RC24769 (W0903-082) March 20, 2009
Computer Science

IBM Research Report

Morphogenesis in Computer Networks

Vasileios Pappas, Dinesh C. Verma
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Morphogenesis in Computer Networks

Vasileios Pappas and Dinesh C. Verma, *Fellow, IEEE*

***Abstract*—Morphogenesis is the process that gives shapes to organisms from an embryonic stage using a process of cell-division. Starting from a simple embryonic cell, the controlled division and transformation of the cells into different types leads to the creation of a complex organism. The growth of complex organisms is completely autonomic, and is of the best examples of self-organizing systems that can be found in nature. In comparison, computer networks of today are much more static and require a significant human intervention in order to take on the shape and size that is desired. However, emerging technology such as virtualization and network computing enables an architecture where computer networks can also develop using morphogenesis. In this paper, we present the benefits of morphogenesis in a computer network, and we describe an architecture for a computer network which self-organizes itself using the principles of morphogenesis.**

***Index Terms*—morphogenesis, self-organization, autonomic networks, self-configuration.**

I. Introduction

The management and operations of distributed computer systems and networks is a complex labor-intensive task that accounts for the lion's share of costs in current computing environment [1]. A significant component of the complexity arises from the heterogeneity that exists in the computing infrastructure. The presence of a variety of different types of machines in a computing environment creates challenges related to the configuration, error diagnosis and repair of the infrastructure that is put into place. In many environments, e.g. in the case of military networks, a variety of instruments implies the need to carry batteries and power equipment, which increases dead weight and increase power consumption.

Although a few vision papers[2][3] have been put forward for the development of self-organizing self-configuring computer systems, an architecture for computer networks that exhibits such behavior remains elusive. If one were to truly attain the vision of self-organization, networking infrastructure at a data center or operational capabilities of a military outpost could be obtained simply by turning on generic machines at the site, and the machines would take on the configuration and installation that is required to fulfill the computing needs of that data center or the military outpost. This paper presents an architecture for computer systems which enables them to realize this vision.

The inspiration for such self-organization is drawn from the ability of biological systems to grow and differentiate themselves into self-organizing patterns [4]. A small acorn can grow into a mighty oak, and a small zygote can differentiate itself to make a complicated human being. This phenomenon of morphogenesis can be replicated to the domain of computer networks using the architecture described in this paper. Our goal would be to obtain well-defined structure for the overall computer network, while avoiding the phenomenon of emergent behavior [5], where unanticipated and unpredictable characteristics are observed. Self-organization without emergence requires restricting the behavior of the system, since even simple cellular automatons [6] can easily result in very complex patterns.

The architecture introduced in this paper would enable computer systems to have the desirable attributes of self-organization and self-configuration, while still resulting in a system that is unlikely to show emergent behavior. Proposed architecture enables constrained and controlled self-organization which is fundamentally different from the uninhibited self-organization in complex systems [7][8]. The architecture is realizable and attainable with current technology in computing systems.

We begin by describing some scenarios that we would like to obtain as a result of this architecture. After the description of the scenario in Section II, Section III describes the biological inspiration of the system. Section IV describes the usage model of the system we develop, and Section V describes the morphogenesis architecture of each node in the self-organizing network that enables morphogenesis. In Section VI we experimentally evaluate the accuracy of our system. Finally, we present the related work and

Manuscript received March 15, 2009. Work was sponsored by U.S. Army Research Laboratory and the U.K. Ministry of Defence and accomplished under Agreement Number W911NF-06-3-0001

we conclude with a summary and the open problems.

II. Target Scenarios

In this section, we look at some target scenarios and what the goal of the morphogenesis architecture would be in each of the scenarios.

A. Commercial Data Center

A commercial data center in current environments is a complex network of devices. Among the different types of devices that are found in the data center, one can enumerate IP routers, firewalls, Ethernet switches, web servers, application servers, database servers, and storage devices such as disk arrays. Additionally, management servers, accounting servers, customer help-desk systems as well as audit compliance systems may be present to ensure smooth operation of the data center. Many of these devices are specialized hardware or software functions, and can rarely be interchanged. In some cases, e.g. on servers, it is possible to manually install database software instead of application software, but the process is usually slow and cumbersome.

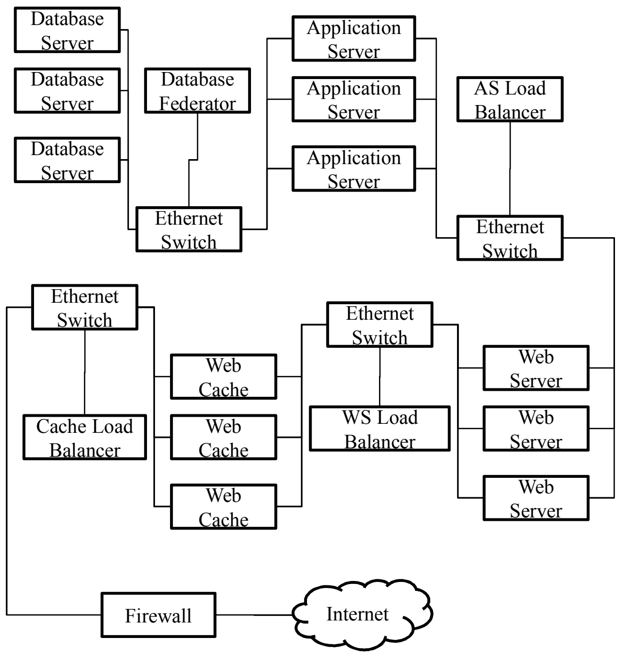


Fig. 1. Typical Tiered Architecture of Data Centers

V. Pappas and D. Verma are with the IBM Watson Research, Hawthorne, NY, USA (telephone: 914-784-7466, e-mail: dverma@us.ibm.com).

