Dealing with Permanent Agent Failure in Dynamic Agents Organisations

Asia Ali Salman Al-karkhi[1] and Maria Fasli[2]
[1]*Computer Science Department, University of Technology, Alsina'a Street, Baghdad, Iraq*
[2]*Institute of Data Analytics and Data Science, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, U.K.*

Keywords: Agent Organisations, Self-organisation, Organisation Recovery, Task Failure.

Abstract: This paper is an exploratory study that focuses on the creation of open and dynamic agent organisations which can sustain the provision of services to requesting customers in the presence of agent failure. In a distributed environment, agents within networks and organisations are prone to failure. This can inevitably lead to decreases in the individual agents' utilisation as well as in the whole system's and to the loss of tasks. Here, we present an approach to tackling this challenge by enabling agents to prevent these kinds of disruptions. In an environment where agents create organisations to increase the execution of tasks, we employ the *Henchman Recovery Protocol* (HRP) within each organisation; this enables the agents within an organisation to maintain its functionality in the presence of agent failures and in particular in the case of the lead (*Head*) of the organisation failing. Furthermore, we explore the stability and evolution of organisations over a period of time and when agents drop out of organisations (due to permanent failure) while new agents may enter the environment and either join existing organisations or create new ones. We conduct our study in the context of a grid-like computing system which was implemented in the Repast Simphony agent-based simulation environment.

1 INTRODUCTION

Dealing with failure in distributed systems is a significant challenge as such systems are large, open and complex. Therefore, providing methods for automatic failure detection is important, and this is still considered an open problem (łgorzata Steinder and Sethi, 2004). However, agent or node failure is unpredictable and therefore a non-trivial problem to tackle.

In order to provide seamless services in a number of domains such as grid environments and cloud computing, a number of often heterogeneous service providers, actors/nodes are clustered behind an interface that is providing access to these services to customers. The main goal of creating grid computing systems has been to be able to provide services and to share resources as fast as possible, just as power is shared across an electric power grid, as noted by Ian Foster in (Foster and Kesselman, 2003).

To study the problem and explore potential solutions, we have simulated a grid-like environment through the use of multi-agent systems. In our system, each agent is essentially a node providing services and resources, i.e. a services provider, to requesting customers. To improve their own utilisation, agents can organise themselves in groups or organisations when certain conditions arise in the system. As we show, through organising the agents in organisations, the system utilisation as a whole improves. Nevertheless, agents within the organisations and within the system as a whole can fail. An agent's failure will almost certainly affect the stability and performance of the organisations that it belongs to. One solution for open and dynamic systems, can be the use of self-organising technique so that depending on the circumstances in the environment, the available agents can still provide the system with the required services. These kinds of techniques are becoming ever more prevalent with respect to networked systems and hence we are interested in exploring techniques that will enable the agents to maintain the effectiveness of the organisations and task execution. Although we show how agent organisations can cope in the presence of failure of individual agents and are able to maintain task execution, the problem of agent failure is more acute in the event where the lead agent of an organisation, termed *Head*, fails. In this paper, we explore the use of the *Henchman Recovery Protocol* (HRP) to enable organisations to continue to function even in the event of the *Head* failing.

In grid computing, solutions for the problems caused by faults, include methods to increase fault tole-

Al-Karkhi, A. and Fasli, M.
Dealing with Permanent Agent Failure in Dynamic Agents Organisations.
DOI: 10.5220/0007399907150723
In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART 2019)*, pages 715-723
ISBN: 978-989-758-350-6

rance, recovery, and removal. The main fault tolerance techniques which have been used in cluster systems and grid computing are checking points, message logging, replication and retry (Haider and Ansari, 2012). A more recent study of agent failure in distributed MAS is (Hayashi, 2017) where the author has addressed agent failure in situations where disaster repeatedly occurs. Consecutive or simultaneous agent failures may occur in the future as a result of the disaster event. A comparison has been carried out to find a viable method for decreasing the number of failed agents in the system.

The rest of the paper is organised as follows. Section 2 Scenario description; section 3 creation of organisations and roles of agents; section 4 presents explanations about HRP, and also introduces the algorithm that is used by each Henchman inside the created organisation to monitor the Head. Sections 5 presents the experimental work. Section 6 evaluation of *HRP*; section 7 discusses the related work in the literature Finally, the paper ends in section 8 with the conclusions and proposal for future work.

