A Formal Specification of the Hormone Loop of an Artificial Hormone System

Mathias Pacher
Institute of Systems Engineering
Leibniz Universität Hannover
Hannover, Germany
Email: pacher@sra.uni-hannover.de

Uwe Brinkschulte
Institute of Informatics
Goethe Universität Frankfurt am Main
Frankfurt am Main, Germany
Email: brinks@es.cs.uni-frankfurt.de

Abstract

The Artificial Hormone System (AHS) is a completely decentralized operation principle for a middleware which can be used to allocate tasks in a system of heterogeneous processing elements (PEs) or cores. Tasks are scheduled according to their suitability for the heterogeneous PEs, the current PE load and task relationships. The AHS also provides properties like self-configuration, self-optimization and self-healing in the context of task allocation. In addition, it is able to guarantee real-time bounds for such self-X-properties.

The operation principle of the AHS is based on the hormone loop. This is a sequence of actions and wait states executed periodically on each PE. We present a formal specification of the hormone loop in this paper. The outcome is to guarantee consistent hormone computation (important for holding the real-time bounds of the self-X properties) and a fast recognition of task or PE failures. Even more, we present an algorithm to terminate single PEs consistently.

1 Introduction

The complexity of technical systems – especially in the area of embedded systems – has increased dramatically. Reasons for the increase are the higher integration of circuits, shorter clock periods and lower power consumption leading to a miniaturization of microprocessors, microcontrollers and Systems on Chip. A result is the development and marketing of ubiquitous devices like small PCs, handhelds, cell or smart phones. In addition, several of these systems and devices are interconnected by busses or via the internet. An example is a modern car which contains up to 100 microcontrollers running crucial tasks like ABS, ESP, engine control, and the navigation system. It is obvious that programming such devices with their multiple interactions gets highly complex, especially if real-time aspects have to be considered. It is a grand challenge to develop and maintain such highly integrated and often distributed systems.

As a response, the Organic Computing (OC) Initiative was founded in 2002. It deals with theoretical and practical foundations to handle the complexity of technical systems described above inspired by mechanisms found in nature and biology. The Artificial Hormone System (AHS) was developed in the scope of the OC initiative as a powerful tool to handle the complexity of assigning tasks to processor cores (so called processing elements, PEs) in a self-organizing and completely decentralized way. Features like self-configuration, self-optimization and self-healing allow an autonomous and robust system operation without user intervention, while the decentralized approach avoids single-points-of-failure. The task assignment done by the AHS uses different kinds of artifical hormones to build up distributed closed control loops. These artificial hormones are represented by messages which are sent periodically by broadcast from a PE to all other PEs or by multicast from a PE to its neighboring PEs. The effect of the artificial hormones is locally defined by the receiving PEs. The assignment of different tasks on different PEs is bounded by antagonistic artificial hormones. Each PE performs sending, receiving and the computation of resulting hormones periodically. This is called the *hormone loop*. The PEs are able to run their hormone loops almost synchronously due to self-synchronisation [5]. Former publications such as [6, 4] show the properties of the AHS considering stability and real-time capability. In this paper, we present a formal specification of the time conditions for the different actions done by each PEs' hormone loops. This is essential to guarantee necessary properties such as strict timely isolation of receiving different types of hormones and consistent hormone computation.

The paper is structured as follows: In Section 2 we shortly present the AHS and explain the hormone loop in detail. The specification of the hormone loop is presented in Section 3 and discussed in Section 4. Section 5

presents the related work and finally, Section 6 concludes the paper.

2 The Artificial Hormone System

The aim of the AHS is to assign tasks to PEs in a self-organizing way i.e., it uses three main types of hormones:

Eager value This hormone type determines the suitability of a PE to execute a task. The higher the hormonal value the better the ability of the PE to execute the task.

Suppressor This hormone type lowers the suitability of a task execution on a PE. Suppressors are subtracted from eager values. There exist several subtypes of suppressors e.g., the *task suppressor* to prevent duplicate task allocation, the *monitoring suppressor* to indicate a deteriorating PE state and the *load suppressor* to indicate the current load of a PE caused by the executed tasks. While the task suppressor is broadcasted to all PEs in the system, the monitoring and load suppressors are only used locally to limit the number of executed tasks on a PE.

Accelerator This hormone type favors the execution of a task on a PE. Accelerators are added to eager values. Like for suppressors there exist several subtypes of accelerators. The most important one is the *organ accelerator* used to cluster cooperating tasks in the neighborhood. This accelerator is multicasted to neighboring nodes to form so called 'virtual organs' of cooperating tasks. Another subtype is the *monitoring accelerator*, which is used locally to indicate improved PE capabilities.

More details on these subtypes of hormones are presented when needed because they are used for fine tuning of the AHS and do not contribute to its basic understanding.

We have to distinguish between received hormones and hormones to be sent and also between tasks and processors. Therefore, we use Latin letters such as i as task indices and Greek letters such as γ as processor indices. A hormone of any type denoted by $H^{i\gamma}$ with superscripted indices signifies that this hormone is dedicated to and will be received by PE γ and task T_i. For the operation of the hormone loop it is not necessary to know who the sender of this hormone is. A hormone of any type denoted by $H_{i\gamma}$ signifies that this hormone is sent by PE γ and task T_i to other PEs. The receiver of this hormone depends on the hormone type, as can be seen below.

The task assignment happens in the following way: Each PE periodically executes the hormone based control loop (*hormone loop*) presented in Figure 1. Each iteration consists of three stages.

Receive stage PE γ receives the modified eager values $Em^{i\gamma}$, suppressors $S^{i\gamma}$ and accelerators $A^{i\gamma}$ for each task T_i from each PE in the network. The communication between the different PEs is depicted by the dashed lines.