A classic arrangement of a commercial data center is that of a tiered architecture where multiple machines perform the same function in each tier. A typical tiered architecture of a web-hosting server is shown in Figure 1 ,with the 0th tier being a set of databases, the 1st tier being a set of application servers, the 2nd tier being a set of web-servers, the 3rd tier being a set of web-caching proxies, the 4th tier being a firewall that acts to protect the other tiers before connecting to the Internet. Each tier (except the firewall) would typically have a load-balancer connected together with a network switch to the other machines in the tier. In some data centers, there may be an additional server or some other application, e.g. LDAP Servers located attached to the side of one of the tiers. Physically, each tier would get its interconnectivity using an Ethernet switch. Note that multiple tiers can make use of the same physical switches but still being isolated at the logical level with the use of virtual local area networks.

Other than the Ethernet switch, the machines that make up the data center consist of servers running various application packages. They can be created dynamically by reconfiguring applications on a generic machine. An Ethernet switch in a data center provides connectivity among a cluster of machines. Assuming that the basic connectivity is available, we would like to achieve a scenario in which a group of generic machines, (e.g. consisting of a base Linux operating system) are connected to different Ethernet switches and booted up. Assuming that each of the machines is aware of the overall blueprint of the system that is available, we would like the generic machines to automatically determine their position in the system, determine their role and then configure themselves to be conformant within that role.

B. Military Camp Scenario

In a military camp, the network interconnecting different devices and machines may be a wireless network instead of a wired network. This is because the physical establishment of a wireless network is easier than that of a wired network in a camp which needs to be dismantled and established rapidly. However, there are many computer systems and components with different persona and roles that need to be established in the military camp.

A typical personal and role of the different servers making up the military are shown in Figure 2. In this figure, the different devices are connected using a wireless network. The different machines are deployed in the role of servers (e.g. to use for electronic mail) while others are used to run different applications such as red/blue force tracking, managing a wireless sensor network or for planning military operations. While the structure in the military camp is not in terms of cleanly delineated tiers such as that of the commercial data center, there are distinct components of the system that perform various functions.

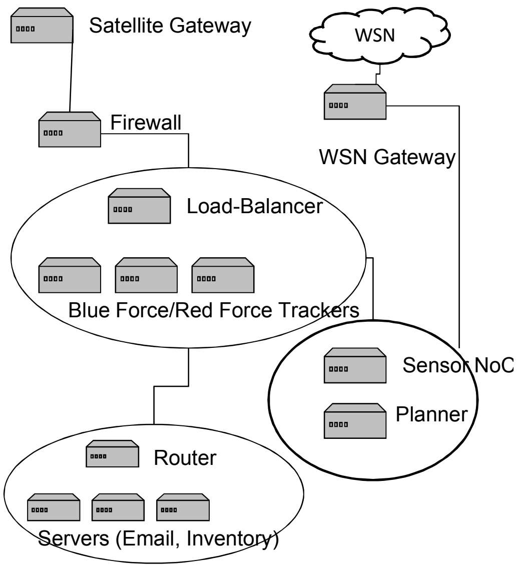


Fig. 2. A typical environment in a military base camp

As in the case of the commercial data center, we would like to attain a scenario is which generic machines are powered on in the base camp. The generic machines, when powered up, are going to understand their position in the overall network, map that position to the role they need to play within the network, and then configure themselves automatically to support and provide those functions.

There are many benefits of using morphogenesis in the design of computer systems and networks. If the computing environment can configure itself automatically and take on the appropriate persona, the effort required in configuring systems would be much reduced. In current environment, e.g. the military camp scenario, the different devices tend to be relatively black boxes that cannot be easily transformed from one to another. This causes problems in the management of logistics, e.g. one may have too many boxes doing blue-force tracking but too few boxes running planning applications, and it is not trivial to convert one to another. Furthermore, as conditions on the ground changes, a system that can change its persona can adjust itself better to the dynamic environmental changes.

The goal of this paper is to define an architecture which will allow us to attain the vision outlined above for the previous environments. The architecture should be usable in many different contexts other than the previous two that we have described.

III. Biologically Inspired Morphogenesis

In order to design morphogenesis in computer systems, we look upon biological cell division for inspiration and insights into a similar function for computer systems. However, our intention is not to mimic the complete biological process, but only to use high-level principles from its operation.

The blueprint for what an organism needs to become is contained in the first cell (embryo) and is copied from that to all other cells that are formed by the process of cell-division. Each new cell determines its position relative to other cells in the entire system and uses its position and environment to determine what it needs to become (i.e. a skin cell, or a blood cell or nerve cell, etc). The determination of the position and the type of cell each one needs to become is determined by a complex set of interactions among the chemicals and the position of the cells.

Without mimicking the complexities of cellular division, we can use the following high-level concepts from the natural morphogenesis process to develop the similar behavior in computer networks:

The Embryo: The embryo is a special cell which is configured with the blueprint of the entire system by means of its genetic code. Similarly, in our development of the morphogenesis of the system, we will select a special node as the embryo which will have the blueprint of the entire system.

The Genetic Code: The genetic code is a concise description of the overall distributed system that needs to be created. Similarly, we will have a concise description of the overall system that ought to be formed. The genetic code will be provided to the embryo node, which will be responsible for disseminating the genetic code to all of the other elements.

Cell Division: The embryo creates new cells using the process of cell division. The structure is obtained by means of the new cells determining their persona and changing themselves into that persona. In the computer systems that we are designing, we would need to have a mechanism to add new cells or nodes. One such process is the creation of new virtual machines on an existing physical node. Another process is the introduction of new physical node.

Position Determination: In the morphogenesis process in biology, the position of the cell and its state with respect to its neighbors is an important factor in determining its behavior. Therefore, each computer node in the biologically inspired network needs to have a way to determine its position.

Blueprint Scaling: As the size of the living organism increases, the sizes of the individual organs and body parts scale accordingly. The same blueprint that defines a toddler defines a full grown man as well. Similarly, as the size of the network increases or decreases we would like to use the same blueprint that can easily scale to various system sizes.

