A Context-Aware Model for the Management of Agent Platforms in Dynamic Networks

Phuong T. Nguyen[1](✉), Volkmar Schau[2], and Wilhelm R. Rossak[2]

[1] Research and Development Center, Duy Tan University, 182 Nguyen Van Linh, Danang, Vietnam
phuong.nguyen@duytan.edu.vn
[2] Department of Computer Science, Friedrich Schiller University Jena, Ernst-Abbe-Platz 2-4, 07743 Jena, Germany
{volkmar.schau,wilhelm.rossak}@uni-jena.de

Abstract. A network infrastructure in a mass casualty incident rescue scenario is normally characterized by multiple of working domains scattered over a wide area. For mobile agent systems working in these networks, the management of agent platforms contributes to achieving fault tolerance and reliability. We employ a honey bee inspired approach for imposing a self-organizing mechanism on the colony of mobile agent platforms. This paper presents our approach as well as introduces some preliminary evaluations of the proposed mechanism.

Keywords: Context-aware systems · Mobile agents · Bio-inspired computing

1 Introduction

For the support of mass casualty incident (MCI) rescue, at the University of Jena a project named SpeedUp[1] has been in development since April 2009. The project aims to develop a technological framework for providing support of rescue forces in MCI situations so that in disaster events, rescue tasks can be performed in a more effective way [1].

In SpeedUp's communication infrastructure the mobile agent concept [2] has been chosen as one of the key technologies. In MCI rescue scenario of the SpeedUp-Type, rescue forces may be distributed widely. Each geographical location forms most likely a technological region in which different forces work together to do rescue tasks (Fig. 1). It is expected that agent platforms are able to adapt to changes that happen at execution time. In this paper, we present an overview to our model, a honey bee inspired mechanism for the management of dynamic mobile agent platforms [6,7]. Afterwards, we introduce some preliminary evaluation results on the key functionalities of the model.

[1] The project was funded by the German Federal Ministry of Education and Research (BMBF), http://www.speedup-projekt.de.

P.C. Vinh et al. (Eds.): ICCASA 2013, LNICST 128, pp. 77–86, 2014.
DOI: 10.1007/978-3-319-05939-6_8,

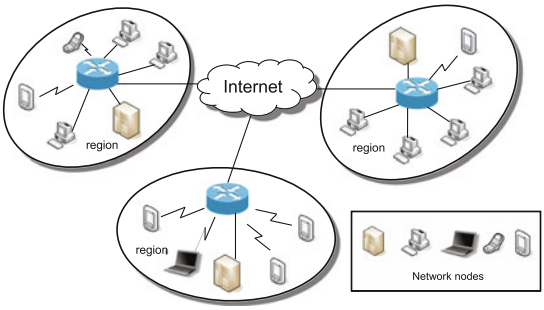

Fig. 1. Network infrastructure in an MCI rescue scenario

2 An Approach for the Management of Dynamic Agent Platforms in MCI Rescue Scenarios

2.1 Network Model

The SpeedUp solution is a communication and data platform for coordination and integration of all rescue teams in catastrophic situations [6]. Because of its characteristics, such as autonomous, reactive, opportunistic, and goal-oriented, the mobile agent technology has been selected for the SpeedUp-Type's MCI communication infrastructure [11]. The main components are as follows:

- Agent Platform: A software platform that provides executing environment for agents.
- Region: A region is made up of agent platforms that have the same authority.
- RegionMaster: An agent platform in a region that undertakes the context-aware tasks for the whole region.

One crucial issue in the SpeedUp context is to manage the dynamics of agent platforms efficiently as they join and quit in an unforeseeable manner. To achieve industrial strength, we propose a context-aware model for the management of agent platforms.

2.2 Biological Background

The main idea of bio-inspired computing in computer systems is to emulate activities in nature so that these systems can have alike features. As a result, they can react to environmental changes. So far, there have been several computational models inspired by colonies in nature [4,5,10]. Regarding the SpeedUp network model, we found some coincidental similarities between the model and the honey bee colony. In addition, honey bees already have good mechanisms to deal with their organizational issues.

