An Artificial Hormone System for Self-organization of Networked Nodes

Wolfgang Trumler, Tobias Thiemann, and Theo Ungerer

Institute of Computer Science
University of Augsburg
86159 Augsburg
Germany
{trumler,ungerer}@email.address, thiemato@web.de

Abstract. The rising complexity of distributed computer systems give reason to investigate self-organization mechanism to build systems that are self-managing in the sense of *Autonomic* and *Organic Computing*. In this paper we propose the Artificial Hormone System (AHS) as a general approach to build self-organizing systems based on networked nodes. The Artificial Hormone System implements a similar information exchange between networked nodes like the human hormone system does between cells. The artificial hormone values are piggy-backed on messages to minimize communication overhead.
To show the efficiency of the mechanism even for large scale systems we implemented a simulation environment in Java to evaluate different optimization strategies. The evaluations show that local information is enough to meet global optimization criterion.

1 Introduction

State of the art computer and software systems raise the level of complexity to unexpected heights. Upcoming ubiquitous environments with their huge amount of different devices and sensors aggravate the situation even more. The need for self-managing systems, as proposed by IBM's *Autonomic Computing* [1] and the German *Organic Computing* [2] initiative, has never been more important than today. Future systems are expected to comprise attributes motivated by self-organizing natural systems as e.g. organisms or societies.

Self-organization seems to be embedded in biological systems in many ways. In the human body it arises at lower levels (e.g. the growth of an organism controlled by hormones) as well as at higher levels (the regulation of the homoeostasis by the autonomic nervous system).

We developed the AMUN middleware [3] to foster the development of ubiquitous computing environments such as Smart Office environments [4]. The aim of the middleware is to build service-oriented applications that can be distributed on networked nodes. The services which can be distributed and relocated between the nodes are monitored to collect information about their runtime behavior and resource consumption.

To implement the autonomic/organic goals within the AMUN middleware we propose a self-organization mechanism based on the human hormone system. The basic idea is to use the messages from services as containers for the information of our Artificial Hormone System like the hormones use the blood circuit in the human body. Each node locally collects information and packs its information onto outgoing messages. The important point is that no extra communication mechanism is used and no explicit messages are generated to exchange the information for the self-organization.

The remainder of the paper is structured as follows: In section 2 the human hormone system is described from an abstract point of view with focus on the mechanisms that were transferred to the Artificial Hormone System. Related work is presented in section 3. The model of the Artificial Hormone System is described in section 4 and evaluated in section 5. The paper closes with a conclusion and future work.

2 Human Hormone System

The human body contains different systems for information transport, depending on the speed of the stimulus transportation and the size of the affected area. The central nervous system is used for fast information transport of consciously given commands whereas the autonomic nervous system and the hormone system are mostly detracted from human's will. Together they regulate the human body to keep the inner environment as constant as possible, called homoeostasis.

The human hormone system differs from the autonomic nervous system in the way information is transported. The autonomic nervous system uses electrical impulses, whereas the hormone system uses chemical messengers (hormones). Beside the mostly known way to spread hormones through the blood circuit to trigger reactions in other parts of the body, a number of other kinds of messengers are known, which do not influence our Artificial Hormone System and therefore are not described here (see [5], [6], and [7] for further details).

The hormones spread to the blood circuit can theoretically reach every cell, but only act on those cells which possess the corresponding receptors. The hormones bind to the receptors and send their information to the inside of the cell on a chemical basis. The usage of receptors has two advantages. First the binding of a hormone can be shortened to designated cells, and second the same hormone can lead to different effects within different cells.

The reaction of the cells is controlled by the distribution of the activator and inhibitor hormones. In many cases the production of activator hormones lead to the production of inhibitor hormones to neutralize the reaction of the cells. This mechanism is called a negative feed-back loop because of the repressive effect of the inhibitors.

3 Related Work

In 1952 Alan Turing published "The Chemical Basis of Morphogenesis" [8]. Inspired by an invertebrate freshwater animal called hydra, he investigated the mathematical foundations for the explanation of the chemical processes during a morphogenesis. The hydra has the remarkable feature, that if it is cut in half, the head end reconstitutes a new foot, while the basal portion regenerates a new hydranth with a mouth and tentacles. During this process the available cells are reformed, moved, and redifferentiated to build new tissue without cell proliferation. One of the application areas of Turing's reaction-diffusion model is the generation of textures as described in [9].