Compute and decision stage PE γ computes the modified eager values $Em_{i\gamma}$ for all of its tasks in the following way. The local static eager value $E_{i\gamma}$ indicates how suited PE γ is to execute task T_i. From this value, all suppressors $S^{i\gamma}$ received by task T_i are subtracted, and all accelerators received by task T_i are added:

$$Em_{i\gamma} = E_{i\gamma} - \sum S^{i\gamma} + \sum A^{i\gamma}$$

The modified eager value $Em_{i\gamma}$ of each task T_i is then waiting to be broadcasted to task T_i on the other PEs in the send stage.

In each iteration a single task T_i is selected and the PE decides on its allocation. For this purpose it compares its own modified eager value $Em_{i\gamma}$ with the received modified eager values $Em^{i\gamma}$ (from all other PEs) for this task. If $Em_{i\gamma} > Em^{i\gamma}$ is true for all received modified eager values PE γ decides to take the task. In case of equality, a second criterion e.g., the smallest position identifier of the PEs, is used to get an unambiguous decision. Otherwise another PE has the highest modified eager value for task T_i and PE γ decides not to take it.

In the next iteration step the PE selects another task and decides whether it will be taken. A PE selects the tasks in a cyclic way i.e., each task will be selected in each m-th iteration if m tasks have to be assigned. By selecting only one task at each iteration the suppressors and accelerators can take effect. Otherwise the decision of taking a task would happen instantaneously and the hormones would have no effect.

Send stage As already mentioned above, PE γ broadcasts the modified eager values $Em_{i\gamma}$ to each task T_i on the other PEs. The strength of these values depends on the results of the computation in the previous phase.

If a task T_i is taken on PE γ, it also broadcasts suppressors $S_{i\gamma}$ dedicated to the same task on all other PEs. On the one hand sending the suppressors indicates the PE has taken the task, and on the other hand it limits the number of allocations of this task.

Furthermore, the PE multicasts accelerators $A_{i\gamma}$ to its neighbored PEs to attract tasks cooperating with task T_i to neighbored PEs thus forming clusters of tasks.

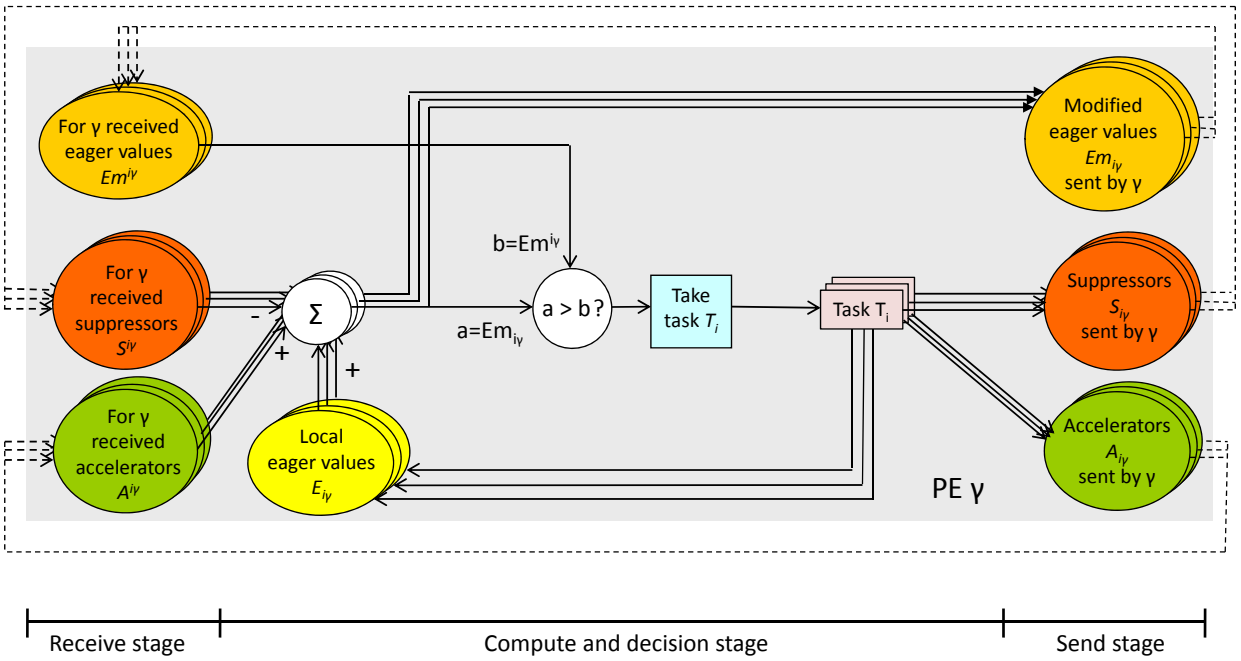

Figure 1: Hormone based control loop

Our approach is completely decentralized, each PE is responsible for its own tasks and the communication with other PEs is realized by a unified hormone concept. As explained in [4] the AHS offers self-X-properties like self-configuration, self-optimization and self-healing. In addition, the self-configuration is real-time capable. Tight upper time bounds are given for self-configuration, these are presented in detail in [6, 25].

The indices of the hormones are not needed in the following chapters. Therefore, we name the PEs with latin indices henceforth for the sake of simplicity.

3 A Formal Specification of the Hormone Loop

The general scheme of the hormone loop is 1) receiving hormones, 2) computation of the new hormone values and decision on the task assignment and 3) sending of the new hormone values. It is obvious that these actions should work synchronously on all PEs of the system because asynchronous execution could lead to misbehavior. As an example we consider two PEs running asynchronously: Let us assume PE P_1 is in the sending stage yet while PE P_2 is in the computation stage yet. As a result, it may happen that PE P_2 did not receive the hormones sent by PE P_1 in time because P_1's hormones were sent too late. This means that the values of the hormones are inconsistent on the two PEs which may lead to an incorrect task assignment. Furthermore, the real-time bounds of self-configuration, self-optimization and self-healing may be harmed in case of PE asynchronity. This can be fatal for hard real-time systems. Therefore, we have to ensure synchronous execution of the participating PEs.