2 SCENARIO DESCRIPTION

There are two types of agents in the system under consideration: the customer agent, which is used to simulate task requests emanating from multiple customers that may exist in reality, and service providers which possess the resources to execute tasks, which we will simply refer to as agents here on.The agents have diverse resources and can execute heterogeneous tasks and provide services; these match the requirements of the customers to various extents. The customer agent simulates the existence of several customers which need to find services or have tasks executed and sends its requests to the network of agents via messages describing the tasks requiring execution and associated resources as well as other conditions such as a deadline by which the result of the task is required.

2.1 Tasks and Resources

In a distributed environment, resources would be heterogeneous. There are different methods for representing resources (Carroll and Klyne, 2004), however, for the purposes of this research the resources associated with a task have been represented in a simpler way as this is not the main focus of the research and using the format:

$RV = < r1, r2, r3 >$.

An agent in the network that receives a task will check whether the task's resources match with its resources. Given the simple way of representing resources, proximity of requested with available resources is calculated between the customer resource vector and an agent's resources using the well-known Manhattan Distance (Manhattan_Dis) Equation 2:

$$Manhattan\_Dis(RV,AR) = \sum_{x=0}^{x=2} |RV_x - AR_x| \quad (1)$$

Where:
RV_x : the requested resources vector for the task.
AR_x : the matched agent resources that performs task successfully.
x = 0 to 2: the index of the requested resource vector, $<r_1, r_2, r_3>$.

The resulting matching value should meet the task's required accuracy; this is a specific value between $(0-12)$. Therefore, in the task delegation protocol as shown in (AL-Karkhi and Fasli, 2017) an agent accepts a task if a matching has occurred between the customer resources and its resources. For example, where a customer issues a task with *RV* is <0, 0, 0> and has required accuracy (*RA*) equal to 6, and the recipient agent's *RV* is <2, 2, 2>, then when applying the Manhattan Distance equation, the match will be positive.

In addition to resources, tasks have *Time To Live* (TTL) and a *Time Deadline* (TD). TTL indicates the number of hops among agents that a message can make in advancing throughout the network, while TD indicates the deadline for the execution of the task. If $TTL = 0$, the task will be considered to have failed, otherwise, the receiving agent will check the TD of the task; if this is sufficient then the task will be executed, but if it is not sufficient (either because the received agent is currently executing a task or its queue of tasks already contains a number of tasks) the agent will then delegate the task to a neighbour agent in the network. Hence, if an agent cannot satisfy at least one of the conditions described above, the task will be either failed or delegated to another agent in the network. The agents are autonomous and self-interested and have the desire to maximise the benefit to themselves; therefore, agents will keep the accepted tasks in their *Accepted Tasks Queue* (ATQ) for as long as they are currently busy executing task.

2.2 Initial Network Formation

We assume a system where agents are created and start execution and as part of this they formulate an initial network through connections with each other.

This initial network formation occurs within the first few simulation cycles. In the initial phase, the agents acquire partial knowledge of their surrounding environment by creating a contact list. A network model of 300 agents visualized using Gephi (Bastian et al., 2009) with (the Force Altas 2) layout which is useful to visualise Small-World network and scale free networks is illustrated in Figure 1. The various colours represent the degree of each node i.e the number of connections for each agent.

Figure 1: Visualisation of network size: 300 agents. The various colours represent the degree of each node; purple=1 connection, light green=2, blue=3, black=4, orange=5, green=6,red=7, light pink=8, grey=9 or more.

As agents receive requests for tasks from the customer, they will check whether these can be serviced or not. They then use their contacts to essentially propagate the tasks that they are unable to execute for different reasons (unable to execute because they are busy; not in possession of required resources, etc.). An agent in the network may fail randomly and permanently. An agent can detect other failed agents through the task delegation process. When an agent delegates a message to another agent, if it does not receive a response from that agent for a number of cycles, then that agent will be considered failed. Therefore, the sender agent will delete its contact information related to that failed agent. Hence, the network could gradually reduce and eventually even collapse. However, as agents may fail, new agent(s) may be created and inserted in the system. A new agent will send messages to randomly-selected other agents – to obtain at least partial knowledge of its surrounding environment – and get connected to the network. New agents can offer their services either directly to the customer or via the task delegation protocol as described in see (AL-Karkhi and Fasli, 2017).