Personal Determination: The final step in the morphogenesis process is the determination of the persona a computer node will take. The persona of a computer node is the set of applications, application data and operating system that is used to provide its functionality.

The architecture that we describe in subsequent sections provides for an implementation of the above concepts borrowed from biology.

IV. USAGE MODEL

In the usage model of the system that we have, each device is booted off as a generic device. By default, each device is assumed to be a generic device and not an embryo. The system administrator can toggle a switch or change configuration parameters of one device to mark a device as an embryo. The embryo is also configured to obtain its genetic code. The embryo communicates with the other devices to establish communication and to disseminate the genetic code.

For simplicity, we will be describing the system as consisting of exactly one embryo. However, a general system may include many different embryos. A simple mechanism to support multiple embryos is to just assume that a generic device would belong to the embryo that it first establishes contact with.

Once a generic device is powered up, it tries to discover other devices in its neighborhood with whom it has connectivity. It waits till it finds an embryo in the discovered cells, or another node which has access to the genetic code from an embryo. The generic device obtains the genetic code from such device. The device in turn would provide the genetic code to any other device that it discovers in the system.

After the generic device has received its genetic code, it determines its position and then its persona by examining the genetic code. Having obtained its persona, the device changes itself to the corresponding set of software as dictated by the persona. Different personas can be prepackaged as different virtual machine images, in which case the adoption of a persona from the generic device becomes just the instantiation of the appropriate image on the generic device.

The operation of an embryo node is very similar to that of the generic device, except that it has its genetic code from the very beginning. After being powered up, the embryo searches for other adjacent nodes in the neighborhood to whom it provides the genetic code.

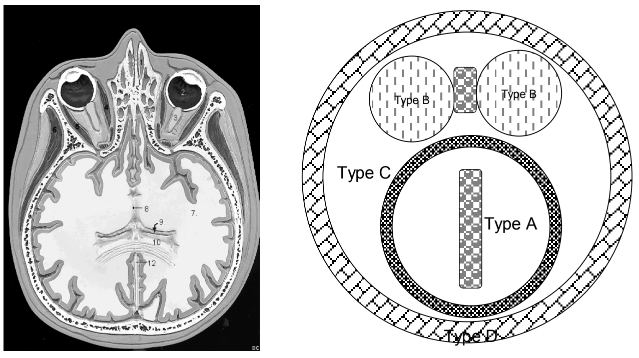
V. MORPHOGENESIS ARCHITECTURE

The key design considerations needed in order to support this architecture are (i) the representation of the genetic code (ii) the position determination algorithm (iii) the mapping from position to persona (iv) the discovery process to find new cells and (iv) the architecture of each generic device which enables it to take on the desired persona.

A. Representation of Genetic Code

Although there are multiple approaches possible to represent the genetic code, we will take the approach of defining the genetic code as a set of mathematical expressions. In specifying the genetic code as a set of mathematical expressions, we assume that the position of a node is specified as a set of coordinates, $\{x_1, x_2,$

.... x_n}, and that a set of mathematical equations define the constraints that map any coordinate to a specific persona. The use of location to define a role of a machine is taken from the observation that most organisms follow a similar paradigm. Figure 3 below shows how the cross-section of a human body can be mapped into a planar description of the regions where different types of personals are assigned to nodes based on position.



(a) Body Cross-Section (b) Simplified Region of Cell-Types

Fig. 3. Presence of Similar Regions in Biological Systems

If we consider the genetic code that represents the system created in Figure 3, we can use a mathematical representation as shown below. A two-point coordinate system is used to represent the position {x_1, x_2} of any point in the plane, and each position is mapped to a persona.

```
If  $r_4 < (x_1^2 + x_2^2)^{1/2} < r_5$
        then persona = D
If  $r_2 < ((x_1 - a_3)^2 + (x_2 - b_3)^2)^{1/2} < r_3$
        then persona = C
If  $((x_1 - a_1)^2 + (x_2 - b_1)^2)^{1/2} < r_1$
        then persona = B
If  $((x_1 - a_2)^2 + (x_2 - b_2)^2)^{1/2} < r_1$
        then persona = B
If $(-w < x_1 < w)$ && $(-h < x_2 < 0)$
        then persona = A
If $(-w < x_1 < w)$ && $(h_1 < x_2 < h_2)$
        then persona = A
```

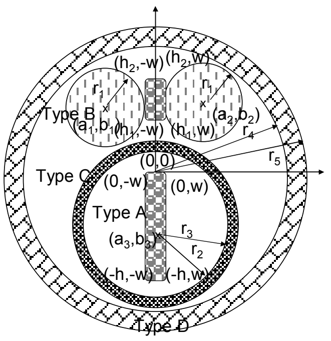


Fig. 4. Example of Genetic Code Description

The same principle can be used to specify the genetic code for any type of computer system. As an example, the genetic code for the commercial data center can be specified simply by using a two dimensional coordinate $\{x_1, x_2\}$ where x_1 measures the distance of the system from the embryo and x_2 is a logical coordinate. The logical coordinate is used in order to assign more than one roles at the same location, mainly in cases where there are multiple nodes at the same location. Each node randomly picks the logical coordinate from a set of possible logical coordinates for that position. The method for location determination is described in subsequent section. Let us assume that the embryo is marked as the firewall to the Internet. The following mapping would be the genetic code specifying the characteristics of the commercial data center.

Distance	Persona
(0,0)	Firewall
(1,0); (2,0); (3,0);	Load-balancer
(1,1)	Web Cache
(2,1)	Web Server
(3,1)	Application Server
(4,0)	Database Federator
(4,1)	Database

Table 1. Datacenter genetic code

Distance	Persona
(0,0)	Firewall
(1,0)	Load-balancer
(1,1)	Red Force Tracker
(2,0)	NoC
(2,1)	Planner
(2,2)	WSN Gateway
(3,0)	Router
(3,1)	Email Server
(3,2)	Asset Inventory

Table 2. Military base camp genetic code

Similarly, a table mapping using a similar description could be used to represent the topology of the system described in the military base camp. The genetic code representation as a mathematical expression is general enough to capture a wide variety of end organisms that are targeted. The representation can be captured in a computer readable manner and transmitted efficiently between different cells.

