Adding Functionality to a Distributed System with Herd

OLIVIER BUECHEL, RUDOLF SCHEURER, BEAT HIRSBRUNNER
DIUF - Department of Informatics
University of Fribourg
1700 Fribourg
SWITZERLAND

Abstract: - This paper presents the deployment feature of herd, a model to cluster mobile agents into a logical unit which is physically distributed. In analogue to the mammals stochastic, non-deterministic herd behavior, a mobile herd agent roams the network vicinity without losing the contact of his herd members. While roaming the network each member function executes its task. We show how the deployment of functions with herd facilitates adding new tasks dynamically to a running network.

Keywords: - Network Programming Environment, Logical Cluster, Network Model

1 Introduction
Modern networks have to cope with dynamic changes during operation. Non-deterministic congestions must be prevented or detected and dissolved. Nodes may be added or removed while the network is still on-line, hence forcing a rerouting of the network. Networks can’t be switched on or off as central systems because there is no central system. This would imply some sort of distributed switch which is contrary to the distributed paradigm.

Without being able to reboot a distributed and thus restart the distributed system with a well-defined state, maintenance becomes complex. Adding new software or just readjusting and patching a distributed system while running is comparable to exchanging the pistols of a Ferrari car during a race (while the pistol is on its lowest position).

We call the introduction of a program in a running distributed environment deployment. In this highly dynamic open distributed situation mobile agent systems are proposed as an adequate mean to control the situation [11,13]. Because of their ability to move code around the network mobile agents overcome the static Client/Server approach of protocols: the client requests information while the Server serves them.

Enabling technologies for mobile agent systems (MAS) are well established [2], [3], [4]. Prototype applications using MAS have been developed [5], [6] for different areas.

The idea of imitating process observed in nature is an old one and called a bionic approach in engineering. Several engineering concepts are copied from nature. In computer science the genetic algorithm reproduces the selection pressure found in natural selection in order to get optimal solutions. Multi agent systems such as ants [1] mimic the social behavior found in an ant-hill. Within an ant system collective behavior emerges from simple stigmergetic interaction. Such ant systems have been shown to be able to route networks.

The herd model too is a system which mimics features observed in nature, namely the clustering behavior of some mammals like Gnus and Lions. Our approach consists in copying the natural herd behavior onto a mobile agent system in order to copy the fault tolerance observed with herds.

Herd [12] is a concept based on mobile agents which emerges to a natural herd-like behavior by communication through local tuple spaces. This paper shows how herd helps in deploying new functionality into a running network. In section 2 we give a shorten description of the herd model explaining herd migration, herd
interaction and herd density. In section 3 the focus will then slide to an application of herd, the deployment of new functions for network management tasks. Section 4 contains a list of further work to do and finally in section 5 the conclusions are drawn.

2 The Herd Model
Gnus do it in herds, bats in colonies, bison in gangs, coyotes in packs, dolphins in flocks, monkeys in troops, sheep in droves and whales in schools: they all cluster together. The reasons in doing so are manifolds, but one reason is the security a herd provides its member respective to carnivore attacks.

Before continuing the description of herd, some basic notations used in this paper have to be stated. A mobile agent or agent for short is a process extended with the commands move(Node), write(tuple), read(tuple), kill(AgentID) and create(AgentURL). All these commands act locally thus avoiding agent topology problems [14]. The Linda tuple space used to communicate with agents on the same node is a well known shared memory concept and several flavors have been implemented and studied [7], [8], [9].

In analogy to natural herds, mobile agent herds migrate, have interactions with other herds and have a specific density of members per node. Those features characterize the behavior of a herd. By choosing appropriate settings for these features, the application programmer can create herd classes fitting his requirements. By instantiating a herd class a logical unit out of mobile agents is deployed in the network.

2.1 Herd-Herd Interaction
Herd-herd interactions (interactions between herds) are arranged along a line from repelling to attraction. In the centre of these two poles lies the non-interaction point. Each type of herd has a interaction function taking as input a herd ID and returning the interaction value between -1 and +1 with the given herd. The value of -1 representing total repelling. The value of 0 represents no interaction at all between both herds. The value of 1 represents a total attraction. This interaction relation is not a symmetric relation: Herd A repelling Herd B doesn't imply herd B repelling herd A (Figure 1).

A given herd A with a setting of repelling herd B will not enter a node which herd B has already occupied. Breaking down this definition to the level of herd members this means that no A member will rest on a node where B members have taken residence. After having verified the existence of a repelling herd member on the same node, it will retreat.

