A DISTRIBUTED STAND-IN AGENT BASED ALGORITHM FOR OPPORTUNISTIC RESOURCE ALLOCATION

Petr Benda, Pavel Jisl
{bendap1,jisl}@labe.felk.cvut.cz

Department of Cybernetics, Czech Technical University in Prague, Technicka 2, Prague 6

Abstract: Mobile ad-hoc networks (MANET) are expected to form the basis of future mission critical applications such as combat and rescue operations. In this context, communication and computational tasks required by overlying applications will rely on the combined capabilities and resources provided by the underlying network nodes. This paper introduces an integrated **FlexFeed**/$\mathcal{A}$-**globe** technology and distributed algorithm for opportunistic resource allocation in resource- and policy-constrained mobile ad-hoc networks. The algorithm is based on agent negotiation for the bidding, contract and reservation of resources, relying primarily on the concept of remote presence. In the proposed algorithm, stand-in Agents technology is used to create a virtual, distributed coordination component for opportunistic resource allocation in mobile ad-hoc networks.

Keywords: agents, multi-agent systems, ad-hoc networks, inaccessibility, defence applications

1. INTRODUCTION

Communications in military battlefield operations are currently one of the most critical technical capabilities for mission success. From a network perspective, tactical military operations are often characterized by highly dynamic ad-hoc wireless environments and include heterogeneous nodes under resource and security constraints.

Furthermore, the communications infrastructure is expected to change its behavior and optimization criteria to adapt to changes in goals and priorities. In recent years, a number of research efforts have focused their attention on this problem, looking for better routing or transport algorithms that would correct the deficiencies observed in the use of traditional wired network protocols.

In this paper, we introduce a novel agent-based communications framework designed to help address this issue in these types of environments. The goal is to provide a framework that will overlay the physical network and transparently provide both services, with minimal changes in current software applications and systems. In our framework, intelligent software agents are used to enable on demand, data-aware capabilities in the network. The agents are mobile, so code and computation can be moved as necessary to opportunistically create capabilities and to react to changes in topology, resource availability, and policies. The framework proposed in this paper leverages from a set of core technologies that have been designed and developed by our research teams for similar types of scenarios. Extensively tested in numerous proof-of-concept applications and demonstrations, these technologies have matured to a point where they complement each other to enable the framework.

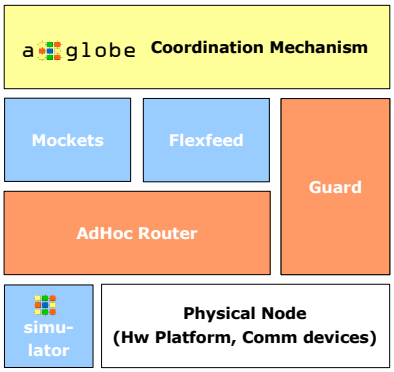

Fig. 1. **FlexFeed**/$\mathcal{A}$-**globe** architecture design

2. INTEGRATED ARCHITECTURE

The **FlexFeed**/$\mathcal{A}$-**globe** integrated infrastructure provides a powerful environment supporting point-to-point messaging, publish-subscribe data streaming, on-demand, opportunistic resource allocation in the dynamics of the environment that is given by (i) mobility of the communication infrastructure (ad-hoc networking) and (ii) changing operational priorities and optimization criteria.

For simulation purposes we use simulator, based on $\mathcal{A}$-**globe** platform (Šišlák*et al.*, 2004) (see section 3). It simulates the behavior of real hardware and is extended with geographical and environment simulators. We can simulate the physics of units, their movements and dynamically add and remove the units.

The architecture has been designed to operate with the real-life hardware (robots, unmanned vehicles or communication devices) and it consists of several mutually linked components (see Figure 1):

