Bio-inspired Error Detection for Complex Systems

Martin Drozda[1], Iain Bate[2], Jon Timmis[23]
[1]*Faculty of Electrical Engineering and Computer Science, Leibniz University of Hannover, Hannover, Germany*
[2]*Department of Computer Science, University of York, York, YO10 5DG, UK*
[3]*Department of Electronics, University of York, York, YO10 5DD, UK*
drozda@sim.uni-hannover.de, {iain.bate,jtimmis}@cs.york.ac.uk

***Abstract*—In a number of areas, for example, sensor networks and systems of systems, complex networks are being used as part of applications that have to be dependable and safe. A common feature of these networks is they operate in a de-centralised manner and are formed in an ad-hoc manner and are often based on individual nodes that were not originally developed specifically for the situation that they are to be used. In addition, the nodes and their environment will have different behaviours over time, and there will be little knowledge during development of how they will interact. A key challenge is therefore how to understand what behaviour is normal from that which is abnormal so that the abnormal behaviour can be detected, and be prevented from affecting other parts of the system where appropriate recovery can then be performed. In this paper we review the state of the art in bio-inspired approaches, discuss how they can be used for error detection as part of providing a safe dependable sensor network, and then provide and evaluate an efficient and effective approach to error detection.**

I. Introduction

Sensornets are a prime example of the complex systems and networks that are being deployed as part of today's dependable systems. Examples are being seen in healthcare and defence as well as many other domains [1]. Sensornets present unique challenges to developers as by their very nature they consist of self-organising components where the individual components feature highly complex often adaptive designs. When combined with uncertainties in their deployment and their operating environment we argue that that traditional approaches to error detection are not appropriate as the errors (their nature and causes) can only be fully understood at run-time and will be continuously varying. Here we define an error to be deviation from expected behaviour, i.e. errors, that may lead to a failure. This leads to the following key objectives being identified, and addressed for sensornets based on a variety of sources, which concur, including the following well-recognised surveys [2], [3].

1) **N1** - Errors need to be detectable from changes in the system itself caused by both internal and external effects, e.g. expected adaptions in protocols or expected changes in the external interference sources [4].
2) **N2** - Errors can occur permanently or temporary (over both short and long durations and frequencies) [5].
3) **N3** - An error is detected then the system should be able to re-establish an acceptable homeostatic state [6].
4) **N4** - Any approach proposed for handling of errors should be achievable in the context of a sensornet system, e.g. with highly-constrained resources [7].

Take for instance a traditional threshold-based test that uses the number of messages that fail to be delivered as a trigger. Without detailed knowledge of exactly how and where the systems are to be deployed it is near impossible to identify a threshold that is not overly conservative or aggressive in terms of its detection. Even if one could be identified, its value might not be suitable over a long period of time. For example as new interference sources are introduced the failed messages may increase and yet no improvement in the way communications are performed is possible. In fact tactics such as turning off nodes suspected of being the source of errors might make things worse.

Other approaches, e.g. error detection using Markov analysis [8] and neural networks [9] would suffer from similar issues. The obstacle facing these approaches is the identification and maintenance of knowledge of what is considered normal behaviour. The common feature of these approaches is they are tuned off-line before the actual deployment of the sensornet and the parameters remain unchanged.

An alternative option is an adaptive learning system where over time a model of the system's behaviour is learnt, e.g. using statistical relational learning [10], or adapted, e.g. using feedback control approaches [11]. Again these approaches have advantages and disadvantages. The advantages include that less precise knowledge is needed before deployment of the sensornet, although some basic knowledge may still be needed, e.g. so that appropriate control parameters can be chosen. The disadvantage of such an approach is the lack of assurance that new errors will be adequately learnt by the system and thus the performance of the system be maintained. This suggests there is a need for an appropriate compromise between statically (innate) pre-defined tactics working in conjunction with a more adaptive system.

The classical approaches, discussed above, to error detection have been applied to WSN with limited success for the reasons that have been raised. For more details refer to [3]. Therefore in this paper a well-established area of work, Artificial Immune Systems (AIS), is introduced as an alternative strategy for dealing with the likely errors. Then in section III a survey of the work on AIS, including their application to sensornets, is provided that gives a strong justification as to why AIS is a suitable technology for use in sensornets that feature as part of a wider dependable system. Section IV provides experimental results which is then followed by the conclusions.

II. Artificial Immune Systems

The immune system is a complex system that undertakes a myriad of tasks. The abilities of the immune system have helped to inspire computer scientists to build systems that *mimic*, in some way, various properties of the immune system. AIS have been defined in [12] as:

> "adaptive systems, inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving."

This field of research, AIS, has seen the application of immune inspired algorithms to a wide range of problems [13]. From a computational point of view, the immune system has many desirable properties such as robustness, adaptability, diversity, scalability, multiple interactions on a variety of timescales and so on, and is thus attractive to many in an engineering context. This has been demonstrated for instance by work on Automated Teller Machines (ATM) [14] where AIS has been used to provide prognostic capabilities that have helped to increase the availability of the machines.

A recent paper [13] highlights that to date, the development of AIS has been *scattergun* i.e. many applications have been tried without a great deal of thought. Indeed, that paper provides a detailed overview of the many application areas that AIS have tried, and this will not be repeated here: the interested reader should consult that paper. The authors go on to propose a number of properties that they feel any AIS should have, and that these properties may help guide the type of application they could be applied to:

- "They will exhibit *homeostasis*
- They will benefit from interactions between *innate* and *adaptive immune models*
- *They will consist of* multiple, interacting, communicating *components*
- *Components can be easily and naturally* distributed
- *They will be required to perform* life-long learning*" [13]*

By homeostasis the authors mean a "steady state" type operation, so that when errors or changes in environmental conditions occur, systems can maintain a certain level of operation, i.e. an acceptable equilibrium is achieved. In the context of this paper, we argue that sensornets, given the above criteria, are a good candidate for the application of ideas from the area of AIS.

III. Immune Systems and their Relationship to Sensornets

From the previous sections, the needs of an error tolerance mechanism have been established and AIS have been introduced. Here we explore how the two are related to deliver a dependable system.

A. Immune Systems

We now briefly discuss several immune mechanisms that we propose can be interpreted and applied to error detection in wireless sensor networks (and similar distributed, resource constrained computing environments). The biological immune system (BIS) employs various mechanisms to combat threats to a host that can cause an immune response to trigger: the general term for something that can cause an immune response is known as a *pathogen* [15]. In general, we consider the BIS to consist of two parts: the innate and the adaptive immune system. This is a simplification as there is a great deal of interaction between both parts of the immune system, but the distinction is useful for our discussion.

The innate immune system incorporates general pathogen defence mechanisms that have evolved over the *germline* of the organism, i.e. these mechanisms are passed from the parents to the offspring. These mechanisms remain essentially unchanged during the lifetime of an individual. The mechanisms of the adaptive immune system also develop as the organism evolves, however they also have the ability to change *somatically* (i.e. during the lifetime of an individual through the production of new cells known as *lymphocytes* and *receptors* on these cells). This results in the ability of the adaptive immune system to recognise previously unseen pathogens (learning) and to remember them for future encounters (memory). The innate and adaptive immune systems typically operate over different timescales. The innate operates on a small time scale often initiating a reaction either instantly or within a matter of minutes, whilst the adaptive immune system operates over a longer time period, taking of the order of days to initiate a reaction. It is the combination and interaction of both the innate and adaptive immune mechanisms that provides us with an effective immune system [16].