In a bee colony, foraging takes place in summer, forager bees spread out to search for food sources. The forager bees may find different food sources, but as

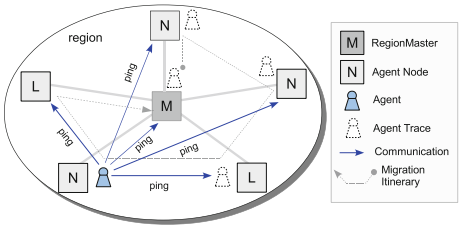

Fig. 2. A scout agent observes nodes and their mutual connectivity

the time goes by, honey bees tend to head for rich food source where they can get more nectar and pollen. A bee, after finding a new food source, returns to the hive and performs a *waggle dance* to notify other honey bees about the food source. As a result, other honey bees are able to know where and how far they must fly in order to get to the food sources [3,9].

Swarming in a bee colony occurs when the bee population increases. Before swarming the queen bee raises a new queen bee. The old queen takes a number of workers with her and they fly to a new temporary location, e.g. a tree branch. The most experienced foragers in the swarm, called scout bees, are then deployed to find new suitable locations. When one round of exploration is finished, each scout bee returns to the bee cluster to inform the whole colony about the place she has found. The scout bee then flies to the site and evaluates it again and then backs to the swarm to dance for it. Other bees either perform their own dance or watch or follow other dances. If the number of scout bees gathered in a site constitutes a quorum then the scouts make a decision, they choose the site as the new hive [3,9].

2.3 A Self-organizing Model for the Management of Dynamic Agent Platforms

We proposed an adaptive mechanism for the management of working regions in the SpeedUp context based on honey bees' activities. In the algorithm, two mappings from the honey bee colony to the SpeedUp's network model are employed.

Mapping 1: RegionMaster plays the role of a beehive, at regular intervals, it deploys scout agents to all platforms of the region to collect information related to connectivity between a platform and others, and platform's performance. When a scout agent arrives at a platform, it sends messages to all platforms to measure the latency between the current platform and other platforms. The process is illustrated in Fig. 2. Once all ping messages have returned, the platform calculates the average latency τ_{avg} as specified below:

$$\tau_{avg} = \frac{1}{n} \sum_{i=1}^{n-1} t(i) \tag{1}$$

In which n is the number of nodes in the region; $t(i)$ is the transfer time between the current platform and the i^{th} neighbour platform. The average

latency represents the level of closeness between the platform and its neighbourhood. A low average latency means the platform has a good connection quality to the remaining platforms. In contrast, a high latency relates to a degraded connection quality between the platform and the others.

A platform holds a boolean value *split* to indicate whether it expects its region to segregate or not. After τ_{avg} has been calculated, it will then be compared to the latency to RegionMaster τ_{RM}. If $\tau_{RM} \gg \tau_{avg}$ then the node expects the region to be splitted; it sets the value *split* to *true*. The platform hands out the two values τ_{avg} and *split* to the scout agent.

After performing its routine at a platform, the agent migrates to the next platform. The process repeats until all nodes of the itinerary have been visited. En route, the scout agent also nominates a candidate node as possible new region master based on platform's performance and connection quality. Once all platforms in its itinerary have been visited, the scout migrates back to RegionMaster to submit all information it has collected. A scout provides the knowledge of each node in its path it has visited by submitting information to RegionMaster like a scout bee dances to notify other bees of a food source. RegionMaster can then build a map of connection quality of the region.

Mapping 2 is inspired by bee swarming. In a region RegionMaster is considered as the queen bee and all others are worker bees. After visiting all nodes, scout agents go back to RegionMaster and submit the information they have collected. RegionMaster counts the number of values that satisfy $split = true$. If the number constitutes a quorum, the region is about to be splitted. RegionMaster promotes a new RegionMaster. The new RegionMaster forms a new region from the nodes it inherits. The two regions are independent from each other, but logically connected.

3 Implementation

The implementation is based on the open source multi-agent system Ellipsis which is developed by our workgroup at the Chair of Software Engineering of the University of Jena.

3.1 Network Monitoring

In the course of time, the performance of an agent platform as well as its connectivity to the neighbourhood can change. The monitoring process takes place at every platform at regular intervals to ensure that environmental stimuli occurring during execution are to be observed and processed adequately.

3.2 Network Organizing

After all nodes of a region have been vesicated, the information gathered by a scout needs to be processed and served as a base for decision making. Self-organizing activities are conducted to maintain an equilibrium between the internal organization of the platforms and the external perturbations. These activities allow agent platforms to recover to a stable state if changes or failures occurred.

