AMP movements during the balancing of loads between locations, and are a result of locally optimal choices made by each AMP.

This paper is the first substantial investigation of thrashing behaviour for collections of distributed agents, and the results of using an agent-based technique, namely negotiation, to ameliorate them. The paper examines properties and features of the greedy effects in collections of AMPs and exposes their causes. We further aim to reduce greedy effects using negotiation, and to estimate the greedy effects in the modified algorithm. The measurement of the greedy effects is implemented both theoretically and by using simulation.

The paper is organised as follows. We discuss AMP related concepts (Section 2), and identify two forms of greedy effects (Section 3). The AMP cost model and simulation are adapted to facilitate an investigation of the greedy effects. We examine the AMP greedy effects on initial distribution, rebalancing and AMP execution time (Section 4). Analysis of types of movements shows that the majority of redundant movements occur because an AMP is unaware of the intentions and movements of other AMPs. So, we discuss ways to reduce the greedy effects and propose the concept of negotiating AMPs (NAMPs) that communicate their intentions with the view to reduce redundant moves.

While a number of negotiation schemes are possible, we have designed and simulated AMPs with a competitive scheme (cNAMPs) (Section 5). cNAMPs announce their intentions to move and compete with each other for opportunity to transfer to the new location. An analysis of simulated cNAMP results shows that even this simple negotiation significantly decreases both the number of redundant movements and the time to rebalance. For example, cNAMPs make no redundant movements during initial distribution which makes initial balancing in the conducted experiments at least three times faster in comparison with AMPs. We establish the properties of *balanced states*, i.e. states where no AMP can gain a greater AMP relative speed by moving [10] (Section 6), and establish the maximum number of redundant movements after a

* Corresponding author.
E-mail addresses: nc75@hw.ac.uk (N. Chechina), P.J.B.King@hw.ac.uk (P. King), P.W.Trinder@hw.ac.uk (P. Trinder).
URLs: http://www.macs.hw.ac.uk/~nc75 (N. Chechina), http://www.macs.hw.ac.uk/~pjbk (P. King), http://www.macs.hw.ac.uk/~trinder (P. Trinder).

0743-7315/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2011.07.003

cNAMP termination using case analysis (Section 7). A summary of the results and discussion of future work are provided in Section 8. We assemble a set of consistent and precise definitions about the behaviour of collections of AMPs and cNAMPs which appears at the end of the paper (Glossary).

Novelty. The paper goes beyond [7] by making the following additional research contributions:

1. We establish the properties of balanced states to estimate the significance of the greedy effects. We analyse AMPs as the general case and examine a number of properties, such as *independent balance*, *singleton optimality*, and *consecutive optimality* (Section 6).
2. We examine the significance of the greedy effects by predicting the worst case (maximum number) of redundant movements after a cNAMP termination from a network of *q subnetworks* (i.e. sets of locations with identical available speeds) using case analyses (Section 7).
3. We define a consistent terminology for reasoning about the behaviour of AMPs and cNAMPs. Some definitions are generalisations of, or more precise versions of, those used in earlier papers [6,10] (Glossary).

2. Related work

2.1. Introduction

This section discusses previous work, first exploring the three core AMP technologies: load balancing (Section 2.2), mobile computing (Section 2.3) and autonomous systems (Section 2.4), before covering mobile agent load balancing (Section 2.5).

The novelty of the AMP approach is that it automates the performance prediction process using compile-time analysis to identify potential migration points in the code, and devolves to the AMP the decision as to whether to migrate or not as each migration point is executed. Effective load management results from emergent behaviour of collections of AMPs. AMP behaviour has previously been investigated using mobile languages like Java Voyager [12] and using simulation [6].

2.2. Distributed load management

The problem of load balancing in distributed computer systems has been widely studied, e.g. [26,29,33]. The main difficulties which load balancing approaches face are minimising execution time, minimising communication delays, and maximising resource utilisation [34]. According to the taxonomy in [2], AMPs are global dynamic load balancers. According to the detailed classification of dynamic load balancers [32], AMPs have decentralised state estimation scheme that involuntary and periodically collects partial information, *and* decentralised, sender initiated, simple decision making policy. AMPs take all movement decisions themselves, independently of other AMPs. They use the information collected by *load servers* (which only collect network state information and are designed to reduce AMP coordination time) to estimate their remaining execution time if they executed at other locations. If the predicted time to complete at the current location is greater than the predicted time at the best available remote location plus the communication time to reach the remote location, then the AMP will relocate.

The way AMPs search for a better location may seem similar to iterative algorithms [9,25]. Indeed, both approaches aim to distribute load among nodes, and the load is estimated by the number of tasks (or programs) per node. There are two distinctions: firstly in iterative algorithms the locations decide when movement is necessary and which tasks to move, whereas in the AMP approach it is the program itself that decides to move to improve its own performance. Secondly, while in iterative algorithms the locations aim with each iteration to approach some mean load, AMPs seek only to reduce the program execution time.

In general redundant movements are a result of locally optimal choices. To reduce the number of redundant movements different techniques are used, such as limiting any particular task to a maximum number of migrations [17], calculating the largest difference between the estimated execution time and the interprocess communication cost [13], applying market mechanisms [16], etc. These techniques aim to reduce the redundant movements where locations have a task scheduler or management agents, whereas we aim to reduce them for independent AMPs without a scheduler.

2.3. Mobility

Mobile computations can move between locations in a network and potentially enable better use of shared computational resources [22]. Basically a mobile program can transport its state and code to another location in a network, where it resumes execution [24].

Fuggetta et al. distinguish two forms of mobility supported by mobile languages [15]: weak mobility is the ability to move only code from one location to another; strong mobility is the ability to move both code and its current execution state. AMPs were constructed using languages with weak and strong mobility. However, a substantial subset of experiments was conducted using Java Voyager [19] which supports only weak mobility.

2.4. Agents and autonomous systems

Agent technology is a high-level, implementation independent approach to developing software as collections of distinct but interacting entities which cooperate to achieve some common goal [37]. With the continuing decline in price and increase in speed of both processors and networks, it has become feasible to apply agent technology to problems involving cooperation in distributed environments. In particular, where agents may change location, typically to manipulate resources in varying locations [28].

Autonomous systems are also called autonomic computing systems, and a definition has been given by IBM: ‘‘autonomic computing system can manage themselves given high-level objectives from administrators’’ [21]. From four aspects of self-management, i.e. are self-configuration, self-optimisation, self-healing, and self-protection, AMPs are primarily self-optimisation systems.

2.5. Mobile agent load balancing

AMPs differ from other mobile agent systems designed to balance load in that each AMP is both autonomous and self aware, i.e. it knows key information like remaining execution time and program size. Another difference is that the approach does not split a program into subtasks as in [4,20,29]. An AMP is the whole program. Thus, AMPs represent a radical point in distributed decision making when not a location server [1], or a load balancing coordinator [35], or a cluster manager [20] decide where and when to move but each agent itself.

3. Greedy effects

An *optimal rebalancing* is a sequence of AMP movements that is the minimum number of AMP movements needed to enter a stable state.

The AMP *greedy effects* are the result of a non-optimal AMP rebalancing which differs from the optimal rebalancing in having additional redundant movements, and is a result of the AMP making a locally optimal choice, i.e. AMPs do not possess sufficient and accurate state information to make the optimal movement decision. There are two types of the AMP greedy effects: location thrashing and location blindness. Both location thrashing and blindness are observed in real [12] and simulated [6] AMP experiments.

3.1. AMP distribution scenarios

To illustrate the greedy effects we first introduce the following AMP scenarios, each of which specifies the number of AMPs and locations, and types of locations:

Scenario 1: 25 AMPs on 15 locations with CPU speeds 3193 MHz (*Loc*1–*Loc*5), 2167 MHz (*Loc*6–*Loc*10), and 1793 MHz (*Loc*11–*Loc*15).

Scenario 2: 20 AMPs on 10 locations with CPU speeds 3193 MHz (*Loc*1–*Loc*5), 2168 MHz (*Loc*6), and 1793 MHz (*Loc*7–*Loc*10).

Scenario 3: 10 AMPs on 3 locations with CPU speeds 3193 MHz.

The scenarios are the same as were used in real experiments. For all scenarios *Loc*1 is the root location. By the *root location* we mean the location where all AMPs start; it is also called either *initiating location* or *first location* in [11]. In the experiments we use large and small AMPs. Large AMPs are programs of matrix multiplication of size 1000×1000, and small AMPs are programs of matrix multiplication of size 500×500.

3.2. Location thrashing

Location thrashing is the greedy effect resulting from an AMP's lack of information about other AMPs intending to move to the same location. That is, two or more AMPs decide to move on the basis of the same information about the target location which causes further AMP retransmission. Location thrashing occurs in dynamic load balancing systems; other terms are *processor thrashing* [23], *task thrashing* [17], *task dumping* [30,31], *transmitting dilemma* [27].

Location thrashing is illustrated in Fig. 1(a) which shows AMP movements in the experiments with the real system [11] based on Scenario 3 (Section 3.1). In Fig. 1(a) each icon denotes an AMP. The locations are specified on the vertical axis and the horizontal divisions represent time intervals. The time intervals are of different lengths, showing states that the system enters as it attempts to reach a stable state. Note that location thrashing incurs two performance penalties, namely the cost of additional communication and the cost of slower execution. By additional *communication cost* we only mean additional AMP movements during a rebalancing and not communication time they may take.

3.3. Location blindness

Location blindness is the greedy effect resulting from an AMP's lack of precise information about the remaining execution time of other AMPs. The problem is not with poor runtime predictions, but rather an inability to obtain accurate AMP runtime predictions at distributed locations, i.e. the more accurate information that is required, the more *expensive* it becomes to collect and process the information in a distributed system [3].

