Specification and Validation of Enhanced Mobile Agent-Enabled Anomaly Detection and Verification in Resource Constrained Networks

Muhammad Usman, Vallipuram Muthukkumarasamy, and Xin-Wen Wu
School of Information and Communication Technology, Griffith University, Gold Coast, QLD 4222, Australia
email: muhammad.usman3@griffithuni.edu.au, v.muthu@griffith.edu.au, x.wu@griffith.edu.au

Abstract—Existing mobile agent-enabled anomaly detection schemes have not considered temporal behavior for their correct functioning and detection of temporal anomalies. This study employs a holistic system approach to design an Enhanced mobile Agent-enabled Anomaly Detection System (EAADS) by designing two new algorithms. The proposed algorithms are not only important for the completeness of the EAADS, but also for the detection of the anomalies caused by the delayed arrival of the in situ verification results. The formal specifications of the individual algorithmic functionalities are addressed by employing the Petri net theory. A bottom-up synthesis of the individual Petri net modules is carried out to formulate a unified model, which has helped in the identification and removal of inconsistencies in the system design. This process formally characterizes the behavioral properties and the overall workflow of the EAADS. The standard unified Petri net model is then extended into a corresponding high class Generalized Stochastic Petri Net (GSPN) model to formalize and analyze the temporal behavior of the EAADS in a highly nondeterministic communication environment. Finally, the GSPN-based temporal behavior is validated through implementation of the functional specifications on a real testbed composed of the resource constrained MICAz motes. The theoretical and experimental results demonstrate the ability of the EAADS to detect certain types of anomalies and the aptness of the temporal behavior of the EAADS for the low resource sensor networks even in a highly nondeterministic communication environment.

Index Terms—Anomaly detection; Generalized stochastic Petri net; In situ verification; Mobile agent; Petri net theory; Wireless sensor networks

I. INTRODUCTION

The advancements in microsensing technologies and architectural designs have opened up a new horizon for the research community to design innovative systems to cater for the ever growing needs of ubiquitous commercial, emergency, and military settings [1]–[5]. Tiny sensor nodes serve as an essential infrastructure, on top of which innovative applications are built for these settings. Software mobile agents offer several benefits such as code and data dissemination, localization, and parallelism to these distributed sensing networks [6]. Over the years, the research community has employed mobile agents for different purposes such as inter-base station control communication and randomly sampling the data over the network in the anomaly detection applications for Wireless Sensor Networks (WSNs) [7]–[9]. Previous works investigated the role of the mobile agents in performing the task of the in situ verification of the nodes in order to identify the source of the anomalies [10], [11]. The previous studies, however, have not analyzed the temporal behavior of the mobile agent-enabled anomaly detection schemes, which is an important factor for their effective post-deployment functionality [7]–[11].

An effective design and verification of the completeness and correctness of the mobile agent-based systems are desired before their deployment [12], [13]. A poorly designed or non-validated system may take a mobile agent-based system into the halt state, or it can adversely impact on the limited resources of sensor nodes. Formal methods are used to address the formal specifications and verification of complex systems [14]. The mathematical foundation of the Petri net theory gives strong assurance on the verification of the specifications of the systems [15]. Regardless of the existence of a few related schemes in the literature of the formally verified security protocols for WSNs by employing the Petri net theory [16]–[19], no study has employed the Petri net theory to address the modeling and verification problem of mobile agent anomaly detection system. To our knowledge, this is the first study which has extended the Petri net theory to modeling and analysis of an agent-enabled anomaly detection system. This study has also employed the Generalized Stochastic Petri Net (GSPN) to formally verify the temporal behavior of the agent-enabled anomaly detection system.

This study extends the initial design of previously proposed mobile Agent-enabled Anomaly Detection System (AADS) [10], [11] into the Enhanced mobile
Agent-Enabled Anomaly Detection System (EAADS). The key contributions, by the design and analysis of the EAADS, in this paper are as follows.