2.3 Agent Failure Modelling

In order to model agent failing in the network, agents are equipped with a parameter that enables them to be switched on/off for a period of time, and when they are off they, in essence, they create the impression to the system that they are offline and unable to execute tasks or respond to messages. We have a probability distribution between[0,1] such that a lower (nearer 0) probability value produces fewer failed agents and a higher probability value (nearer 0.9) will produce a larger number of failed agents in the system. During the initialization step of the simulation, all agents are assigned the same probability of failure. Applying a high failure rate simulates the harsh operational conditions which exist in some systems and enables us to study effects on the system overall and individual agent behaviour in extreme circumstances.

3 CREATION OF ORGANISATIONS AND ROLES OF AGENTS

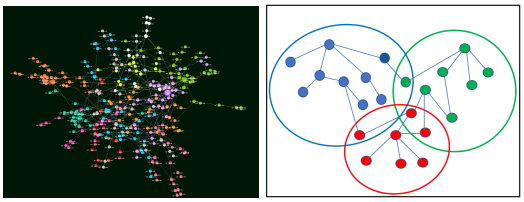
While the initial network formation enables agents to create links and propagate tasks to each other that they are unable to execute themselves through this network and therefore increase task execution, it still leaves the system fairly under-utilised with tasks remaining un-executed or failing due to the unfocused propagation of task messages through an agent's essentially random connections. In order to increase the system utilisation, we are creating organisations within the network to enable more targeted and faster task delegation. To enable the creation of organisations, agents are taking on roles. An agent role can be defined as an agent behaviour that can affect, enhance and/or change a system's structure. Typically a role will posit expectations, skills and duties and hence an agent which takes on a particular role must be able satisfy these requirements (Ferber et al., 2003). Each agent a_i, at any one time, has a number of roles R_i.

- R_i is defined as a set of roles, $< R_1, R_2, R_3, R_4 >$
- An agent a_i can assume a role in relation to at least one other agent, which can be an individual or an organisation.
- The roles an agent may take on within an organisation are $\{Member, Henchman, Head\}$, each agent may have one of these roles or more than one, depending on a set of conditions.
- If a_i has a role it may accept consequential role(s), this means that an agent that is a *Member* in an

organisation later on may accept the $Head's$ message to be the organisation's *Henchman* agent.

- If Org_p and Org_q are two created organisations, then a_i can belong to both of them. In turn, any two or more organisations in the network can share a number of *Members*, so an agent can have different or the same Roles: if the overlapped organisations(Org_p, Org_q):
  1. a_i can be *Member* in Org_p, *Member* in Org_q.
  2. a_i can be *Head* in Org_p, *Member* in Org_q.
  3. a_i can be *Henchman* in Org_p, *Member* in Org_q.
  4. a_i can be *Head* in Org_p, Not *Henchman* in Org_q.
  5. a_i can be *Henchman* in Org_p, Not *Henchman* in Org_q.
  6. a_i can be *Henchman* in Org_p, Not *Head* in Org_q.
- The *Head* of each organisation is responsible for the activities of its agents. The presence of a *Head* in an organisation of agents is necessary in order to coordinate the work.
- A *Henchman* in an organisation will become a temporary *Head* when the *Head* of its organisation has failed.
- Agents are expected to take on and change roles while they are operating in the environment.