Fig. 2 shows an example of location blindness in some simulated AMP experiments. The experiment is undertaken on the basis of Scenario 1 (Section 3.1). Numbers identify the number of AMPs on a location. After an AMP termination on *Loc*14 in state S1 and the system enters state U1 an AMP from *Loc*5 discovers the opportunity for faster execution first and moves (state U2). Then an AMP from *Loc*7 discovers the opportunity for faster execution on *Loc*5 and also moves (state S2). In contrast to the location thrashing, location blindness only causes redundant communication and causes no additional computation cost. Each AMP will have reduced its execution time by moving. Thus, among two types of the AMP greedy effects the location thrashing is more harmful, because it causes an increase in AMP execution time, and hence decreases AMP efficiency.

4. Simulating AMP greedy effects

Earlier simulation experiments of AMPs on a LAN showed that the simulation closely models real AMPs on LANs, and is an effective tool to analyse AMP behaviour [6]. We provide essential calculations and discuss properties and features of AMPs in Section 4.1. To simulate the greedy effects some changes to the previous simulation model [6] have been proposed, such as bounds on information transfer times and delays on the transfer of state information [5]. We further discuss greedy effect experiments and investigate the redundant movement causes in Section 4.2.

4.1. Cost model and parameterisation

The key equations of the AMP cost model defined in [10] are repeated here, and the parameter values are given in Table 1.

A *homogeneous network* is a set of locations with the same available speeds, except the root location which may be different, because of the overheads of initiating the remote processes. A *heterogeneous network* is a set of locations with different available speeds.

Available speed is the execution speed of a single AMP on a location, i.e. $S = (\textit{CPU speed}) \cdot (1 - \textit{non AMP load})$, is used to differentiate the total resources of a location from the resources available for AMPs. The current research, like the previous AMP investigations [7,11,12], assumes that all resources of a location are available for AMPs, and the CPU speed coincides with the available speed, except the case of the root location where the external load is higher.

An *AMP relative speed*, R, is available speed equally divided between the AMPs at the location, x_{loc}, i.e.[1]

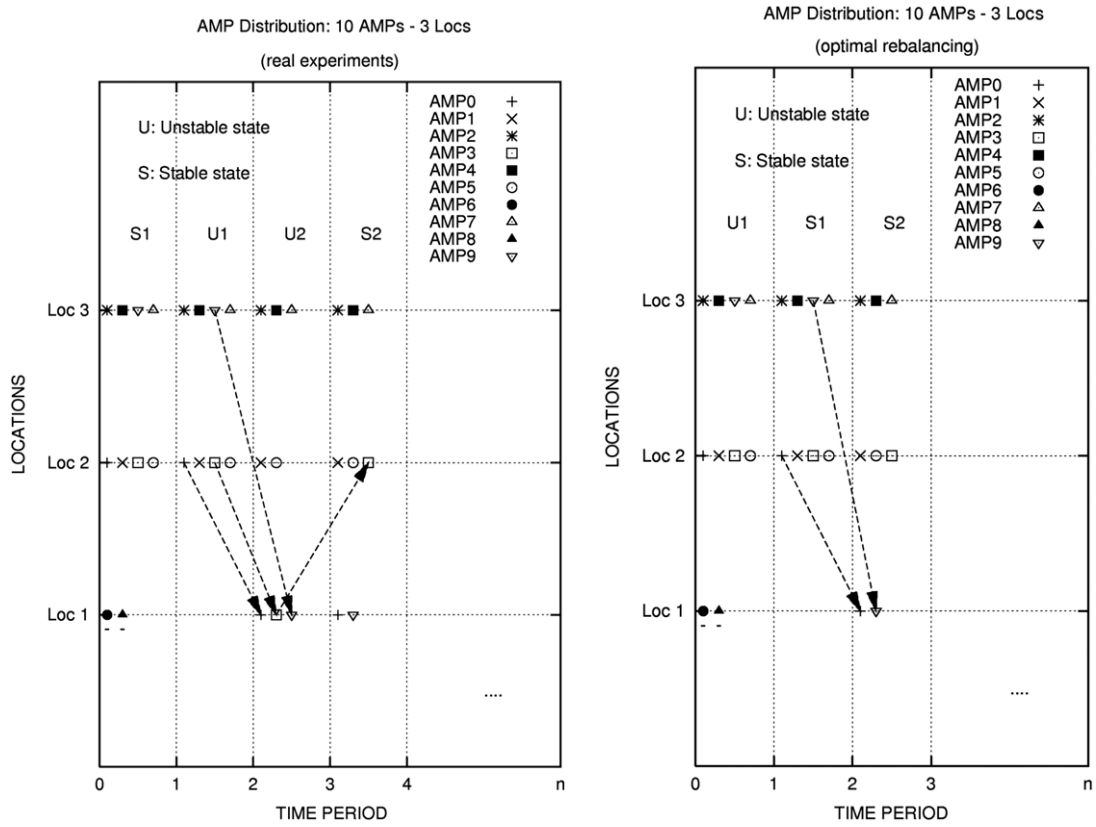
$$R = \frac{S}{x_{loc}}. \tag{1}$$

An AMP executes for a time $T_{gran} = \frac{T_{coord}}{O}$ before it tests the relative speeds of other locations to see if a move will improve its completion time. The time parameters for our simulation are taken from Java Voyager AMP measurements on a LAN [10], as validated in [6]. The coordination time, T_{coord}, is determined experimentally to be 0.011s for a load server architecture. The overhead, O, is 5%.

After executing for T_{gran} an AMP makes a request to the load server of the current location about states of other locations in the network. If an AMP decides to stay on the current location, then it continues execution for a further T_{gran} seconds, otherwise it moves to a new location taking $T_{send} = 0.029 + 5.07 \cdot 10^{-6} \cdot d^2$ seconds. The AMP studied here performs square matrix multiplication and d is the dimension of the matrix. Experiments with other programs, e.g. coin counting or ray tracing, show consistent behaviour, and in the following we use matrix multiplication.

As soon as an AMP makes a decision to move to a new location, the load server of the current location decreases its number of AMPs. In turn, the new location increases its number of AMPs after

[1] In [11] the *available speed* is called *relative speed*, and the *AMP relative speed* is called *average relative speed*.



(a) Redundant rebalancing.

(b) Optimal rebalancing.

Fig. 1. Location thrashing greedy effect [12].

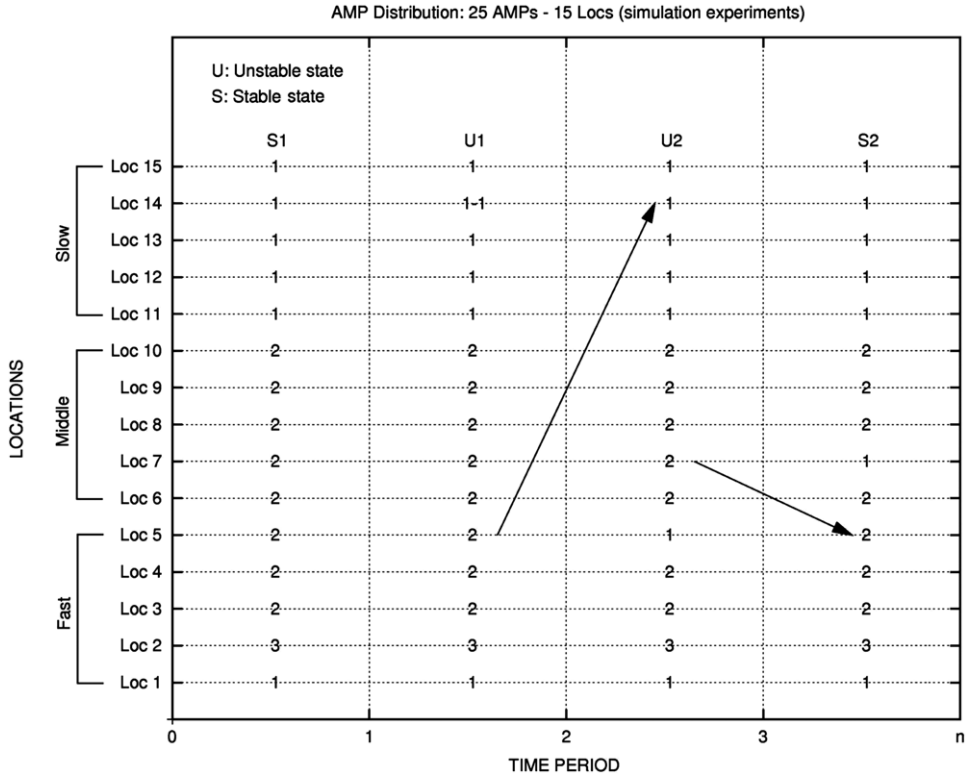


Fig. 2. Location blindness greedy effect [6].

Table 1
Parameter definitions.

d	Dimension of square matrix
$gran$	Fragment of work which must be executed between searches for a better location
N	Number of locations in a network
O	Overhead
R	AMP relative speed
S	Available speed
T_{comm}	Time for single communication
T_{coord}	Coordination time in the load server architecture
T_{gran}	Execution time of fragment of work $gran$
T_h	Execution time on the current location
T_n	Execution time on the new location
T_{renew}	Time of total state information renewing by load server
T_{req}	Time to send a request to a remote load server and receive a response
T_{send}	Time to transmit an AMP to the new location
x_{loc}	Number of AMPs on a location

the AMP has completely arrived and is ready to execute. To obtain state information from other locations in a network, a load server sends requests to locations in a sequential order following [10]. T_{req}, the time taken to send a request to a remote Java process and receive a response has been measured using Java Voyager, and is equal to 0.25s [10, p. 80]. Thus, a load server completely renews state information about the remaining $N-1$ locations in the network every $T_{renew} = T_{req}(N-1)$ seconds.