The main idea is to launch the PEs with a delay of η time units (TUs) at most ($0 \leq \eta < \infty$) and to introduce wait states after each critical action. Critical actions are 1) sending of suppressors and accelerators, 2) sending of eager values and 3) computation of the new hormone values and decision on the task assignment. The wait states must be large enough to guarantee the completion of each critical action on each PE. Figure 2 shows the approach which also respects jitter in the wait states as well as jitter in the execution of the actions.

We assume three conditions in the following:

1. The communication time (the time it takes from sending a hormone by one arbitrary PE up to its receiving by any other arbitrary PE) is bounded and we know its upper limit t_K time units ($0 < t_K < \infty$).

2. The PEs are launched with a *pairwise delay* of η time units at most.

3. The jitter of the wait states and of the actions is bounded. The jitter of each wait state i is quantified by a factor p_i with $0 \leq p_1 < 1$. This results in a minimal jitter of 0 time units and a maximal jitter of the duration of the wait state itself which is a reasonable assumption.

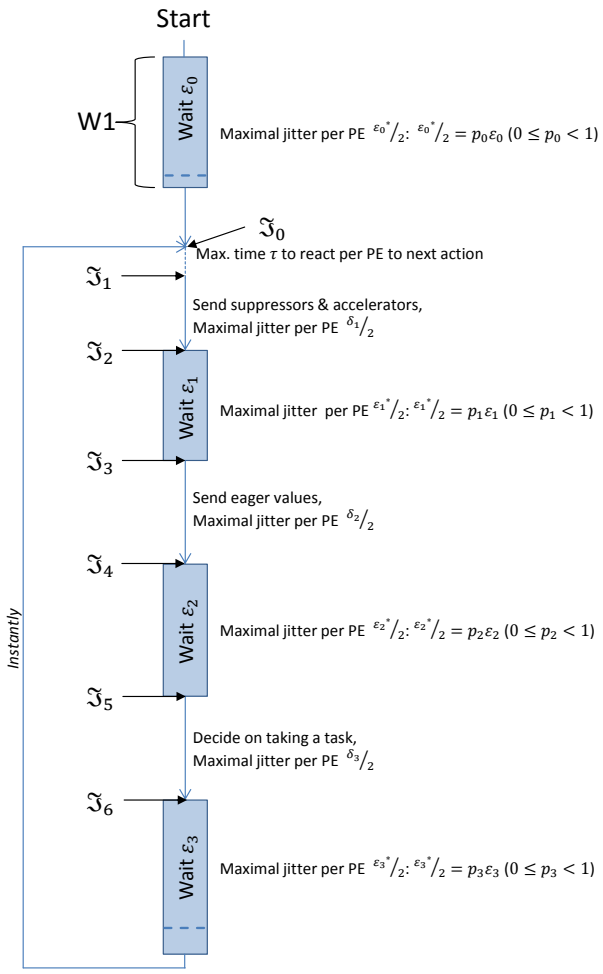

Figure 2: Specification of the hormone loop

The jitter of the actions j is quantified by δ_j in the following.

In detail, the hormone loop of our approach works as follows: The PEs are launched with a delay of η time units at most and each of them waits for ε_0 time units at most (wait state W1); $\varepsilon_0 := \eta + \gamma_0$. Here, a maximal jitter of $\frac{\varepsilon_0^*}{2}$ is allowed, see Fig. 2, and let $\gamma_0 > \frac{p_0\eta}{1-p_0}$ be true.

There are two ways to exit wait state W1: The first is by timeout. After waiting the time ε_0 each PE is in the point $\mathfrak{I}_0$ in Fig. 2. It immediately jumps to $\mathfrak{I}_1$. The second way to exit W1 is that a PE receives a hormone (illustrated by the dashed line in Fig. 2). Then, it immediately exits W1 and is also at the point $\mathfrak{I}_0$ in Fig. 2. It reaches $\mathfrak{I}_1$ with a maximal delay of τ time units (they may occur due to hormone recognition time).

When a PE reaches $\mathfrak{I}_1$ it immediately starts sending suppressors and accelerators. This takes s_1 time units including a maximal jitter of $\frac{\delta_1}{2}$. Obviously, $\frac{\delta_1}{2} < s_1$ is true. After sending these hormones, each PE reaches $\mathfrak{I}_2$ and begins a wait state of ε_1 time units; $\varepsilon_1 := 2t_K + \tau + \delta_1 + \gamma_1$. Here, a maximal jitter of $\frac{\varepsilon_1^*}{2}$ is allowed, see Fig. 2, and let $\gamma_1 > \frac{p_1(2t_k+\tau+\delta_1)}{1-p_1}$ be true. After exiting the wait state each PE starts sending its eager values. This takes s_2 time units including a maximal jitter of $\frac{\delta_2}{2}$. Obviously, $\frac{\delta_2}{2} < s_2$ is true.

This is continued up to point $\mathfrak{I}_6$ in the same manner. Therefore, we only provide the numbers for the wait states and jitter duration:

$$\begin{aligned} \varepsilon_2 &:= 2t_K + \tau + \delta_1 + \delta_2 + \varepsilon_1^* + \gamma_2 \\ \gamma_2 &> \frac{p_2(2t_K + \tau + \delta_1 + \delta_2 + \varepsilon_1^*)}{1-p_2} \end{aligned} \tag{1}$$

$$\begin{aligned} \varepsilon_3 &:= 2t_K + \tau + \delta_1 + \delta_2 + \delta_3 + \varepsilon_1^* + \varepsilon_2^* + \gamma_3 \\ \gamma_2 &> \frac{p_3(2t_K + \tau + \delta_1 + \delta_2 + \delta_3 + \varepsilon_1^* + \varepsilon_2^*)}{1-p_3} \end{aligned} \tag{2}$$

Each PE exits the last wait state (beginning from $\mathfrak{I}_6$) either because of timeout or because it receives a hormone (illustrated by the dashed line in Fig. 2). This is the same process as in wait state W1.