The Digital Hormone Model [10] adopts the idea of hormone diffusion to the distributed control of robot swarms. The equations investigated by Turing combined with the limited wireless communication capabilities of robots can be mapped to the reaction-diffusion model known from tissues. The information of a robot can diffuse to the neighboring robots in vicinity. Simulations showed that such a distributed control algorithm can be used to control a robot swarm to fulfill different tasks. The challenges were to find and size targets, to spread and monitor a region, and to detour obstacles during mission execution.

For peer-to-peer and sensor networks so called Bio-Inspired approaches are currently investigated. In [11] an approach for a self-organizing peer-to-peer system in terms of load balancing is described. The authors show by simulation, that only local knowledge and the distribution of local information to randomly chosen nodes are sufficient to balance the load of the nodes nearly optimal. For sensor networks an approach exists that mimics the control loop for the blood pressure of the human body [12]. Information is disseminated in the network like hormones in the human body and a feedback loop is used to decide whether the requested task is already accomplished.

4 Model of the Artificial Hormone System

The AHS consists of four parts that can be directly mapped to their human counterparts. First we have metrics to calculate a reaction (transfer of a service in our case), which can be compared to the functions of the cells. Second, the hormone producing tissues of the human body are influenced by receptors which observe the environment to trigger hormone production. This behavior is modeled by monitors which collect information locally about the running services and piggy-back this information onto outgoing messages. Third, monitors for incoming messages collect the piggy-backed information and hand them over to the metrics, which is the same as the receptors of the cells does. And fourth, the digital hormone values carrying the information.

To further reduce the amount of information needed to exchange, we assume that the digital hormone value enfolds both, the activator as well as the inhibitor

hormone. If the value of the digital hormone is above a given level, it activates while a lower value inhibits the reaction.

The AHS will be included into the AMUN middleware to implement the self-optimization. The services can be relocated during runtime to meet the optimization criterion of the self-optimization as good as possible. Thus the model of the AHS must take some requirements into account given by AMUN. In AMUN services run on the nodes and can be transferred to other nodes. The services consume resources and the optimization should distribute the services such that the resource consumption on all nodes is nearly equal. Resources can be CPU, memory, communication bandwidth, battery status, radio range or any other resource suitable for the optimization of an application.

The idea of the AHS is to define the resources used for the optimization process and a corresponding digital hormone value. The model assumes that the values of the hormone values are normalized to fit into an interval from 0 to 100. The hormone value describes the consumption of a resource. In combination this can be interpreted that the hormone value represents the percentage consumption of a resource. As mentioned above these values are calculated by the monitors which observe the services in the middleware. In the model we have to define these values initially for the simulation process.

4.1 Mathematical Model

Given k resources, the consumption of resource i for service s is characterized by $r_i(s)$, $1 \leq i \leq k$, which is normalized such that all resources i satisfy the condition $0 \leq r_i(s) \leq 100$. The allocated amount of a resource i for all services $s \in \mathcal{S}(n)$ (the set of running services on node n) is computed as follows:

$$R_i(n) = \sum_{s \in \mathcal{S}(n)} r_i(s) \tag{1}$$

$R_i(n)$ is the total consumption of resource i of all services on node n. In our model the maximum capacity for each resource i on every node n is always 100. So any time the inequation $0 \leq R_i(n) \leq 100$ holds for all resources i and all nodes n. The number of services assigned to one node is limited by the nodes capacity and must not be violated by a service transfer.

To balance the load within the network it is assumed that services can be transferred from one node to another. The cost for the transfer to another node is modeled as the service transfer cost $tc(s)$, $0 \leq tc(s) \leq 100$ which can be different for every service.

Simulation initialization The initial state of our simulation model is chosen randomly to ensures that our optimization strategies are evaluated under a great variety of start conditions.

To initialize a node, a random maximum capacity value between 0 and $c_{\max}$ is chosen for every resource of the node. Then services are created with random

values for each resource. The maximum value for $c_{\max}$ of a node and the resources $r_{i_{\max}}$ of a service are defined prior to the initialization of the simulation. Services are generated as long as the maximum capacity of the node's resources are not exhausted as defined by equation 1. If a node is filled with services the procedure is repeated for the next node until all nodes are initialized. Regarding the mean value of the nodes resource consumption, this guarantees that for l nodes we generate a total load of $\approx l\frac{c_{\max}}{2}$ randomly distributed on all nodes.