The main rule on the basis of which AMPs make a decision to move to a new location is whether execution time on the current location, T_h, exceeds execution time on the next location, T_n, and communication delay, T_{comm}:

$$T_h > T_n + T_{comm}. \tag{2}$$

If condition (2) is satisfied, then an AMP moves. Here, communication time is time to transfer an AMP, i.e. $T_{comm} = T_{send}$.

4.2. Analysing greedy effects

The analysis investigates frequency and significance of greedy effects in the three scenarios from Section 3.1 on homogeneous and heterogeneous networks. We further classify the redundant movements and identify the primary cause.

Experiment 1 *(E*1*): Initial distribution.* This experiment investigates the greedy effects as large AMPs distribute over the network from a single location in each scenario. Column *Initial distribution* in Table 2 shows the mean number of redundant movements and the time required for the system to enter a stable state.

Experiment 2 *(E*2*): Rebalancing after an AMP termination.* The experiment measures the number of movements and time required for a system to rebalance after an AMP termination in each scenario. Column *Rebalancing after an AMP termination* in Table 2 shows the mean number of redundant movements and rebalancing time.

Experiment 3 *(E*3*): Large AMP execution time.* This experiment estimates large AMP execution time and measures its variability in each scenario. The total number of AMPs corresponds to the relevant scenario. All AMPs, two of which are small and the rest are large, start on the root location. The results are presented in column *Large AMP execution time* in Table 2.

Fig. 3 shows the initial AMP distribution between locations as the system rebalances from initial (unstable) state U1 to stable state S. As all AMPs move from *Loc*1 in state U1 to *Loc*2–*Loc*5 in state U2, we do not indicate them with lines. There are 88 movements in total, but there would only be 24, if each AMP moved directly to the location it reaches in state S, i.e. the system makes 64 redundant movements.

The analysis of AMP movements allows them to be classified into the following main types of redundant movements: (a) two or more AMPs move from one location to another, and then some of them rebalance; (b) two or more AMPs move from a location, and then some AMPs move back to the location; (c) two or more AMPs move from different locations to one location, and then some of them immediately move again. Therefore, we conclude that *redundant AMP movements are mainly caused by AMP ignorance of intentions and actions of other AMPs in the network and, hence, lack of information to make an efficient decision*, i.e. location thrashing.

5. Negotiating AMPs and cNAMPs

As the main reason for poor AMP movement decisions is a lack of communication between AMPs via the load servers, we propose to use negotiation to reduce the greedy effects. Possible methods of negotiation between AMPs and/or load servers are discussed in Section 5.1. We introduce a simple negotiation scheme in which AMPs announce their movement intentions and compete for opportunity to transfer. AMPs using this scheme are called cNAMPs. The modifications to AMP algorithm to provide this behaviour are described in Section 5.2. The comparative cNAMP and AMP performance is presented in Section 5.3.

5.1. Negotiating AMPs

The analysis of the greedy effects in the simulated AMP experiments in Section 4.2 shows that the majority of redundant movements occur because an AMP makes a decision on the basis of currently available information and is unaware of impending movements of other AMPs.

To understand AMPs better we draw an analogy between AMPs and human society where an AMP acts as an individual. Then AMP behaviour corresponds to *autistic* behaviour. Such an AMP does not request information about other AMPs, does not provide information itself, and does not communicate with other locations to make efficient movement decision. AMPs only rely on the current information and are not concerned about actions of other AMPs on the same information about the target location.

In order to reduce the greedy effects AMPs must negotiate with each other, i.e. communicate more information. There can be different types of negotiation, such as malicious, honest, etc. A *malicious* strategy would be for a load server to misrepresent the load so that other AMPs were deterred from moving to a location. An *honest* strategy requires AMPs and load servers to share information to reduce wasted movements and is more effective for load balancing. A range of honest AMP negotiation tactics are possible, e.g. *competitive* where AMPs compete with each other to move to the target location; *queueing* where each AMP has a sequence number in a queue, and moves to the new location only if an earlier AMP in the queue rejected to move; *probabilistic* where an AMP makes a decision to move on the basis of calculating the probability of simultaneous AMP movements from other locations; *relationship* where a network is logically divided into groups, locations first share information within their group, a location can be a member of more than one group, thus information is spread like a rumour.

5.2. The design of cNAMPs

cNAMPs are negotiating AMPs with a competitive scheme which announce their intentions to move and compete with each other for opportunity to transfer to the new location. In the context of cNAMPs the negotiation is a simple coordination among competitive and self-interested agents [36]. cNAMPs do not negotiate directly with each other, but only by means of a load server.

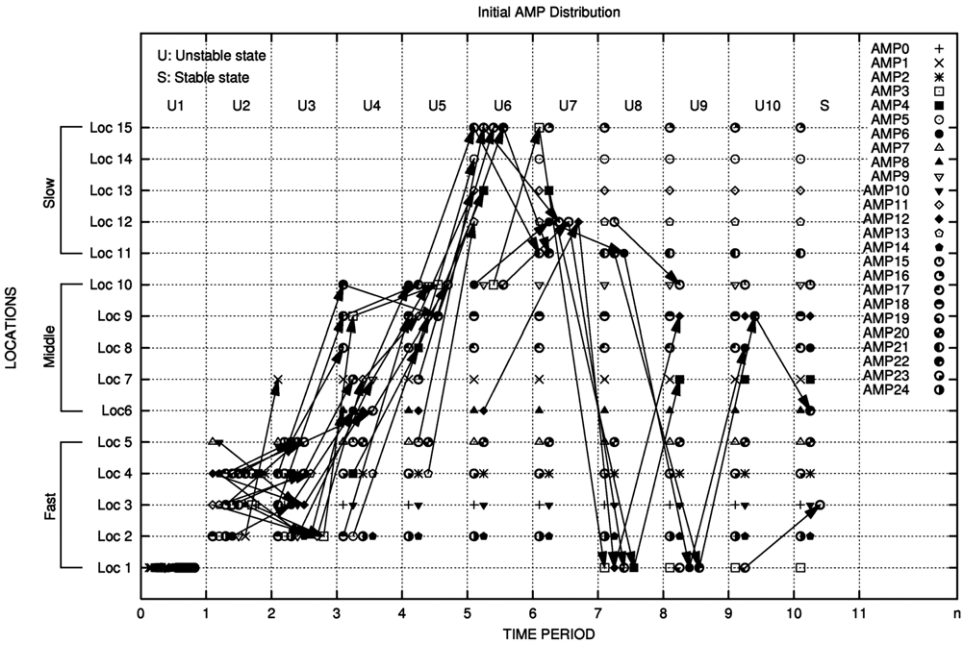


Fig. 3. AMP movements during initial distribution (Scenario 1).

Table 2
AMP greedy effect experiment summary.

Config.	E1: Initial distribution		E2: Rebalancing after an AMP termination		E3: Large AMP execution time (s)	
	Mean No. of redun. moves	Mean time (s)	Mean No. of redun. moves	Mean time (s)	Mean	Standard deviation
Scenario 1	64	60.4	6	22.5	173.8	7.66
Scenario 2	43	50.5	11	28.2	182.1	11.5
Scenario 3	13	26.8	6	14.1	232.6	9.91

Table 3
cNAMP and load server pseudocode.

cNAMP	Load server

`while work remains to execute`
`{if outstanding request & positive response`
`{inform local load server about movement`
`move to target location }`
`else if no local cNAMP awaits a response`
`{for` n `from 1 to total number of loc-s`
`find` *minimum* `of` $T_n + T_{comm}$
`if` $T_h >$ *minimum*
`{send request to` L_n
`inform local load server about`
`request sent } }`
`continue execution }`
	`forever do`
	`case local cNAMP sent a request to` Loc_i`:`
	`lock information about` Loc_i
	`case local cNAMP received response from` Loc_i`:`
	`{renew and unlock information about` Loc_i
	`if positive response`
	`reduce actual and committed loads }`
	`case arrival notification from remote cNAMP:`
	`increase committed load`
	`case cNAMP arrived:`
	`increase actual load`

cNAMPs are designed only to reduce location thrashing. Eradicating location thrashing eliminates redundant movements during initial distribution and significantly reduces the number of redundant movements during rebalancing (Table 4 in Section 5.3). In addition, reduction of location blindness requires that cNAMPs and load servers possess even more information about locations and cNAMPs of the network. In Section 7 we show that both the probability of redundant movements and their number are very small.

Unlike an AMP, a cNAMP first sends a request to the selected target location declaring the intentions to move. The request is also an agent which can be seen as a cNAMP representative. On arrival to the target location the representative recalculates parameters, informs the target load server if the decision is positive and moves back. Only if the representative confirms the movement decision does the cNAMP actually move; otherwise the cNAMP continues execution locally. When a request informs the target load server about a cNAMP movement the load server reports the load as if the transferring cNAMP had already arrived. Each load server maintains two values for the load: (a) the *actual load* which is the number of executing cNAMPs and is used for local cNAMP calculations; (b) the *committed load* which represents the actual load of a location together with the cNAMPs that confirmed their transferring to the location, and is used by remote load servers. Pseudocode for cNAMP and load server implementations are presented in Table 3. The way cNAMPs confirm a movement decision with the target location has some small similarities to a two-phase commit protocol [18].