- $\mathcal{A}$-**globe simulator** – Due to the fact that the real-life experiments can be costly and in some cases it can be impossible to implement, an inseparable component of this architecture is the agent-based simulator (presented in section 3) that models the communication and interaction environment.
- **ad-hoc Router** – The simulator and/or real hardware is connected with highest levels of architecture over ad-hoc Router. Ad-hoc router supports multiple routing algorithms and is implemented at application level of ISO/OSI specification (ISO, 2005). For connected applications the ad-hoc router acts as normal UDP sockets. This extension provides restrictions in connection between units and components depending on visibility or policies.
- **Mockets** – Mobile Sockets are an application level transport layer, designed to transparently provide resource redirection and cross-layer interaction in mobile ad-hoc network environments.
- **FlexFeed** – Technology used for stream-oriented communication between agents. While **FlexFeed** is designed for stream planning and controlling, Mockets provide an infrastructure used for stream transmission.
- **Guard** – This component is used for controlling the resource allocation and policy checking. Extended with KAoS (Bradshaw, 1997; Bradshaw *et al.*, 1999), the policy framework for **FlexFeed** and NOMADS, it can provide the mechanism for definition, verification, distribution and enforcement of policies restricting agent access to sensor data, bounding agent resources and governing the mode of notification to users.
- $\mathcal{A}$-**globe coordination** – The above listed components are closely linked to $\mathcal{A}$-**globe** coordination mechanism. This is a critical component which provides both central and distributed high-level planning and resource allocation algorithms for coordinating the communication flows. The coordination mechanisms represent the main contribution of this paper and presented in sections 3.2 and 4.1.

The **FlexFeed**/$\mathcal{A}$-**globe** integrated architecture was designed on background knowledge of complex frameworks, designed in IHMC (eg. **FlexFeed**, KAoS, NOMADS) and the $\mathcal{A}$-**globe** project, designed in Gerstner Laboratory at Czech Technical University. Both technologies have its advantages and disadvantages and their connection can provide easy implementable architecture for mobile agents in environments with constrained accessibility.

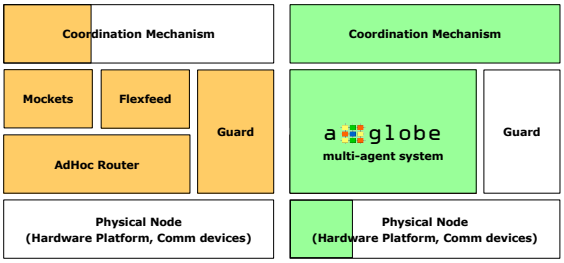

Fig. 2. IHMC provided (left) and Gerstner Laboratory provided (right) components

The provided technologies are complementary and in part overlapping (see Figure 2). The GL provided $\mathcal{A}$-**globe** multi-agent platform that implements its own messaging and routing services and supports full agent migration. This is why it can replace the infrastructure provided by **FlexFeed**, Mockets and the ad-hoc router. The policy enforcement is provided only by Guard and is not implemented in $\mathcal{A}$-**globe**. As the $\mathcal{A}$-**globe** component is in charge of the coordination mechanisms, in **FlexFeed** only a centralized coordination is

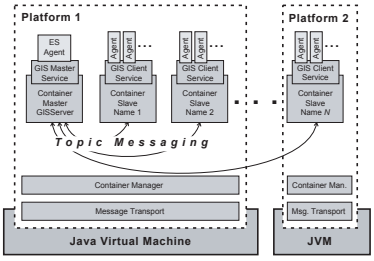

Fig. 3. $\mathcal{A}$-**globe** simulation architecture

implemented, using the ULM algorithm (Carvalho *et al.*, 2005). This centralized coordination is implemented in $\mathcal{A}$-**globe** as well, however $\mathcal{A}$-**globe** also provides partially distributed and fully distributed coordination algorithms and motion oriented coordination planning. The physical nodes can be used in both architectures, but $\mathcal{A}$-**globe** provides the support for simulation of these real-life physical nodes (discussed in section 3).

3. $\mathcal{A}$-GLOBE SIMULATION

$\mathcal{A}$-**globe** is Java-based, fast, scalable and lightweight agent development platform with environmental simulation and mobility support. Beside the functions common to most agent platforms (such as JADE, COUGAAR, FIPA-OS or JACK agent platform) it provides a position-based messaging service, so it can be used for experiments with extensive **environment simulation** and **communication inaccessibility**. Communication in $\mathcal{A}$-**globe** is very fast and the platform is relatively lightweight (Šišlák*et al.*, 2004).

$\mathcal{A}$-**globe** is an agent platform designed for testing experimental scenarios featuring agents position and communication inaccessibility, but it can be also used without these extended functions. The platform provides functions for the residing agents, such as communication infrastructure, store, directory services, migration function, deploy service, etc.

$\mathcal{A}$-**globe** platform is not fully compliant with the FIPA specifications, e.g. it does not support inter-platform communication. This interoperability is not necessary when developing closed systems, where no communication outside these systems is required (e.g. agent-based simulations).

$\mathcal{A}$-**globe** is suitable for real-world simulations including both static (e.g. towns, ports, etc.) and mobile units (e.g. vehicles). In such case the platform can be started in extended version with Geographical Information System (GIS) services and Environment Simulator (ES) agent. The ES agent simulates dynamics (physical location, movement in time and others parameters) of each unit.

