Autonomous Distributed Systems and their Simulation

Valeriy Kolesnikov [0000-0002-1991-3614]

Sumy State University, Sumy, Ukraine

v.kolesnikov@cs.sumdu.edu.ua

Abstract. The development and actual testing of autonomous distributed systems is difficult, long, and expensive. Simulators can save time, the development and testing efforts, as well as bring the overall costs down. This paper describes an approach to designing and testing autonomous distributed systems. The design of such systems is based on organization model. For testing such systems, we developed DiSy-Sim simulator. Specifically, DiSy-Sim simulator simplifies testing of communication between system components by providing statistics on messages being used in communication. We demonstrate the use of the simulator for a swarm system of autonomous drones used for disaster relief aid delivery. DiSy-Sim simulator is applicable for a variety of distributed systems.

Keywords: autonomous distributed systems, swarms, simulation, organization model, DiSy-Sim, software engineering

CEUR-WS.org/Vol-2608/paper27.pdf

1 Introduction

Autonomous distributed systems have found their application in many areas. For example, such systems can enable autonomous drones to perform a task together as a team. A swarm of such drones can work towards a common goal. Each drone in a swarm is autonomous, but drones can communicate among themselves in a distributed manner.

A study of autonomous systems has become important especially in recent years as such systems can be used, for instance, for humanitarian aid and disaster relief, search and rescue and search and track of a target. To this end, swarm based autonomous systems should be self-directing, with reliable communication among its parts, resilient to various interferences and self-adjusting in the face of failures of some of its parts.

It is a difficult task to develop autonomous distributed systems that are robust, flexible and fault tolerant. This paper describes how such systems can be designed utilizing an organization model approach and tested with the help of DiSy-Sim simulator. These efforts are a part of a bigger ecosystem for design, development, verification, validation, and deployment of autonomous distributed systems that the author is working toward.

Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

The approach used in this study is the continuation of our efforts to model autonomous distributed systems utilizing the organization model. This approach provides a framework for defining goals that the organization exists to achieve, entities to achieve these goals and relationships among the entities, and finds its motivation in the Multiagent Systems Engineering methodology (MaSE) [1,2] to derive goals from the system description, and then design roles, capabilities and agents to achieve the organization goals.

For testing an autonomous distributed system with DiSy-Sim simulator we utilize the example of a swarm of autonomous drones for disaster relief aid delivery.

The remainder of this paper is organized as following: in section 2, we describe related work to designing and simulating autonomous distributed systems; next, we present an organization model approach to design autonomous distributed systems; we follow with an example of a swarm of autonomous drones for disaster relief aid delivery and present on organization instance for this example; next, we present DiSy-Sim simulator and discuss its application to the example of a swarm of autonomous drones for disaster relief aid delivery; finally, we end with conclusion and directions for our future work.

2 Related Work

We chose designing and implementing autonomous distributed systems using organization-based approach because research shows that: 1) organizational design is easier to understand as it is close to human organizations [3], 2) organization-based systems tend to be efficient and can efficiently adapt to changing conditions [3,4], 3) organizations are flexible [3,4], and 4) best designs utilizing organization-based approach are application and situation dependent [5]. Organizations can be centralized or decentralized. In a distributed scenario, decentralized organizations tend to exhibit higher performance and reliability [1].

For testing purposes, there are many simulators available that range from general in purpose to specific contexts. A performance comparison of recent network simulators is given in [6]. In our previous research [7-9] we studied and used NS2, NS3 [10] and J-Sim [11] simulators extensively.

NS3 a discrete-event network simulator for internet systems. It is built using C++ and Python and provides scripting capability. The following are the key strong features of NS3: topology definition, model development, node and link configuration, execution environment with logging capability, performance analysis and graphical visualization. Four main drawbacks for us in using NS2 or NS3 are: 1) these are network simulators (they don't capture application layer communications), 2) they cannot simulate the movement of application entities over time, 3) they are written in C++ and Python (the language of choice for our ecosystem for design, development, verification, validation and deployment of autonomous distributed systems is Java) and 4) we want our simulator to be integrated in our ecosystem briefly described below.