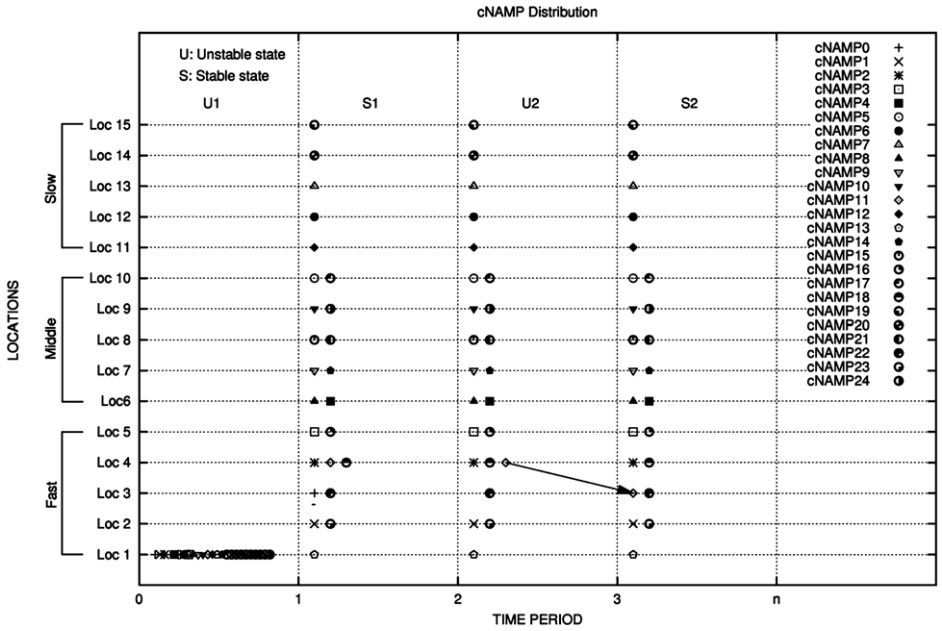


Fig. 4. Initial distribution and rebalancing after a cNAMP termination (Scenario 1).

Table 4
Comparative summary of AMP and cNAMP greedy effects.

Configuration and type of experiment	Initial distribution		Rebalancing after an AMP/cNAMP termination		Large AMP/cNAMP execution time (s)	
	Time (s)	Mean number of redundant movements	Time (s)	Mean number of redundant movements	Mean	Standard deviation
Scenario 1						
AMPs	60.4	64	22.5	6	173.8	7.66
cNAMPs	14.7	–	5.9	–	104.8	12.9
Reduction	4.11	64 moves	3.81	6 moves	1.65	
Scenario 2						
AMPs	50.5	43	28.2	11	182.1	11.5
cNAMPs	12.4	–	7.8	1	113.6	9.43
Reduction	4.07	43 moves	3.62	10 moves	1.6	
Scenario 3						
AMPs	26.8	13	14.1	6	232.6	9.91
cNAMPs	8.5	–	5.6	–	142.2	4.97
Reduction	3.15	13 moves	2.52	6 moves	1.64	

5.3. Comparative cNAMP and AMP performance

We compare the greedy effects exhibited by AMPs and cNAMPs in Table 4 using the experiment design presented in Section 3.1. For each scenario the first and the second rows show results of AMP and cNAMP experiments respectively.

The results show that even simple negotiation in cNAMPs significantly reduces the number of movements and time to rebalance. During rebalancing after an AMP/cNAMP termination, cNAMPs make far fewer redundant movements. cNAMPs require less execution time than AMPs. Fig. 4 shows initial cNAMP distribution and system rebalancing after a cNAMP termination. Arrows show cNAMP movements, and as before we do not show the cNAMP movements from state U1 to state S1. Fig. 4 should be compared with Fig. 3 in Section 4.2. More examples of cNAMP movements are provided in [8]. The results show that cNAMPs only display location blindness (Section 3) which does not increase cNAMP execution time.

Table 5 presents the overhead of cNAMPs compared with AMPs, and we see that although cNAMPs send more messages (columns 2 and 6), most of them are small request/response messages. Therefore, overall AMP communication is larger than overall cNAMP communication (columns 3 and 9). As a consequence cNAMPs execute faster than AMPs (column 7 in Table 4).

6. Properties of balanced networks

To analyse the significance of greedy effects we establish the properties of balanced states. The properties and definitions developed in this section are essential for the proofs in Section 7, and are summarised in the Glossary. Note that the properties of balanced states apply to all AMPs, not only cNAMPs. The balanced state are investigated using a balanced state checker, i.e. a special program to explore the state space the system enters. The balanced state checker uses identical logic to the simulated and to real AMPs. Hence, it is not surprising that the predicted balanced states exactly reproduce simulated results for all scenarios considered. A homogeneous network is analysed as a special case of a heterogeneous network where the root location is taken as a *singleton subnetwork*, i.e. a subnetwork with one location.

There is a similarity between AMP's balanced states and Nash equilibria [14]. Participants in both Nash equilibrium and AMPs aim to get as much *profit* as possible. However, in game theory participants have a choice to play cooperatively or not, whereas

Table 5
AMP/cNAMP and request/response messages.

Conf.	AMPs		cNAMPs					
	No. Moves	Size (Mb)	No. Req/ Resp	No. Moves cNAMPs	Total No. Msg	Req/Resp Size (Mb)	cNAMP Size (Mb)	Total Size (Mb)
Sc. 1	106	954	78	35	113	36	315	351
Sc. 2	77	693	66	28	94	30	252	282
Sc. 3	28	252	26	11	37	12	99	111

AMPs/cNAMPs have no such choice. Though cNAMPs communicate more information than AMPs, cNAMPs are still *selfish*, and they never *care* about others' profit but only about decreasing their own execution time. Another difference is that AMPs/cNAMPs do not predict or estimate other AMP/cNAMP behaviour, i.e. as soon as an AMP/cNAMP finds a location where it can reduce its execution time it moves.

We first discuss empirical balanced state checker in Section 6.1 which is used to investigate balanced states, then present the *independent balance property* in Section 6.2, and discuss optimal and near-optimal balanced states in Sections 6.3 and 6.4 respectively. Furthermore, we characterise balanced states for homogeneous networks in Section 6.5 and heterogeneous networks in Section 6.6.

6.1. Balanced state checker

The properties of balanced states were investigated using a program to explore the states entered as AMPs are added to the system one by one. The program assumes that communication time is negligibly small relative to computation time, i.e. $T_{comm} \ll T_{comp}$. The sequence of stable states that are entered are all balanced.

The algorithm of the balanced state checker is as follows. To allocate a new AMP when a distribution of x AMPs is given ($0 \leqslant x \leqslant \infty$), we add one AMP to each location. Then AMP relative speeds are calculated, and one AMP is removed from each location except those with the highest AMP relative speed. The resulting distribution is the distribution $x+1$ AMPs where this one AMP can be on one of locations with the highest AMP relative speed. More detailed description and pseudocode are given in [5].

6.2. Independent balance property

The analysis of balanced states shows the following property.

Property 1 (*Independent Balance*)**.** *For a balanced state, the relationship between the number of AMPs x_i and x_j on locations in any two subnetworks i and j is independent of the number of locations in those subnetworks and independent of the presence or absence of other subnetworks, subject only to the sum $x = x_i + x_j$ being constant. The only exception to this rule is the case when distribution of $x' = x_i + x_j + 1$ AMPs results in all x' AMPs having the same relative speed. In this case the partition may have two variants.*

Independent balance property holds in all scenarios investigated and can be observed in the following experiments.

Experiment 1

(a) Distribute 11 AMPs over a heterogeneous network of two singleton subnetworks. Without loss of generality, assume that the distribution places 8 AMPs on the location of subnetwork 1 and 3 AMPs on the location of subnetwork 2 as Table 6(a) shows.

(b) Add further singleton subnetworks to the system and also add enough AMPs to the new system so that balanced state is achieved and there are 11 AMPs distributed between singleton subnetworks 1 and 2. The distribution between singleton subnetworks 1 and 2 will *always* be 8 AMPs in subnetwork 1 and 3 AMPs in subnetwork 2 (Table 6(b)). It will *never* be distributed 7 AMPs in subnetwork 1 and 4 AMPs in subnetwork 2, or 9 AMPs in subnetwork 1 and 2 AMPs in subnetwork 2.

(c) Adding any number of locations to a subnetwork will not change the distribution between a pair of locations from subnetworks 1 and 2 so long as the system is in a balanced state and a sum of AMPs in the pair of locations from subnetworks 1 and 2 is 11 (Table 6(c)).

Table 6
AMP distribution in a pair of subnetworks for a given sum of AMPs.

Available speed of locations	Number of AMPs
(a) Scenario A	
S_1	8
S_2	3
(b) Scenario B	
S_1	8
S_2	3
S_i	x_i
(c) Scenario C	
S_1	8
	8
	8
S_2	3
	3
S_i	x_i

Experiment 2

Assume that distribution of 12 AMPs between subnetworks 1 and 2 *would* result in all AMPs having the same relative speed, and the distribution would be 8 AMPs in subnetwork 1 and 4 AMPs in subnetwork 2. Then distribution of 11 AMPs has two alternatives: 8 AMPs in subnetwork 1 and 3 AMPs in subnetwork 2, *and* 7 AMPs in subnetwork 1 and 4 AMPs in subnetwork 2. The distribution of 11 AMPs may result only in these two partitions independently of presence or absence of other locations and subnetworks.

Let there be two subnetworks, with the available speeds S_1 and S_2, and let each location of subnetworks 1 and 2 have x_1 and x_2 AMPs in a balanced state respectively. Then the following inequalities hold in *any* balanced state:

$$\begin{cases} \dfrac{S_1}{x_1} \geqslant \dfrac{S_2}{x_2 + 1} \\ \dfrac{S_2}{x_2} \geqslant \dfrac{S_1}{x_1 + 1}. \end{cases} \tag{3}$$

We distinguish two balanced states which a system can enter: optimal and near-optimal balanced states. In Sections 6.3 and 6.4 we investigate their properties and generalise some of the definitions given in [10].