B. Position Determination Algorithm

The position of a node is determined by means of three independent metrics, the first measuring the distance of the node from the embryo in terms of hops, the second (an optional component) measuring the absolute difference in position of the node relative to the embryo, and the third being a customizable definition of position that is broadcast by the embryo.

The distance of node from the embryo is relatively trivial to determine. Each node maintains a distance counter for itself, the distance counter being the lowest among the distance counters of all its neighbors incremented by 1. The simple distributed algorithm measures the smallest distance in hops between a node and the embryo.

The absolute difference in position requires that the devices be capable of obtaining an absolute position reading for themselves. This may be obtained by using a GPS system [9], or for indoor systems similar triangulation [10] [11] techniques that can be used from different access points located inside a building. The ability to read the absolute position is dependent on the capabilities of the generic device base software infrastructure.

The third component of the position is a set of directions that comes from the embryo. This component provides instructions from the embryo on how to compute any additional components of the position. The distance from the embryo and the absolute difference in position provide the first few coordinates of the position vector. The other coordinates can be computed logically by means of programmatic directions that come with the embryo along with the genetic material. These programmatic instructions include information such that exactly one of the equidistant nodes should take on a logical coordinate of zero, and others take the logical coordinate of one, or that a coordinate may be selected randomly to be either 0 or 1 with an equal probability. These instructions come as executable software in the mode of active networks [12], thereby providing the ability to compute position metrics dynamically by the embryo.

Taken together, these three mechanisms provide a position coordinate for each node where the first coordinate marks the distance from the embryo, the second few mark the absolute difference in position and the remainder are computed as per directions coming from the embryo.

Once the position and the genetic code are received from the embryo by the process of dissemination, the computation of the persona from the position vector is a routine operation from the specified genetic code.

C. Blueprint Scaling Algorithm

The definition of the genetic code as a mathematical expression is done in a way that is agnostic of the size of the network. That means that the same genetic code can be used to describe a small network, confined in an area as small as a few square meters, as well a large network, in an area as large as a few square kilometers. In essence the genetic code describes the configuration of the network in a unit-free way. Then for each specific instantiation of the genetic code, nodes scale the mathematical expression of the genetic code accordingly, so as to adjust the network configuration to the appropriate size.

The main challenge in scaling the mathematical expression is the knowledge of the actual area that is covered by the network. For example if we consider the genetic code description of Figure 4, given in a unit-free manner, and we know that the total area of the network where the code is going to be instantiated is one square kilometer, then we can appropriately scale the mathematical expression. Unfortunately the total area may not be known in advance. Even worse, the total area can vary considerably during the lifetime of a network, as the network may expand or shrink during its lifetime.

To deal with these issues nodes continuously estimate the size of the network and scale the mathematical expression accordingly. The estimation is done in a completely distributed manner with the use of local measurements only. Each node i measures the number of nodes N_i that it can listen within its local neighborhood, a circle that covers an area of A_c. Given these measurements node i computes the local density D_i, defined as $D_i = N_i/A_c$. Assuming a homogeneous network, each node independently estimates the density of the whole network, with node's i estimate being D_i. Given the total number of nodes N that participate in the network is known in advance by every node, then the total area of the network estimated by node i is: $A_i = N/D_i = A_c{*}N/N_i$. Given this estimation of the total area, and the fact that the genetic code is defined within one square unit area, each node computes each scaling parameter S_i as follows:

$$S_i = \mathrm{A_i}^{1/2} = (A_c{*}N/N_i)^{1/2}.$$

This final expression shows that the scaling parameter is computed only with the use of local information: the size of the listening area (A_c), which mainly depends on predefined radio parameters, the number of local neighbors (N_i) that are within the listening area, and the total number of nodes (N), that is known to every node in advance. In the case that precise location information is available, with the use of GPS, the accurate determination of the neighbors' location can further improve the density estimation. Note that the above process yields different scaling results for each node. In the evaluation section we study experimentally how substantial these differences are as well as their impact on the accuracy of the persona determination algorithm.

D. Discovery Process

Unlike biological cells, computer nodes cannot subdivide. Thus, the discovery process relies on existence of generic devices which are customized to obtain specific persona. The discovery process of each device begins by discovering other devices that are on the same network.

As part of this discovery, an implementation of a zeroconf protocol would be needed to eliminate the need for specialized servers to configure machines. Assuming that the generic devices support the IP protocol, we propose a variation of the Apple mDNS protocol for zeroconfiguration. Each machine picks a randomly generated domain name and an IP address randomly from the reserved private address space (e.g. the subnet 10/8). The IP address picked is always selected to have a 0 in the last bit, so in effect the machine is reserving a network address block capable of supporting two individual machines. The rationale for reserving two addresses will be apparent when we look at the design of elements in a single generic device.

The device then starts listening on a reserved port, and sends a broadcast message on the local subnet on the same reserved ports to other devices in the subnet. When the devices receiving the message respond, it is able to identify other nodes. If it finds a conflict on the subnet with any other machine, it selects another random name and address that is not already taken, and broadcasts its change of name to all of the nodes in the subnet.

Once the node has discovered other devices and negotiated/finalized its domain name and network address, it looks for any node that may have already been contacted by an embryo node. If any node it has discovered has already found the embryo node, it obtains the genetic code, the location computation code and distance from the node of the embryo. When the node finds any new node in turn, it transmits the information over to that node as well.