With respect to the adaptive immune system, lymphocytes are of two general types: B and T cells. These cells have to go through a selection process to ensure a minimal amount of what is known as *self-reactivity* where cells of the host attack the host: this is also known as *autoimmunity*. The selection process in the immune system has two basic forms: positive selection and negative selection. Positive selection selects individual immune cells with high success rate when recognising a pathogen during an immune response. Successful cells are cloned and B cells undergo a process known as clonal expansion where new receptors are generated, or mutated in an attempt to improve recognition ability of the overall system [12]. T cells are also cloned, but do not undergo any mutation process. Negative selection selects T-cells that do not bind any self cell. This process is done before T-cells are allowed into the lymphatic system, and the assumption is if a cell passes this selection phase, it is only reactive with pathogen. Of course, this process is not perfect, and self-reactive cells do make it into the lymphatic system. However, negative selection process plays a key role in decreasing the possibility of an autoimmune reaction [15].

One of the most important events triggered by the activation of innate immunity is the expression of what is known as *co-stimulatory molecules* which help support the process of recognition in the adaptive immune system. This co-stimulation can form a feedback loop that leads to pathogen elimination [16]. Later, we will argue that splitting an error detection system into two functional units, one being responsible for error classification and the other one responsible for providing context on the ability of a

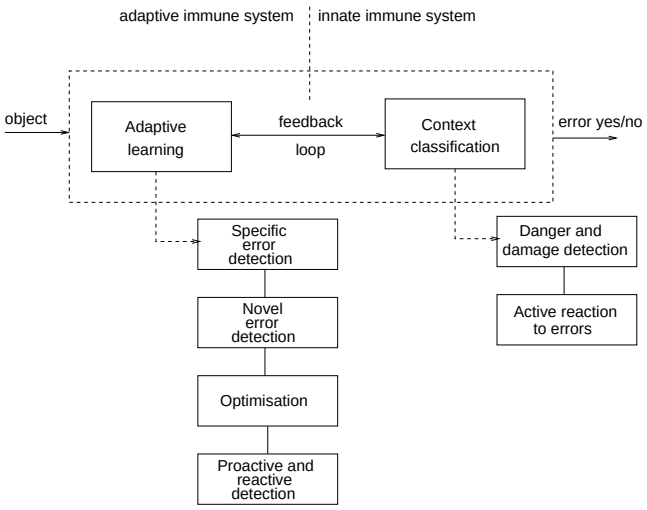


Figure 1. Immune inspired error detection - Architecture overview

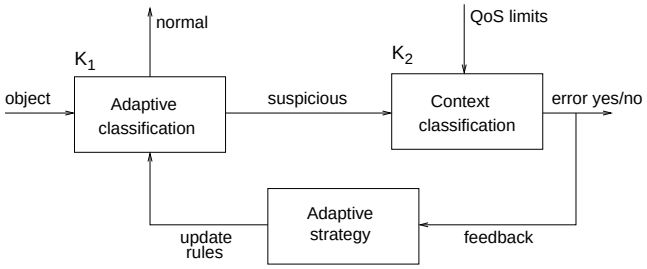


Figure 2. Immune inspired error detection

given error to cause damage is essential for introducing many desirable properties such as novel error detection, false positives control, reactive and proactive error detection or adaptivity to changes in external environment.

B. Immune Inspired Architecture

As we have discussed above, the BIS is a biological protection system that specialises in recognition and elimination of a range of threats to a host (pathogens). For the purposes of this paper, we focus on architectural properties of the BIS and their interpretation in the scope of error detection, rather than offering a computational interpretation of each process within the BIS. We assume that these individual processes can be well interpreted applying standard machine learning and statistical approaches, as well as other bio-inspired approaches. Our immune inspired error detection architecture should reflect the following properties of the BIS:

- Adaptivity: in response to novel error as well as to any modification of known error.
- Context evaluation: any detected error is evaluated with respect to whether it can impact quality of service.
- Feedback loop: context evaluation provides a feedback on the success rate of adaptivity.

Figure 1 provides an overview of our proposed immune inspired architecture. It consists of two modules, each of them having a different focus. The purpose of the adaptive learning module is to reflect the capabilities of adaptive immune system: selection, memory and learning, whereas the purpose of context classification module is to reflect the capabilities of innate immune system: responses to non-specific threats. The context classification module is capable of signalling whether any error detected by the adaptive learning module, in our study, could result in decreased quality of service. This module thus provides the necessary feedback after any adaptation occurs within the adaptive learning module.

We will now discuss the interpretation of this architecture shown in Figure 2. We assume that a set of Quality of Service (QoS) limits is provided. Exceeding these limits implies that an error negatively impacts the monitored sensor network. We will later explain how Adaptive classification and Context classifications can be implemented. For the ease of the discussion we assume that these two types of classification can be implemented as classifiers K_1 and K_2, respectively.

C. Problem Formulation

Let $\Omega = \{o_1, ..., o_n\}$ be a finite set of objects, where n is the number of these objects (representing the behaviour of nodes or sensors). Each classifier $K_i \in \{K_1, K_2\}$ takes an object $o_h \in \Omega$ and assigns it to a class C_j. Let $\epsilon(K_i)$ be the classification error resulting from applying K_i to Ω. Classification error is defined as $\epsilon(K_i) = \frac{p}{n}$, where p is the number of objects incorrectly classified. Let $\xi(K_i)$ be the cost of such a classification process.

Let $K_1 \leftrightarrow K_2$ be a classifier resulting when K_1 and K_2 are combined according to Figure 2. Let $\epsilon(K_1 \leftrightarrow K_2)$ and $\xi(K_1 \leftrightarrow K_2)$ be the classification error and cost of this classifier, respectively. The objectives of this architecture with respect to error detection can be formulated as follows:

$$minimise : \epsilon(K_1 \leftrightarrow K_2), \quad \xi(K_1 \leftrightarrow K_2) \tag{1}$$

subject to:

$$\epsilon(K_1) > \epsilon(K_2), \quad \xi(K_1) < \xi(K_2) \tag{2}$$

Eq. 1 defines our immune inspired error detection approach as a multiobjective optimisation problem. Note that if either of the following holds: (i) $\epsilon(K_1) \leq \epsilon(K_2)$ and $\xi(K_1) < \xi(K_2)$ or (ii) $\epsilon(K_1) > \epsilon(K_2)$ and $\xi(K_1) \geq \xi(K_2)$ then the inequalities formulated in Eq. 2 are not satisfied. This would imply that there is a classifier that has both lower classification error as well as lower cost. In other words, only classification applying either K_1 or K_2 would be necessary.

To achieve the objectives formulated in Eq. 1, our goal is to apply an adaptive strategy such that:

$$\lim_{t \to \infty} \epsilon(K_1) - \epsilon(K_2) = \delta \tag{3}$$

where t is time and δ is a fixed adaptivity error. Since K_2 reflects the detection capabilities of innate immunity with a negligible adaptation rate, $\epsilon(K_2)$ can be considered fixed. This form of adaptivity thus implies that applying an adaptive strategy results in K_1 with classification error converging to that of K_2. The adaptivity error can either

be directly provided by the user or it can be a function of cost, i.e. $\delta = g(\xi(K_1 \leftrightarrow K_2))$, where $g(.)$ is monotonically decreasing.

In the following we show how our classifier reflects requirements such as responsiveness, good split between true positives and true negatives, ability to detect novel types of error and energy efficiency.