Definition 1. *The triple (P_i, $\mathfrak{I}_x$, n) indicates PE P_i which reached the point $\mathfrak{I}_x$ in its n-th run of the hormone loop ($x \in \{0, 1, 2, 3, 4, 5, 6\}$).*

Definition 2. *The map C assigns each triple (P_i, $\mathfrak{I}_x$, n) the point in time at which PE P_i reached $\mathfrak{I}_x$ in its n-th run of the hormone loop.*

We now provide seven invariants in the following theorem. Their main outcome is that the critical actions are timley separated.

Theorem 1. *Let M be a set of PEs launched with a maximal delay of η time units. Let $\mathfrak{P}(n)$ be the set containing the PEs which are in their n-th run since the launch of M. Then the invariants $I_0, I_1, I_2, I_3, I_4, I_5, I_6$ are true in the following order:*

$$I_0 \to I_1 \to I_2 \to I_3 \to I_4 \to I_5 \to I_6 \to I_0 ...$$

The invariants are:

$$\begin{aligned} I_0 ::= &\forall P_i, P_j \in \mathfrak{P}(n) : \\ &|C(P_i, \mathfrak{I}_0, n) - C(P_j, \mathfrak{I}_0, n)| \le t_K \quad \wedge \\ &(P_i, \mathfrak{I}_0, n) \textit{ by timeout} : C(P_i, \mathfrak{I}_1, n) = C(P_i, \mathfrak{I}_0, n) \quad \wedge \\ &(P_i, \mathfrak{I}_0, n) \textit{ by hormones} : |C(P_i, \mathfrak{I}_1, n) - C(P_i, \mathfrak{I}_0, n)| \le \tau \quad \wedge \\ &[(P_i, \mathfrak{I}_0, n) \wedge (n > 1)] \longrightarrow \big[[\forall P_k \in \mathfrak{P}(n-1)] \longrightarrow \\ &\qquad (P_k, \mathfrak{I}_6, n-1) \textit{ passed since more than } t_K \textit{ TUs}\big] \end{aligned}$$

$$
\begin{aligned}
I_1 ::= & \forall P_i, P_j \in \mathfrak{P}(n): \\
& |C(P_i, \mathfrak{I}_1, n) - C(P_j, \mathfrak{I}_1, n)| \leq t_K + \tau \quad \wedge \\
& |C(P_i, \mathfrak{I}_2, n) - C(P_i, \mathfrak{I}_1, n) - s_1| \leq \frac{\delta_1}{2}
\end{aligned}
$$

$$
\begin{aligned}
I_2 ::= & \forall P_i, P_j \in \mathfrak{P}(n): \\
& |C(P_i, \mathfrak{I}_2, n) - C(P_j, \mathfrak{I}_2, n)| \leq t_K + \tau + \delta_1 \quad \wedge \\
& |C(P_i, \mathfrak{I}_3, n) - C(P_i, \mathfrak{I}_2, n) - \varepsilon_1| \leq \varepsilon_1^*
\end{aligned}
$$

$$
\begin{aligned}
I_3 ::= & \forall P_i, P_j \in \mathfrak{P}(n): \\
& |C(P_i, \mathfrak{I}_3, n) - C(P_j, \mathfrak{I}_3, n)| \leq t_K + \tau + \delta_1 + \varepsilon_1^* \quad \wedge \\
& |C(P_i, \mathfrak{I}_4, n) - C(P_i, \mathfrak{I}_3, n) - s_2| \leq \frac{\delta_2}{2} \quad \wedge \\
& (P_i, \mathfrak{I}_3, n) \longrightarrow (P_j, \mathfrak{I}_2, n) \textit{ passed since more} \\
& \qquad \textit{than } t_K \textit{ TUs}
\end{aligned}
$$

$$
\begin{aligned}
I_4 ::= & \forall P_i, P_j \in \mathfrak{P}(n): \\
& |C(P_i, \mathfrak{I}_4, n) - C(P_j, \mathfrak{I}_4, n)| \leq t_K + \tau + \delta_1 \\
& \qquad + \delta_2 + \varepsilon_1^* \quad \wedge \\
& |C(P_i, \mathfrak{I}_5, n) - C(P_i, \mathfrak{I}_5, n) - \varepsilon_2| \leq \varepsilon_2^*
\end{aligned}
$$

$$
\begin{aligned}
I_5 ::= & \forall P_i, P_j \in \mathfrak{P}(n): \\
& |C(P_i, \mathfrak{I}_5, n) - C(P_j, \mathfrak{I}_5, n)| \leq t_K + \tau + \delta_1 \\
& \qquad + \delta_2 + \varepsilon_1^* + \varepsilon_2^* \quad \wedge \\
& |C(P_i, \mathfrak{I}_6, n) - C(P_i, \mathfrak{I}_5, n) - s_3| \leq \frac{\delta_3}{2} \quad \wedge \\
& (P_i, \mathfrak{I}_5, n) \longrightarrow (P_j, \mathfrak{I}_4, n) \textit{ passed since more} \\
& \qquad \textit{than } t_K \textit{ TUs}
\end{aligned}
$$

$$
\begin{aligned}
I_6 ::= & \forall P_i, P_j \in \mathfrak{P}(n): \\
& |C(P_i, \mathfrak{I}_6, n) - C(P_j, \mathfrak{I}_6, n)| \leq t_K + \tau + \delta_1 \\
& \qquad + \delta_2 + \delta_3 + \varepsilon_1^* + \varepsilon_2^* \quad \wedge \\
& |C(P_i, \mathfrak{I}_0, n+1) - C(P_i, \mathfrak{I}_6, n) - \varepsilon_3| \leq \varepsilon_3^*
\end{aligned}
$$

Proof.
Base case ($n = 1$).
$\mathbf{I_0}$: If a PE has launched its wait state W1 takes at least the time

$$
\begin{aligned}
\varepsilon_0 - p_0\varepsilon_0 &= (1-p_0)(\eta + \gamma_0) \\
&> (1-p_0)\left(\eta + \frac{p_0\eta}{1-p_0}\right) \\
&= \eta.
\end{aligned}
$$