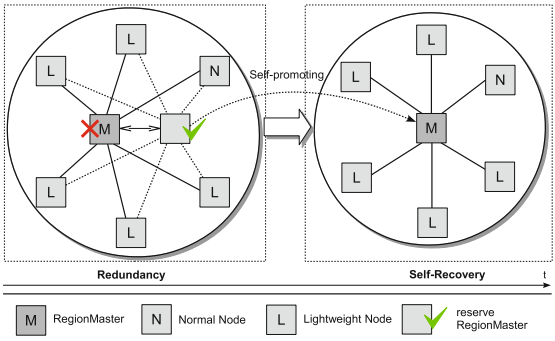

Fig. 3. Fault tolerance by using redundancy

3.3 Fault Tolerance

In the proposed approach, RegionMaster plays a decisive role, which may generate a single point of failure. Given that RegionMaster breaks down or disconnects suddenly, the node community has no information about the network and, therefore, cannot re-organize wherever necessary. To eliminate the effects of network failure, a reserve for RegionMaster is voted based on information fetched by the scouts. En route each scout agent compares information it has collected from the visited platforms, and connection quality of each platform, it nominates a reserve RegionMaster. The reserve RegionMaster senses the presence of RegionMaster by periodically sending pings to it. If RegionMaster is no longer available, the reserve RegionMaster replaces this node (Fig. 3).

4 Evaluation Parameters

For the software prototype, there are many important functionalities that need to be evaluated. However, due to space limitations, in this paper we concentrate on validating the feasibility and the ability of being context-aware. Interested readers are referred to [7] for more evaluation results.

4.1 The Feasibility

In a dynamic network environment the cost for monitoring network might be considerably high. The proposed mechanism is beneficial only if network monitoring gains a good performance whilst keeping a reasonable running cost. The feasibility of the proposed mechanism means that it maintains a reasonable cost for monitoring while providing necessary information for the self-organizing tasks. To validate the feasibility, scouts are deployed to monitor network, the parameters regarding processing time and exploration time are measured. One round of exploration happens when a scout is created, travels through all nodes of the region, performs its routine and migrates back to RegionMaster. The following information is going to be acquired:

– The time a scout needs for accomplishing its tasks at a platform: $t_{processing}$.
– The time a scout needs for performing one round of exploration: $t_{exploration}$.

The processing time at a platform is the duration from when an agent arrives until it completes its tasks and leaves for the next node. It is calculated as follows:

$$t_{processing} = t_l - t_a \tag{2}$$

where t_a is the time when a scout arrives at a platform and t_l is the time when it leaves for the next node of its itinerary. The processing time is:

$$t_{processing} = t_{RTT} + t_s + t_d \tag{3}$$

in which: t_{RTT} is the period from the first ping message sent until the last response received; t_d is the time to deserialize a scout agent from an incoming stream of byte; t_s is the time to serialize a scout agent into a byte stream. The average exploration time is computed:

$$t_{exploration} = t_{end} - t_{begin} \tag{4}$$

where t_{begin} and t_{end} are the time when a scout starts and completes one round of exploration, respectively. This parameter indicates how often a scout agent supplies a RegionMaster with up-to-dated information.

4.2 Context Awareness

For agent systems working in highly dynamic networks, being context aware is an important feature. Agent platforms should be able to take appropriate measures to counteract adverse effects happening to them. In the scope of this paper, we investigate the ability of the framework to detect and provide the system with adequate measures to deal with perturbations happening to RegionMaster.

5 Evaluation

5.1 Experiment Setup

To evaluate the software prototype, we build a laboratory scale test system where conditions of a real network are imitated, without a real scenario being present. Characteristics of a dynamic network are simulated using other softwares. The test network consists of eight computers connected through a local Gigabit Ethernet LAN 1000 Mbps (Table 1).

5.2 The Feasibility

Figure 4 shows the logical connection for the first test. In this scenario, one scout is created at RegionMaster. RegionMaster assigns the list of agent platforms $\{D1, D2, D3, D4, FJ, PT, T1, T2\}$ to the scout. Figure 5 depicts the processing

Table 1. Hardware configuration for the experiments

Computer	Alias	OS	Kernel	RAM	Processor
Desktop	D1	Fedora 12	2.6.31	2.0 GB	AMD 2.2 GHz
Desktop	D2	Debian 6.0.4	2.6.32	4.0 GB	Intel 2*2.0 GHz
Desktop	D3	Debian 6.0.4	2.6.32	4.0 GB	Intel 2*2.0 GHz
Desktop	D4	Debian 6.0.4	2.6.32	4.0 GB	Intel 2*2.0 GHz
Fujitsu	FJ	Fedora 12	2.6.31	3.0 GB	Intel 2*2.8 GHz
Portégé	PT	Windows 7	N/A	4.0 GB	Intel 2*2.4 GHz
Thinkpad	T1	Fedora 12	2.6.31	2.4 GB	Intel 2*2.4 GHz
Thinkpad	T2	Windows XP	N/A	1.0 GB	Intel 1.7 GHz