6.3. Optimal balance

An *optimal balanced state* is a state when locations with the same available speed have equal numbers of AMPs. The total number of AMPs x in an optimally balanced network with q subnetworks is: $x = \sum_{i=1}^{q} x_i N_i$, where there are N_i locations in subnetwork i and each location has x_i AMPs.

Property 2 (*Singleton Optimality*)**.** *All balanced states which a network of singleton subnetworks enters are optimally balanced.*

The *singleton optimal property* is a direct corollary of the optimal balanced state definition. Furthermore, from optimal balanced state definition and independent balance property it follows that:

(a) finding optimal balance states of arbitrary heterogeneous networks only requires the finding of the optimal balance states for networks with the same number of singleton subnetworks:

$$k = \sum_{i=1}^{q} x_i, \quad k \in [1; +\infty). \tag{4}$$

(b) Any optimally balanced network is a composition of optimally balanced pairs of subnetworks.

Solving inequations (3) for a heterogeneous network of two subnetworks with available speeds S_1 and S_2, and $k = x_1 + x_2$, each location has the following number of AMPs:

$$x_i \approx \frac{S_i(k+1)}{S_1+S_2} - \frac{1}{2}, \quad i = 1, 2. \tag{5}$$

Here, by $\approx$ we mean rounding to the nearest value. If the fraction part of x_i is exactly .5, then either x_i is rounded up and x_j is rounded down, *or* x_i is rounded down and x_j is rounded up. To indicate the rounding to the nearest value in the paper we use double square brackets, [[]]. These brackets also imply that both adjacent integer values should be considered if the fractional part of the contents is exactly .5.

This analysis of balanced and optimal balanced states enables testing of an assumption made in [10], i.e. it is said that AMPs tend to have equal AMP relative speed in optimal balance. However, the observations show the following results.

Lemma 3. *An AMP relative speed on a faster location tends to be slightly lower than the mean AMP relative speed; and an AMP relative speed on a slower location tends to be slightly higher than the mean AMP relative speed.*

Proof. This can be observed from the following analysis. First, it is important to recall that though a system is equilibrium in an optimal balanced state, AMP relative speeds on locations from different subnetworks are not the same, because the number of AMPs on a location is integer.

According to the assumption in [10], i.e. $\frac{S_1}{x_1} \approx \frac{S_2}{x_2}$, and the optimal balance property a location of subnetwork 1 must have $[[\frac{S_1 k}{S_1+S_2}]]$ AMPs. Hence, AMP relative speed on the basis of (1) must be

$$R_1' = \frac{S_1}{\left[\left[\frac{S_1 k}{S_1+S_2}\right]\right]}. \tag{6}$$

However, the number of AMPs on a location is given by (5), and an AMP relative speed on a location of subnetwork 1 is

$$R_1 = \frac{S_1}{\left[\left[\frac{S_1 k}{S_1+S_2} + \frac{S_1-S_2}{2(S_1+S_2)}\right]\right]}. \tag{7}$$

As $S_1 > S_2 > 0$ and $\frac{S_1-S_2}{S_1+S_2} < 1$, then $0 < \frac{S_1-S_2}{2(S_1+S_2)} < 0.5$. Thus, $\frac{S_1-S_2}{2(S_1+S_2)}$ increases the number of AMPs assumed in [10] by nought or one during the rounding. Therefore, an AMP relative speed on a location of subnetwork 1 either coincides with the AMP relative speed assumed in [10] or is slightly slower. The same analysis sequence is made to investigate AMP relative speed in subnetwork 2.

The difference between AMP relative speeds on locations from different subnetworks is less when the difference in the available speeds of locations is small; and it decreases with the increase of the total number of AMPs. □

6.4. Near-optimal balance

For any optimal balanced state, the *optimal number of AMPs* for a subnetwork is the number of AMPs on each location in the subnetwork. *Nearest upper (lower) optimal balanced state* is the optimal balanced state which the system enters by adding (removing) the minimum number of AMPs.

We define a *near-optimal balanced* network be the network where some subnetworks have near-optimal number of AMPs. The locations of these underloaded subnetworks have either the optimal number of AMPs or one less than the optimal number. The underloaded subnetworks are determined by being the subnetworks with the slowest AMP relative speed in the nearest upper optimal balanced state.

Property 4 (*Consecutive Optimality*)**.** *If a system with a total of x AMPs is optimally balanced, and the subnetwork with the highest AMP relative speed is a singleton, then the system with a total of $x+1$ AMPs is also optimally balanced.*

The consecutive optimal property is derived from the near-optimal balance definition as a corollary.

From the independent balance property (Section 6.2) we conclude that *a system has no balanced states other than optimal and near-optimal balanced states*. That is, only subnetworks that have the slowest AMP relative speed in the nearest upper optimal balanced state can be in a near-optimal balanced state. If other networks were simultaneously in a near-optimal balance state, then there would be an immediate transfer of AMPs from the slower to the faster subnetwork (in terms of AMP relative speed).

As the near-optimally balanced subnetworks are determined for each optimally balanced state, we can identify a subnetwork which may change its number of AMPs when we add/remove an AMP:

(a) when an AMP is *added* to an optimally balanced network, the number of AMPs in the subnetwork which would have the highest AMP relative speed if one AMP is added to each location increases by one.

(b) when an AMP is *removed* from an optimally balanced network, the number of AMPs in the subnetwork with the slowest AMP relative speed decreases by one.

6.5. Characterising balanced states in a homogeneous network

Let a homogeneous network have N locations, the available speed of non-root locations be S. As only a part of the root location capacity is available for an AMP execution, let the load factor, $0 < f < 1$, define available resources for AMPs on the root location, i.e. available speed of the root location is $f \cdot S$.

In an *optimal balanced state* on the basis of (5) the system has the following numbers of AMPs on the root, x_{rt}, and non-root, x_{nrt}, locations:

$$x_{rt} \approx \frac{f(2k+1)-1}{2(f+1)}, \tag{8}$$

$$x_{nrt} \approx \frac{2k-f+1}{2(f+1)}. \tag{9}$$

In a *near-optimal balanced state* the root location has $[[\frac{f(2k+1)-1}{2(f+1)}]]$ AMPs and non-root locations have either $[[\frac{2k-f+1}{2(f+1)}]]$ or $[[\frac{2k-3f-1}{2(f+1)}]]$ AMPs. For further distinguishing non-root locations with a different number of AMPs, those which have an optimal number of AMPs, i.e. $[[\frac{2k-f+1}{2(f+1)}]]$, we call *heavy* locations, and those which have one or two AMPs less than a heavy location, i.e. $[[\frac{2k-3f-1}{2(f+1)}]]$ and $[[\frac{2k-5f-3}{2(f+1)}]]$, we call *light* and *very light* locations respectively.

6.6. Characterising balanced states in a heterogeneous network

Let $k_{i,j}$ be the sum of AMPs in singleton subnetworks i and j, i.e.

$$k_{i,j} = x_i + x_j, \quad i, j = 1, 2, \ldots, q,\ i < j. \tag{10}$$

To calculate AMP distribution in optimally balanced heterogeneous networks the following algorithm is used.

(a) We calculate $k_{i,j}$ for all i and j denoted in (10) using the following equation:

$$k_{i,j} = \left[\left[\frac{(2k+q)(S_i+S_j)}{2\sum_{m=1}^{q} S_m} - 1\right]\right]. \tag{11}$$

(b) Then for each $k_{i,j}$ we find x_i and x_j using (5), i.e.

$$\begin{cases} x_i = \left[\left[\dfrac{S_i(k_{i,j}+1)}{S_i+S_j} - \dfrac{1}{2}\right]\right], \\ x_j = \left[\left[\dfrac{S_j(k_{i,j}+1)}{S_i+S_j} - \dfrac{1}{2}\right]\right]. \end{cases} \tag{12}$$

(c) Using results from (12) we make a list of all possible distributions, and delete distributions where $k \neq \sum_{m=1}^{q} x_m$.

(d) In the remaining distributions we check inequality (3). The inequality is strictly larger for all pairs i and j, i.e.

$$\frac{S_i}{x_i} > \frac{S_j}{x_j + 1}, \quad i, j = 1, 2, \ldots, q. \tag{13}$$

The only exception is for pairs where x_i and x_j result in .5. In this case (3) is as follows:

$$\begin{cases} \dfrac{S_i}{\lceil x_i \rceil} = \dfrac{S_j}{\lfloor x_j \rfloor + 1}, \\ \dfrac{S_j}{\lceil x_j \rceil} = \dfrac{S_i}{\lfloor x_i \rfloor + 1}. \end{cases} \tag{14}$$

These distributions are the only distributions of k AMPs on the network of q singleton subnetworks. According to discussion in Section 6.4, numbers of AMPs on locations in a *near-optimal balanced state* coincides with the numbers in the nearest upper optimal balanced state, except the subnetworks with the slowest AMP relative speed. Locations of these subnetworks have number of AMPs given in the algorithm or one AMP less.

7. cNAMP greedy effect analysis

This section examines the cost of cNAMP location blindness, by predicting the maximum number of redundant movements in homogeneous networks (Section 7.1) and heterogeneous networks (Section 7.2). To estimate the number of redundant movements, we analyse the conditions under which cNAMPs transfer. The components of cNAMP cost model are as follows:

$$T_h = \frac{W_r}{R_h}, \tag{15}$$

$$T_n = \frac{W_r}{R_n}. \tag{16}$$

Fig. 5 shows the cNAMP cost model parameters. A network with AMPs, and hence with cNAMPs, enters one of two balanced states: optimal and near-optimal balance. We analyse the maximum number of movements and its probability after a cNAMP termination from optimally and near-optimally balanced homogeneous and heterogeneous networks.

f - the load factor on the root location;
R_h - the current location relative speed;
R_n - the new location relative speed.
T_h - execution time on the current location;
T_n - execution time on the new location;
W_r - the work remaining;
x - the total number of cNAMPs;
x_{nrt} - the number of cNAMPs on a non-root location;
x_{rt} - the number of cNAMPs on the root location;

Fig. 5. cNAMP cost model parameters.