J-Sim simulator solves two of the issues. First, it is written in Java. Also, it can be used in almost any system, in which object states change discretely and therefore capture application layer communications. The basic building blocks of J-Sim simulations are processes and queues. Whereas processes are active, queues and other elements of the simulation are passive. The simulation is executed step by step. During each step, only one process is given an opportunity to run. J-Sim provides a rich set of features, such as simulation of Dijkstra semaphores, simulation of message passing (blocking and non-blocking send and receive operations, symmetric, asymmetric, and indirect communication), independent random number generators that can be initialized by user-defined seed and therefore guarantee repeatability of experiments and console mode and batch and interactive GUI modes [11]. Still, J-Sim cannot simulate movements of application entities over time. Integration with InDiGO framework remains the main hurdle. Therefore, we opted to create our own simulator and add additional features valuable for InDiGO framework and for our ecosystem in general.

Of interest for us is a framework or a whole ecosystem for the ease of developing, testing, and deploying autonomous distributed systems. Several attempts have been done at creating and formalizing systems or frameworks to capture the concept of teamwork within an organization [12-16]. Although valuable for design of autonomous distributed systems, they fail to provide a wide accepted use.

In our previous work we proposed InDiGO, an infrastructure for the development of distributed systems [7]. InDiGO allows design of generic but customizable algorithms and provides tools to customize such algorithms for distributed applications [7,17]. We demonstrated advantages of using such a framework for distributed systems development [7-9] and plan to merge this research into InDiGO framework.

3 Organization Model

To ease the design of autonomous systems, we propose an organization model (O). This model describes the structure of the organization and combines all entities of the autonomous system and the relationships among them. Fig. 1 shows organization model using standard UML notations.

Organization model O contains a set of goals (G) that the system is trying to achieve, a set of roles (R) that achieve the stated goals, a set of agents (AG) that are capable of playing specific roles, a set of capabilities (C) that agents possess and that are required by the roles to achieve the goals, a set of assignments (A) that the organization performs to assign agents to play specific roles to achieve specific goals and a set of policies (P) that constrain the assignments.