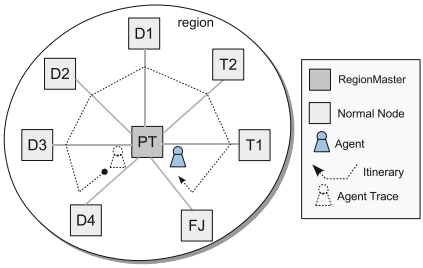

Fig. 4. Logical representation of the experiments

time of the platforms. This parameter represents the time that the agent needs to perform its tasks at a platform. It is dependent on the processing power of agent platforms and the latencies to the other platforms, which are in turn dependent on the network speed. It can be seen that, except $T2$ that has a higher processing time because of its limited processing power (Table 1), the processing time for the other platforms is considerably low. It guarantees that the processing activities place comparatively little burden on the system performance.

To measure $t_{exploration}$, the agent is sent around the network for different number of rounds r. In the second experiment r is set to different values, i.e. $r = \{100; 200; 300; 500; 1000; 2000; 3000; 5000\}$. The average exploration time is:

$$t_{exploration} = \frac{t_{end} - t_{begin}}{r} \tag{5}$$

This parameter demonstrates how fast the scout supplies RegionMaster with up-to-date information of the network. If the agent needs a long time to perform its tasks at the platforms, the information submitted to RegionMaster might be out-of-date. As a consequence, the reactions produced by RegionMaster would not be adequate. However, the measurement results show that this is not the case. In Fig. 6, the curve represents the accumulated exploration time. The straight line which depicts the average time for finishing one round of surveillance provides evidence that the parameter is stable, no matter how many rounds the

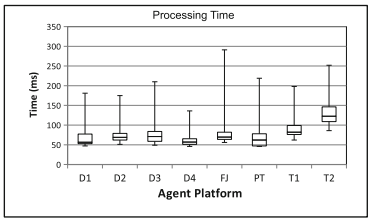

Fig. 5. Processing time for every platform

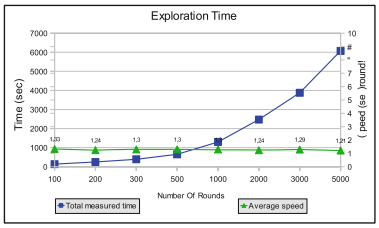

Fig. 6. Average time for a scout agent to explore the region

Table 2. Metrics measured at the time of self-organizing

Platform	D1	**D2**	D3	D4	FJ	PT	T1	T2
τ_{avg}/τ_{RM} (%)	47,8	47,6	37,5	27,5	—	37,5	44,4	29,7
Split	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*true*

agent has migrated. This means that the scout produces no overhead when it works in the long run.

5.3 Context Awareness

FJ is RegionMaster, a scout is deployed to survey the region. In this scenario, the connectivity between RegionMaster and the remaining platforms is degraded using software. This aims to investigate the countermeasures of the system given that the quality of the connection to RegionMaster has declined. Given the circumstances, it is expected that the software helps the platforms recover from the degradation.

In this experiment, a certain network traffic between RegionMaster and the other platforms is created, resulting in a smaller bandwidth left to the remaining platforms. To produce network traffic, we utilize the open source software Iperf [8]. This tool is used to produce both TCP and UDP data streams over networks; data sent by the client will be received and eventually discarded by the server. Since Iperf consumes a certain bandwidth on the connection between RegionMaster and the rest of the region, there is smaller bandwidth left for other applications. As a result, each platform experiences effects from the traffic generator. The latencies between the platforms and RegionMaster grow sharply. These changes are sensed by the scout. Every platform sets the value *split* based on the ratio $\Theta = \tau_{avg}/\tau_{RM}$; where τ_{avg} is the average latency and τ_{RM} is the latency to RegionMaster, respectively. The splitting threshold is set to $\Theta < 50\,\%$. Table 2 shows the ratio and the corresponding value *split* for every platform.