7.1. Homogeneous network

cNAMP termination in an optimally balanced network

From (8) and (9), in an optimal balanced state the root location has $[[\frac{f(2k+1)-1}{2(f+1)}]]$ cNAMPs and every non-root location has $[[\frac{2k-f+1}{2(f+1)}]]$ cNAMPs.

Theorem 5. *There is no greedy effect when a cNAMP terminates in an optimally balanced homogeneous network.*

Proof. The proof proceeds by case analysis on the location where termination occurs, and where the first movement is initiated.

(a) *Termination at a Non-root Location.* From (8) and (9) after the cNAMP termination, the system has $[[\frac{f(2k+1)-1}{2(f+1)}]]$ cNAMPs on the root location, $[[\frac{2k-3f-1}{2(f+1)}]]$ cNAMPs on the non-root location where termination occurred (the light location), and $[[\frac{2k-f+1}{2(f+1)}]]$ cNAMPs on the remaining non-root locations (heavy locations). Hence, the system is in the near-optimal balanced state. We analyse possible movements which may occur from the root and heavy locations to the light location.

- *cNAMP movement from the root to the light location.* The main rule on the basis of which cNAMPs make the decision about a movement is presented in (2), i.e. $T_h > T_n + T_{comm}$. According to it, to make the decision the cNAMP needs to know its execution time on the current location, T_h, and its execution time on the new location after the cNAMP arrival, T_n.
  First, on the basis of (15) in Fig. 5 and (1) cNAMP execution time on the root location before a cNAMP movement is calculated:
  $$T_h = \frac{W_r}{f \cdot S} \cdot \left[\left[\frac{2fk + f - 1}{2(f+1)}\right]\right]. \tag{17}$$
  The light location becomes a heavy location after a cNAMP arrival, and has $[[\frac{2k-f+1}{2(f+1)}]]$ cNAMPs. According to (16) in Fig. 5 and (1) execution time is
  $$T_n = \frac{W_r}{S} \cdot \left[\left[\frac{2k - f + 1}{2(f+1)}\right]\right]. \tag{18}$$
  Substituting (17) and (18) in (2) gives
  $$\frac{W_r}{f \cdot S} \cdot \left[\left[\frac{2fk + f - 1}{2(f+1)}\right]\right] > \frac{W_r}{S} \cdot \left[\left[\frac{2k - f + 1}{2(f+1)}\right]\right] + T_{comm}. \tag{19}$$
  If condition (19) holds then the cNAMP moves from the root to the light location. The system enters another optimal balanced state and *cannot have any more movements*.
- *cNAMP movement from a heavy to the light location.* cNAMP execution time on a heavy location is
  $$T_h = \frac{W_r}{S} \cdot \left[\left[\frac{2k - f + 1}{2(f+1)}\right]\right]. \tag{20}$$
  After a cNAMP movement to the light location, the light location becomes heavy and execution time on it is (18). Substituting

(18) and (20) in (2) gives the following cNAMP movement condition:

$$\frac{W_r}{S}\cdot\left[\left[\frac{2k-f+1}{2(f+1)}\right]\right] > \frac{W_r}{S}\cdot\left[\left[\frac{2k-f+1}{2(f+1)}\right]\right] + T_{comm}$$

or

$$T_{comm} < 0. \tag{21}$$

In fact, (21) shows that *the number of cNAMPs on locations with the same available speed must differ by at least two before cNAMPs will move between them*. We call this condition the *minimum difference criterion*.

(b) *cNAMP Termination at the Root Location*. After the termination the root location has $[[\frac{f(2k-1)-3}{2(f+1)}]]$ cNAMPs, and non-root locations have $[[\frac{2k-f+1}{2(f+1)}]]$ cNAMPs, again from (8) and (9). Applying a similar analysis we find that *only one movement may occur, and hence there is no greedy effect*. □

cNAMP termination in a near-optimally balanced network. In near-optimal balance a system has $[[\frac{f(2k+1)-1}{2(f+1)}]]$ cNAMPs on a root location, and either $[[\frac{2k-3f-1}{2(f+1)}]]$ or $[[\frac{2k-f+1}{2(f+1)}]]$ cNAMPs on non-root locations (Section 6.5).

Theorem 6. *The greedy effect causes at most one redundant movement, when a cNAMP terminates in a near-optimally balanced homogeneous network.*

Proof of Theorem 6 follows directly from Lemma 7.

Lemma 7. *A redundant movement occurs only in two cases: of a cNAMP termination in near-optimal balance on the root location which is discovered first by a cNAMP from a light location, and of a cNAMP termination in near-optimal balance on a light location which is discovered first by a cNAMP from the root location.*

Recall that a light location has one cNAMP less than the optimal number (Glossary). The proof of Lemma 7 again proceeds by case analysis on the location where termination occurs, and where the first movement is initiated [8].

Probability of the greedy effect after a cNAMP termination. In a homogeneous network the greedy effect occurs only in two cases after a cNAMP termination from a near-optimal balanced state:

- a cNAMP terminates at the root location, and then a cNAMP from a light location discovers the opportunity to move first, i.e. the following condition must hold:

$$\frac{W_r}{S}\cdot\left[\left[\frac{2k-3f-1}{2(f+1)}\right]\right] > \frac{W_r}{f\cdot S}\cdot\left[\left[\frac{2fk+f-1}{2(f+1)}\right]\right] + T_{comm}.$$

- a cNAMP terminates at a light location, and then a cNAMP from the root location discovers the opportunity to move first, i.e. the following condition must hold:

$$\frac{W_r}{f\cdot S}\cdot\left[\left[\frac{2fk+f-1}{2(f+1)}\right]\right] > \frac{W_r}{S}\cdot\left[\left[\frac{2k-3f-1}{2(f+1)}\right]\right] + T_{comm}.$$

Thus, the greedy effect probability, P, in a near-optimally balanced homogeneous network is the sum of probabilities of these two events:

$$P = P_1 + P_2. \tag{22}$$

Probability P_1 that a cNAMP terminates from the root location, and then a cNAMP from a light location discovers the opportunity to move first is a product of the probability of a cNAMP termination from the root location, P_{termR}, and the probability of the discovery of the better opportunity for execution first by a cNAMP from a light location, P_l:

$$P_1 = P_{termR}\cdot P_l. \tag{23}$$

To calculate the probability of a cNAMP termination at the root location, assume that cNAMP execution time on locations follows an exponential distribution. The mean cNAMP execution time on the root, heavy and light locations is given by:

$$T_{loc} = \frac{W\cdot x_{loc}}{S_{loc}} = \frac{W}{R_{loc}}. \tag{24}$$

Hence, the rate of cNAMP terminations at the root, heavy and light locations is:

$$\nu_{loc} = \frac{R_{loc}}{W}.$$

Assume, that there are N_l light locations and N_h heavy locations in the system. Then probability that a cNAMP terminates at the root location is:

$$P_{termR} = \frac{\nu_{root}}{\nu_{root} + N_h\cdot\nu_h + N_l\cdot\nu_l}$$

or

$$P_{termR} = \frac{R_{root}}{R_{root} + N_hR_h + N_lR_l}. \tag{25}$$

cNAMPs from non-root locations have an equal probability of discovering a better opportunity for execution first. The total number of cNAMPs on non-root locations, $x_{non-root}$, is

$$x_{non-root} = N_hx_h + N_lx_l.$$

The probability that a cNAMP from a light location is the first to discover a better opportunity for execution is the ratio of the total number of cNAMPs on light locations to the total number of cNAMPs on non-root locations:

$$P_l = \frac{N_lx_l}{N_lx_l + N_hx_h}. \tag{26}$$

Thus, substituting (25) and (26) in (23), gives the following probability, P_1, of cNAMP termination on the root location and further discovery of the opportunity to move by a cNAMP from a light location first:

$$P_1 = \frac{R_{root}}{R_{root} + N_hR_h + N_lR_l}\cdot\frac{N_lx_l}{N_lx_l + N_hx_h}. \tag{27}$$

Applying the same principle results in the following probability, P_2, of cNAMP termination on a light location and further discovery of the opportunity to move by a cNAMP from the root location first:

$$P_2 = \frac{N_lR_l}{R_{root} + N_hR_h + N_lR_l}\cdot\frac{x_{root}}{x_{root} + N_hx_h}. \tag{28}$$

Substituting (27) and (28) in (22) gives the following probability of the greedy effect after a cNAMP termination from a near-optimally balanced homogeneous network:

$$P = \frac{N_l}{R_{root} + N_hR_h + N_lR_l}\cdot\left(\frac{R_{root}x_l}{N_lx_l + N_hx_h} + \frac{R_lx_{root}}{x_{root} + N_hx_h}\right). \tag{29}$$

The range of values that P can take is calculated for homogeneous networks of locations (3193 MHz) by changing the total number of locations from 3 to 50, the number of light locations from 1 to $N-2$, the load factor, $0.05 \leqslant f \leqslant 0.95$, and the number of cNAMPs, k, from 1 to 200. The calculation considers only cases when $N_h\cdot N_l \neq 0$, because a system must have both heavy and light locations in order the greedy effect can occur. The root and light locations must have at least one cNAMP.