2.2 Migration of a Herd
A second feature of a herd is the collective movement through a network called migration. We distinguish two migration types a herd can apply, namely alpha type migration and swarm type migration. The former is called alpha migration because of the herd agents following a specially signaled herd member, the alpha agent. Although a special designated agent introduces a single point of failure this can be circumvented by leader election algorithms. The second migration type is called swarmed migration because as in a bird swarm no obvious leader exists.

With swarmed migration, each herd member will inform other members at the same node about their last location and the members seen there. This gives a local view of the herd that will do to calculate the centre of gravity of the

![Figure 1: Herd-herd interaction example: Gnus are repelled by Lions. Lions are attracted by Gnus. Flies are attracted by both. Neither Gnus nor Lions care about fleas.](attachment://herd-herd-interaction.png)
herd. The resulting node is used as the goal to be hopped on.

In both migration types a herd holds a function representing a gradient indicating the general direction of migration. The gradient is freely definable to suit the herd’s purpose and applied on every new node.

Examples of gradients are measurements of CPU capacity, traffic capacity etc. Thus a herd using a CPU capacity gradient will always tend to hang around the most powerful nodes (or with negative sign hanging around the less powerful node without disturbing the powerful ones).

### 2.3 Herd density

Although a herd can not guarantee a globally fixed number of members it can guarantee some local constraints concerning the size of a herd. We define the formal density of a herd as the number of members on the same node. This number can be adapted by the corresponding alpha agent to suit the network. Such herds can increment their density if for example spare CPU capacity is available.

We define the actual density as the number obtained by effective counting the members of a herd on a specific node. The actual density of a herd is constantly changing due to movements of its members and is oscillating around the value of the formal density. It is the herd’s objective to keep oscillating the actual density near the formal density, incorporating the stochastic aspects of a herd.

### 3 Deployment Mechanism

We differentiate two methods of deployment of new functionality with herd according to the two possibilities of instantiation. The first method is to add a new member to an already running herd. Therefore the ID of respective herd is needed. This information is provided along with the code representing the new functionality to be added to the network.

The herd deployment mechanism wraps the given code into a new herd member type including the herd ID. This type is then instantiated and "injected" into the network (Figure 2). Given the stochastic behavior of a herd the herd member will be eventually found by his herd.

The second deployment mechanism is to create and add a new herd to the running system. The functionality to be added is implemented in several herd members. When the new functionality is to be added to the network the corresponding herd in instantiated. That is a new (unique) Herd ID is generated by the underlying system and each of its herd members are instantiated into the system. The herd specific code is wrapped around the code implementing the functionality to be added. Once instantiated on the same node, the herd code will coordinate the mobile agents to emulate the herd behavior as described above. The function code wrapped into the mobile agents will be executed. It is in the programmes responsibility to verify the integrity of the code respective to the location.

Until now we have only mentioned how to add functionality to a running, distributed system. However deleting functionality is its natural counterpart and as important as adding functionalities. When talking about deleting functionality we have to ask how much functionality
we want to remove. That’s when we must take account of the coarseness of herd. Only members or whole herds can be removed. There is no way to remove just parts of a member in order to remove a part of functionality. The decision when to deploy with a herd member (first method) or when to use the second method of deploying a whole herd is not easy to make. One can compare it with the decision a C programmer takes when he wants to add new functionality to his program: Should he just add a new procedure to an existing file or should he create a new file containing the implementation? Another analogy arises from OO programming with methods and objects. There are no rules to those choices and it’s up to the programmer and his experience to judge.

3.1 Wrapping tasks
The herd model provides a mechanism to include external code into a herd member through wrapping. Thereby the external code is also called operational code because it is the code representing the wanted network operation. The code wrapped around the operational code like an external skeleton is called skeleton code. It provides the needed functionality to emulate the herd behavior. The skeleton code is the part of code of an herd member that is inherited from its herd type description. This code makes an agent behave as a member of the specific herd. Through specification of the herd type, the herd framework generates the code for further use. The application programmer has to specify the operational code. This code describes what to do on any node. No move operation is allowed in operational code in order not to disturb the skeleton effort to keep the members reasonable together. However, the application programmer is aware of the fact that even if the operation on a particular node is not executable, the agent will be moved by the skeleton to another node, where the operation might succeed. Through the stochastic approach, each node where other members are located will be visited by any other member. And, each member will meet any other member of the same herd. However the node where the meeting happens is not predetermined due to stochastic effects. We can think of several ways of integrating the herd skeleton with the application programmers operational code. The ways heavily depend on the chosen implementation language. A simple include as in C or a more sophisticated inheritance mechanism as in object oriented language like Java can do the work. However a more suitable procedure for mobile entities like mobile agents may simplify the herd skeleton handling. Such methods have been investigated and several agent systems have their solution proposed. Within the Ecomobile system [10] a system has been developed which is used in our herd implementation to decouple the operational code from the coordination code. The herd scheme of clustering agents and its application however are independent of the way the skeleton is integrated. Basically, once the migration problem (i.e. code mobility) is resolved, herds can be used in most major programming languages.