Typically an agent will initiate the process of creating an organisation when certain circumstances in the system arise and the agent essentially becomes overly busy and unable to handle additional requests. The initiator agent, becomes the *Head* of the organisation and start asking other agents in the network to join its organisation, see (AL-Karkhi and Fasli, 2017). Organisations can be created as either consisting of a set of agents holding similar resources (within boundaries specified by the *Head*) or heterogeneous resources, leading to heterogeneous organisations. The *Head* is responsible for delegating tasks to the *Member* agents and keeping track of task execution within the system communicating with the customer. The creation of organisations in the system can lead to increased utilisation of the resources of the agents(Dignum et al., 2004), but as has been noted in an organisation that is part of a grid computing system, a computational node may still be underutilised and not fulfil its potential; this can be only 5% or even less of the time (Haider and Nazir, 2016). This motivated us to create overlapped organisations; we aimed to increase the agents' resource utilisation by allowing agents to join more than one organisation. However, in practice, depending on how busy it is, an agent can only be committed to a limited number of organizations at any one time. We also limit the maximum number of organisations that an agent can join by setting a parameter and this varies from agent to agent. So, this will lead to the creation of overlapping communities/organisations as shown in Figure 2 (a) and Figure 2 (b).



(a) Several overlapped organisations structure.

(b) Simplified structure, an example of overlapping organisations where q=3

Figure 2: Illustration of the concept of overlapping organisations. In Figure (b), the Blue organisation overlaps with the Green and the Red organisations in terms of a single node, whereas the Green overlaps with the Red in relation to two nodes. These overlapping regions are in the intersection of the large circles.

4 HENCHMAN RECOVERY PROTOCOL (HRP)

Although agent utilisation may improve through the creation of organisations, agents within organisations may fail. The failed agent could be a *Head* or a *Member*. If it is a *Head*, then when a *Member* wants to send messages to the *Head* and there is no response from the *Head* and after a number of attempts, the *Member* will consider the *Head* to have failed. When a *Member* agent has failed, the *Head* will detect this after a number of attempts at sending messages; in this latter situation, the *Head* will remove the *Member* from its database (*DB*). Although task execution will be affected by a *Member* failing, the case of *Head* failure is more acute as it essentially means that the organisation becomes disconnected and this leads to further task failures and under-utilisation of resources. In this section, we describe how we have deployed the HRP protocol in each organisation in order to recover more of the customer tasks. In the literature, most of the available methods that provide fault tolerance, use reactive techniques; these provide solutions to failure only after its occurrence. In contrast, we argue that using multi-agent systems with self-organising capabilities represented by *HRP* can provide a proactive methodology which can improve task execution in open, dynamic and distributed environments.

The *Head* is the agent which is responsible for distributing the tasks to its *Members* in an organisation, so that if the *Head* fails, the tasks will also fail,

and this will affect the performance of the system as a whole and lead to the loss of more tasks. To avoid this loss of tasks and enhance the performance of the system, a new role, that of the *Henchman* is used to address this problem. When a *Head* fails, the role of the *Henchman* is to detect the failure of its *Head* and act as a substitute to the failed *Head*, providing further self-organised capability to the system, maintaining the functionality of the organisation. Each *Member* has the ability to decide to accept a new role or to reject it. The following describes the mechanism of the *Henchman* role in more detail.

The first stage of the *HRP* is performed during the runtime process of the gossip algorithm – Algorithm 1, which is based on the multi-cast "Push Gossip" algorithm (Serugendo et al., 2011). The process is started by the *Head* agent sending out a message to the first agent a_i that joined the $Head's$ organisation, asking the agent whether it will agree to be its (the $Head's$) *Henchman* (HM). The *Head* will send the same message to all the other agents that joined in sequence until one of the *Members* accept the $Head's$ message and becomes the organisation's *Henchman* and this is what we called it "BeMyHenchman" acknowledgement messaging technique.

After assigning a *Henchman* for its organisation, the *Head* has the responsibility to synchronize its database *DB* with that of the *Henchman*; this contains the contact details of the *Members*, and will, of every newly joined agent. This means that the $Henchman's$ *DB* will always be maintained to be identical to the $Head's$ *DB*. When the *Head* goes offline, the *Henchman* will immediately replace the *Head* to maintain the functionality of the organisation.

If the *Henchman* does not receive any acknowledgement back from the *Head*, the *Henchman* will declare to all the other *Members* and the customer agent that the *Head* has failed and the new *Head* is the *Henchman*; this is in order to redirect the traffic to itself instead. When the *Head* recovers, i.e. the *Henchman* receives an "I'm alive" messages again from the *Head*, the *Henchman* will inform the organisation *Members* as well as the customer that the *Head* is now alive and the organisation should be back to its normal condition. However, there is also the chance that a disruption will affect both the *Head* and its *Henchman* at the same time. Here, if the customer agent sends tasks while this is the case, these tasks will fail – until the *Head* and/or the *Henchman* return to an active state.