In total the probability is calculated for 3,100,872 states. The analysis shows that the number of cNAMPs, k, has no significant effect on the probability. The total number of locations, N, the number of light locations, N_l, and the load factor of the root location, f, have a direct impact on the probability. The maximum probability in the conducted calculations is 32%. It occurs in a homogeneous network of 50 locations where 47 locations are light, load factor of the root location $f = 0.95$ and $k = 195$ cNAMPs, i.e. the total number of cNAMPs is 4948. The mean probability is 4%.

Summary In the analysis of a cNAMP movement on homogeneous networks, we have defined dependence between number of cNAMPs and locations in optimal and near-optimal balanced states. The analysis shows the following results of cNAMP behaviour after a cNAMP termination in an optimal balanced network:

(a) there is no greedy effect (Theorem 5);

(b) cNAMPs do not move, when a system is in a near-optimal balanced state (minimum difference criterion).

When a cNAMP terminates in a near-optimal balanced network the greedy effect only occurs in two case and causes only one redundant movement (Theorem 6). The mean probability of this movement in the conducted experiments is 4%.

7.2. Heterogeneous network

cNAMP termination in optimally and near-optimally balanced networks. The analysis is made for a heterogeneous network of q subnetworks one of which is the root location singleton subnetwork.

Theorem 8. *The number of redundant movements in a heterogeneous network after a cNAMP termination does not exceed $q - 1$.*

Lemma 9. *A system makes at most $q - 2$ redundant movements after a cNAMP termination from an optimally balanced heterogeneous network.*

Lemma 10. *A system makes at most $q - 1$ redundant movements after a cNAMP termination from a near-optimally balanced heterogeneous network.*

The proofs of Theorem 8, Lemmas 9 and 10 again proceed by case analysis on the state (i.e. optimal or near-optimal balanced) and location where termination occurs, and then where the first movement is initiated [8].

Probability of the greedy effect after a cNAMP termination. The probability of the maximum number of redundant movements after a cNAMP termination from optimally balanced heterogeneous network is calculated using the same principle as in Section 7.2 (the detailed discussion is presented in [8]). The calculation shows that *the median probability of $q-2$ redundant movements after a cNAMP termination from optimally balanced heterogeneous network does not exceed 1%.*

As it is difficult to estimate mean and maximum values of the probability for a near-optimally balanced subnetwork, the results of conducted experiments show that the probability is less than 30%, and it rapidly decreases as the number of subnetworks increases.

Summary The analysis of cNAMP movements after a cNAMP termination from optimally balanced heterogeneous network shows that: (a) cNAMPs never move from locations which in optimal balance have higher cNAMP relative speed to locations which in optimal balance have lower cNAMP relative speed; (b) a system makes at most $q - 2$ redundant movements to rebalance (Lemma 9); (c) the probability median value of maximum number of redundant movements does not exceed 1%.

Results of analysis after a cNAMP termination from near-optimal balance are as follows: (a) to rebalance a system makes at most $q - 1$ redundant movements (Theorem 8); (b) in the experiments the probability of the maximum number of movements does not exceed 30% and rapidly decreases with the increase of a number of subnetworks.

8. Conclusion and future work

8.1. Summary

We have undertaken the first substantial investigation of thrashing, or greedy effects, in distributed collections of autonomous mobile agents. We have identified two types of greedy effects in AMP systems: location thrashing causes additional movements *and* increase in AMP execution time; location blindness causes only additional movements, as all transferred AMPs improve their execution environment. Both greedy effects appear in the load balancing literature (Section 3).

We have simulated the greedy effects in an AMP implementation, and shown that each AMP makes on average two redundant movements during execution for the scenarios considered. Although greedy effects have limited impact on networks with a small number of AMPs, few locations, or small AMPs, their effects increase as any of these factors scale. The analysis of the redundant movement types and the reasons they occur have showed that redundant movements are mainly caused by location thrashing (Section 4).

To reduce location thrashing we have introduced the agent-oriented concept of negotiating AMPs, described and implemented AMPs that negotiate with a competitive scheme, so called cNAMPs. By negotiation we only mean a coordination among competitive and self-interested agents. cNAMP simulation results show that cNAMPs exhibit only the location blindness. cNAMPs do not make redundant movements during initial distribution, and all scenarios show at least three times faster initial balancing in comparison with AMPs, e.g. dropping from 60.4 to 14.7 s in Scenario 1. During rebalancing after an AMP/cNAMP termination, cNAMPs make far fewer redundant movements, and the cNAMP rebalancing takes less than half of the time of AMP rebalancing, e.g. to rebalance 19 AMPs take 28.2 s, and 19 cNAMPs take 7.8 s in Scenario 2. cNAMPs require less execution time than AMPs. Mean cNAMP execution time is at least 1.6 times less than mean AMP execution time, e.g. mean execution times of 10 AMPs and 10 cNAMPs in Scenario 3 are 232 and 142 s respectively (Section 5).

To analyse the significance of the greedy effect we have established the properties of balanced states. Here, we have analysed AMPs as the general case and described properties including *independent balance*, *singleton optimality*, *consecutive optimality*, and characterised optimal and near-optimal balanced states for homogeneous and heterogeneous networks (Section 6).

We have established the significance of the greedy effect by predicting the worst case (maximum number) of redundant movements after a cNAMP termination from a network of q subnetworks. The results show that a difference in the number of cNAMPs needs to be at least two before a movement will occur between locations of a same subnetwork. A system with q subnetworks makes at most $q - 2$ redundant movements after a cNAMP termination from optimally balanced network with median probability less than 1% (Lemma 9). The number of movements after a cNAMP termination in a network of q subnetworks does not exceed $q - 1$ (Section 7).

We have assembled a set of consistent and precise definitions about the behaviour of collections of AMPs and cNAMPs. These definitions are essential for reasoning about AMP/cNAMP behaviour, and often generalise the definitions given in earlier papers [6,10] (Glossary).

Table 7
Glossary.

Term & Definition	Source	Related Concept	Section
The *actual load* is the number of executing cNAMPs on a location. It is used by local cNAMPs.		Committed load	5.2
An *AMP relative speed*[a], R, is an available speed, S, equally divided between the AMPs at a location, x_{loc}, i.e. $R = \frac{S}{x_{loc}}$.	3	Available speed	4.1
Autonomous mobile programs (AMPs) are mobile agents that improve execution efficiency by managing load balancing; AMPs are aware of their resource needs, sensitive to the execution environment and periodically seek a better location to reduce execution time.	1	cNAMP, load server	1
The *available speed*[b] of a location is the execution speed of a single AMP on that location, i.e. $S = (\textit{CPU speed}) \cdot (1 - \textit{non AMP load})$.	3	AMP relative	4.1
In a *balanced state* no AMP can gain a greater AMP relative speed by moving.	1	Near-opt. balance, optimal balance, stable state	1
cNAMPs are negotiating AMPs with a competitive scheme which announce their intentions to move and compete with each other for opportunity to transfer to the new location.		AMP, load server	5.2
The *committed load* represents the actual load of a location, together with the cNAMPs that have received confirmation to transfer to the location. It is used by remote load servers.		Actual load	5.2
A *communication cost* is the number of AMP movements during a rebalancing.		Greedy effects, opt. rebalancing	3.2
Consecutive optimal property. If a system with a total of x AMPs is optimally balanced, *and* the subnetwork with the highest AMP relative speed is singleton, then the system with a total of $x + 1$ AMPs is also optimally balanced.		Optimal balance, singleton subnetwork	6.4
Greedy effects are the result of a non-optimal AMP rebalancing which differs from the optimal rebalancing in having additional redundant movements, and is a result the AMP making a locally optimal choice.		Location blindness, location thrashing, optimal rebalancing	3
A *heavy location* is a location with the optimal number of AMPs.		Light location, optimal number of AMPs, root location	6.5
A *heterogeneous network* is a set of locations with different available speeds.	2	Homogeneous network, subnetwork	4.1
A *homogeneous network* is a set of locations with the same available speed, except the root location, which may have different, because of the communication with the remote processes shipping from it.	2	Heterogeneous network, subnetwork	4.1
Independent balance property. For a balanced state, the relationship between the number of AMPs x_i and x_j on locations in any two subnetworks i and j is independent of the number of locations in those subnetworks and independent of the presence or absence of other subnetworks, subject only to the sum $x = x_i + x_j$ being constant. The only exception to this rule is the case when distribution of $x' = x_i + x_j + 1$ AMPs results in all x' AMPs having the same relative speed. In this case the partition may have two variants.		balanced state, subnetwork	6.2
A *light location* is a location which has one AMP less than the optimal number of AMPs in a heavy location.		heavy location, very light location	6.5
A *load server* on a location collects network state information to reduce AMP coordination time.	1	AMP, cNAMP	2.2
Location blindness is the greedy effect resulting from an AMP's lack of precise information about the remaining execution time of other AMPs.		greedy effects, location thrashing, opt. rebalancing	3.3
Location thrashing is the greedy effect resulting from an AMP's lack of information about other AMPs intending to move to the same location.		greedy effects, location blindness, opt. rebalancing	3.2
Minimum difference criterion. The number of cNAMPs on locations with the same available speed must differ by at least two before cNAMPs will move between them.		balanced state, stable state	7.1
In *near-optimal balance* some subnetworks have near-optimal number of AMPs. The locations of these underloaded subnetworks have either the optimal number of AMPs or one less than the optimal number. The underloaded subnetworks are determined by being the subnetworks with the slowest AMP relative speed in the nearest upper optimal balanced state.	3	balanced state, nearest upper (lower) optimal balanced state, optimal balance, subnetwork	6.4
Nearest upper (lower) optimal balanced state is the optimal balanced state which the system enters by adding (removing) the minimum number of AMPs.		near-opt. balance, optimal balance	6.4
In *optimal balance* locations with the same available speed have equal numbers of AMPs.	2	balanced state, near-opt. balance	6.3
For any optimal balanced state, the *optimal number of AMPs* for a subnetwork is the number of AMPs on each location in the subnetwork.		heavy location, optimal balance, subnetwork	6.4
An *optimal rebalancing* is a sequence of AMP movements that are the minimum number of AMP movements needed to enter a stable state.		greedy effects	3

Table 7 (*continued*)

Term & Definition	Source	Related Concept	Section
The *root location*[c] is the location where all AMPs start.		heavy location, light location	3.1
Singleton optimal property. All balanced states which a network of singleton subnetworks enters are optimally balanced.		balanced state, optimal balance, singleton subnetwork	6.3
A *singleton subnetwork* is a subnetwork with one location.		subnetwork	6
In a *stable state* no AMP can reduce its execution time by moving.		balanced state	1
A *subnetwork* is a set of locations with identical available speeds.		heterogen. network, homogen. network	1
A *very light location* is a location which has two AMPs less than the corresponding heavy location.		heavy location, light location	6.5

[a] Termed *average relative speed* in [10].
[b] Termed *relative speed* in [10].
[c] It is either called *initiating location* or *first location* in [10].