3.2 Herd injection into a network
Injection is the procedure of instantiating agents members of a specific herd. After programming the operational code and the integration with the skeleton code, a herd name must be specified. This herd name will be used when injecting the herd member to inform him to which herd he belongs. The combination of instantiating a herd agent and attaching the herd name defines injection. As we have already seen above, there are two instantiations methods: instantiation of a single herd member and instantiation of a herd, i.e. of several herd members creating one and the same herd.

The first case is called the hermit case because the herd member instantiated has no contact to his herd like a hermit in the desert. As long as the designated herd does not pass the node where the hermit has been initially instantiated, the hermit keeps its state unchanged, doesn’t move and waits to be activated. Alone, the herd member would roam the network vicinity without the protection of his herd. In addition, herd
members may work together to achieve a result where one member alone would fail. A pure herd is an agent family without an actual goal. Its only purpose is to keep roaming around the network. A herd wandering around a network may lose some herd agents if a node failure occurs. However, the missing agents are only a part of the herd and the herd as an entity still exists. The herd will reconstitute them. Only the type of the lost agent is needed to instantiate a new one. The work done previously by the lost or dead agent however has to be redone. In order to increase efficiency, herd members can store their actual work on the tuple space so that a new instance simply takes over where the lost agent stopped. Each herd member will stochastically meet any other member, each member will pass stochastically through any node available to the herd. In this highly dynamic case we know that all mobile agents will meet each other, but we don't know for sure when this will happen or where. We can take advantage of the pure herd by adding tasks to the members of a specific herd. Each herd agent inherits from the herd coordination code plus the task submitted to be executed on every node. Thus specific functionality is added to the herd and consequently to the network. For example, whenever a new node is dynamically added to the network, no notification other than towards the adjacent nodes is needed in order to integrate the new node into the network. The herd mechanism will automatically result in a visit to the new node of a herd member agent. The corresponding functionality will therefore be expanded automatically within the new node. No explicit network management operation execution is needed.

Another feature considered to be implemented is the “while on node do .. end” construct. This construct would ensure the execution of code on a single node or signal a failure to do so. This would help to ensure the integrity of the operational code.

5 Conclusion
Modifying and enhancing a network while running is a complex task where herd has its advantages through the stochastic behavior. Eventually every node will be visited and the corresponding enhancement will be installed. The herd exempts the application programmer from thinking how to keep the mobile agents together and how to coordinate them through the network. He can focus on the operational functionality and simply apply his code with herd. Throughout this paper no security aspects are considered although they are critical in real world applications. Our working hypothesis consists of a friendly agent universe where no malicious agent tries to abuse or to unauthorized kill a mobile agent. Based on above mentioned assumptions any collection of mobile agents can form a cluster with herd alike features. This makes herd a flexible model, adaptable to most MAS. The nature of a herd, i.e. its stochastic non-centralized architecture, is designated to scale well. Like procedures and objects, herds capsule functionality. However, a herd is a capsuling method for agent oriented programming, especially for mobile agent. Ultimately, in the future we may have techniques which let us grow or nurture networks with meta constructs of mobile agents instead of managing them through classical protocols.

4 Future work
More work has to be done when enhancing the herd model with meta-herds. A meta-herd is a herd of herds or a herd of herd-herds and so forth. This would enhance the possibilities and coarseness respective to adding or removing tasks.

References:


[8] David Gelernter; Generative communication in Linda; ACM Transactions on Programming Languages and Systems 1985

[9] D. Campbell, H. Osborne, and A. Wood; Characterizing the design space for Linda semantics; Techreport YCS-97-277, University of York


[14] Xiaolong Jin, Jiming Liu; Agent Network Topology and Complexity; Autonomous Agents and Multi Agent Systems AA-MAS’03, Melbourne, Australia