- Two algorithms, namely, status update on the cluster heads and base station are proposed. These algorithms are not only important for the completeness of the anomaly detection system in order to handle the in situ verification results, but also for the detection of anomalies caused by the delayed arrival of the in situ verification results.
- The Petri nets-based formal specifications of the individual algorithmic functionalities of the EAADS are addressed. A bottom-up synthesis of the individual net modules is then carried out to formulate a unified formal model which characterizes the behavioral properties such as the boundedness and liveness, and also the overall workflow of the EAADS. This holistic system approach assisted us in the identification of the missing specifications in the initial design of the system. After an integration of the missing specifications into the system design, the EAADS not only can detect temporal anomalies, but also allows sensor nodes to go into the sleep mode after performing their designated tasks in order to save energy.
- The standard Petri net model is extended into a corresponding high class GSPN model to formalize and analyze the temporal behavior of the EAADS in the highly nondeterministic communication environment of the WSNs. The GSPN-based temporal behavior of the EAADS is then validated through the experimentation on a real testbed composed of the resource constrained sensor nodes, namely, MicaZ. Five different cases are experimentally examined to thoroughly analyze the temporal behavior of the EAADS in different scenarios and also for different applications. The results indicate the suitability of the temporal behavior of the EAADS even when the communication environment of WSNs is highly nondeterministic.

Note that this study has adopted a complementary approach for the evaluation of the EAADS [20]. That is, the anomaly detection process is formally verified through the standard Petri nets and the temporal behavior is formalized through the GSPN and validated through implementation on the real motes.

This paper is organized as follows. Section 2 elaborates on the definitions and terminologies, which are drawn upon in this study. The network model and assumptions are highlighted in Section 3. Section 4 describes the functional specifications of the EAADS. The Petri net-based model formulation is carried out in Section 5. Section 6 characterizes and analyzes the formal model of the EAADS. The GSPN-based unified model is described in Section 7 and then validated in Section 8. Section 9 describes the comparative study. Finally, Section 10 concludes the paper.

II. DEFINITIONS AND TERMINOLOGIES

A few basic definitions and notations are elucidated in this section. Some key notions which are important to elaborate and analyze the work presented in this paper are also discussed.

Definition 1: (Petri net [21]) A Petri Net, PN, is a 5-tuple net: \( PN = (P, T, F, W, M_0) \), where \( P = \{p_1, p_2, ..., p_m\} \) is a non-empty and finite set of places, \( T = \{t_1, t_2, ..., t_n\} \) is a non-empty and finite set of transitions, such that \( P \cup T \neq \emptyset \) and \( P \cap T = \emptyset \). \( F \subseteq (P \times T) \cup (T \times P) \) is the set of arcs. \( W : F \to \{1, 2, ..., n\} \) denotes the weight function. \( M_0 : P \to N_0 \) is initial marking.

Definition 2: (Generalized Stochastic Petri net [21]) A Generalized Stochastic Petri Net (GSPN) is an 8-tuple net: \( GSPN = (P, T, I(\cdot), I^-(\cdot), O^+(\cdot), H(\cdot), W(\cdot), M_0) \), where \( (P, T, I(\cdot), I^-(\cdot), O^+(\cdot), H(\cdot), M_0) \) is underlying confusion free PN with priority and inhibition functions denoted by \( I(\cdot) \) and \( H(\cdot) \), respectively. The \( I^-(\cdot) \) and \( O^+(\cdot) \) are input and output functions. Similarly, \( W(\cdot) : T \to R \) denote rate or weight for timed and immediate transitions, respectively.

Definition 3: (Reachability [14]) The set of reachable markings from the initial state, \( M_0 \), is denoted by \( RM(M_0) \) if \( M_j \in RM(M_0) \) and \( M_j[t_i > M'_j] \) for some \( t_i \in T \), then \( M'_j \in RM(M_0) \). The marking \( M'_j \) is reachable from the initial state \( M_0 \) if there exists a firing sequence that takes \( M_0 \) to \( M'_j \).

Definition 4: (Safety and Boundedness [14]) For any \( p_i \in P \) of PN, the PN is safe if \( M_j(p_i) \leq 1 \) and \( k \)-bounded if \( \forall M_j \in RM(M_0) : M_j(p_i) \leq k \).