On the other hand this assures that the number of generated services differ from every initialization depending on the number of nodes, the maximum initial capacity of the nodes $c_{\max}$, and the maximum values given for the service resource parameters $r_{i_{\max}}$.

Service communication model As the digital hormone values are piggy-backed on the outgoing communication, it is important to model the communication paths between the services. The communication model relies on the assumption that certain services prefer to communicate with each other while others don't. This behavior can be observed by many systems where the usage dependencies between different modules or classes are given due to the structure of the application and the resulting method calls.

Hence each service is assigned a constant set of other services acting as its communication partners. In each step of the simulation a given number of services are chosen to be senders and for each of these services s_{send} a receiving service s_{rec} is randomly picked out of their communication partners. Assuming that service s_{send} is running on node n_{send} and service s_{rec} is running on node n_{rec} the information necessary for the optimization is piggy-backed on the message sent from n_{send} to n_{rec}.

If there is a node which had no services assigned during the initialization, it could not participate in the optimization process thus loosing the free capacity of this node. In a networked system this situation might occur if a node was not present during the initial resource allocation phase. Such nodes will connect other nodes of the network sooner or later which is modeled by a certain probability to choose empty nodes as sender nodes n_{send}. As these nodes do not have a predetermined set of receivers a receiver node n_{rec} is randomly chosen out of all other nodes.

4.2 Transfer Strategies

In the optimization process node n_{send} adds information about its resource consumption to a message sent by one of its services to node n_{rec}. Based on this information node n_{rec} uses a *Transfer Strategie* (TS) to decide whether to transfer one of its services to n_{send} or not. To avoid confusion it should be mentioned that the terms n_{send} and n_{rec} refer to the direction of message passing. However, a service is transferred in the opposite direction from n_{rec} to n_{send}. In the following we present different Transfer Strategies from very simple to more sophisticated ones. All TS have in common that they only have local

knowledge and those information received from its communication partners. Furthermore there is no central instance to control the optimization process. Only the simulation environment has the possibility to check the state of the nodes after every simulation step to produce evaluation output.

Weighted Transfer Strategy The first TS is called *Weighted Transfer Strategy* because it uses weights for every resource to weight them in the decision process. The load of all resources multiplied by the corresponding weights are added and divided by the total amount of the weights. This value is the load value of a node. If all weights are one, all resources are treated equally, if one resource is weighted higher than the others evaluations show that the higher weighted resource is optimized better at the expense of the others.

If the node n_{rec} got a message from node n_{send} it subtracts the load value of n_{send} from its own. If the result is positive this means that the sender has a lower load than this node and that a service might be transferred to the sender. If the result is equal or less than zero, nothing will be done, as the sender's load is already equal or higher than the local load. To find a service for the transfer the loads of all services are compared to the half of the load difference and the service with the minimal distance to that value will be chosen for the transfer, as it best adjusts the load values of both nodes.

Transfer Strategy with Dynamic Barrier This TS uses a barrier to suppress the transfer of services. Evaluations showed, that the Weighted Transfer Strategy produces a high amount of service transfers beyond the estimated value. The TS with Dynamic Barrier continuously calculates the gain that could be obtained if a service would have been transferred. The dynamic barrier rises if the gain falls and vice versa.

The barrier can be calculated with a linear, an exponential, and a logarithmic function, while the latter produces the best results in terms of responsiveness due to load changes and attenuation behavior.

Transfer Strategy with Load-Estimator The problem of the Weighted TS is the missing knowledge about the total system load. If every node would know the load of all other nodes the mean load could be calculated and every node could trigger or prevent transfers depending on the load of the receiver and the sender node. With this TS a node keeps the last n load values of its sender nodes and calculates a mean load value. A service will only be transferred, if the senders' load is below and the local load above this value. A transferred service would not only optimize the load between the two nodes but also the overall load of the system, which is not always the case with the Weighted TS.

The TS with Load-Estimator is the best TS in terms of required service relocations. It nearly reaches the calculated theoretical optimal value. The drawback of this Strategy is the missing tolerance to dynamic service behavior. If the services change their load dynamically the strategy tries to further optimize the system and thus produces unwanted service relocations. The same applies for the Weighted TS.

Hybrid Transfer Strategy The fourth TS combines the advantages of the TS with Load-Estimator and the TS with Dynamic Barrier. At the beginning of the optimization process only the TS with Load-Estimator is used. If the change of the calculated estimated load is less than 10% within two consecutive steps, the TS with Dynamic Barrier is used additionally which results in an attenuated transfer behavior especially in case of dynamically changing service loads.