In some environments, the discovery process needs to be secured since a morphogenesis approach like the one described in this manner introduces obvious security vulnerabilities. A malicious node pretending to be an embryo can easily subvert the operation of the entire system. In order to secure the discovery process, an initial configuration needs to be done on all the generic devices. However, the configuration values are the same for all of the nodes. The configuration parameter includes the public key for the embryo node which is to be used for communication. Only location information and genetic code that comes signed with the public key of the embryo node is accepted by any node.

Furthermore, a shared secret key is provided to each of the embryo to encrypt their communication with each other during the discovery process. The shared secret key allows only properly configured generic devices to communicate with each other and the embryo.

These two relatively simple security mechanisms can be used effectively to securely discover neighbors and to start off the process of morphogenesis.

E. Structure of Generic Device

An essential element in the morphogenesis architecture is the structure of the generic device. The generic device should be capable of taking on a generic application module and reformat itself accordingly. At the same time, it should enable the operation and functioning of existing applications.

We use the concept of hypervisors [13] and virtual machines in order to implement a customizable generic device. The generic device is a computing element that needs to be capable of taking on different personas at different times. Each such node can be developed according to the architecture shown in the next figure.

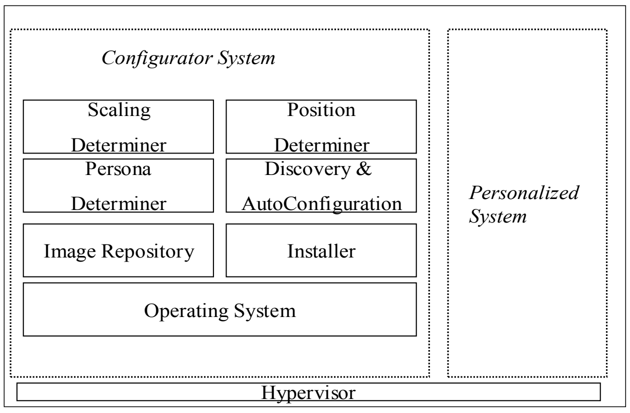


Fig. 5. Architecture of the generic device

The node consists of a base hypervisor which is always operational. A hypervisor allows multiple virtual machines to be running simultaneously on a single node. In our generic machine design, two virtual machines are always active, one virtual machine (the configurator system) implementing the initial auto-discovery and persona selection algorithm, while the other virtual machine (the personalized system) implements the software that corresponds to the persona that the machine is required to operate on after transforming itself. The existence of the two virtual machines is the reason the configuration performs the zeroconfig process for a subnet capable of supporting two machines. The configuration system uses one of the assigned network addresses for itself and the other addresses for the personalized system.

The configurator system has access to a privileged interface of the hypervisor, which allows it to specify the images to be used for the personalized system. The configuration system is also loaded with the images of the various possible personas the personalized system may take. Depending on the outcome of the persona determination process, the configuration system provides the appropriate image to the hypervisor for it to launch and start the personalized system.

The separation of the configurator system and the personalized system enables many existing images to run on the machine without the danger of running into any type of interference from the configuration system processes. Along with the structure of the generic node described in this manner, the protocols described in the previous section would be able to support a system in which nodes automatically determine their position, blueprint scaling parameter and associated persona and create the desired topology of the overall system in a completely distributed manner.

VI. Evaluation

In this section we study experimentally the accuracy of the persona determination algorithm under various conditions, such as the network size, the node density and node mobility. We assume that nodes are equipped with a GPS chipset, or they have other means of accurately estimating their current location. On the other hand, nodes do not need to know the location of any other node in the network (with the exception of the embryo node) and thus they do not need to know their actual placement in the network with respect to all other nodes. They use the blueprint scaling algorithm of Section V.*C* in order to estimate their relative position in the network and consequently determine their persona.

The accuracy of the persona determination algorithm is mainly a function of the accuracy of the scaling parameter as it is estimated by the blueprint scaling algorithm. The scaling parameter is mainly a function of the network density that each node estimates by using only local information, mainly the number of nodes that it can listen to in an area of a certain radius. Next we investigate how accurate these density measurements are under the following two varying conditions: the size of the sensing area and the number of nodes in the network. In both experiments we create networks with an average density of 0.5 nodes per square meter. The networks are created with the following procedure. Nodes are placed randomly in a square area and each node is connected with all the other nodes that are within its sensing radius. Note that the resulted network has the properties of a random geometric graph.

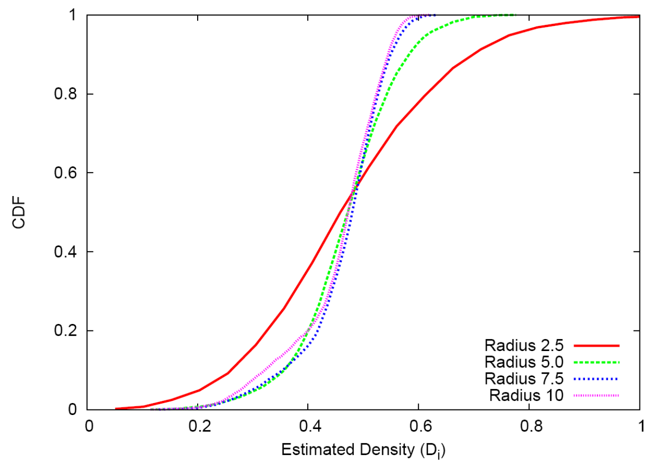


Fig. 6. Density estimations at different antenna radius sensitivity, for a network of 5000 nodes, in a square area of 100m width (actual density of 0.5)