The platform ensures the functionality of the rest of the system using these main components (see fig. 3):

- **Container Manager.** One or more agent containers can run within single agent platform. Container Manager takes care of starting, execution and finishing of these containers. Containers are mutually independent except for the shared part of the message transport layer. Usage of single agent platform for several containers running on one computer machine is beneficial because it rapidly decreases system resources requirements (use of single JVM), e.g. memory, processor time, etc.
- **Message Transport.** The platform-level *message transport* component ensures an efficient exchange of messages between two agent containers running in a single agent platform (single JVM).
- **Agent Container.** The agent container hosts two types of entities that are able to send and receive messages: agents and services. Agents do not run as stand-alone applications. Instead, they are executed inside the agent containers, each agent in its own separate thread. The schema of general agent container structure is shown in Figure 3. Container provides the agents and services with several low level functions (message transport, agent management, service management).

3.1 $\mathcal{A}$-globe simulation support

$\mathcal{A}$-**globe** is suitable for real-world simulations including both static and mobile units (e.g. logistics, ad-hoc networking simulation), where the core platform is extended by a set of services provided by a Master GIS (Geographical Information System) container hosting various environment simulator (ES) agents.

For integration with **FlexFeed** infrastructure the Geographical Information (GIS) Server and Environment Simulator were adopted from the NAIMT project (Rollo *et al.*, 2005) and extended for **FlexFeed** framework.

Simulation Server integrates the GIS Server and Environment Simulator into one server, which provides the simulation of environments. Simulation server handles logins and logouts of environments, computes visibility among these environments and handles movements of environments.

Environmental Simulator connects the $\mathcal{A}$-**globe** platform with **FlexFeed** framework. It provides the **FlexFeed** with environment simulation information like positions of environment, informs

about other visible environments and can be used for sending movement commands to Simulation Server.

3.2 $\mathcal{A}$*-globe/FlexFeed Coordination*

The proposed integrated architecture proposes four different approaches to coordination:

- **centralized** – A single process (coordinator) searches *F-Graph* – a centralized knowledge structure that represents communication accessibility in the network. By means of Dijkstra-like algorithm, the coordinator finds the optimal feed. This centralized coordination is implemented in **FlexFeed** and uses the ULM algorithm (Carvalho *et al.*, 2005).
- **partially decentralized** – The centralized approach can be partially distributed among regions, where the coordinators maintain their own *F-Graphs* representing communication accessibility particular region. For planning feed and coordination, the region coordinators need to negotiate the path for routing to another region.
- **fully decentralized** – In fully decentralized coordination approach each agent maintains a local *F-Graphs* representing communication accessibility in its direct neighborhood. This information is used for planning and coordinating the feed in peer-to-peer manner.
- **motion enforcement coordination** – Yet another approach to coordination is used in the situations with higher level of communication inaccessibility among the agents. This approach plans and controls not only the communication traffic but also movement and change of location of the particular nodes (robots).

At this moment the **FlexFeed** framework implements only the centralized coordination. In $\mathcal{A}$-**globe** simulation scenarios the partially and fully decentralized coordination as well as the motion enforcement coordination were implemented in the NAIMT (Rollo *et al.*, 2005) project.

In the remaining part of this document we will be discussing the fully decentralized approach to coordination. In principle, there are two fundamental approaches how such coordination can be implemented – by means of the *remote awareness* or *remote presence* agent concepts. While in the former case, each node is expected to autonomously maintain the information about the accessible nodes and perhaps their accessibility in the *acquaintance model*, in the latter case each individual node creates so called *stand-in agents*. The stand-in agents represent its owner and are disseminated across the network in order to represent its owner accessibility.

4. FULLY DISTRIBUTED COORDINATION

4.1 Stand-in Agents

In standard situations, agents in a multi-agent community need to communicate with each other to accomplish cooperation and coordination of joint activities. Normally it is expected that each agent can contact any other using a reliable communication infrastructure. Often one has to think about situations when a communication subsystem becomes disintegrated and some agents become isolated from the rest of the community. Stand-in agents solve inaccessibility in two ways: the first is the routing communication protocol based on swarming and micropayments within agent community and the second is distribution of social knowledge. In this paper we talk about stand-ins without social knowledge functionality (Šišlák*et al.*, 2005), because we want to build only a message passing system there. These stand-ins provide top-level communication API in mobile network.

An important attribute of the stand-ins is their passive role in the network. These agents are meant to be carried on a physical device and they aren't able to affect position of the device in mobile network anyway.