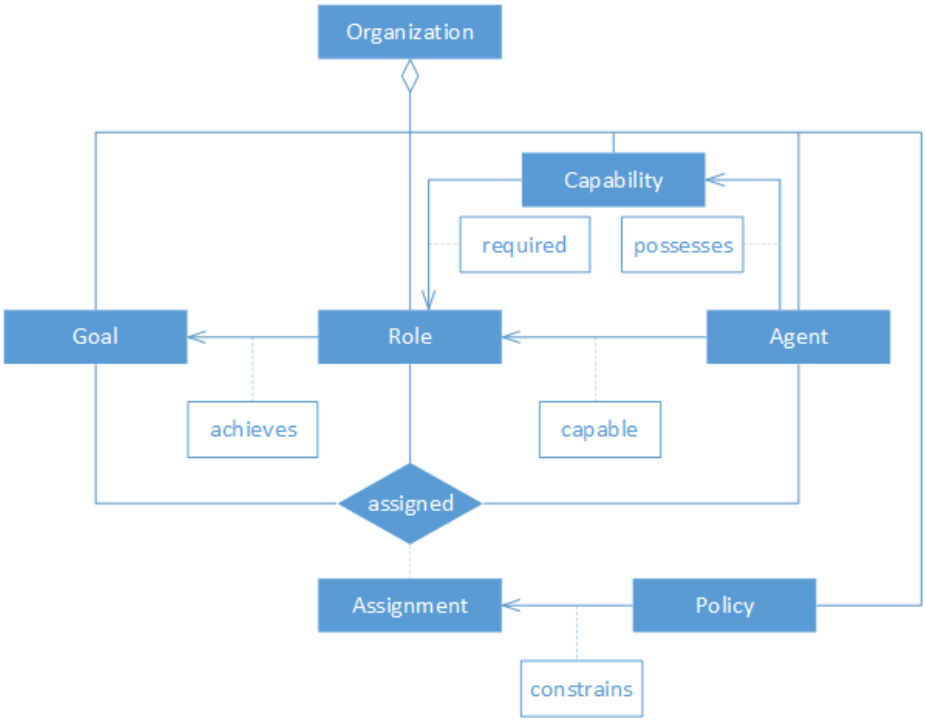

Fig. 1. Organization model

The organization model also describes relationships that exist among entities of the autonomous system. Among them are the following relationships: achieves relationship that exists between roles and goals, capable relationship that exists between agents and roles, possesses relationship that exists between agents and capabilities, required relationship that exists between capabilities and roles, constrains relationship that exists between policies and assignments and assigned relationship that exists between agents, roles and goals.

Formally, organization model O is a tuple.

$$O = < G, R, AG, C, A, P, achieves, capable, possesses, required, constrains, assigned > \quad (1)$$

where

$$achieves: R \times G \rightarrow Boolean \quad (2)$$

$$capable: AG \times R \rightarrow Boolean \quad (3)$$

$$possesses: AG \times C \rightarrow Boolean \quad (4)$$

$$required: C \times R \rightarrow Boolean \quad (5)$$

$$constrains: P \times A \rightarrow Boolean \quad (6)$$

$$assigned: AG \times R \times G \rightarrow Boolean \quad (7)$$

Every autonomous system is built to achieve certain goals. In our model each goal has a definition. All goals are placed in a set G. Goals can be critical or noncritical. Function critical determines whether a goal is critical. At this point we do not consider various relationships among goals, for example, such as subgoals, parent and child goals, conjunctive and disjunctive goals and precedence between goals and leave this to our future work.

$$critical: G \rightarrow Boolean \quad (8)$$

The model is said to be complete if all the goals can be achieved. The model is said to be incomplete if at least one goal cannot be achieved. It is certainly desirable for a model to be complete and in some cases it is absolutely necessary. There are noncritical situations though, in which a model can be incomplete but sufficient for a system. We will research such scenarios in our future work.

A goal is directly achieved by a role or a set of roles that each agent is capable of playing. Therefore, each goal is indirectly achieved by an agent or a group of agents through the roles, which those agents are capable of playing. An agent can be capable of playing several roles depending on the inherent capabilities that the agent possesses. Each role requires one or several capabilities that agent/s possess.

Completeness of the model is validated through assignments. Assignments are used to capture information about which agents play which roles to achieve which goals. Assignments are constrained by the organization policies that dictate which assignments are allowed and which are not.

Along with general constraints that each system must satisfy, an organization might impose additional constraints through policies. An example of such a policy constraint could be one that states that certain roles must be achieved by several agents to provide redundancy. This redundancy might be important for agent's capabilities that degrade under certain conditions, such as capabilities that depend on a battery life for autonomous drones.

4 Organization Instance

In this section we present examples of what various entities of an organization model could be in a case of a system of autonomous drones and then apply organization model approach to designing a swarm system of autonomous drones that are used for disaster relief aid delivery.

Examples of agents could be autonomous drones, parts of autonomous drones or whole groups or swarms of autonomous drones.

Examples of capabilities are data acquisition capabilities, data access capabilities, data manipulation capabilities, data transmission capabilities, computational capabilities, sensor capabilities, GPS capabilities, etc. During the system life span capabilities can degrade or improve. For example, sensors usually degrade over time and some-

times their accuracy becomes an issue. Another example of a degrading capability, albeit for possibly a short period of time, would be a limited vision capability during a cloudy or rainy weather. On the other hand, learning algorithms can improve a certain capability. In these cases, the system reorganization could be advisable or necessary.

Each organization has a set of policies that constrain the assignment of agents to roles to goals. These policies could be general in nature or very specific to the autonomous system. An example of a general policy could be a policy that states that each role must be played by at least one agent. An example of a system specific policy could be a policy that states that no fewer than three agents ought to be involved in working on a specific goal.