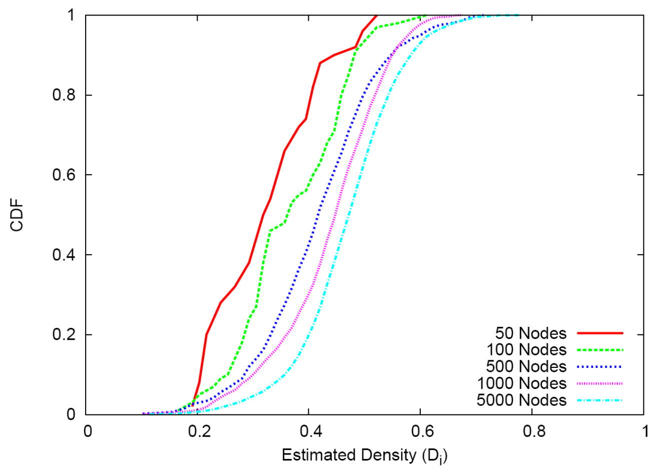


Fig. 7. . Density estimations at different network sizes, with actual density being constant for all networks (actual density of 0.5)

Figure 6 gives the cumulative distribution function (CDF) of the density estimated by each node of a random geometric graph of 5000 nodes in a square area of 10000m^2, for different radius sizes. Given that the average density is 0.5 we see that by increasing the radius the estimation accuracy improve considerably. The main reason behind that is that as the radius increases the number of neighbors increases as well, and thus the sampling size becomes larger, which leads to better density estimation. One side effect of the increased radius size is that nodes close to the limits of the square area underestimate the network density, given that there are no nodes outside of the square area. This can be seen in Figure 6 by the larger number of estimation with a density less than 0.4 for the 10 radius curve, compared to the 7.5 and 5 radius curves. The effects of this underestimation on the accuracy of the persona determination algorithm will be more evident in an experiment presented later in this section.

Figure 7 gives the density estimation results for various network sizes. The average density for all these networks is 0.5 and is achieved by appropriately changing the size of the square area. The radius size of the sensing area for all these experiments is 5m. The graph shows that as the size of the network increases the estimation accuracy increases. Intuitively someone would expect that the accuracy should be the same given that the average density and the sensing radius is the same for all network size. On the other hand, for smaller network sizes, the square area becomes smaller and the percentage of nodes that are closer to the border increases, which leads to a larger number of underestimations for the average density.

From the previous results we conclude that the estimation accuracy of the scaling parameter from each node increases as the size of the network increases, as the sensing radius becomes larger and as the estimating node is further away from the edges of the network. Next we investigate the accuracy of the persona determination algorithm when nodes are moving and when the network is expanding. We are adopting the genetic code of Figure 4 for all of the following experiments. The following table gives the actual values that we used for the parameters of the genetic code.

a_1	a_2	a_3	b_1	b_2	b_3	w	h	h_1	h_2	r_1	r_2	r_3	r_4	r_5
0.120	-0.120	0.000	0.175	0.175	-0.150	0.025	0.300	0.150	0.250	0.100	0.150	0.200	0.350	0.400

Table 3. Parameter setting for the genetic code of figure 4

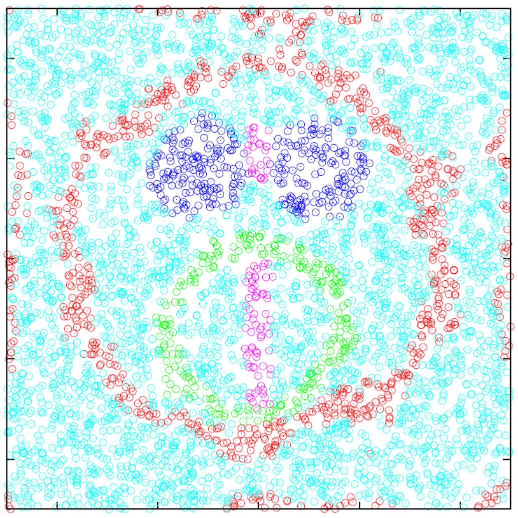


Fig. 8. One experimental instantiation of the genetic code of figure 4 in a network of 5000 nodes, using the parameters of table 3

Figure 8 presents graphically an instantiation of the genetic code at snapshot of time in a randomly generated network. With visual inspection we can see that the persona assignments are very close to the desired ones. Notable exceptions are the nodes that are close to the border of the square area. These nodes, as explained before, underestimated the density of the network and in consequence overestimate the scaling parameter. As a result they are under the impression that they are close to the embryo node than they really are, which leads some of them to adopt persona D. The next table shows the target percentage of nodes that should adopt one of the 4 possible personas (based on the parameter settings of table 1).

Persona	A	B	C	D
Nodes (%)	2.000	6.283	5.498	11.781

Figure 9 shows the percentage of nodes that have adopted one of the four personas as a function of time. During the simulation time, nodes move freely around the square area, following a waypoint mobility model. The graph also shows the targeted number of nodes for each persona. We can see that for personas

A, B and C the actual numbers of nodes are very close to the targeted ones through out the duration of the simulation. For persona D, which is assigned to the nodes that are close to the border of the square area, the actual number is above the targeted one due to the border effects in their density estimation.

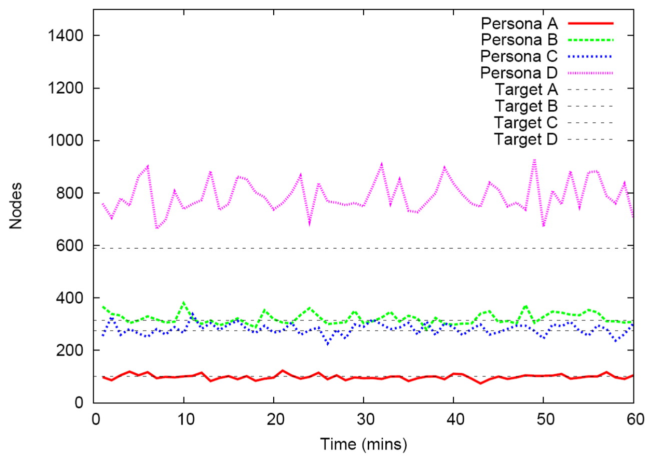


Fig. 9. Distribution of personas as a function of time, in a network of 5000 mobile nodes