D. Energy Cost Model

When evaluating energy efficiency of $K_1 \leftrightarrow K_2$, we assume that any sensor will work reliably, most of the time. The rationale is that error occurrence is unpredictable and therefore it is necessary to analyse the running cost of the dominant case, i.e. the running cost of error detection in an error free sensor network.

Inspecting the architecture shown in Figure 2, it can be seen that when classifying an object, two cases can occur:

1) K_1 is applied: prediction is "normal" behaviour.
2) K_1 is applied: prediction is "suspicious". Then K_2 is applied. If prediction of K_2 is an "error" then the object in consideration is classified as erroneous. Subsequently, a new adaptive strategy can be computed and applied.

The application of the later classification stage K_2 is conditional upon the outcome of K_1 classification. In an error free sensor network, the energy cost model can be formulated as follows:

$$\xi(K_1 \leftrightarrow K_2) = \xi(K_1) + \epsilon_0(K_1) \cdot (\xi(K_2) + \xi_{adapt}) \quad (4)$$

where $\epsilon_0(K_1)$ is the rate at which normal behaviour is mistakenly predicted to be an error and ξ_{adapt} is the cost related to computing and applying a new adaptive strategy. Notice that since $\xi(K_1) < \xi(K_2)$, the rate $\epsilon_0(K_1)$ controls the frequency at which the costlier K_2 is being applied.

E. False Positives Control

Let us now discuss the effects of our architecture on false positives control. Considering a class $c \in \{norm, mis\}$, let $det.\ rate^{\Omega}_{c}(K_i)$ be the detection rate and let $FP\ rate^{\Omega}_{c}(K_i)$ be the false positives rate resulting when classifier K_i is applied to classify the objects Ω. These two measures are defined as follows:

$$det.\ rate^{\Omega}_{c}(K_i) = \frac{t_c}{n_c} \qquad FP\ rate^{\Omega}_{c}(K_i) = \frac{\neg t_c}{\neg t_c + t_c} \quad (5)$$

where n_c is the number of objects in Ω belonging to the class c, t_c is the number of objects correctly predicted to belong to c and $\neg t_c$ is the number of objects incorrectly predicted to belong to c.

Let us first assume that $\epsilon(K_2) = 0$. With respect to the error class *mis*, after each classification stage the following holds:

1) K_1 is applied to Ω: $det.\ rate^{\Omega}_{mis}(K_1) = a$ and $FP\ rate^{\Omega}_{mis}(K_1) = b$
2) K_2 is applied to $\Omega' = \Omega - \Omega_{K_1}$: $det.\ rate^{\Omega'}_{mis}(K_1 \leftrightarrow K_2) = a$ and $FP\ rate^{\Omega'}_{mis}(K_1 \leftrightarrow K_2) = 0$

where $a, b \in R,\ 0 \leq a, b \leq 1$ and Ω_{K_1} are the objects that were predicted to represent normal behaviour in the first stage. The final FP rate equals 0 due to the fact that after K_2 with $\epsilon(K_2) = 0$ is applied, all misclassified objects in Ω' are removed, i.e. $\forall c\ (\neg t_c = 0)$.

Let us now consider the case when $\epsilon(K_2) > 0$. If $\forall o_h \in \Omega_{K_1}$ ($o_h \in norm$) and $\epsilon^{\Omega_{K_1}}(K_2) = 0$ then the following holds:

$$FP\ rate^{\Omega' + \Omega_{K_1}}_{mis}(K_2) = FP\ rate^{\Omega'}_{mis}(K_2) \quad (6)$$

Since $\Omega = \Omega' + \Omega_{K_1}$, then:

$$FP\ rate^{\Omega}_{mis}(K_2) = FP\ rate^{\Omega'}_{mis}(K_2) \quad (7)$$

And finally, for the final false positives rate and the final detection rate, it holds that:

$$FP\ rate^{\Omega}_{mis}(K_1 \leftrightarrow K_2) = FP\ rate^{\Omega}_{mis}(K_2) \quad (8)$$

$$det.\ rate^{\Omega}_{mis}(K_1 \leftrightarrow K_2) \leq det.\ rate^{\Omega}_{mis}(K_1) \quad (9)$$

Applying K_2 with $\epsilon(K_2) > 0$ may decrease the final detection rate, since K_2 can misclassify objects that were correctly predicted by K_1 to belong to the class *mis*.

If $\exists o_h \in \Omega_{K_1}$ ($o_h \in mis$) and o_h can be correctly classified by K_2 then in order to keep the final FP rate unchanged, the false positives rate of K_2 has to be adjusted as follows:

$$FP\ rate^{\Omega'}_{mis}(K_2) = \frac{\triangle \neg t_{mis}}{\triangle \neg t_{mis} + \alpha \cdot t_{mis}} \quad (10)$$

Solving this equation for $\triangle \neg t_{mis}$ yields:

$$\triangle \neg t_{mis} = \frac{\alpha \cdot t_{mis} \cdot FP\ rate^{\Omega'}_{mis}(K_2)}{1 - FP\ rate^{\Omega'}_{mis}(K_2)} \quad (11)$$

where $\alpha \in R,\ 0 \leq \alpha \leq 1$ and $\alpha \cdot t_{mis}$ is the decreased number of objects correctly classified as error, i.e. α reflects the fact that some objects that have errors could not enter the second phase of classification. $(t_{mis} - \triangle \neg t_{mis})$ is the adjustment that is necessary in order to keep the outcome of Eq. 10 constant. A special case happens when $FP\ rate^{\Omega'}_{mis}(K_2) = 0$, in which case $FP\ rate^{\Omega}_{mis}(K_1 \leftrightarrow K_2)$ is independent from the classification capability of K_1.

With respect to the above said, one option for achieving false positives control is that the outcome of K_1 classification is $\Omega' \subset \Omega$ such that $FP\ rate^{\Omega'}_{mis}(K_2) = 0$. This implies that Ω_{K_1} contains all objects representing normal behaviour that K_2 would misclassify. Whether K_1 can deliver such an effect is one of the goals of our experimental analysis.

Additionally, in order to stimulate energy efficiency K_1 must remove from Ω a large number of objects representing normal behaviour, i.e. $|\Omega'| \ll |\Omega|$. Let us now discuss the implications of the above discussed form of false positives control on the design of adaptive error detection systems.

F. Adaptivity

Taking Eqs. 8 and 11 into consideration it can be seen that the final false positives rate is independent of $FP\ rate^{\Omega}_{mis}(K_1)$. Given that K_1 should exhibit a large degree of adaptivity, this fact simplifies the way to a suitable adaptive strategy. Let us now formulate the implications of this fact:

Adaptive strategy: since the final false positives rate is solely determined by K_2, the applied adaptive strategy is allowed to increase $FP\ rate^{\Omega}_{mis}(K_1)$. This increase however

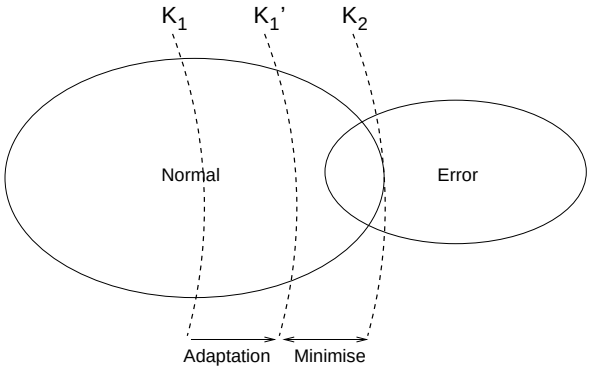


Figure 3. Immune inspired error detection - adaptation

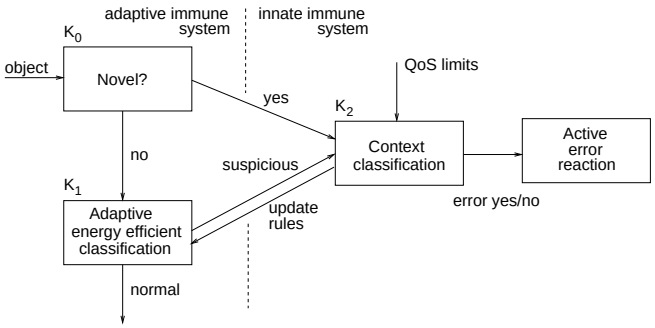


Figure 4. Immune inspired error detection - novel error detection

must stay time bounded, since any increase in $\neg t_{mis}^{K_1}$ negatively impacts the total cost as formulated in Eq. 4.