This means that all PEs are in their wait state W1 at least when the first PE leaves its wait state W1 (especially, $\mathfrak{P}(1) = M$ is true). As mentioned above, this first PE jumps from $\mathfrak{I}_0$ to $\mathfrak{I}_1$ immediately and starts sending hormones. These hormones are received by each PE with a delay of t_K time units at most. If they are in W1, they jump to $\mathfrak{I}_0$ immediately. This means the first clause of I_0 is true considering the first PE to reach $\mathfrak{I}_0$. The absolute value of the difference of the arrival times of all other PEs is obviously smaller or equal than t_K, too. This proves the first clause of I_0. The second and third clause of I_0 are obviously true due to the assumptions in the detailed description of the hormone loop. The last clause of I_0 does not match here because $n = 1$.
$\mathbf{I_1}$: We know from I_0 that each PE of $\mathfrak{P}(1)$ has passed the point $\mathfrak{I}_0$ with a maximal delay of t_K. Each PE needs τ time units at most to reach the point $\mathfrak{I}_1$. This means all PEs will eventually reach $\mathfrak{I}_1$ (for the first time, $n = 1$). Thus, each PE will reach $\mathfrak{I}_1$ at most $t_K + \tau$ time units after the first PE has reached $\mathfrak{I}_1$. This is obviously true for the absolute value of the differences of the arrival times of all other PEs. This proves the first clause of I_1. The second clause is true due to the the assumptions in the detailed description of the hormone loop.
$\mathbf{I_2}$: We know from I_1 that each PE of $\mathfrak{P}(1)$ has passed the point $\mathfrak{I}_1$ with a maximal delay of $t_K + \tau$ time units. Each PE will reach the point $\mathfrak{I}_2$ due to the assumptions in the detailed description of the hormone loop after the time s_1 (a jitter of $\frac{\delta_1}{2}$ time units is possible). As shown in the proofs of I_0 and I_1 and due to the maximal jitter of $\frac{\delta_1}{2}$ per PE, the first clause of I_2 holds. The second clause is true due to the assumptions in the detailed description of the hormone loop.
$\mathbf{I_3}$: The proof of the first and the second clause is similar to the proofs in I_2. Now we consider the third clause. The first PE to reach $\mathfrak{I}_3$ (for the first time) has been waiting in its waiting state the time $\varepsilon_1 - p_1\varepsilon_1$ at least. It holds:

$$
\begin{aligned}
\varepsilon_1 - p_1\varepsilon_1 &= (1-p_1)\varepsilon_1 = (1-p_1)(2t_K + \tau + \delta_1 + \gamma_1) \\
&> (1-p_1)\left(2t_K + \tau + \delta_1 + \frac{p_1(2t_K + \tau + \delta_1)}{1-p_1}\right) \\
&= (1-p_1)(2t_K + \tau + \delta_1) + p_1(2t_K + \tau + \delta_1) \\
&= 2t_K + \tau + \delta_1.
\end{aligned}
$$

Each PE of $\mathfrak{P}(1)$ reaches the point $\mathfrak{I}_2$ (see I_2). It waits for more than $2t_K + \tau + \delta_1$ time units then. We know (from I_2) that the PEs pass $\mathfrak{I}_2$ with a maximal delay of $t_K + \tau + \delta_1$ time units: Therefore, it holds that if any PE is at $\mathfrak{I}_2$ any other PE will be in its waiting state for more than t_K time units. Thus, the time to reach $\mathfrak{I}_3$ takes more than t_K time units for these other PEs. We conclude that if any PE reaches the point $\mathfrak{I}_3$ any other PE has passed the point $\mathfrak{I}_2$ since more than t_K time units.
$\mathbf{I_4}$: Similar to the proof of I_2 using I_3 as precondiction.
$\mathbf{I_5}$: Similar to the proof of I_3 using I_4 as precondiction.
$\mathbf{I_6}$: Similar to the proof of I_2 using I_5 as precondiction.

Inductive step ($n \to n+1$).
We know from I_6 that all of $\mathfrak{P}(n)$'s PEs will eventually reach $\mathfrak{I}_0$ for the $(n+1)$-th time (induction hypothesis). Thus, $\mathfrak{P}(n+1) = \mathfrak{P}(n)$ is true. We know from I_6 that each PE of $\mathfrak{P}(n)$ is in its waiting state for $\varepsilon_3 - p_3\varepsilon_3$ time units at least. It holds

$$\begin{aligned}
\varepsilon_3 - p_3\varepsilon_3 &= (1-p_3)\varepsilon_3 \\
&= (1-p_3)(2t_K + \tau + \delta_1 + \delta_2 + \delta_3 + \varepsilon_1^* + \varepsilon_2^* \\
&\qquad + \gamma_3) \\
&> (1-p_3)\Bigg(2t_K + \tau + \delta_1 + \delta_2 + \delta_3 + \varepsilon_1^* + \varepsilon_2^* + \\
&\qquad \frac{p_3(2t_K + \tau + \delta_1 + \delta_2 + \delta_3 + \varepsilon_1^* + \varepsilon_2^*)}{1-p_3}\Bigg) \\
&= (1-p_3)(2t_K + \tau + \delta_1 + \delta_2 + \delta_3\varepsilon_1^* + \varepsilon_2^*) + \\
&\qquad p_3\,(2t_K + \tau + \delta_1 + \delta_2 + \delta_3\varepsilon_1^* + \varepsilon_2^*) \\
&= 2t_K + \tau + \delta_1 + \delta_2 + \delta_3 + \varepsilon_1^* + \varepsilon_2^*.
\end{aligned}$$

Each PE of $\mathfrak{P}(n)$ reaches the point $\mathfrak{I}_6$ (because of I_6 of the induction hypothesis) and then, it waits more than $2t_K + \tau + \delta_1 + \delta_2 + \delta_3 + \varepsilon_1^* + \varepsilon_2^*$ time units and reaches $\mathfrak{I}_0$ for the $(n+1)$-th time. We also know (from the induction hypothesis) that the PEs passed $\mathfrak{I}_6$ with a delay of $t_K + \tau + \delta_1 + \delta_2 + \delta_3 + \varepsilon_1^* + \varepsilon_2^*$ time units at most. Thus, if a PE passes $\mathfrak{I}_6$ (at the n-th time) we know that each other PE is in the corresponding waiting state for t_K time units at least.