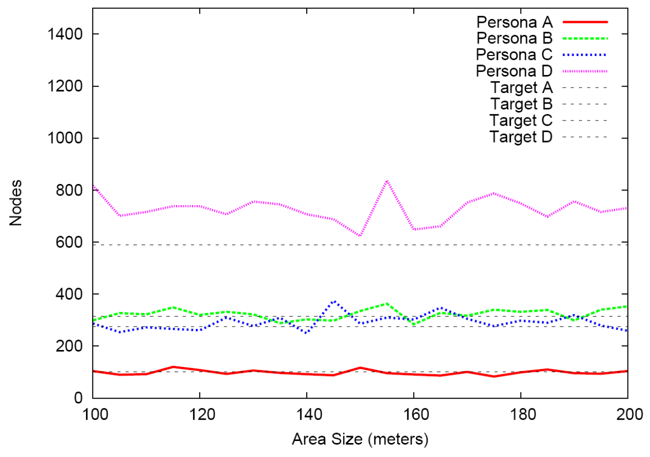


Fig. 10. Distribution of personas as a function of the area size, in a network of 5000 mobile nodes that expands from a square area of 100m to an area of 200m

Figure 10 provides similar results with the difference that during the simulation time the network expands from a square area of size 100m to a square area of size 200m. As in the previous case nodes move randomly within the square, by following the waypoint mobility model. While we observe similar results as before, we should point out this graph demonstrates the ability of the persona determination algorithm to adjust appropriately to network size changes. Throughout the simulation time the actual number of nodes for each persona is close to the targeted one.

VII. Related Work

Self-organization principles applied to highly dynamic computing systems have been the focus of recent research efforts in the mobile ad-hoc networks (MANET) and peer-to-peer (P2P) fields. In the MANET routing domain, protocols such as AntHocNet [15] and SAMPLE [16] have used biological metaphors to identify robust and efficient routes for packet delivery. The inspiration for AntHocNet comes from the process of stigmergy employed by ants in their quest to identify shorter (or higher quality) paths while foraging for food. The SAMPLE protocol makes use of collaborative reinforcement learning to coordinate the solution to discrete optimization problems in multi-agent systems, in order to optimize desirable system properties such as throughput or robustness. Both of these protocols apply the principles of short- and long-term feedback, however their framework does not emphasize the possibly different roles that the nodes can play in the MANET, as the network service under consideration (routing) is relatively homogeneous in functionality among nodes.

Localized structures and positive feedback loops are self-organizing properties also exhibited by P2P systems such as Freenet [17], whose similarities to the emerging behavior and mechanisms of MANET routing services are further highlighted in [18]. However, as in the case of self-organizing MANET routing protocols, the network nodes are rather homogeneous, in contrast to the different roles and functionalities encountered in more complex services such the ones we consider in this work.

Adaptation by continuous monitoring of the environment, often referred to as “execution context”, and composite services that are further decomposed into sets of more primitive ones offered by a plurality of nodes, and for which a role selection process must be in place, have been recently studied in the nascent field of modeling and composition of mobile web services [19] [20] [21] [22]. Various approaches and formalisms are employed for the description, coordination and adaptation of the mobile services, however the main assumption of self-organizing systems -- that of completely localized decision making -- is typically not met: the orchestration, i.e., the binding of the composite service specification to nodes for execution, monitoring and supervision, is performed by an orchestration service that assumes access to global knowledge of the environment, the nodes, and their capabilities.

Autonomous operation based on partial observability of the execution environment and adaptive behavior that makes use of feedback have been advocated for use in multi-agent systems [23] [24]. Key abstractions such as agents, interactions and organizational structures are used for designing and engineering complex software systems in a manner that is more natural and closer to the conceptual model of the programming task under examination than existing object-oriented or procedural approaches. The multi-agent field embodies several of the concepts presented in this paper. However, we should note that sharing global knowledge in a large software system is rather straightforward and imposes minimal overhead, compared to a similar task in a MANET environment, in which the underlying physical infrastructure is significantly more dynamic.

Design of network services based on principles that have been inspired by the operation of biological and social systems is also described in the adaptive networking architecture for service emergence (ANA-SE) in [25] and [26]. As in multi-agent systems, the main abstraction in this proposal is autonomous entities called cyber-entities. Emergent services created from such autonomous cyber-entities are able to interact with one another and adapt to the dynamics of the environment. The main difference between ANA-SE and our approach is that the former models services with agents that move around between nodes, so when they move their intelligence/memory comes with them, whereas in our work we decompose the services into roles, and let the nodes choose what role to play.

Closed to the idea of morphogenesis presented in this paper is the work by Kruger and Dressler [27], [28] and [29], in which they have designed self-organizing networks inspired by the interaction of individual cells. The applications that they consider are network security and wireless sensor networks. The biological inspiration of their work mainly focuses on the similarities between the signaling that happens in biological cells and between nodes in technological networks, as well as the feedback loops introduced due to these signaling mechanisms. In contrast our work while also inspired by biological cells, focuses mainly on the developmental aspects of cells, as described by the process of morphogenesis and its application on the design of networks where nodes can independently and distributedly can differentiate their roles.

VIII. Conclusions And Future Work

The architecture described in this paper is part of the enabling architecture that can be used to develop a variety of computing networks. The extent to which the system can model a variety of systems depends on the richness of the mathematical expression that is used to represent the genetic code. In the initial architecture of the system, we have opted to use simple deterministic mathematical equations to represent the genetic code. The representation can be extended to include stochastic and probabilistic functions to determine the persona of individual machines. The extent to which such functions can augment the types of distributed system that can be represented is the subjective of active investigation in the ITA [14] program.

The architecture as described above can be implemented readily on by modifying traditional hypervisors such as VMWare or Xen to support the capabilities of the configurator machines. The distributed algorithms required to attain the architecture are relatively simple and can be implemented readily on current platforms. Thus, the architecture provides a simple, but highly effective scheme to obtain self-configuration behavior in a machine.