If the failed agent is one which has a *Henchman* role in one of the existing organizations, the *Head* will remove it from its *DB* and the *Members* will be informed by the *Head* that the *Henchman* is no longer functioning. The *Head* agent will then assign a new *Henchman* for its organisation.

If both the *Head* and then the *Henchman* of the organisation have failed, the organization will have neither a *Head* nor a *Henchman*, and in such cases, when the *Members* want to access them by sending messages they will receive no reply, and after a number of attempts the *Members* will consider the organisation to have been disbanded.

Algorithm 1: Gossip Protocol.

Input: a_i: is the most busy agent (*Head*) that will create an organisation, a_x is one of the a_i contacted list neighbour, $TTL > 0$.

Output: a_i completed organisation.

1: $Cycle\_e$
2: a_i.SendMessage ($Contactlist$(1,N))
3: **if** (a_x ! = busy) **then**
4: a_x infected with the gossip message of a_i
5: // a_x has the option to join or not to the created organisation.
6: applying "BeMyHenchman" acknowledgement messaging technique to select *aMember* to be *aHenchman*.
7: **else if** (a_x == busy) **then**
8: a_x.SendMessage (1,(N))
9: **end if**
10: **if** ($TTL > 0$) **then**
11: $TTL = TTL - 1$
12: **end if**
13: $Cycle\_e + 1$
14: End

5 EXPERIMENTAL WORK

We have developed three models (*HRP*, Organisation_Ver1 and No Organisation) which all use the setting parameters, as in Table 1. In relation to these three models, we have shown the results of various network sizes (500, 5000) agents, task distributions (the task distribution is used to specify the number of tasks sent from customer to the network of agents in each cycle) and simulation times. For all of the three models, the number of tasks sent from the customer agent follows a normal distribution with specific values for the mean and variance. The simulation time was set at 7000 cycles for all of the three models.

- No Organisation model is a network model with only task delegation property.
- Organisation_Ver1 model is a model that contains the organisations of agents.

- *HRP* model is the organisations model with HRP protocol capability.

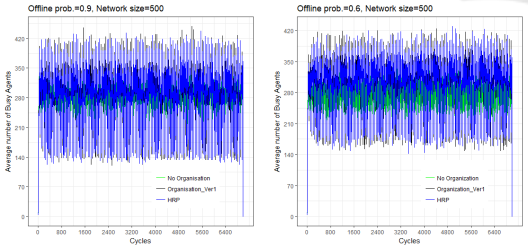
Table 1: Experimental Setting Parameters.

Agent Network Size	Task distribution
500	Mean=1000,variance=10
2500	Mean=1000,variance=10
5000	Mean=1000,variance=10

$$ANSET = \frac{1}{N_R} \sum_{i=1}^{N_R} \sum_{x=0}^{x=2} |RV_x - AR_x| \qquad (2)$$

Where:
N_R: the number of runs $= 10$.
RV_x : the requested resources vector for the task.
AR_x : the matched agent resources that performs task successfully.
$x = 0$ to 2 : the index of the tuple that represent the requested resource vector $< r_1, r_2, r_3 >$.

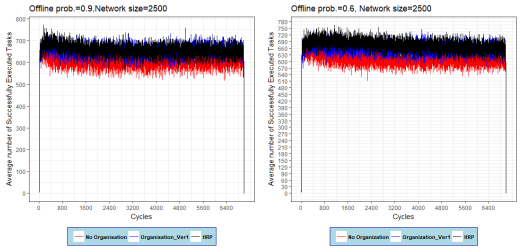
Figure 3 (a) and Figure 3 (b) show the average number of successfully executed tasks, ANSET, within the cycles, for each of the three models which have been implemented, see Equation 2. Note that the two models (Organisation_Ver1 and HRP) have a specific structure in terms of their organisations and also have gradual responsibilities for their agents: agents have different roles(*Head*, *Henchman*, *Member*). However, the roles of agents may change over the simulation time and agents disappear and new agents appear; these changes will affect the execution of tasks and the system's ability to schedule tasks.