Definition 5: (Liveness [23]) The transition \( t_i \) is live if and only if there is a marking \( M'_j \) which is reachable from every \( M_j \in RM(M_0) \) such that \( M'_j \) enables \( t_i \). The Petri net, PN, is live if for all \( t_i \in T : t_i \) is live. Transition \( t_j \) has the following five levels of liveness.

- Level 0: If \( t_j \) can never be fired.
- Level 1: If \( t_j \) is potentially fireable, i.e., if there exists \( M_j \in RM(M_0) \) such that \( t_j \) is enabled in \( M_j \).
- Level 2: If for every integer \( n \geq 1 \) there is a firing sequence in which \( t_j \) occurs at least \( n \) times.
- Level 3: If \( \exists \) infinite firing sequence in which \( t_j \) occurs an infinitely number of times.
- Level 4: If for each \( M_j \in RM(M_0) \) there is a firing sequence \( \sigma \) such that \( t_j \) is enabled in \( \eta(M_j, \sigma) \).

Definition 6: (Deadlock [23]) The transition \( t_i \in T \) is a dead transition at marking \( M_j \in RM(M_0) \) if there is no marking reachable to enable transition \( t_i \). The
marking $M_j \in RM(M_0)$ is deadlocked if $\forall t_i \in T, t_i$ is dead. The PN model is deadlock free if there is no deadlock.

III. Network Model

The WSN is assumed to be a hierarchically clustered network. The $r_{th}$ cluster $C_r$ is composed of an $s$ number of the cluster member sensor nodes, $msn$, at leaf level, i.e., $C_r = \{msn_i | 1 \leq i \leq s\}$. Let $msn_q$ denote the node in question from the $C_r$ cluster of the network and the notation $MSN$ represents all of the $msn$ in the network. The resource rich nodes of the clusters are called the cluster heads, CHs, which are in charge of the nodes in their respective regional territories. A $u$ number of cluster heads are connected with the base station, BS. The BS is an overall controlling entity of the network which stores the application data. The overall network functionality has the following characteristics.

- The communication among entities in the network, i.e., MSN, CH, and BS, is nondeterministic because of factors such as environmental influence and channel errors [27].
- The CH nodes are aware of the resource status (such as battery and memory) of the respective MSN nodes through the Coordinated Resource Management, CRM, mechanism. The TinyOS, an operating system for the sensor nodes, facilitates the CRM mechanism through low level interfaces to manage and share the resource status of the nodes [24].
- The mobile agent, $MA$, has the embedded watermark in its code. Thus, it is secure from in transit and in situ attacks [25].
- To efficiently utilize the energy budget of the network, the inter-node itinerary of the $MA$ is kept between the two nodes, i.e., only between the CHs and their MSNs.
- The leaf level nodes, i.e., all MSNs of the network are vulnerable to the in situ and in transit faults, errors, and attacks. The CHs and BS are assumed as secure in this study due to the fact that they can employ more sophisticated security and fault tolerance measures because of their resource rich property.
- At the time of the system deployment, each MSN has a bootstrap trust value, $TR$, which varies within the closed interval $[0,1]$ with the passage of the time according to the behavior of the sensor node.

IV. THE ENHANCED AGENT-ENABLED ANOMALY DETECTION SYSTEM

The reliability of the WSN applications is greatly dependent on the accuracy of the received data. The data transmitted by the sensor nodes are, however, susceptible to the in situ and in transit faults and attacks. Therefore, an effective anomaly detection system must be able to identify the source of the anomalies in addition to their detection before initiating appropriate action against the anomalous sensor node. The proposed system is not only capable of detecting anomalies in the received CRM-based observations, but also uses the previously received values for the in situ verification of the suspicious sensor nodes by employing a mobile agent.

This section first presents three algorithms, namely, features collection, anomaly detection, and in situ verification, with their updated functional specifications. Two new algorithms, namely, the status update on the CH and BS are also then presented.