$\neg t_{mis}^{K_1}$ is the number of objects incorrectly classified by K_1 as belonging to class *mis*. Note that $\epsilon_0(K_1)$ can be expressed as:

$$\epsilon_0(K_1) = \frac{\neg t_{mis}^{K_1}}{n_{norm}} \tag{12}$$

In order to apply the above formulated adaptive strategy we are limited by the fact that only the objects in Ω' and their class prediction by K_2 can be used as a feedback for updating the decision rules of K_1.

The effects of this form of adaptation are illustrated in Figure 3. Since we assume that K_2 has a negligible adaptation rate, any adaptation is done by improving K_1 classification, more formally:

$$minimise : |\Omega'| \tag{13}$$

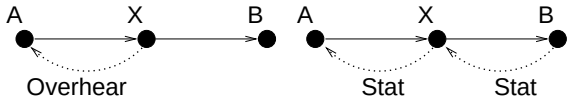
subject to:

$$FP\ rate_{norm}^{\Omega}(K_1) \leq \gamma \tag{14}$$

where γ is a cost parameter, i.e. a well bounded fraction of objects that have errors should end up in Ω_{K_1}. Note that $|\Omega'| = \neg t_{mis}^{K_1} + t_{mis}^{K_1}$. On Figure 3, $|\Omega'|$ is the area to the right of K_1 (or K_1' after adaptation).

G. Feature Sets

A challenge when implementing our immune inspired architecture shown in Figure 2 is to design two distinct feature sets so that the requirements formulated in Eq. 2 can be fulfilled. A *feature* in the context of sensor networks is understood to be a performance measure that allows for an efficient reasoning about whether a node (sensor) has errors or works normally. Let us demonstrate how this can be done on a simple example where the goal is to detect whether a node X timely forwards data packets; see Figure 5.



(a) f_2: Overhearing (b) f_1: **Stat** sent over 2 hops

Figure 5. Feature sets f_1, f_2: An example

Let us contrast two approaches that can be applied for this task: (a) node A can overhear the data traffic between X and B and thus detect whether data packet forwarding is not being delayed by X or (b) both node A and node B compute inter-arrival delay statistic; the statistic computed by B is sent over two-hops to A and then compared with the statistic computed by A. Let us denote feature sets computed through overhearing and over two hops as f_2 and f_1, respectively. Observe that if X does not forward **Stat**, then this node is classified as having errors, since there is no inter-arrival information received by A within a predefined time limit which we denote *win. size*.

Let us suppose there is a mapping Φ that maps feature set f_2 to f_1:

$$\Phi : f_2^s \mapsto f_1^d \tag{15}$$

where s and d, $s \leq d$, is the dimensionality of feature set f_2 and f_1, respectively. Note that for the features and delaying errors shown in Figure 5, $s = 1$ and $d = 2$. In the case of f_1, $d = 2$ since two features (that may have an identical definition) are computed, each by a different node. In general, $f_1^d = f_1(A) \cup f_1(B)$, where $f_1(A)$ and $f_1(B)$ is the f_1 feature set applied by A and B, respectively. **Stat** is a numerical instance of $f_1(B)$.

Let $K_i(f_i)$ be classifier K_i applying features f_i to reason about errors. Since we aim at fulfilling the requirements formulated in Eq. 2, we are looking for a feature space transformation such that:

Definition *(γ_1, γ_2)-transformation* is a mapping $\Phi : f_2^s \mapsto f_1^d$ such that the following holds:

$$\epsilon(K_1(\Phi(f_2))) - \epsilon(K_2(f_2)) = \gamma_1 \tag{16}$$

$$\xi(K_2(f_2)) - \xi(K_1(\Phi(f_2))) = \gamma_2 \tag{17}$$

where $\gamma_1, \gamma_2 \in R^+$.

A (γ_1, γ_2)-transformation, when provided a feature set f_2, returns f_1 that induces a higher classification error and a lower cost. Notice that $\xi(K_1(f_1))$ as well as $\epsilon(K_1(f_1))$ are a function of *win. size*. This is due to the fact that

the frequency at which a statistic originating at node B is received by A influences the cost (**Stat** has to be sent more often) as well as the classification error assumed that the statistic is averaged over *win. size*. Notice that for a given Φ to check whether Eqs. 16 and 17 are fulfilled may require an experimental analysis.

Assuming $\xi(K_1(f_1))$ and $\xi(K_2(f_2))$ are dominated by communication costs, these two costs can be expressed as follows:

$$\xi(K_1(f_1)) = 2 \cdot (\xi_{TX}(\textbf{Stat}) + \xi_{RX}(\textbf{Stat})) \quad (18)$$

$$\xi(K_2(f_2)) = k \cdot \xi_{RX}(\textbf{Data}) \quad (19)$$

where $\xi_{TX}(\textbf{Stat})$ and $\xi_{RX}(\textbf{Stat})$ is the communication cost of sending and receiving **Stat**, respectively, and $\xi_{RX}(\textbf{Data})$ is the cost of overhearing a data packet **Data**. The fixed constant 2 in Eq. 18 reflects the fact that **Stat** must hop twice to reach A. We assume that k data packets need to be overheard in order to detect an erroneous node.

The experimental results in [17], [18] show that considering data packet dropping and data packet delaying errors, it is possible to construct a (γ_1, γ_2)-transformation that fulfils the conditions stated in Eqs. 16 and 17. Those results are based on energy consumption models of several wireless devices including TI CC2420.

Mapping a data set into a higher dimensional space has been successfully applied in several other areas, most notably, in databases where such a mapping is used to facilitate nearest neighbour searches [19]. In our case, a (γ_1, γ_2)-transformation is a tool for finding a feature set f_1 with more favourable energy cost and classification error trade-off than f_2. Unlike in [19], such a transformation is limited by the fact that each feature in f_1 must be observable, i.e. it can be directly computed taking as input states and events observable locally by any node.

As mentioned before, applying thresholds can lead to either overly conservative or aggressive detection pattern. To circumvent this problem, we apply mapping Φ to project any threshold based features (and the related QoS limits) into a feature space of a higher dimension. This adds the necessary flexibility and results in a detection system that can recognise a larger set of error types, however, only errors that violate any QoS limit would trigger a reaction. Observe that projecting threshold based features applying Φ also allows the detection system to become more specific, i.e. error types that would violate a single predefined threshold can possibly match different decision rules of K_1.

Even though, we demonstrated that for a certain type of error, finding a suitable Φ might be straightforward, for other error types it can prove challenging. This means that the success of our approach is closely tied with our ability to find Φ for any expected error and this has to fulfil the requirements of Eqs. 16 and 17 to become a useful feature set transformation.