(a) (b)

Figure 3: Average numbers of successfully executed tasks computed across 7000 simulation cycles, p= 0.9 and p= 0.6.

The No Organisation model consistently delivers lower numbers of successfully executed tasks over the simulation time (than the other two). So, using this model, the system loses a significant number of tasks due to frequent agent failures and the restrictions imposed by the messages' Time To Live (TTL) and the tasks' deadlines. In this model, the resources are distributed and the delegation for the task message may take more time to reach a desired agent that



(a) (b)

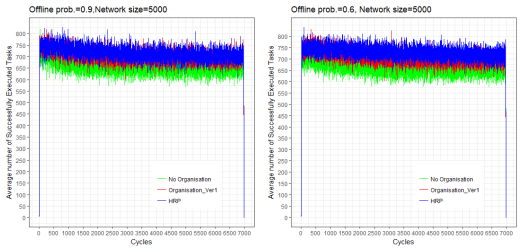
Figure 4: Average numbers of successfully executed tasks computed across 7000 simulation cycles, p= 0.9 and p= 0.6.

can accept the task. Hence, the other two models, Organisation_Ver1 and *HRP*, show better performance; in these cases, fewer hops are required in order to find agents which can accept the tasks because, even though there is a probability of failure, the organisations are created with heterogeneous resources and therefore, in the event of failure, other agents are readily available to execute the tasks. The ANSETs of these two models fluctuate due to the presence of the probability of failure, which causes disruption in the system such that agents will start to fail inside the created organisations leading to significant changes in the execution rates of the tasks. The No organisations model shows a smaller percentage of variation in its ANSET as compared to the other two models. It is clear that the No Organisation model executes tasks within the same ranges of time, on average, in each and every simulation cycle. This is because the model contains no structure and new agents may connect randomly with other existent individual agents. Where the task may be executed is dependant only on delegation across the entire network, and this quickly consumes the tasks' TTL values. Hence, the Organisation_Ver1 and *HRP* models perform better on average than does the "No Organisation" model.

In Figure 4 (a) and Figure4 (b), it is noteworthy that, with network size 2500, the system activity becomes more stable, showing less fluctuation in the ANSET values than in a network size of 500 agents. Even in the presence of significant numbers of failures, the *HRP* model achieves the highest average number of successfully executed tasks. This is because, first, the higher the number of agents in the system the more organisations can be created with a greater variety of resources. Second, when failures occur in the system, there are other agents which are able to join the network structure and so can also join the existing organisations.

In Figure 5 (a), and Figure 5 (b), where increasing the network size leads to there being higher numbers of tasks executed within the simulation cycles: the ANSET is more than with the other two network sizes

(2500, 500). In models Organisation_Ver1 and *HRP*, the created organisations consist of a large number of heterogeneous agents, so even with the failure of an agent the *Head* will probably still able to find a *Member* that can satisfy the required resources. Moreover, the existence of the *Henchman* in the *HRP* has added benefit to the system.

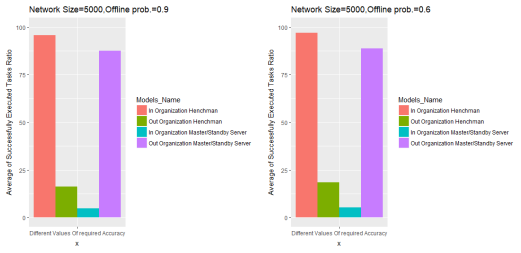


(a) (b)

Figure 5: Average numbers of successfully executed tasks computed across 7000 simulation cycles, p=0.9 and p=0.6.

6 EVALUATION OF THE HRP MODEL

We have compared *HRP* with other methods from the literature. The results from the *HRP* model with network sizes (500, 2500, 5000). However, due to the limited space we have included only the experimental work of 5000 agents. were compared against the master-standby system (Chen, 2007). The Master (MAS)/standby (SBS) model is a fault tolerance model in which the system has two servers. The first one is called the MAS and all the clients are connected to it. The second is called the SBS – the clients are only connected to this when the MAS has failed. Furthermore, there is a checking message transmitted between the MAS and the SBS which enables the SBS to switch to active mode and serve the clients' requests in place of a failed MAS. We have implemented this architecture within our simulation set-up, and the results are depicted in the following Figures:



(a) (b)

Figure 6: Average number of successfully executed task ratio in/out organisations with probability p= 0.9 and p= 0.6.