There are a few potential extensions to the architecture that can be made to enhance its capability. Instead of having a single personalized system on a physical node, it is possible to have more than one personalized system running on a single node. That allows for a single node to be able to take on the desired topology of the distributed system, or multiple nods to collaborate together to obtain the desired topology depending on which configuration is feasible at any time. The scalability of the system can also be increased by using an image server to provide the images for the personalized system, rather than storing all images on a local machine.

The architecture that we have presented works well when the final desired node is relatively static. However, in the context of a mobile ad-hoc network, the resulting system may tend to be unstable due to frequent changes in persona assignments. More research is needed to determine how this system could be made more stable in a mobile dynamic environment.

As we gain more experience with the architecture and mathematical models of the morphogenesis approach, the self-organizing distributed network would become a reality and ease the burden of configuring and installing a large variety of machines and appliances in the current IT environment.

REFERENCES

[1] http://www.gartner.com/4_decision_tools/measurement/measure_it_articles/july01/mit_spending_history1.html

[2] A. Ganek and T. Corbi, "The Dawning of the Autonomic Computing Era", IBM Systems Journal, vol. 42, no. 1, January 2003.

[3] J. O. Kephart and D.M Chess, ``The vision of autonomic computing'', , *IEEE Computer,* 36(1), 41-50, Jan 2003.

[4] S. Camazine, J.-L. Deneubourg, N.R. Franks, J. Sneyd, G. Theraula, and E. Bonabeau, Self-Organization in Biological Systems: (Princeton Studies in Complexity), 2003.

[5] S. Forrest, Ed., . Emergent Computation: Self-organizing, Collective and Cooperative Phenomena in Natural & Artificial Computing Networks, MIT Press, 1991.

[6] S. Wolfram, Cellular automata as simple self-organizing systems, 1992.

[7] M. Prokopenko, Ed., Advances in Applied Self-organizing Systems, Springer, 2007.

[8] F. Dressler Self-Organization in Sensor and Actor Networks, Wiley, 2008.

[9] E. Kaplan and C. Hegarty, "Understanding GPS: Principles And Applications", Artech House, 2006.

[10] N. Bulusu, J. Heidemann and D. Estrin, "GPS-less low-cost outdoor localization for very small devices", IEEE Personal Communications, vol 7. no. 5, pp. 28-34, Oct. 2000.

[11] H. Chu and R. Jan, "A GPS-less, outdoor, self-positioning method for wireless sensor networks", Ad-hoc Networks, vol. 5, no. 5, pp. 547-557, July 2007.

[12] D. Tennenhouse and D. Wetherall, Towards an active network architecture, ACM SIGCOMM Computer Communication Review, vol 37, no 5, October 2007.

[13] Sapuntzakis C. and M. S. Lam. ''Virtual appliances in the Collective: A Road to Hassle-free Computing,'' Proceedings of the 9th Hot Topics in Operating System, May, 2003.

[14] G. Cirincione and J. Gowens, "The International Technology Alliance In Network And Information Science A U.S.-U.K. Collaborative Venture", IEEE Comms. Mag., Vol 45, Issue 3, pp. 14-18, March 2007.

[15] G. Di Caro, F. Ducatelle, and L. M. Gambardella. AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. European Transactions on Telecommunications, Special Issue on Self-organization in Mobile Networking, 16(5):443-455, 2005.

[16] E. Curran and J. Dowling. Sample: Statistical network link modeling in an on-demand probabilistic routing protocol for ad hoc networks. International Conference on Wireless on Demand Network Systems and Service, 0:200-205, 2005.

[17] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. Lecture Notes in Computer Science, 2009:46, 2001.

[18] B. Biskupski, J. Dowling, and J. Sacha. Properties and mechanisms of self-organizing MANET and P2P systems. ACM Trans. Auton. Adapt. Syst., 2(1):1, 2007.

[19] M. de Leoni, M. Mecella, and G. D. Giacomo. Highly dynamic adaptation in process management systems through execution monitoring. Lecture Notes in Computer Science, 4714:182-197, 2007.

[20] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani. A framework for qos-aware binding and re-binding of composite web services. J. Syst. Softw., 81(10):1754-1769, 2008.

[21] G. De Giacomo, M. de Leoni, M. Mecella, and F. Patrizi. Automatic workflows composition of mobile services. IEEE International Conference on Web Services, 2007. ICWS 2007, pages 823-830, July 2007.

[22] Z. Maamar, Q. Z. Sheng, and B. Benatallah. On composite web services provisioning in an environment of fixed and mobile computing resources. Information Technology and Management Journal, Special Issue on Workflow and e-Business, 5:2004, 2004.

[23] P. F. Carles Sierr and N. R. Jennings. A service-oriented negotiation model between autonomous agents. In Human and Artificial Societies, LNAI, pages 201-219. Springer-Verlag, 1999.

[24] N. R. Jennings. An agent-based approach for building complex software systems. Commun. ACM, 44(4):35{41, 2001.

[25] T. Itao, T. Nakamura, M. Matsuo, and T. Suda. Adaptive networking architecture for service emergence. IEEE/IPSJ International Symposium on Applications and the Internet Workshops, 0:9, 2001.

[26] T. Nakano and T. Suda. Self-organizing network services with evolutionary adaptation. IEEE Transactions on Neural Networks, 16(5):1269-1278, 2005.

[27] B. Kruger, F. Dressler, Molecular Processes as a Basis for Autonomous Networking, in: Symposium on Challenges in the Internet and Interdisciplinary Research (IPSI), 2004.

[28] F. Dressler, B. Kruger, G. Fuchs, R. German, Self-Organization in Sensor Networks using Bio-Inspired Mechanisms, in: Proc. ACM/GI/ITG ARCS Workshop on Self-Organization and Emergence, 2005, pp. 139-144.

[29] F. Dressler, I. Dietrich, R. German, B. Kruger, Efficient operation in sensor and actor networks inspired by cellular signaling cascades, in: Proc. Autonomics, 2007, pp. 1-10.