4.1 A Swarm System of Autonomous Drones Used for Disaster Relief Aid Delivery

In this section we apply the organization model approach to designing a swarm system of autonomous drones that are used for disaster relief aid delivery. This example is intentionally kept to a minimum for simplicity and clarity of presentation.

Defining Goals. The purpose of this system is to deliver disaster relief aid to the disaster area. Therefore, we define the following two goals for this system:

- G1 – deliver disaster relief aid.
- G2 – navigate to site.

Defining Roles. Goals are achieved by roles that various agents play. For simplicity we define only two roles that will be directly mapped to achieve the two goals stated above. In general, an organization can have many roles that achieve various goals.

- R1 – carrier.
- R2 – navigator.

Defining Capabilities. Each role requires a set of capabilities that an agent must possess in order to be capable to play the role. For this system we define four capabilities and list them below:

- C1 – carrying capability.
- C2 – navigation capability.
- C3 – data processing capability.
- C4 – communication capability.

Defining Agents. We also define seven agents with various capabilities and list them below:

- A1 - A4 – carrier agents.
- A5 – navigator agent.
- A6 – A7 – data processing agents.

Organization Instance. After we define goals, roles, capabilities and agents, the organization algorithms make assignments taking into consideration policy constraints to create an organization instance that would achieve all the stated goals. We call such an organization instance a system. The organization also produces information to the user whether the system is complete. If a complete system cannot be formed, the organization tries to create a viable system. In all cases, initial system has to be valid.

An organization instance for our example is shown in Fig. 2. It is both complete and valid. It is complete because all goals are achieved. It is also valid because all organization policies are satisfied.

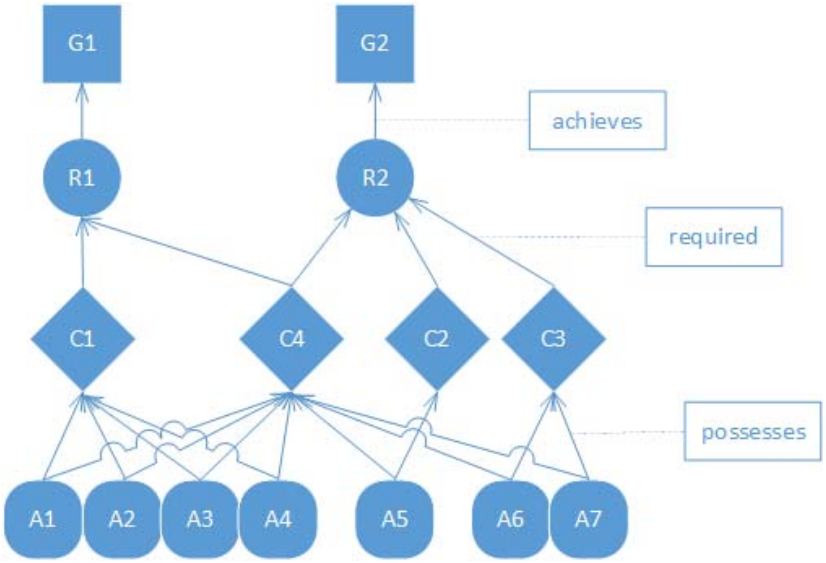

Fig. 2. Organization instance

5 Simulator

We developed a simulator DiSy-Sim (*Di*stributed *Sy*stems *Sim*ulator) to test message passing algorithms in middleware and application layers of autonomous distributed systems, as well as to examine the work of a system in real time. The statistics allows distributed systems developers and testers to measure the efficiency of their message passing algorithms and compare various optimization techniques. The ability to see the work of a system in real time allows developers and testers to bring their development cost down and improve the overall quality. The end users of actual physical systems can use the simulator to make decisions on the quantity and the characteristics of the system entities to achieve the goals for the system.