In Figure 6 (a), and Figure 6 (b), we have computed the number of executed tasks ratio, ANETR, in relation to the following: various different required accuracies; execution inside and outside of organisations; a network size of 5000 agents; and two probabilities of failure – 0.9 and 0.6. As shown, a network where *HRP* operates within organisations outperforms the MAS/SBS model. This is because the MAS/SBS model depends for its operation on its delegation process, and a smaller number of tasks have been executed inside its organisations as a result of the fact that this model has no structural roles such as those in the *HRP* model's organisations. *Members* inside the organisations have roles and they can delegate tasks to the $Head's$ of organisations that they are part of. Also, the *Head* and the *Henchman* of an organisation each have a significant role to perform within it. However, in MAS/SBS the nodes delegate tasks to other nodes in the emerged organisations, and if no agent can accept the task the task will be delegated across the network which consumes its TTL values.

7 RELATED WORK

Real world networks can consist of multi-layers that can all be affected by unpredictable changes (disruptions) in their structure (De Domenico et al., 2014). Therefore, to investigate the tolerance to failure of our system, we created three different models: the first is a straightforward network of agents model; the second model implements a virtual layer of organisations which provides a self-organised multi-agent and the application of roles and protocols for the agents; and the third is the HRP model.

In the literature, many studies have used bio-inspired methods to solve NP hard problems because they are considered an appropriate mechanism to study complex systems. In (Stamatopoulou et al., 2004) studies have used bio-inspired, bird flocking, method in creating dynamic organisations of multi-agents with the ability to reconfigure the system individual connections. The authors have used two formal mechanisms, first P-systems (Nematollahi Mahani, 2012) with active members and second software engineering communication language X-machines with active membranes in order to model flocking agents. Agents can have various roles (leader, donor, incubator). The produced system is very complex and cannot be animated.

Researches in (Dignum et al., 2004) have discussed the different organisational structure: social organisations and emerging organisations; that can be deployed to organise agents. They have also explai-

ned how it is difficult to describe at what point the organisations may reorganise for better performance and utilization but without showing any solution. In our work the dynamic organisations may re-organise because of agents failure which may lead to create another organisation or participate in already existed ones and with the aid of the *HRP* the task execution output has increased.

Many works in literature, such as (Kota et al., 2009) that use the agent organisations and the created organisations reconfigured again depend on the type of task been sent to the organisations without claiming and type of technique that will lead to this action of creation. Unlike our work the organisations are created based on triggering conditions to provide proactive solution to the failure case. In our work, the created organisations have the ability to reconfigure themselves as an occurrence with critical event (failure). Hence, organisations may be created or disbanded depending on the network real problems whatsoever are the types of coming tasks.

Researcher in (Ferber et al., 2003) presented a method to reorganise organisations by applying a method that can be carried by the individual agents in the system. A pair of agent can decide when to reorganise and with whom they can create the new organisation based on their utility values. The new connections will not change the inner characteristic of agents. In our work when new agents are added to the system some new roles will emerge in the system. The update to the already existed organisations can lead to better performance since the new agents can be Henchman, Member or they can be Heads of new organisations when the triggering conditions met.

In (Mathieu et al., 2002) the authors have presented self-adaptation of a multi-agent systems for organisations. The dynamic interaction between agents and their decision-making capabilities may lead to either an agent decision to keep itself connected to its organisation or change its connection for better performance with other organisations. Aiming to reduce the message flow in the organisations to enhance the system behaviour. In our work, agents may be added to the system leading to update the existing connections and enhancing the system output.

8 CONCLUSIONS

This paper focuses on studying the creation of open and dynamic agent organisation formations which can provide services to requesting customers in the presence of failure. This can lead to the loss of tasks and to decreases in the effectiveness and utilisation of agent networks. We have presented a framework whereby agents and organisations can be counted on to provide remedies which can avert these kinds of disruptions. Our aim was to deploy the *Henchman* Recovery Protocol. *HRP*, within each organisation; this is a viable solution for maintaining the functionality of the organisations. After that, we explore the performance and stability of the created organisations in the situation where we have agents malfunctioning and others appearing. The new agents may simply become part of existent organisations or their presence may result in the emergence of new organisations.