Since most of the platforms set the value *split* to *true* the region is splitted. There is only RegionMaster staying at the old region. The remaining nodes join the new region with $D2$ promoted to be the new RegionMaster. In this case, an

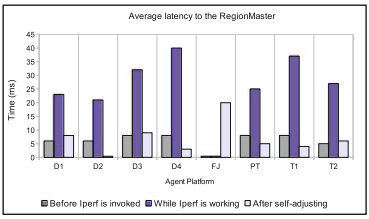

Fig. 7. Average latencies to RegionMaster

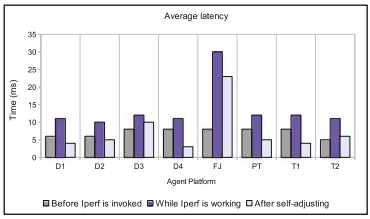

Fig. 8. Average latencies of all platforms

adaptation has been made, the old RegionMaster relinquishes its leadership in the only-one platform region and becomes an inferior node of the new region. A new region emerges from the original region.

Figure 7 displays the latency to RegionMaster τ_{RM} of every platform in three phases. For a platform, the left column is the latency before Iperf is activated; the middle column represents the time while Iperf is operating and the right column is the latency of the platform after the self-adapting process has occurred. In the beginning and while Iperf was working FJ was RegionMaster so $\tau_{RM}(FJ) = 0$. Similarly, after $D2$ has taken its job as RegionMaster $\tau_{RM}(D2) = 0$. As usual expected, while Iperf is working, the latencies to RegionMaster for every platform increase significantly. However, once $D2$ takes over as RegionMaster, the latencies decrease proportionally. Figure 8 shows the average latencies τ_{avg} of every platform in the corresponding phases. These latencies also shift in the same pattern as by τ_{RM}. Before additional bandwidth was produced, the average latencies had been at a normal level. While Iperf was operating the latencies rose markedly. After the region has been restructured, these values resume to a normal level. The network monitoring activities supply the framework with up-to-date information about network situation, thereby facilitating the decision making process. The measurement results suggest that the swap in role of RegionMaster from FJ to $D2$ brings a more stable connectivity to every platform compared to that of the old arrangement, right after Iperf started producing bandwidth.

6 Conclusion

In this paper we have introduced a mechanism for the management of agent platforms in highly dynamic networks based on the organizational model of honey bees.

Experimental results show that the software framework has an acceptable operating overhead as well as a practical processing speed. The test scenarios demonstrated that the framework is able to detect degradations in connectivity once they occurred. Based on the information gathered by scout agents, the framework provides the system with a measure to adequately overcome the

problem that adversely affects the platform colony. The countermeasures appear to be effective since they help the colony to promote a new equilibrium in connectivity between the platforms. The connection qualities from a platform to RegionMaster as well as from a platform to the others have been improved. From our perspective, the evaluation validates that the software framework principally fulfils the requirements regarding the feasibility and being self-adaptive.

References

1. FSU Jena, The SpeedUp Project (2011)
2. Braun, P.: The migration process of mobile agents. Ph.D. dissertation, Friedrich Schiller University, Jena (2003)
3. Seeley, T.D., Kirk Visscher, P.: Group decision making in nest-site selection by honey bees. J. Apidologie **35**, 101–116 (2004)
4. Karaboga, D., Bahriye, A.: A survey: algorithms simulating bee swarm intelligence. J. Artif. Intell. Rev. **31**, 61–85 (2009)
5. Pham, T., Afify, A., Koc, E.: Manufacturing cell information using the bees algorithm. In: Proceedings Innovative Production Machines and Systems Virtual Conference (2007)
6. Nguyen, P.T., Schau, V., Rossak, W.R.: Towards an adaptive communication model for mobile agents in highly dynamic networks based on swarming behaviour. In: The 9th European Workshop on Multi-agent Systems (EUMAS) (2011)
7. Nguyen, P.T., Schau, V., Rossak, W.R.: An adaptive communication model for mobile agents inspired by the honey bee colony: theory and evaluation. In: The 10th European Workshop on Multi-agent Systems (EUMAS) (2012)
8. Network Performance Measurement. http://sourceforge.net/projects/iperf/
9. Seeley, T.D., Buhrman, S.C.: Nest-site selection in honey bees: how well do swarms implement the best-of-N decision rule? J. Behav. Ecol. Sociobiol. **49**, 416–427 (2001)
10. Brocco, A.: Exploiting self-organization for the autonomic management of distributed system. Ph.D. thesis, University of Fribourg, Switzerland (2010)
11. Buford, J.F., Jakobson, G., Lewis, L.: Multi-agent situation management for supporting large-scale disaster relief operations. Int. J. Intell. Control Syst. **11**, 284–295 (2006)