5 Evaluation

To evaluate the self-organization of the AHS we developed a simulation environment in Java. For the simulations we chose three resource parameters for the optimization. The different TS were simulated and the service transfers and the mean error of the average resource consumption have been measured. The simulator uses steps to perform the calculation and produces output after every step. We ran every simulation with 1000 nodes and 2000 steps a 100 times to take into account different starting conditions due to the random initialization process of the simulator. Thus we can show that the TS are independent of the initialization and produce good and stable results.

5.1 Service Transfers

Figure 1 shows the amount of service transfers of the different Transfer Strategies and the mean error of the average resource consumption. The mean error is the mean difference of a node's resource consumption from the expectation of the resource consumption. This value shows how good the algorithm reaches the theoretical optimum. The charts show that all four TS achieve good optimization results with a mean error ranging from 0.62 to 1.47 with varying amounts of service transfers. The Weighted TS needs about 5300 transfers, the Dynamic Barrier 3000, the Load-Estimator needs about 2300, and the Hybrid TS uses about 1750 transfers after 2000 simulations steps.

The mean resource consumption of a node for the simulations was 41,25. The lowest mean error reached with the TS with Load-Estimator is 0,62. This means that the optimization achieves 98,5% of the theoretical optimum. The least service transfers are generated by the Hybrid TS which achieves a mean error of 1,47 resulting in an optimization quality of 96,5% of the theoretical optimum. The Hybrid TS offers the best trade-off between service transfers and optimization quality.

With the amount of produced services during the initialization phase and the mean error at the beginning of the simulation the theoretical amount of service transfers needed to optimize the network can be calculated. For the above simulations this value is about 2300 service transfers. This implies that the TS with Load-Estimator produces the best result with the least error and that the Hybrid TS does not utilize all transfers as the dynamic barrier suppresses those transfers with a low gain.

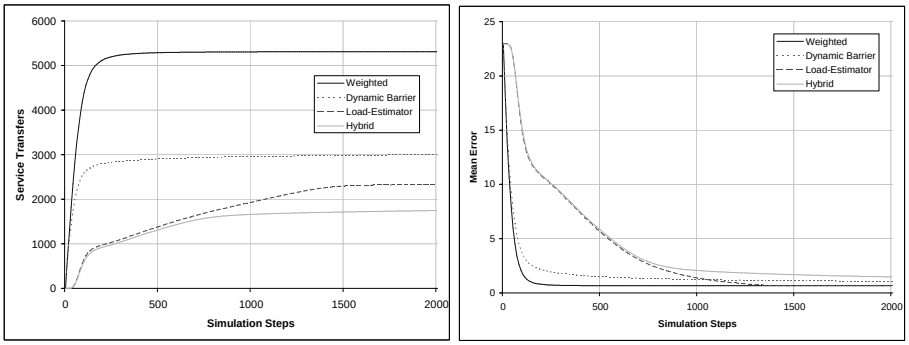

Fig. 1. Service transfers and the corresponding mean errors for the different TS.

Variation of Node Number To test the scalability of the systems we simulated networks with 100 to 5000 nodes. The number of services created is directly proportional to the number of nodes as expected from the initialization as described in section 4.1. Figure 2 shows that the service transfers increase only linear with the amount of nodes and services. The mean error remains nearly constant for all TS.

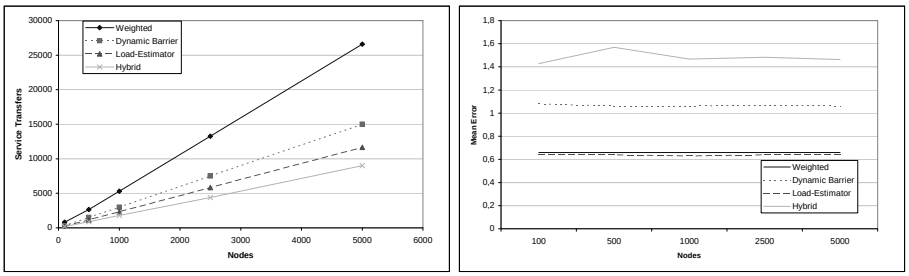

Fig. 2. Service transfers and mean error of the TS at varying node numbers.

Dynamic Services To evaluate the TS in a dynamic environment, we simulated a dynamic behavior of the services. Therefore a predefined amount of services change their resource consumption such that they raise or lower their resource consumption, hold that new value for a while and return to the initial value.