5.1 Simulator goals

We set the following goals for the creation of DiSy-Sim simulator:

1. Ability to model message passing in a distributed system of autonomous entities.
2. Ability to provide statistics in terms of number of messages used by the algorithms and application itself; statistics should be easily available:
   (a) for the whole system,
   (b) for a particular algorithm,
   (c) for a set of algorithms,
   (d) for application layer,
   (e) for a single entity,
   (f) for a group of entities,
   (g) for an entity type.
3. Simulation of the system in real time.
4. Graphical representation of messages over time.
5. Logging.

5.2 Simulator components

DiSy-Sim simulator was written in Java and tested on a Windows platform. Its main control window contains 3 panels as can be seen in Fig. 3: system settings/system view panel, simulation view panel and graphical stats panel. The panels can be resized within the main window. Each panel can also be shown in a separate window. Simulation view and graphical stats panels can be duplicated to show different parts of the system/statistics. A separate window for a single system entity can be opened by selecting the entity from either the simulation view panel or system view panel. Multiple windows for multiple entities can be opened concurrently.

5.3 Simulating organizational instance in DiSy-Sim

Below in Fig. 3 is a screenshot from the simulation in DiSy-Sim of the organizational instance described above.

Seven agents A1-A7 are moving to their destination. Communication protocols are based on the agents' roles and the message statistics is shown in the graphical statistics panel. The whole system consists of seven agents as can been seen from the system view panel. The agents' arrows show the direction, in which the drones are moving. The system successfully logged messages used by all the entities in the process of system execution.

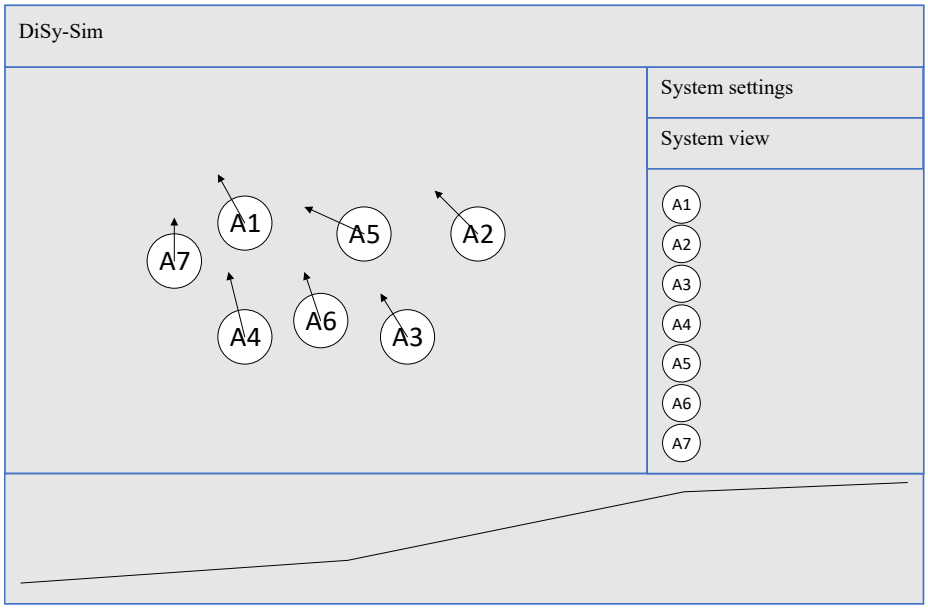

Fig. 3. DiSy-Sim simulator main window

To check the correctness of message calculation, we tested the system on a simple communication protocol that consisted of one navigator drone sending a broadcast message to all other drones during every time increment and other drones responding with an acknowledgement message sent back to the navigator drone upon receiving the navigation message. We tested the system using 100-time increments. The system correctly calculated the number of messages as can be seen in Fig. 4.

We also simulated this scenario with a different number of drones in a swarm. The system correctly calculated the number of messages in each test runs as can be seen from Fig. 5.