H. Adaptivity Revisited

Let $\Omega_i \subset \Omega$ such that for arbitrary i, j: $\Omega_i \neq \Omega_j$ and $\Omega = \bigcup_i \Omega_i$. Let $K_i(f_j)[o_h]$ be the *label* of $o_h \in \Omega_i$ as predicted by $K_i(f_j)$. Let $\Omega'_i \subseteq \Omega_i$ be a set of objects such that $\forall o_h \in \Omega'_i$ $(K_1(f_1)[o_h] = error)$. At last, let $\Gamma_i = \Gamma_{i-1} \cup (o_h, K_2(f_2)[o_h])$, i.e. Γ_i is the set of objects that were subject to K_2 classification including their class membership as predicted by K_2. Γ_i is a basis of feedback from K_2 to K_1.

Since K_1 and K_2 work over different feature spaces and any K_1 adaptation can only benefit from Γ_i, Eq. 3 becomes:

$$\lim_{i \to \infty} \epsilon^{\Omega}(K_1^{\Gamma_i}(f_1)) - \epsilon^{\Omega}(K_2(f_2)) = \delta \quad (20)$$

where $K_1^{\Gamma_i}(f_1)$ denotes classifier K_1 over feature space f_1 applying Γ_i for adaptation. In practical terms, we would like to see that $\epsilon^{\Gamma_i}(K_1(f_1))$ converge to $\epsilon^{\Omega}(K_2(f_2))$ within a reasonable number of steps, i.e. we would like to see $i < c; c \in Z^+$. Convergence results are going to be another goal of our experimental analysis. Note that classifier K_2 is based on user provided QoS limits. Since they do not change once defined by the user, we omit a superscript in the case of K_2.

I. Novel Error Detection

To understand how to detect novel error detection, let us consider the architecture shown in Figure 4. In this architecture any object is first classified by $K_0(f_1)$, i.e. this classifier applies the same feature set as K_1. The only task of this classifier is to decide whether this object is novel, i.e. whether this or a similar object has not been seen in the past. The classification then proceeds as follows:

1) If the object is novel, then it is classified by $K_2(f_2)$.
2) If the object is not novel then cascading classification by $K_1(f_1)$ and $K_2(f_2)$ is done. In this case, $K_1(f_1)$ is expected to have a rule capable of class prediction with respect to this object.

K_0 takes inspiration from the role of negative selection in the BIS. The purpose of negative selection is to produce immune cells capable of detecting any non-self cells, i.e. cells that are not building blocks of the host. Since K_1 can only classify objects already seen in the past, any novel object is directly sent to K_2. Algorithm 1 is a formal description of the architecture shown in Figure 4.

J. Summary

We discussed how several basic properties can be achieved applying an immune inspired architecture. Let us now summarise our findings:

- Energy efficiency is determined by the capability of K_1 to classify normal behaviour and in doing so objective **N4** is achieved.
- Final false positives rate is determined by K_2.
- Adaptivity is a matter of feedback between K_1 and K_2, where any adaptation is only based on $\Omega' \subset \Omega$. Any adaptive strategy applied to K_1 is allowed to increase false positives rate of this classifier at the cost of increased energy cost. The adaptivity allows objectives **N1** and **N2** to be achieved.
- Novel error detection is done by evaluating whether any user defined QoS limits are violated. The accumulative result of such an evaluation Γ_i is used to adapt K_1. As such objective **N3** can be achieved as the system can then be tailored so that all parts meet the QoS target.

Algorithm 1 Immune inspired error detection.

Require: Set of objects $\Omega_i \subset \Omega$

```
1: procedure DETECT_ERROR
2:     if i == 1 then Γ_i ← ∅
3:     else
4:         Γ_i ← Γ_{i−1}
5:     end if
6:     for h := 1 → |Ω_i| do
7:         suspicious ← false
8:         if K_0(f_1)[o_h] == novel then suspicious ← true
9:         else
10:            if K_1(f_1)[o_h] == error then suspicious ← true
11:            end if
12:        end if
13:        if suspicious == true then
14:            Γ_i ← Γ_i ∪ (o_h, K_2(f_2)[o_h])
15:            if K_2(f_2)[o_h] == error then error_detected
16:            end if
17:        end if
18:    end for
19:    COMPUTE_K_1 (Γ_i)    ▷ Classifier K_1 computation using e.g. a decision tree algorithm
20: end procedure
```

IV. Experimental Analysis

In this section we show preliminary experimental results on the feasibility of Algorithm 1.

A. Experimental Setup

The experimental setup is similar to that used in [18]. We consider a network with 1718 nodes. The network topology is based on a snapshot from the movement prescribed by the Random waypoint movement model [20]. The physical area size was 3,000m × 3,000m.

We modelled data traffic as Constant bit rate (CBR), i.e. there was a constant delay when injecting data packets. This constant delay in our experiments was 2 seconds (injection rate of 0.5 packet/s); the packet size was 68 bytes. CBR data packet sources correspond to e.g. sensors that transmit their measurements in predefined constant intervals.

In our simulations we used 50 concurrent data connections. The connection length was 7 hops. In order to represent a dynamically changing system, we allowed connections to expire. An expired connection was replaced by another connection starting at a new random source node. Each connection was scheduled to exist 15 to 20 minutes. The exact connection duration was computed as $\tau + r_U\lambda$, where τ is the minimum duration time of a connection, r_U a random number from the uniform distribution $[0, 1]$ and λ the desired variance of the connection duration. In our experiments, we used $\tau = 15$ *min* and $\lambda = 5$ *min*.

We used the JiST/SWANS network simulator [21] with the AODV routing protocol and the default settings provided by JiST/SWANS. We used the IEEE 802.11 MAC protocol. The RTS-CTS-DATA-ACK handshake was enabled for all data communication. The channel frequency was set to 2.4 GHz. The transmission rate was set to 54 Mbit/s. We used the two-ray signal propagation model [22]. Antenna and signal propagation properties were set so that the resulting radio radius equals 100 meters.

K_1 classification was done using a decision tree classifier. To decide whether a node within the decision tree should be further split (impurity measure), we used the information gain measure. As the decision tree classifier is a well-known algorithm, we omit its discussion. We refer the interested reader to [23]. We used the decision tree implementation from the Rapidminer tool [24].

For the purpose of our experimental evaluation we consider *two error types*: data packet dropping (qualitative error) and data packet delaying (quantitative error). For the data packet dropping, the erroneous node drops a given data packet randomly and uniformly with probability 0.1. For the data packet delaying, the erroneous node delays the forwarding of a given data packet randomly and uniformly with probability 0.3 by a fixed delay amount $20ms$. For each error type and normal behaviour, there were 20 simulation runs with a different simulation seed done. The simulation time was 4 hours.

There were 236 nodes randomly chosen to execute data packet dropping or delaying erroneous. Our intention was to model *random error occurrences*, assuming an uniform error distribution in the network. As it is hard to predict the routing of data packets, many of the 236 nodes could not originate errors as there were no data packets to be forwarded by them. In our case, about 20-30 erroneous nodes were concurrently active.

In order to simplify our experiments, we only considered 20 nodes with the highest amount of data traffic. For each of the 20 nodes we split its vector set into two sets: training and test. A node's vector set comprises numerical instances of f_1 and f_2. The size of training set and test set is 1500 and 500 vectors, respectively. Both sets are obtained by random stratified sampling. Training set constitutes the input Ω of our immune inspired architecture, whereas test set Ω^T is used to monitor progress after each round i. Training set Ω is further split into Ω_i that constitutes the input at each round.