If the first PE of $\mathfrak{P}(n)$ passes $\mathfrak{I}_0$ for the $(n+1)$-th time by timeout it is obviously in $\mathfrak{P}(n+1)$. Furthermore, we know that all other PEs of $\mathfrak{P}(n)$ have passed $\mathfrak{I}_6$ for the n-th time since t_K time units at least. This proves the fourth clause of I_0.

If the first PE reaches $\mathfrak{I}_0$ for the $(n+1)$-th time all other PEs are in the last waiting state of their hormone loop n at least. Thus, the first clause of I_0 holds. The second and third clauses of I_0 hold due to the assumptions in the detailed description of the hormone loop.

The invariants I_1 up to I_6 of the induction step are proven as in the base case. □

4 Discussion

The invariants of Theorem 1 make the understanding of the hormone loop's workflow simpler. Let us consider the third clause of I_3: It says that each PE has passed $\mathfrak{I}_2$ since more than t_K time units if any PE passes $\mathfrak{I}_3$. This means that each PE has finished sending suppressors and accelerators for more than t_K time units if any PE exits its wait state at $\mathfrak{I}_3$. Therefore, any suppressors and accelerators sent in run n are received by any PE before it exits its wait state at $\mathfrak{I}_3$. As a result, each PE can calculate its eager values based on the up-to-date values of these hormones.

The sending and receiving of the eager values of each PE is finished in its n-th run before any PE launches its hormone computation and task assignment process. This follows from the third clause of I_5 with a similar argumentation as for the suppressors and accelerators.

The last clause of I_0 states that each PE has passed $\mathfrak{I}_6$ (thus, it is in its last wait state in run n) if the first PE reaches $\mathfrak{I}_0$ for the $(n+1)$-th time. This is the reason that the PEs can be re-synchronized to a maximal delay of t_K time units: Either they hold it by timeout or they hold it by activation due to hormone receiving. The latter option is possible as each PE of $\mathfrak{P}(n)$ is in its last wait state at least.

These observations show that the PEs obeying the specification provided in the last section are able to run with a maximal determined delay. Furthermore, we can guarantee that all relevant hormones are received by each PE before it starts to execute resulting actions or decisions. Thus, the AHS runs in a consistent way[1].

We also observe that the hormone loop presented in Fig. 2 differs from the characterisation in Section 2. The sending of suppressors and accelerators ($\mathfrak{I}_1$ up to $\mathfrak{I}_2$) is separated from the sending of eager values ($\mathfrak{I}_3$ up to $\mathfrak{I}_4$). As a result the receiving of suppressors and accelerators ($\mathfrak{I}_2$ up to $\mathfrak{I}_3$) is separated from the receiving of eager values ($\mathfrak{I}_4$ up to $\mathfrak{I}_5$). This means that we have two independant sending stages and two independant receiving stages considering the formalized hormone loop whereas there is only one sending and one receiving stage in Section 2.

A major advantage of this hormone loop is the simple detection of failing PEs. If a PE misses the suppressors and accelerators of another PE until it reaches $\mathfrak{I}_3$ or it misses the eager values of another PE until it reaches I_5 it marks the other PE as failing. Then it may skip the hormones of the failing PE and start with the bidding for the tasks of the failing PE from the beginning of the next cycle.

Consistent termination of single PEs. A user of the AHS may want to deactivate a single PE or several PEs. We assume that the maximal communication time to send a message between the user and any PE (and vice versa) is t_K.

A user could send a termination signal to a PE at any point of time. The PE would be deactived at the moment it received the signal. The correct workflow of the overall AHS is not affected by this way of deactivating a PE. The reason is that it is identical to the failing of the deactived PE to the other PEs. However, this is not a nice way because the running PEs may still have hormones of the deactived PE in their buffers.

Therefore, we propose the following algorithm to deactivate one or several PEs:

- The user snoops on eager values.
- As soon as the user snoops eager values from any PE, it sends the termination signals to the PEs to be deactivated.
- Each PE receiving a termination signal continues its hormone loop until it reaches $\Im_0$ the next time. At this point it stops running the hormone loop.

This algorithm works properly because if the user receives eager values, all PEs will be in their wait state before the point $\Im_5$. If the user sends the termination signal now, all PEs will be in their n-th run of the hormone loop. This is true because their last wait state takes more than t_K time units. This means that a termination signal sent in the n-th run of the PEs (according to the algorithm) will be received in the n-th run. Each receiving PE stops its execution consistently at the point $\Im_0$ of the $(n+1)$-th run. A result of this algorithm is that the user is able to terminate PEs consistently and no hormones are still in the buffers of the remaining PEs.

Hormone loop duration. In this paragraph, we want to study the impact of the hormone loop specification on the hormone loop duration. Former publictions like [2] provide a time limit of about $3t_K$ time units for the hormone loop duration. We use this value as a reference.

The duration t_{HL} of the hormone loop as specified in Section 3 *without jitter* can be calculated by summing up the initial duration of the reaction time, the duration of the sending and computing actions and the duration of the wait states:

$$t_{\mathrm{HL}} = \tau + s_1 + \varepsilon_1 + s_2 + \varepsilon_2 + s_3 + \varepsilon_3$$

First, we calculate a lower limit for t_{HL}. It is reasonable to assume the reaction time to be $\tau = 0$ time units because a PE has to set a launch flag only if it receives a hormone. Furthermore, we assume the sending and computing action time to be 0 according to the hormone loop specification and as we search a lower limit. An estimation of the lower limit of the wait states is provided in the proof of Theorem 1. We obtain

$$\begin{aligned} t_{\mathrm{HL}} &\geq (2t_K + \delta_1) + (2t_K + \delta_1 + \delta_2 + \varepsilon_1^*) \\ &\quad + (2t_K + \delta_1 + \delta_2 + \delta_3 + \varepsilon_1^* + \varepsilon_2^*) \qquad (3) \\ &\geq 6t_K. \end{aligned}$$

The estimation in (3) shows that a single hormone loop as specified in Section 3 takes $6t_K$ time units at least. This means the hormone loop duration is about twice the duration (at least) as in the reference literature.