5.4 Applicable contexts

The DiSy-Sim simulator was designed for autonomous distributed systems, specifically for swarms of drones modeled using organizational approach. However, the simulator was designed for a general case of autonomous component based distributed systems and is applicable for such distributed systems as: sensor networks, distributed applications running on fixed computers/devices and distributed applications running on moving devices (ex., mobile phones), to name a few. We will demonstrate it with case studies in our future work.

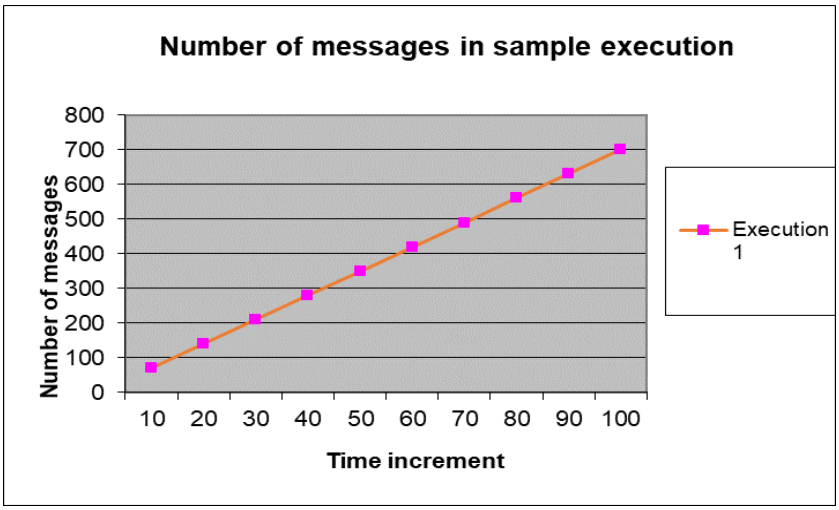

Fig. 4. Number of messages in sample execution

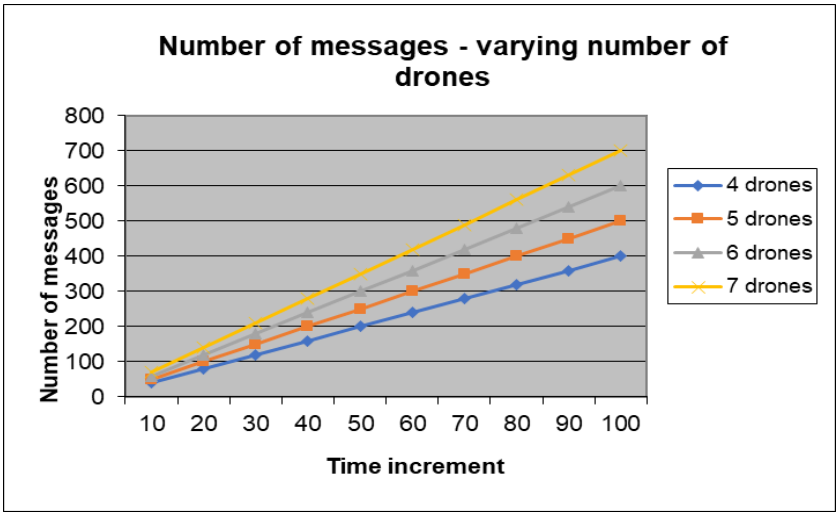

Fig. 5. Number of messages – varying number of drones

6 Conclusions and Future Work

In this paper, we presented an approach to designing and testing autonomous distributed systems. The design of such systems is based on organization model. This ap-

proach simplifies modeling of such systems and allows for better understanding of the system overall.

For testing autonomous distributed systems, we developed DiSy-Sim simulator. DiSy-Sim simulator simplifies testing of communication between system components by providing statistics on messages being used in communication.

The approach to designing and testing autonomous distributed systems presented in this paper is applicable for a variety of distributed systems. We demonstrated it for a swarm system of autonomous drones used for disaster relief aid delivery. Even though the example of an organizational instance for a swarm system of autonomous drones used for disaster relief aid delivery is simplified, it still shows the benefits of using such an approach.