The applied features are listed in Appendix. The features belonging to f_1 are more complex than in the example shown in Fig. 5. The reason is that our simulation setup allows for multiple connections running over each node. Note that the duration of monitoring when applying f_2 can be, depending on data traffic pattern or expected number and severity of errors, shorter than *win. size*. For simplicity, in our simulations we assume that this monitoring period is equal to *win. size*.

Assuming that f_1 and f_2 computation costs are dominated by communications costs, these feature sets fulfil the requirement of a (γ_1, γ_2)-transformation for any *win. size* where the cost of transmitting **Stat** over two hops becomes cheaper than computation of watchdog features f_2. Since *win. size* is a sampling period for f_1, $\epsilon^{\Omega}(K_1(f_1))$ increases as *win. size* increases; see [17]. It is straightforward to couple the energy

model introduced in Eq. 4 with energy consumption model of a wireless device in order to get a quantitative estimate; see [18].

The average class distribution for the top 20 nodes was as follows: i) *win. size* = 100*s* resulted in 29.01% vectors to represent data packet dropping or data packet delaying and ii) *win. size* = 50*s* resulted in 28.94% vectors to represent data packet dropping or data packet delaying.

QoS limits that we applied are $q_1 = 0.5\%$ for data packet dropping and $q_2 = 2.6ms$ for data packet delaying. If observed data dropping or data delaying exceeds during the monitoring period q_1 or q_2, respectively, then K_2 assigns class "error" to the classified object. QoS limits q_1, q_2 can be in general set to any value (being physically possible), however, since q_1, q_2 partition the set of objects, it is necessary to choose q_1, q_2 so that none of the partitions is empty. Should the choice of q_1, q_2 result in a single non-empty partition, K_2 will assign each incoming object the same class.

B. Experiments

Experiment 1: $|\Omega_1| = 5$, $|\Omega_i| = 25, 2 \leq i \leq 30$. Each Ω_i is a result of stratified sampling applied to Ω. The experiment was done applying *win. size* set to either 50s or 100s. The rationale of this experiment was to show the performance of $K_1 \leftrightarrow K_2$ upon start up. This experiment assumes that an error is present from the first moment.

Experiment 2: $|\Omega_1| = 150 + 2 + 2$ where 150 objects represent normal behaviour, 2 represent data dropping errors and 2 represent data delaying errors. $|\Omega_i| = 2, \quad 2 \leq i \leq 30$, where 1 objects represents data dropping errors and 1 objects represents data delaying errors. We contrast this with another sub-experiment with $|\Omega_1|$ set to 750+2+2. Objects are chosen randomly from Ω. *win. size* is set to 100s. The rationale of this experiment is to show the performance of $K_1 \leftrightarrow K_2$ after a large number of objects representing normal behaviour were part of the process, i.e. in this case the network was running normally and after a certain time period an error emerged.

In order to streamline the experiments, we decided to apply a simplified model for deciding whether an object is novel. We mark all objects in Ω_1 as novel, i.e. they are all sent directly to K_2. The objects belonging to $\Omega_i, i > 1$ are assumed to be not novel therefore being sent to K_1. This was necessary in order to investigate the effects of adaptivity in isolation.

Performance evaluation of $K_1 \leftrightarrow K_2$ was done with respect to δ, $|\Gamma_i|$, detection rate and FP rate including partial results for K_1 and K_2. Additionally, we consider $FP(K_2, K_1 \leftrightarrow K_2) = FP\ rate^{\Omega^T}_{mis}(K_2) - FP\ rate^{\Omega^T}_{mis}(K_1 \leftrightarrow K_2)$. For each of these measures, we compute 95% confidence intervals.

C. Experimental Results

The experimental results for Experiment 1 are reported in Table I. It can be seen that after 30 rounds, $|\Gamma_{30}|$ compared to $\sum_i |\Omega_i|$ is 25.97% and 26.40% for *win. size* set to 100s and 50s, respectively. $\epsilon_0^{\Omega^T}(K_1)$ decreased for *win. size* = 100*s* from 47.67% to 4.50% and *win. size* = 50*s* from 36.76% to 3.49%. This means that K_1 could adapt to the two types of error that we consider and only a small fraction of objects representing normal behaviour is incorrectly misclassified. This also implies that we achieved our adaptivity objective to decrease the size of Ω'; see Eqs. 13 and 14. This coincides with increasing $det.\ rate^{\Omega^T}_{mis}(K_1)$ as adaptivity positively impacts classification performance of K_1.

i	1	5	10	20	30
$\sum_i \lvert\Omega_i\rvert$	5	105	230	480	730
win. size = 100*s*					
$\lvert\Gamma_i\rvert$	5.00±0.00	37.45±0.61	68.10±2.08	126.10±4.06	189.55±4.68
δ	44.36±0.06	15.98±0.59	14.02±0.38	11.60±0.11	10.22±0.35
$det.\ rate^{\Omega^T}_{mis}(K_1)$	46.34±0.01	75.58±0.44	87.57±0.67	87.75±0.05	89.66±0.24
$det.\ rate^{\Omega^T}_{mis}(K_2)$	99.94±0.06	99.94±0.06	99.94±0.06	99.94±0.06	99.94±0.06
$det.\ rate^{\Omega^T}_{mis}(K_1 \leftrightarrow K_2)$	44.62±0.28	73.83±0.42	86.07±0.65	86.02±0.12	88.08±0.32
$FP\ rate^{\Omega^T}_{mis}(K_1)$	62.07±2.08	23.01±2.54	18.01±1.98	11.51±1.27	11.64±1.70
$FP\ rate^{\Omega^T}_{mis}(K_2)$	3.53±0.54	3.53±0.54	3.53±0.54	3.53±0.54	3.53±0.54
$FP(K_2, K_1 \leftrightarrow K_2)$	2.57±0.00	2.37±0.13	2.01±0.17	2.92±0.07	2.55±0.35
$\epsilon_0^{\Omega^T}(K_1)$	47.67±0.51	8.94±0.99	6.14±0.68	4.50±0.50	4.50±0.13
win. size = 50*s*					
$\lvert\Gamma_i\rvert$	5.00±0.00	36.70±0.85	67.80±0.97	130.20±0.46	192.75±0.63
δ	37.89±1.25	15.33±0.57	11.72±0.09	9.57±0.22	8.86±0.30
$det.\ rate^{\Omega^T}_{mis}(K_1)$	43.08±0.50	86.06±1.22	84.72±0.36	90.40±0.83	92.24±0.40
$det.\ rate^{\Omega^T}_{mis}(K_2)$	99.02±0.04	99.02±0.04	99.02±0.04	99.02±0.04	99.02±0.04
$det.\ rate^{\Omega^T}_{mis}(K_1 \leftrightarrow K_2)$	41.14±0.45	83.10±1.33	81.86±0.47	88.40±0.82	90.45±0.37
$FP\ rate^{\Omega^T}_{mis}(K_1)$	56.35±2.90	20.94±1.75	14.52±1.21	9.95±0.58	8.88±0.46
$FP\ rate^{\Omega^T}_{mis}(K_2)$	4.01±0.21	4.01±0.21	4.01±0.21	4.01±0.21	4.01±0.21
$FP(K_2, K_1 \leftrightarrow K_2)$	2.03±0.30	2.48±0.04	2.59±0.00	3.11±0.12	2.64±0.07
$\epsilon_0^{\Omega^T}(K_1)$	36.76±2.35	8.59±0.64	6.43±0.52	3.84±0.16	3.49±0.12

Table I
EXPERIMENT 1: PERFORMANCE OF $K_1 \leftrightarrow K_2$.