Second, we try to estimate an upper limit for t_{HL}. We have to estimate the upper limit for the duration of the wait states ε_1, ε_2 and ε_3. They depend on the jitter parameters p_1, p_2 and p_3 in the absorption summands γ_1, γ_2 and γ_3. Each of the summands is of the form $\frac{p \cdot \mathrm{const.}}{1-p}$ which diverges if p tends towards 1. Therefore, there is no finite upper limit for the duration of t_{HL}[2].

The result of this section is that we can ensure a proper workflow of the hormone loop on the distributed PEs. This has numerous advantages: The consistency of the hormone communication is guaranteed even in presence of jitter, the fast recognition of PE or task failures and the consistent termination algorithm. However, we have to pay the costs in return: The hormone loop duration takes $6t_K$ time units at least and can take even longer depending on the strength of the jitter. Nonetheless, the approach in this paper is a major improvement regarding previous publications on the AHS as they neither do guarantee consistency of the hormones nor do they deal with jitter and other reaction times.

5 Related Work

Self-organization has been a research focus for several years. Publications like [16] or [26] deal with basic principles of self-organizing systems, like e.g. emergent behavior, reproduction etc. Regarding self-organization in computer science, several projects and initiatives can be listed.

IBM's Autonomic Computing project [13, 17] deals with self-organization of IT servers in networks. Several so called self-X properties like self-optimization, self-stabilization, self-configuration, self-protection and self-healing have been postulated. The MAPE cycle consisting of *M*onitor, *A*nalyze, *P*lan and *E*xecute was defined to realize these properties. This MAPE cycle is executed in the background and in parallel to normal server activities similar to the autonomic nervous system.

The German *Organic Computing* Initiative was founded in 2003. Its basic intention is to improve the controllability of complex embedded systems by using principles found in organic entities [24, 23]. Organization principles successful in biology are adapted to embedded computing systems. The DFG priority programme 1183 "Organic Computing" [9] has been established to deepen research on this topic.

Self-organizing and organic computing is also followed on an international level by a task force of the IEEE Computational Intelligence Society (IEEE CIS ETTC OCTF) [14]. Several other international research programs have also addressed self-organization aspects for computing systems, e.g. [10, 7].

So far, there are several approaches for clustered task allocation in middleware.

The authors of [3] present a scheduling algorithm for distributing tasks onto a grid. It is implemented in the

Xavantes Grid Middleware and arranges the tasks in groups. Their approch is completely different from ours because it uses central elements for the grouping: the Group Manager (GM), the Process Manager (PM) and the Activity Managers (AM). The GM is a single point of failure. If it fails, there is no possibility of obtaining group information from this group anymore. Our approach does not apply a central task distribution instance and therefore single point of failures are avoided.

Another approach is presented in [21]. The authors propose two algorithms for task scheduling. The first algorithm, Fast Critical Path (FCP), ensures that time constrains are kept. The second one, Fast Load Balancing (FLB), schedules the tasks equally on processors according to current loads. Different from our approach, task relationships are not regarded to allocate cooperating tasks closely together. Furthermore, these algorithms do not consider the failing of processing elements.

[22] presents a load balancing scheme for task allocation based on local workpiles (of processors) storing the tasks to be executed. The authors propose a load balancing algorithm which is applied to two processors to balance their workload. The algorithm is executed with a probability inversely proportional to the length of the workpile of a PE. Although this approach is distributed it does not consider aspects like self-healing or real-time constraints.

Other approaches of load balancing are presented in [1, 8, 12, 11, 27]. None of them cover the whole spectrum of self-X properties, task clustering, and real-time conditions like our approach.

A research project regarding self-organizing task allocation with respect to real-time properties is the CAR-SoC project [18, 19]. This project uses agent based principles and an auction mechanisms to achieve self-X features. This approach is not completely decentralized like the AHS, since an auction manager is responsible for a certain set of tasks.

The DoDOrg project [15] researches the use of bio-inspired principles to build a new, self-organizing robust processor architecture. Within this project, we invented a first version of the AHS to assign software tasks to distributed processing cells. In [20] we improved this first approach to distribute time dependent tasks in a distributed system.

6 Conclusion

We presented an approach to formally specify the workflow of the hormone loop of the artificial hormone system in this paper. The specification includes time limits for the actions to be executed and wait states to be passed. We proved seven invariants regarding important properties of the hormone loop guaranteeing its correct progress. They also ensure a strict timely isolation of the sending (and receiving) of suppressors and accelerators on the one hand and the sending (and receiving) of eager values on the other hand. This is important to ensure the correct computation of resulting hormone values. In addition, the specification allows to recognize the failing of a task or a PE in a fast way. We also provided an algorithm to terminate PEs in a consistent way. The specification also allows to deal with jitter in the action exection and in the waiting states. The cost of this advantages is a longer duration of the hormone loop depending on the strength of the jitter.

The future work will be two-fold: One the one hand we want to reduce the hormone loop duration by reducing the duration of the wait states. On the other hand we want to include additional features in the specification of the hormone loop such as launching additional PEs while other PEs are running the loop yet. This is a complex task because a just launched PE has to recognize if it is launched concurrently with all other PEs (it is a system start) or if there are other PEs just running yet (the PE is added to a running system). In addition, we plan to specify the hormone loop allowing to launch additional PEs concurrently to terminating running PEs.