We will address the integration of the algorithm with the generic stand-in agent architecture. Unlike classical middle agent architectures (Sycara *et al.*, 1999) where the prime functionality is devoted towards matchmaking and negotiation, we would like to extend the concept of the middle agent by its capability to autonomously migrate in the network, clone and destruct copies. We present an abstract architecture of the stand-in agent, that is composed of the following components

- **Swarming controller** – consists of two modules: population manager ensures cloning, migration and destruction of stand-in agents in the system while the information propagator manages information flows through the agent, more specifically the messages or knowledge to transfer or actions to take.
- **Knowledge base**, a domain specific knowledge structure of the stand-in agent, consists of three parts: activity knowledge, information evaluator and timeout checker. While the activity knowledge contains the domain specific knowledge and the meta-data provided by the propagator, the information evaluator and timeout checker are the algorithms working on this knowledge. The information evaluator classifies and indexes the knowledge, so that the index values can be used by information propagator to manage its activity and further propagation. It also evaluates the knowledge usefulness. The

timeout checker module implements forgetting of the activity knowledge.
- **Stand-in agent functionality** – universal interface between modules and agent platform. It provides fundamental agent functions (clone, migrate and die), message interface and monitoring listeners, as well as original stand-in agent code. This code depends on the actual type of the stand-in agent. Only this part of relay agent needs to be changed to work properly with another agent platform.

4.2 Stand-in based Coordination

One of the key issues in stand-in operation is their proper location in the network. The distributed stand-in agent allocation mechanism uses only locally accessible information. It does so not only to minimize the network maintenance communication, but it also enables operation in the disruptive or partially inaccessible environment. Locally accessible information is obtained by monitoring stand-ins neighborhood – identifying currently visible targets and other stand-in agents.

In principle there are two key approaches to controlling the efficiency of the stand-in agents allocation:

(1) **forward swarming control** – where the stand-in agent migrates its clone only to the locations with higher possibility of future inaccessibility and higher interaction expectancy and
(2) **backward swarming control** – where the stand-in agents dispatch their clones to every reachable destination and the useless ones are eliminated in the future, reflecting the actual state of inaccessibility.

Each of the approaches has its advantages and disadvantages. The forward swarming control is computationally efficient, as it tries to minimize the number of stand-in agents in the system and prevent the possible swarming explosion. This is why this approach seems to be particularly suitable for domains with high scalability and operational efficiency requirements. On the other hand, the backward swarming control has an important advantage. This approach is substantially more domain independent, demands less knowledge about the environment nature and is more robust, as it doesn't explicitly use any prediction about the future of the community.

We have opted for the use of the backward swarming, as this approach is more robust and domain independent. Abstract criteria of the system quality defined in the introduction were also formulated in a precise manner, with descending priority: (i) provides connection between any two system elements through the minimum number of stand-in agents, (ii) minimize the number of stand-in agents in the system and (iii) minimize the number of messages for system operation and/or knowledge maintenance.

Population manager is driven by a biology inspired algorithm. Social dominance and altruism models (Thomas *et al.*, 2004) were successfully used to partition the group of agents into those who work for the good of the community and the others, who profit from the altruism of the first group.

To ensure the target coverage, stand-in agents can be reproduced in the system using two main propagation strategies:

- **full flood fill** – any stand-in agent initiates full flood filling reproduction strategy when it identifies a new unserved knowledge target in its reach. To decide whether the target is really new, all agents keep a set of served targets, that includes both the other stand-in agents and the knowledge about final users. A target is removed from the set when it is not used for a specified period – forget time.
- **bounded flood fill** – this is depth-limited version of the previous reproduction strategy. After the initiation, the stand-in agents are successively cloned only up to the depth specified by FloodFillDepth constant. This reproduction strategy is triggered by a local accessibility change when the source agent holds relevant, non-expired knowledge.

Both flooding strategies are time limited. There is a specified constant flood duration during which the stand-in agent retains reproduction intention. When this period expires, the agent no longer reproduces until the new reproduction is started by the agent itself or the others.