Observe that $FP\ rate^{\Omega^T}_{mis}(K_1)$ remains high after all 30 rounds are completed. This is inline with our adaptive strategy introduced in Section III-F, where we state that $FP\ rate^{\Omega^T}_{mis}(K_1)$ does not impact $FP\ rate^{\Omega^T}_{mis}(K_1 \leftrightarrow K_2)$. Therefore, other than energy efficiency there is no need to optimise for this performance measure. $det.\ rate^{\Omega^T}_{mis}(K_1 \leftrightarrow K_2)$ is lower than $det.\ rate^{\Omega^T}_{mis}(K_1)$. This is inline with Eq. 9.

The results for $FP(K_2, K_1 \leftrightarrow K_2)$ indicate that $FP\ rate^{\Omega^T}_{mis}(K_1 \leftrightarrow K_2)$ is lower than $FP\ rate^{\Omega^T}_{mis}(K_2)$. This decrease is statistically significant and suggests that $FP\ rate^{\Omega}_{mis}(K_2) > FP\ rate^{\Omega'}_{mis}(K_2)$, i.e. the objects in Ω' are simpler to classify than the objects in Ω. As discussed in Section III-E, this means that a portion of objects representing normal behaviour that K_2 would misclassify ended up in Ω_{K_1}.

At last notice that δ which benchmarks the adaptive process in terms of classification error, decreased by 34.14% and 29.03% for *win. size* set to 100s and 50s, respectively. The impact of *win. size* on δ is statistically significant. This is expected as *win. size* is effectively the sampling period for f_1.

The experimental results for Experiment 2 are reported in Table II. The results are similar to the results reported for Experiment 1. It can be however seen that for the sub-experiment with $|\Omega_1| = 750 + 2 + 2$, $\epsilon_0^{\Omega^T}(K_1)$ increases from 2.10% to 2.54%, which can be in absolute terms considered a very low misclassification rate. Comparing $\epsilon_0^{\Omega^T}(K_1)$ for

i	1	5	10	20	30
$\|\Omega_1\| = 150 + 2 + 2$					
$\sum_i \|\Omega_i\|$	154	162	172	192	212
$\|\Gamma_i\|$	154.00±0.00	159.00±0.22	165.05±0.89	177.40±1.26	187.55±1.05
δ	20.89±0.23	16.15±0.14	15.33±0.47	13.77±0.40	12.23±0.28
$det.\ rate_{mis}^{\Omega^T}(K_1)$	91.78±0.42	92.52±0.64	89.23±0.28	90.18±1.96	88.21±2.12
$det.\ rate_{mis}^{\Omega^T}(K_2)$	99.84±0.11	99.84±0.11	99.84±0.11	99.84±0.11	99.84±0.11
$det.\ rate_{mis}^{\Omega^T}(K_1 \leftrightarrow K_2)$	89.53±1.05	89.88±1.09	86.76±0.74	87.82±2.20	86.08±2.26
$FP\ rate_{mis}^{\Omega^T}(K_1)$	18.60±0.29	13.82±0.16	12.51±0.90	13.29±1.04	13.42±0.61
$FP\ rate_{mis}^{\Omega^T}(K_2)$	3.60±0.09	3.60±0.09	3.60±0.09	3.60±0.09	3.60±0.09
$FP(K_2, K_1 \leftrightarrow K_2)$	2.46±0.32	2.64±0.42	2.86±0.20	2.67±0.20	2.68±0.20
$\epsilon_0^{\Omega^T}(K_1)$	2.35±0.16	2.78±0.15	3.23±0.48	4.13±0.69	4.67±0.46
$\|\Omega_1\| = 750 + 2 + 2$					
$\sum_i \|\Omega_i\|$	754	762	772	792	812
$\|\Gamma_i\|$	754.00±0.00	759.50±0.06	765.70±0.30	777.35±0.59	788.60±0.73
δ	20.74±1.13	17.66±1.49	15.26±1.76	13.63±0.99	12.58±0.53
$det.\ rate_{mis}^{\Omega^T}(K_1)$	97.70±0.25	97.10±0.33	96.90±0.20	96.40±0.24	96.94±0.08
$det.\ rate_{mis}^{\Omega^T}(K_2)$	99.85±0.02	99.85±0.02	99.85±0.02	99.85±0.02	99.85±0.02
$det.\ rate_{mis}^{\Omega^T}(K_1 \leftrightarrow K_2)$	94.85±0.33	94.39±1.00	94.51±0.05	94.19±0.32	94.82±0.05
$FP\ rate_{mis}^{\Omega^T}(K_1)$	16.46±2.22	11.43±3.05	10.34±3.17	8.43±1.22	8.58±0.57
$FP\ rate_{mis}^{\Omega^T}(K_2)$	3.60±0.09	3.60±0.09	3.60±0.09	3.60±0.09	3.60±0.09
$FP(K_2, K_1 \leftrightarrow K_2)$	1.10±0.53	2.14±0.08	2.35±0.10	2.16±0.22	2.22±0.25
$\epsilon_0^{\Omega^T}(K_1)$	2.10±0.29	2.11±0.40	2.32±0.37	2.32±0.34	2.54±0.24

Table II
EXPERIMENT 2: PERFORMANCE OF $K_1 \leftrightarrow K_2$.

the two sub-experiments, it can be seen that the presence of normal behaviour helps improve energy efficiency of $K_1 \leftrightarrow K_2$.

V. CONCLUSION

We introduced and evaluated an adaptive approach to error detection. Our architecture reflects basic components of the biological immune system. We argued that our architecture can address several key challenges of error detection systems: adaptivity, false positives control and energy efficiency. Our approach to adaptivity is based on a feedback loop between two classifiers, one representing the capability of innate immune system whereas the other one representing the capability of adaptive immune system. In order to achieve flexibility, we project any applied threshold features onto another feature set having a higher dimension.

We did experimental performance analysis comprising two distinct scenarios. The goal of the first scenario was to estimate performance of our adaptive detection approach when errors occur at network start-up, whereas the goal of the other scenario was to estimate performance when an error occurs after the network was running error free over a longer time period. The common goal of these two scenarios was to show whether our adaptive strategy can be exploited to improve the classification capability of our detection architecture without sacrificing false positives control and with a limited impact on energy efficiency.

The advantage of our approach is that it maps the various objectives onto different modules of our architecture. This makes it easier to tune in isolation with respect to error detection rate, false positives rate, energy efficiency and adaptivity rate.

ACKNOWLEDGEMENTS

Martin Drozda was supported by the German Research Foundation (DFG) under the grant no. SZ 51/24-3 (Survivable Ad Hoc Networks – SANE).

REFERENCES

[1] J. Stankovic, I. Lee, A. Mok, and R. Rajkumar, "Opportunities and obligations for physical computing systems," *IEEE Computer*, pp. 23–31, Nov 2005.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Survey*, vol. 41, no. 3, pp. 1–58, 2009.

[3] S. Rajasegarar, C. Leckie, and M. Palaniswami, "Anomaly detection in wireless sensor networks," *IEEE Wireless Communications*, vol. 15, no. 4, pp. 34–40, 2008.

[4] P. Vicaire, T. He, Q. Cao, T. Yan, G. Zhou, L. Gu, L. Luo, R. Stoleru, J. Stankovic, and T. Abdelzaher, "Achieving long-term surveillance in VigilNet," *ACM Transactions on Sensor Networks*, vol. 5, no. 1, pp. 1–39, 2009.