References

[1] BECKER, W. Dynamische adaptive Lastbalancierung für große, heterogen konkurrierende Anwendungen. Dissertation, Universität Stuttgart, Fakultät Informatik, Dezember 1995.

[2] BETTING, B., PACHER, M., AND BRINKSCHULTE, U. Development and evaluation of a self-adaptive organic middleware for highly dependable system-on-chips. In *8th International Conference on Autonomic and Autonomous Systems (ICAS 2012)* (2012), St. Maarten, Netherlands Antilles.

[3] BITTENCOURT, L. F., MADEIRA, E. R. M., CICERRE, F. R. L., AND BUZATO, L. E. A path clustering heuristic for scheduling task graphs onto a grid. In *3rd International Workshop on Middleware for Grid Computing (MGC05)* (Grenoble, France, 2005).

[4] BRINKSCHULTE, U., PACHER, M., AND RENTELN, A. An artificial hormone system for self-organizing real-time task allocation in organic middleware. In *Organic Computing*, Understanding Complex Systems. Springer Berlin Heidelberg, 2008, pp. 261–283.

[5] BRINKSCHULTE, U., PACHER, M., RENTELN, A., AND BETTING, B. Organic real-time middleware. In *Self-Organization in Embedded Real-Time Systems*, M. T. Higuera-Toledano, U. Brinkschulte, and A. Rettberg, Eds. Springer New York, 2013, pp. 179–208.

[6] BRINKSCHULTE, U., PACHER, M., AND VON RENTELN, A. Towards an artificial hormone system for self-organizing real-time task allocation. *5th IFIP Workshop on Software Technologies for Future Embedded & Ubiquitous Systems (SEUS)* (2007), 339–347.

[7] CSIRO. Centre for Complex Systems.

[8] DECKER, T., DIEKMANN, R., LÜLING, R., AND MONIEN, B. Universelles dynamisches task-mapping. In *Konferenzband des PARS'95 Workshops in Stuttgart, PARS-Mitteilung 14* (1995), pp. 122–131.

[9] DFG SCHWERPUNKTPROGRAMM 1183. Organic Computing.

[10] EU. Program Future Emerging Technolgies FET - Complex systems.

[11] FINKE, J., PASSINO, K., AND SPARKS, A. Stable task load balancing strategies for cooperative control of networked autonomous air vehicles. In *IEEE Transactions on Control Systems Technology* (2006), vol. 14, pp. 789– 803.

[12] FINKE, J., PASSINO, K. M., AND SPARKS, A. Cooperative control via task load balancing for networked uninhabited autonomous vehicles. In *42nd IEEE Conference onDecision and Control, 2003. Proceedings* (2003), vol. 1, pp. 31 – 36.

[13] IBM. Autonomic Computing.

[14] IEEE. Organic Computing Task Force.

[15] J.BECKER, K.BRÄNDLE, BRINKSCHULTE, U., HENKEL, J., KARL, W., KÖSTER, T., WENZ, M., AND WÖRN, H. Digital On-Demand Computing Organism for Real-Time Systems. In *Workshop on Parallel Systems and Algorithms (PASA), ARCS 2006* (Frankfurt, Germany, March 2006).

[16] JETSCHKE, G. *Mathematik der Selbstorganisation*. Harry Deutsch Verlag, Frankfurt, 1989.

[17] KEPHART, J. O., AND CHESS, D. M. The Vision of Autonomic Computing. *IEEE Computer* (Jan. 2003), 41–50.

[18] KLUGE, F., MISCHE, J., UHRIG, S., AND UNGERER, T. CAR-SoC - Towards and Autonomic SoC Node. In *Second International Summer School on Advanced Computer Architecture and Compilation for Embedded Systems (ACACES 2006)* (L'Aquila, Italy, July 2006).

[19] NICKSCHAS, M., AND BRINKSCHULTE, U. Guiding Organic Management in a Service-Oriented Real-Time Middleware Architecture. In *6th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS 2008)* (Capri, Italy, October 2008).

[20] PACHER, M., AND BRINKSCHULTE, U. Real-Time Distribution of Time-Dependant Tasks in Heterogeneous Environments. *First IEEE Workshop on Self-Organizing Real-Time Systems (SORT 2010)* (2010).

[21] RADULESCU, A., AND VAN GEMUND, A. J. C. Fast and effective task scheduling in heterogeneous systems. In *IEEE Computer - 9th Heterogeneous Computing Workshop* (Cancun, Mexico, 2000).

[22] RUDOLPH, L., SLIVKIN-ALLALOUF, M., AND UPFAL, E. A simple load balancing scheme for task allocation in parallel machines. In *ACM Symposium on Parallel Algorithms and Architectures* (1991), pp. 237–245.

[23] SCHMECK, H. Organic Computing - A New Vision for Distributed Embedded Systems. In *8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2005)* (Seattle, USA, May 2005), pp. 201–203.

[24] VDE/ITG (HRSG.). VDE/ITG/GI-Positionspapier Organic Computing: Computer und Systemarchitektur im Jahr 2010. *GI, ITG, VDE* (2003).

[25] VON RENTELN, A., AND BRINKSCHULTE, U. Reliablity of an Artificial Hormone System with Self-X Properties. In *Parallel and Distributed Computing and Systems* (Cambridge, Massachusetts, USA, November 19 - 21 2007).

[26] WHITAKER, R. Self-Organization, Autopoiesis, and Enterprises.

[27] XU, C., AND LAU, F. Decentralized remapping of data parallel computations with the generalized dimension exchange method. In *Proceedings of Scalable High-Performance Computing Conference* (1994), pp. 414 – 421.

Notes

[1]Note that the AHS described in Section 2 only guarantees coherency: It ensures the receiving of either suppressors and accelerators or the receiving of eager values. The AHS avoids false multiple task assignements in this way, however, sub-optimal decisions are possible. Our new approach in this paper also ensures optimal decisions.

[2]Obviously, this is only a technical restriction: In a real-world scenario, we assume that we can estimate the parameters p_i. Therefore, we can compute a finite upper limit for t_{HL} as a result.