To keep the number of stand-in agents close to optimum, the population manager contains also the methods that decrease the number of stand-in agents in the system:

In random duels the attacking stand-in agent randomly selects an adversary between accessible agents and launches an attack with force proportional to its profit during specific period, as determined by information propagator (see bellow). Besides the attack force, the attack also includes the information about its active target set at the time of the attack. The attacked agent evaluates the attack and decides whether it won or lost. If the attacked agent loses, it removes itself from the system; losing attacker is not penalized for the attack. Attack evaluation compares the active targets first and when one is a subset of the other, its owner loses the fight. When the sets

are identical, force of the attack decides the fight – the stronger agent wins. Active target set size is evaluated differently for new and old agents. For the young agents that are not yet completely adapted to the environment, the set contains all directly accessible targets, while it contains only the really used ones. Besides this advantage, the youngest agents benefit from the immunity period, during which they can not lose a fight while attacked.

Information propagator manages knowledge propagation and use in the system. This component uses virtual payments to reward the other agents for the knowledge, receives payments from the others for the information provided and generates the profit also from acting on behalf of the represented agent. Each agent optimizes its profit, ensuring the overall information flow efficiency.

Historical data (represented as probabilities assigned to knowledge characteristics, origins and targets) that are used to identify the targets to which we send the information are periodically updated and the old data is discarded.

5. CONCLUSIONS

The main contribution of this paper is in reporting on the integration of the stand-in technology into the **FlexFeed** framework. This integration provides all stand-in behavior and many changes in NOMADS and **FlexFeed** core were applied.

The stand-ins implementation was divided into stand-ins Core and platform-dependent implementation of underlying infrastructure. The stand-in Core is platform independent and provides all the above mentioned behavior. For easy implementation, the interfaces for stand-in required infrastructure were deployed. Platform-dependent part of stand-ins provides infrastructure for stand-in creation and cloning, messages propagation and solving visibility of containers and environments.

A set of experiments was carried out in order to study the stand-in behavior in environment with limited communication ranges. We measured the evolution of the number of stand-in agents at three levels of inaccessibility dynamics. To determine the optimal number of stand-in agents in each moment, we have implemented an efficient centralized algorithm (Šišlák*et al.*, 2005). In our domain, where the accessibility is distance-based, this algorithm behaves optimally.

Future work will involve the integration of coordination components with the rest of the framework to build a prototype for tests and demonstrations. We also plan an extensive set of tests and experiments to validate the concept and characterize the prototype.

REFERENCES

Bradshaw, J. M. (1997). Kaos: Toward an industrial-strength generic agent architecture. In: *Software Agents, AAAI Press/MIT Press*. pp. 375–418.

Bradshaw, Jeffrey M., Mark Greaves, Heather Holmback, Tom Karygiannis, Wayne Jansen, Barry G. Silverman, Niranjan Suri and Alex Wong (1999). Agents for the masses?. *IEEE Intelligent Systems (March/April)* **14**(2), 53–63.

Carvalho, Marco, Florian Bertele and Niranjan Suri (2005). The ulm algorithm for centralized coordination in flexfeed. In: *Proceedings of the 9th World Multi-Conference on Systemivcs, Cybernetics and Informatics - Orlando, USA*.

ISO (2005). Open Systems Interconnection Reference Model. `http://en.wikipedia.org/wiki/OSI_model`.

Rollo, M., P. Novákand P. Jisl (2005). Simulation of underwater surveillance by a team of autonomous robots. In: *Holonic and Multi-Agent Systems for Manufacturing* (Mařík, Brennan and Pěchouček, Eds.). number 3593 In: *LNAI*. Springer-Verlag, Heidelberg. pp. 25–34.

Šišlák, D., M. Rollo and M. Pěchouček(2004). A-globe: Agent platform with inaccessibility and mobility support. In: *Cooperative Information Agents VIII* (M. Klusch, S. Ossowski, V. Kashyap and R. Unland, Eds.). number 3191 In: *LNAI*. Springer-Verlag, Heidelberg.

Šišlák, David, Martin Rehák, Michal Pěchoučekand Petr Benda (2005). Optimizing agents operation in partially inaccessible and disruptive environment. In: *Intelligent Agent Technology, 2005 IEEE/WIC/ACM International Conference*. number P2416 In: *IEEE*.

Sycara, K., J. Lu, M. Klusch and S. Widoff (1999). Dynamic service matchmaking among agents in open information environments.. *ACM SIGMOID Record* **28**(1), 211–246.

Thomas, Vincent, Christine Bourjot, Vincent Chevrier and Didier Desor (2004). Hamelin : A model for collective adaptation based on internal stimuli. In: *From animal to animats 8 - Eighth International Conference on the Simulation of Adaptive Behaviour 2004 - SAB'04, Los Angeles, USA* (Stefan Schaal, Auke Ijspeert, Aude Billard, Sethu Vijayakumar, John Hallam and Jean-Arcady Meyer, Eds.). pp. 425–434.