Weeding out failure from distributed systems requires sound theories and efficient solutions that can be applicable in order to maintain systems stabilities (Bao and Garcia-Luna-Aceves, 2003),(Haider and Nazir, 2016). Grid computing is the target domain for this work because it can provide researchers with a suitable environment in which to apply our virtual organisations as well as in which to study node failure. Our solution is to apply a heuristic protocol, *HRP*, in order to recover customer tasks and preserves the organisations' formation structure. *HRP* has been shown to have a more acceptable performance as compared to the MAS/SBS model. The existence of roles inside the heterogeneous organisations plays an important role in the self-organisation of the systems and provides a pro-active technique for dealing with failure. The experiments have shown that the *HRP* produces fewer traffic messages than the other models: (No Organisation, Organisation_Ver1, MAS/SBS). As part of future work, we will explore adding the ability to learn to the agents in the emerging organisations to explore how this can affect system performance.

REFERENCES

AL-Karkhi, A. and Fasli, M. (2017). Deploying self-organisation to improve task execution in a multi-agent systems. In *2017 3rd IEEE International Conference on Cybernetics (CYBCONF)*, pages 1–8.

Bao, L. and Garcia-Luna-Aceves, J. J. (2003). Topology management in ad hoc networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 129–140. ACM.

Bastian, M., Heymann, S., Jacomy, M., et al. (2009). Gephi: an open source software for exploring and manipulating networks.

Carroll, J. J. and Klyne, G. (2004). Resource description framework ({RDF}): Concepts and abstract syntax.

Chen, C.-W. (2007). Dual redundant server system for transmitting packets via linking line and method thereof.

De Domenico, M., Solé-Ribalta, A., Gómez, S., and Arenas, A. (2014). Navigability of interconnected networks under random failures. *Proceedings of the National Academy of Sciences*, 111(23):8351–8356.

Dignum, M., Sonenberg, E., and Dignum, F. (2004). Dynamic reorganization of agent societies. In *Proceedings of workshop on coordination in emergent agent societies*.

Ferber, J., Gutknecht, O., and Michel, F. (2003). From agents to organizations: an organizational view of multi-agent systems. In *International Workshop on Agent-Oriented Software Engineering*, pages 214–230. Springer.

Foster, I. and Kesselman, C. (2003). *The Grid 2: Blueprint for a new computing infrastructure*. Elsevier.

Haider, S. and Ansari, N. R. (2012). Temperature based fault forecasting in computer clusters. In *Multitopic Conference (INMIC), 2012 15th International*, pages 69–77. IEEE.

Haider, S. and Nazir, B. (2016). Fault tolerance in computational grids: perspectives, challenges, and issues. *SpringerPlus*, 5(1):1991.

Hayashi, H. (2017). Comparing repair-task-allocation strategies in mas. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART,*, pages 17–27. INSTICC, SciTePress.

Kota, R., Gibbins, N., and Jennings, N. R. (2009). Self-organising agent organisations. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 797–804. International Foundation for Autonomous Agents and Multiagent Systems.

łgorzata Steinder, M. and Sethi, A. S. (2004). A survey of fault localization techniques in computer networks. *Science of computer programming*, 53(2):165–194.

Mathieu, P., Routier, J.-C., and Secq, Y. (2002). Dynamic organization of multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 451–452. ACM.

Nematollahi Mahani, M. (2012). *Strategic structural reorganization in multi-agent systems inspired by social organization theory*. PhD thesis, University of Kansas.

Serugendo, G. D. M., Gleizes, M.-P., and Karageorgos, A. (2011). *Self-organising software: From natural to artificial adaptation*. Springer Science Business Media.

Stamatopoulou, I., Gheorghe, M., and Kefalas, P. (2004). Modelling dynamic organization of biology-inspired multi-agent systems with communicating x-machines and population p systems. In *International Workshop on Membrane Computing*, pages 389–403. Springer.