8.2. Future work

The current paper assumes that the network is totally reliable, and has a flat structure, i.e. all location are equidistant from one another in a fully connected graph (a LAN). The available speed at a location is constant, and is shared equally by the AMPs or cNAMPs at that location. Other directions include relaxing the assumptions, e.g. network reliability, and using game theoretic techniques to analyse AMP behaviour. As AMPs and cNAMPs have been constructed and analysed only on LANs we also plan to adapt the cNAMP cost model and investigate cNAMP properties in wide area networks.

Glossary

Table 7 gives rigorous definitions for the concepts defined and used in the paper and associated proofs. In column *Source* '1' denotes a definition from [10], '2' a generalisation of a definition from [10], and '3' a more precise definition than in [10]. In column *Related Concept* we provide lists of related concepts to clarify the difference and similarity of the concepts. Column *Section* gives the numbers of the sections where the notions are introduced in the paper.

References

[1] M. Backschat, A. Pfaffinger, C. Zenger, Economic-based dynamic load distribution in large workstation networks, in: Euro-Par '96, vol. 2, Springer-Verlag, London, UK, 1996, pp. 631–634.
[2] T.L. Casavant, J.G. Kuhl, A taxonomy of scheduling in general-purpose distributed computing systems, IEEE Trans. Softw. Eng. 14 (2) (1988) 141–154.
[3] T.L. Casavant, J.G. Kuhl, Analysis of three dynamic distributed load-balancing strategies with varying global information requirements, in: DCS '87, IEEE Press, New York, USA, 1987, pp. 185–192.
[4] A.J. Chakravarti, G. Baumgartner, Self-organizing scheduling on the organic grid, Int. J. High Perform. Comput. Appl. 20 (2004) 130.
[5] N. Chechina, Autonomous Mobility in Multilevel Networks, (Expected) Ph.D. Thesis, School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK, 2011.
[6] N. Chechina, P. King, R. Pooley, P. Trinder, Simulating autonomous mobile programs on networks, in: PGNet'09, Liverpool, UK, 2009, pp. 201–206.
[7] N. Chechina, P. King, P. Trinder, Using negotiation to reduce redundant autonomous mobile program movements, in: IAT'10, IEEE Computer Society, Toronto, Canada, 2010, pp. 343–346.
[8] N. Chechina, P. King, P. Trinder, Complete experimental and theoretical analysis of greedy effects in autonomous mobility, Tech. Rep. 0073, Heriot-Watt University, Edinburgh, UK, 2010.
[9] G. Cybenko, Dynamic load balancing for distributed memory multiprocessors, J. Parallel Distrib. Comput. 7 (2) (1989) 279–301.
[10] X.Y. Deng, Cost Driven Autonomous Mobility, Ph.D. Thesis, School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK, June 2007.
[11] X.Y. Deng, G.J. Michaelson, P.W. Trinder, Cost-driven autonomous mobility, Comput. Lang. Syst. Struct. 36 (1) (2010) 34–59.
[12] X.Y. Deng, P.W. Trinder, G.J. Michaelson, Autonomous mobile programs, in: IAT'06, IEEE Computer Society, Washington, DC, USA, 2006, pp. 177–186.
[13] A.E. El-Abd, M.I. El-Bendary, A neural network approach for dynamic load balancing in homogeneous distributed systems, in: HICSS'97, IEEE Computer Society, Washington, DC, USA, 1997, p. 628.
[14] D. Fudenberg, J. Tirole, Game Theory, The MIT Press, USA, 1991.
[15] A. Fuggetta, G.P. Picco, G. Vigna, Understanding code mobility, IEEE Trans. Softw. Eng. 24 (5) (1998) 342–361.
[16] C. Georgousopoulos, O.F. Rana, Combining state and model-based approaches for mobile agent load balancing, in: SAC'03, ACM, New York, NY, USA, 2003, pp. 878–885.
[17] A. Ghafoor, I. Ahmad, An efficient model of dynamic task scheduling for distributed systems, in: COMPSAC'90, IEEE Computer Society Press, 1991, pp. 442–447.
[18] J. Gray, Notes on data base operating systems, in: Operating Systems, An Advanced Course, Springer-Verlag, London, UK, 1978, pp. 393–481.
[19] Recursion Software, Inc., Voyager Technical Documentation, 2010 http://www.recursionsw.com/Products/voyager.html.
[20] L.V. Kale, S. Kumar, M. Potnuru, J. DeSouza, S. Bandhakavi, Faucets: efficient resource allocation on the computational grid, in: ICPP'04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 396–405.
[21] J.O. Kephart, D.M. Chess, The vision of autonomic computing, Computer 36 (1) (2003) 41–50.
[22] Z.D. Kirli, Mobile Computations with Functions, Kluwer Academic Publishers, Norwell, MA, USA, 2002.
[23] H. Kuolin, Allocation of processors and files for load balancing in distributed systems, Ph.D. Thesis, University of California at Berkeley, USA, 1985.
[24] D.B. Lange, M. Oshima, Seven good reasons for mobile agents, Commun. ACM 42 (3) (1999) 88–89.
[25] A. Legrand, H. Renard, Y. Robert, F. Vivien, Mapping and load-balancing iterative computations, IEEE Trans. Parallel Distrib. Syst. 15 (6) (2004) 546–558.
[26] F.C.H. Lin, R.M. Keller, The gradient model load balancing method, IEEE Trans. Softw. Eng. 13 (1) (1987) 32–38.
[27] M. Livny, M. Melman, Load balancing in homogeneous broadcast distributed systems, in: CNP '82, ACM, New York, NY, USA, 1982, pp. 47–55.
[28] D. Milojičić, F. Douglis, R. Wheeler, Mobility: Processes, Computers and Agents, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999.
[29] N. Miyata, T. Ishida, Community-based load balancing for massively multi-agent systems, 2008, pp. 28–42.
[30] L.M. Ni, C.-W. Xu, T.B. Gendreau, A distributed drafting algorithm for load balancing, IEEE Trans. Softw. Eng. 11 (10) (1985) 1153–1161.
[31] A. Ross, B. McMillin, Experimental comparison of bidding and drafting load sharing protocols, in: DMC '90, vol. 2, IEEE Computer Society Press, 1990, pp. 968–974.
[32] H.G. Rotithor, Taxonomy of dynamic task scheduling schemes in distributed computing systems, IEE Proc. Comput. Digit. Tech. 141 (1) (1994) 1–10.
[33] T. Schlegel, P. Braun, R. Kowalczyk, Towards autonomous mobile agents with emergent migration behaviour, in: AAMAS'06, ACM, New York, NY, USA, 2006, pp. 585–592.
[34] B.A. Shirazi, K.M. Kavi, A.R. Hurson, Scheduling and Load Balancing in Parallel and Distributed Systems, IEEE Computer Society Press, Los Alamitos, CA, USA, 1995.
[35] J. Stender, S. Kaiser, S. Albayrak, Mobility-based runtime load balancing in multi-agent systems, in: SEKE '06, Reedwood City, CA, USA, 2006.
[36] G. Weiss (Ed.), Multiagent Systems, in: A Modern Approach to Distributed Artificial Intelligence, The MIT Press, Massachusetts, USA, 1999.
[37] M. Wooldridge, Agent-Based Softw. Eng. 144 (1) (1997) 26–37.

Natalia Chechina received the Engineering degree from Kyrgyz State Technical University in Kyrgyzstan in 2008. She is now a Ph.D. student at Heriot–Watt University under the direction of Dr. Peter King and Prof. Phil Trinder. Her main research interest is simulation modelling of multiprocessor structures and networks.

Peter King has a first degree in Mathematics from University College London, and a Ph.D. from the University of Newcastle upon Tyne. He has been at Heriot–Watt University since 1982. He has taught a wide range of Computer Science topics to all levels of undergraduate and postgraduate students. His main research interests are quantitative models of performance of computer and communication networks, teaching of computer programming, and predictive models of software performance. Away from the computer, he likes to climb hills, ride his bicycle, and explore his family tree.

Phil Trinder is a (full) professor of Computer Science at Heriot–Watt University in Edinburgh. His primary research interest is the design, implementation and evaluation of high-level parallel and distributed programming languages. He has held eight major research grants to support this activity, and has over 80 refereed journal and conference publications. He received his B.Sc. (Hons) degree from Rhodes University in South Africa in 1983 and a D.Phil. from Oxford University in 1989 for research into parallel functional databases.