As expected the TS with Dynamic Barrier and the Hybrid TS tolerate the changes of the services' resource consumption while the other TS begin to oscillate resulting in a dramatic increase in service transfers. For both evaluations we defined 10% of the services to change their resource consumption up to 50% of the current value. The left chart of figure 3 shows the results if the change takes 10 simulation steps and 100 steps for the service to rest at its normal

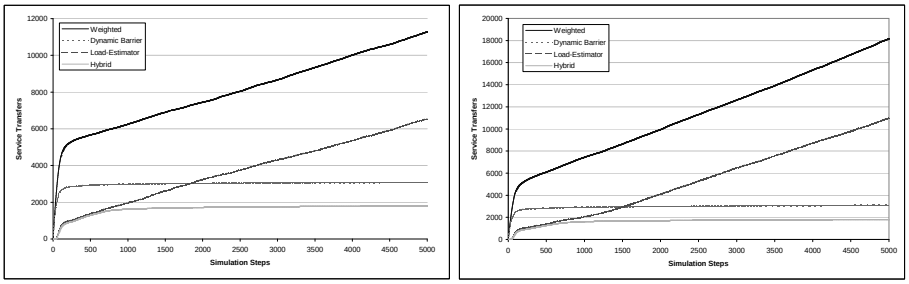

Fig. 3. Service transfers of the different TS with dynamic services.

level before the next change. The left chart shows the results for a 50 steps long change while the idle time is 100 steps which.

6 Conclusion and Future Work

In this paper we proposed the Artificial Hormone System for self-organization in networked systems. Derived from the human hormone system we developed a distributed algorithm that has neither a central control nor complete knowledge about the system. The organizational information is piggy-backed on top of the messages exchanged between the nodes of the networked system. The general approach of the AHS is described and implemented in a Java simulaton environment.

Evaluations showed that the AHS distributes services of networked nodes nearly optimal in terms of resource consumption and that the effort needed for larger networks increases only linear with the amount of nodes. Further evaluations regarding dynamic process behavior showed that the Hybrid Transfer Strategy tolerates load changes. The Hybrid Transfer Strategy is able to adapt the barrier to the mean load change, thus only few additional service transfers are produced due to dynamic process behavior.

Future work will be to implement and evaluate the AHS in AMUN and to further investigate the latencies produced by a service transfer and the corresponding costs for the optimization algorithm.

References

1. Horn, P.: Autonomic Computing: IBM's Perspective on the State of Information Technology. http://www.research.ibm.com/autonomic/ (2001)
2. VDE/ITG/GI: Organic Computing: Computer- und Systemarchitektur im Jahr 2010. http://www.gi-ev.de/download/VDE-ITG-GI-Positionspapier Organic Computing.pdf (2003)
3. et al., W.T.: AMUN - An Autonomic Middleware for the Smart Doorplate Project. In: UbiSys '04 - System Support for Ubiquitous Computing Workshop, Nottingham, England (2004)

4. Trumler, W., Bagci, F., Petzold, J., Ungerer, T.: Smart Doorplate. Personal and Ubiquitous Computing **7** (2003) 221–226
5. Silbernagl, S., Despopoulos, A.: Taschenatlas der Physiologie. Georg Thieme Verlag und Deutscher Taschenbuch Verlag, Stuttgart (2001)
6. Trepel, M.: Neuroanatomie Struktur und Funktion. Urban und Fischer Verlag, München,Jena (1999)
7. et al., F.H.: Biochemie des Menschen. Georg Thieme Verlag, Stuttgart (2002)
8. Turing, A.M.: The chemical basis of morphogenesis. In: Philosophical Trans. of the Royal Society of London. Volume 237 of Series B, Biological Sciences. (1952) 37–72
9. Witkin, A., Kass, M.: Reaction-diffusion textures. Computer Graphics **25** (1991) 299–308
10. Shen, W.M., Will, P., Galstyan, A., Chuong, C.M.: Hormone-inspired self-organization and distributed control of robotic swarms. Autonomous Robots **17** (2004) 93–105
11. et al., M.J.: A modular paradigm for building self-organizing peer-to-peer applications. In: Engineering Self-Organising Systems. Lecture Notes in Artificial Intelligence (2004) 265–282
12. et al., F.D.: Self-organization in sensor networks using bio-inspired mechanisms. In: ARCS'05: Workshop Self-Organization and Emergence. (2005) 139–144