[5] H. Liu, J. Li, Z. Xie, D. Siu, S. Lin, K. Whitehouse, and J. Stankovic, "Automatic and robust breadcrumb system deployment for indoor firefighter applications," in *Proceedings of the Conference on Mobile Systems, Applications, and Services (MobiSys 2010)*, 2010, pp. 21–34.

[6] S. Lin, G. Zhou, K. Whitehouse, Y. Wu, J. Stankovic, and T. He, "Towards stable network performance in wireless sensor networks," in *Proceedings of the IEEE Real-Time Systems Symposium*, 2009, pp. 227–237.

[7] Y. Wu, K. Kapitanova, J. Li, J. Stankovic, S. Son, and K. Whitehouse, "Run time assurance of application-level requirements in wireless sensor networks," in *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN 2010)*, 2010, pp. 197–208.

[8] N. Ye, "A Markov Chain Model of Temporal Behavior for Anomaly Detection," in *Proceedings of the Workshop on Information Assurance and Security*, 2000, pp. 166–169.

[9] S. Han and S. Cho, "Evolutionary neural networks for anomaly detection based on the behavior of a program," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 36, no. 3, pp. 559–570, 2005.

[10] E. Eskin, "Anomaly detection over noisy data using learned probability distributions," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML '00)*, 2000, pp. 255–262.

[11] Y. Zhang and Y. Xue, "Study of immune control computing in immune detection algorithm for information security," in *Proceedings of the 4th International Conference on Intelligent Computing: Advanced Intelligent Computing Theories and Applications - with Aspects of Artificial Intelligence*, 2008, pp. 951–958.

[12] L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 2002.

[13] E. Hart and J. Timmis, "Application areas of AIS: The past, the present and the future," *Applied Soft Computing*, vol. 8, no. 1, pp. 191–201, 2008.

[14] R. de Lemos, J. Timmis, S. Forrest, and M. Ayara, "Immune-inspired adaptable error detection for automated teller machines," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 5, pp. 873–886, 2007.

[15] I. Roitt, *Roitt's Essential Immunology*, 10th ed. Oxford, UK: Blackwell Science, 2001.

[16] M. Neal and B. Trapnel, *In Silico Immuonology*. Springer, 2007, ch. Go Dutch: Exploit Interactions and Environments with Artificial Immune Systems, pp. 313–330.

[17] M. Drozda, S. Schildt, S. Schaust, and H. Szczerbicka, "An Immuno-Inspired Approach to Misbehavior Detection in Ad Hoc Wireless Networks," *Computing Research Repository (CoRR)*, 2010. [Online]. Available: http://arXiv.org/abs/1001.3113

[18] M. Drozda, S. Schaust, S. Schildt, and H. Szczerbicka, "Priming: Making the Reaction to Intrusion or Fault Predictable," *Natural Computing*, vol. 10, no. 1, pp. 243–274, 2011.

[19] C. Faloutsos and K. Lin, "Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1995, pp. 163–174.

[20] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, 1996, vol. 353, pp. 153–181.

[21] R. Barr, Z. Haas, and R. van Renesse, "JiST: an efficient approach to simulation using virtual machines," *Software Practice and Experience*, vol. 35, no. 6, pp. 539–576, 2005.

[22] T. Rappaport, *Wireless communications: principles and practice*. Prentice Hall, 2001.

[23] E. Alpaydin, *Introduction To Machine Learning*. MIT Press, 2004.

[24] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, "Yale: Rapid prototyping for complex data mining tasks," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 935–940.

Appendix

Definitions of features belonging to f_1 and f_2

Let $s_s, s_1, ..., s_i, s_{i+1}, s_{i+2}, ..., s_d$ be the path between s_s and s_d determined by a routing protocol, where s_s is the source node, s_d is the destination node. Let connection be defined as a tuple $cn = (s_s, s_d)$. Let $pcts_{TX}$ and $pcts_{RX}$ be the number of data packets sent and received by s_i in a time window of the size *win. size*, respectively.

f_1 **features:**

1) **Out-of-order packet index:** Number of DATA packets that were received by s_i out of order.

$$A_1 = \frac{\#OO}{win.\ size}$$

where *#OO* is the number of data packet received out-of-order. *#OO* is incremented, if node s_i receives on the connection *cn* a data packet p_j such that $seq.\ number(p_j) - 1 \neq seq.\ number(p_{j-1})$, where $seq.\ number(p_j)$ is the sequence number of the data packet p_j. This assumes that the connection source uses an incremental (or similar easily predictable) scheme for computing $seq.\ number(p_j)$. If more than 1 connection is running over s_i then average value for A_1 is computed.

2) **Interarrival packet delay index 1:** Average delay between data packets p_j and p_{j+1} sequentially received by s_i. The delay was computed separately *for each connection* and then a master average was computed.

$$A_2 = \frac{\sum_{cn=1}^{\#connect} avg\_delay_{cn}}{\#connect}$$

where avg_delay_{cn} is the average delay for data packets belonging to the connection *c* defined as:

$$\frac{\sum_{j=1}^{pcts_{RX}^{cn}} delay_{cn}(p_{j+1}, p_j)}{pcts_{RX}^{cn}}$$

where $pcts_{RX}^{cn}$ is the number of data packets received by s_i on the connection *cn*. $delay_{cn}(p_{j+1}, p_j)$ is the delay between the data packets p_{j+1} and p_j transported by the connection *cn*.

3) **Interarrival packet delay variance index 1:** Variance of delay between DATA packets received by s_i. The variance was computed separately for *each connection* and then a master average was computed.

$$A_3 = \frac{\sum_{cn=1}^{\#connect} avg\_var\_delay_{cn}}{\#connect}$$

where $avg\_var\_delay_{cn}$ is the variance of the delay for data packets belonging to the connection *cn*. It is defined as:

$$\frac{\sum_{j=1}^{pcts_{RX}^{cn}} (delay_{cn}(p_{j+1}, p_j) - avg\_delay_{cn})^2}{pcts_{RX}^{c} - 1}$$

4) **Interarrival packet delay index 2:** Average delay between DATA packets received by s_i.

$$A_4 = \frac{\sum_{j=1}^{pcts_{RX}} delay(p_{j+1}, p_j)}{pcts_{RX}}$$

where $delay(p_{j+1}, p_j)$ is the delay between any two data subsequent packets p_{j+1} and p_j received by s_i.

5) **Interarrival packet delay variance index 2:** Variance of delay between DATA packets received by s_i.

$$A_5 = \frac{\sum_{j=1}^{pcts_{RX}} (delay_(p_{j+1}, p_j) - A_4)^2}{pcts_{RX} - 1}$$

f_2 **features:**

6) **Forwarding index (watchdog):** Ratio of data packets sent from s_i to s_{i+1}, TX_{s_i} and then subsequently forwarded to s_{i+2}, $TX_{s_{i+1}}$.

$$B_1 = \frac{TX_{s_{i+1}}}{TX_{s_i}}$$

The data packets that have s_{i+1} as the destination node are excluded from this statistic.

7) **Processing delay:** Time delay that a data packet accumulates at s_{i+1} before being forwarded to s_{i+2}.

$$B_2 = \frac{\sum_{p=1}^{pcts_{TX}} delay_{s_{i+1}}}{pcts_{TX}}$$

The data packets that have s_{i+1} as the destination node are excluded from this statistic.