In future work, we plan to enhance the organization model with additional features to account for failures in the system and the reorganization to take place.

We will add additional features to DiSy-Sim simulator, such as: single entity view with stats, replay capability, various terrains/obstacles for moving entities and failure simulation (loss of messages, failure of entity/equipment, battery power, etc.). We will apply DiSy-Sim simulator to other contexts, for instance, for sensor networks, distributed applications running on fixed computers/devices and distributed applications running on moving devices (ex., mobile phones).

We also plan to merge organization-based approach to modelling autonomous distributed systems into InDiGO framework and provide tools to extract optimization information from the model for middleware distributed algorithms of a distributed systems. We also have plans to research a possibility of expressing communication protocols through the organization-based modelling so that optimizations for communication in a distributed system could be realized based on the information from a design stage of a distributed system.

References

1. DeLoach, S.A., Wood, M.F., Sparkman, C.H.: Multiagent Systems Engineering. The International Journal of Software Engineering and Knowledge Engineering 11(3): 231–258 (2001)
2. DeLoach, S.A., Wood, M.F.: Developing Multiagent Systems with agentTool. In: Agent Theories Architectures and Languages, 7th International Workshop. LNCS, vol. 1986, Springer Verlag, Berlin (2001)
3. Carley, K.M.: Computational and Mathematical Organization Theory: Perspective and Directions. Computational and Mathematical Organization Theory 1(1): 39–56 (1995)
4. Carley, K.M.: Organizational Adaptation. Annals of Operations Research 75: 25–47 (1998)
5. Lawrence, P.R., Lorsch, J.W.: Organization and Environment: Managing Differentiation and Integration. Journal of Artificial Intelligence Research 7: 83–124 (1997)
6. Weingartner, E., Lehn, H., Wehrle, K.: A Performance Comparison of Recent Network Simulators, In: 2009 IEEE International Conference on Communications, pp. 1–5. Dresden (2009)

7. Kolesnikov, V., Singh, G.: InDiGO: An infrastructure for optimization of distributed algorithms. In: 7th International Symposium on Parallel and Distributed Computing, pp. 401–408 (2008)
8. Kolesnikov, V., Singh, G.: Utilizing model checking for automated optimization information discovery in InDiGO. In: 8th International Symposium on Parallel and Distributed Computing, pp. 91–98 (2009)
9. Kolesnikov, V.: Realizing optimization opportunities for distributed applications in the middleware layer by utilizing InDiGO framework. In: 9th International Symposium on Parallel and Distributed Computing, pp. 85–92 (2010)
10. Kumar, A.R., Rao, S.V., Goswami, D.: NS3 Simulator for a Study of Data Center Networks, In: 12th IEEE International Symposium on Parallel and Distributed Computing, pp. 224–231 (2013)
11. Sobeih, A., Hou, J., Kung, L., Li, N., Zhang, H.: J-Sim: a simulation and emulation environment for wireless sensor networks. Wireless Communications 13: 104–119 (2006)
12. Cohen, P.R., Levesque, H.J.: Intention is Choice with Commitment. Artificial Intelligence 42(3) (1990)
13. Jennings, N.R.: Commitments and Conventions: The Foundation of Coordination in Multiagent Systems. The Knowledge Engineering Review 8(3): 223–250 (1993)
14. Jennings, N.R.: Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems Using Joint Intentions. Artificial Intelligence 75(2): 195–240 (1995)
15. Grosz, B., Kraus, S.: Collaborative Plans for Complex Group Action. Artificial Intelligence 86(2): 269–357 (1996)
16. Kinny, D., Ljungberg, M., Rao, A.S., Sonenberg, E., Tidhar, G., Werner, E.: Planned Team Activity. Artificial Social Systems 830: 226–256. LNAI, Springer-Verlag (1992)
17. Singh, G., Kolesnikov, V., Das, S.: Methodologies for optimization of distributed algorithms and middleware. In: The 22nd IEEE International Parallel and Distributed Processing Symposium (2008)