A. Features collection by sensor node

At $t$ time, the cluster member sensor node in question, $msn_q$, wakes up and performs a battery check. If its current battery level, $B_i^l$, is greater than that of the predefined battery threshold level, $B_t^l$, then it collects the values for the $v$ number of features, $F_q$. The $v = 3$ in this case and $F_0 = \{SR, MS, BS\}$, where $SR$, $MS$, and $BS$ denote sensor reading, memory status, and battery status, respectively. After collection of the values of the $F_q$, the $msn_q$ stores them in its reserved stack memory segment, $S_{msn_q}$, for the process of the in situ verification. These values are then transmitted to the corresponding cluster head, $ch_q$, for further processing. After finishing its designated task, the $msn_q$ again goes into sleep mode. This process is repeated after every $t$ time as per the specifications of the application. The high level pseudocode for this process is given in Algorithm 1.

B. Anomaly detection by cluster head

The cluster head, $ch_q$, receives the data packet containing the values of the features of the normal profile, i.e., $F_q = \{SR, MS, BS\}$, from the $msn_q$ within the allocated timeslot, i.e., $T_i^{ch} \leq T_{ar}(F_q) := T_{ar}^{msn} \leq T_i^{ub}$, where $T_i^{ch}$ denotes the start time of the timeslot, $T_i^{ub}$ represents the finish time of the timeslot, and $T_{ar}(F_q)$ is the function which computes the time of arrival,

Algorithm 1 Features collection by the $msn_q$

<table>
<thead>
<tr>
<th>Input: $B_i^l$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: $F_q$</td>
</tr>
<tr>
<td>1: At $t$ time</td>
</tr>
<tr>
<td>2: if $B_i^l \geq B_t^l$ then</td>
</tr>
<tr>
<td>3: while $i \leq j$ do</td>
</tr>
<tr>
<td>4: $Clet \left( F_q[i] \right)$</td>
</tr>
<tr>
<td>5: $S_{msn_q}[i] \leftarrow F_q[i]$</td>
</tr>
<tr>
<td>6: $F_q \leftarrow F_q + F_q[i]$</td>
</tr>
<tr>
<td>7: $i \leftarrow i + 1$</td>
</tr>
<tr>
<td>8: end while</td>
</tr>
<tr>
<td>9: $TRNSMT F_q$ to $ch_q$</td>
</tr>
<tr>
<td>10: go to sleep mode</td>
</tr>
<tr>
<td>11: else</td>
</tr>
<tr>
<td>12: $msn_q$ fails to wake up</td>
</tr>
<tr>
<td>13: end if</td>
</tr>
</tbody>
</table>
Figure 1: An example scenario of the temporal anomaly

The IEEE 802.15.4 standard divides the communication period into two modes, namely, Contention Access Period (CAP) and Contention Free Period (CFP) [31]. In the CFP mode, the msn, the MAR, acquires the guaranteed timeslot to communicate with the ch, the MA. The anomaly detection algorithm employs the CFP mode for receiving the Fq. The msn, is treated as anomalous if the Fq is received outside the allocated timeslot. A typical example scenario is depicted in Figure 1, where the ch, has received an Fq outside the allocated timeslot. Thus, in this case the msn should be considered as anomalous.

In the case if the chq does not receive the Fq from the msn within the allocated timeslot, then it transmits the mobile agent, MA, to the msn, to perform the in situ verification of its anomalous behavior. The MA carries the previous values of the Fq to verify whether the msn, is anomalous or if an anomaly has taken place during the transmission of the Fq between the msn and ch. In this case, the chq also decrements the trust count of the msn, by a $\varsigma$ factor (i.e., a user defined value) and transmits an alarm, $d^{al}_{aq}$, to the BS node. In the case when the chq receives the Fq within the allocated timeslot, it performs the process of the anomaly detection by comparing the received values of the Fq with those of the corresponding normal profile bounds, $Prf_q = \{ (SR^{lb}, SR^{ub}) , (MS^{lb}, MS^{ub}) , (BS^{lb}, BS^{ub}) \}$, in order to detect the anomalies. If the comparison is true, then the sensor reading, SR, is aggregated by the aggregation unit, Aqnt, for a specified period of time before its transmission to the BS node. On the other hand, if the chq detects an anomalous value, then it triggers the MA for the in situ verification of the anomalous behavior of the msn,; transmits an alarm, $d^{al}_{aq}$, to the BS node, and also decrements the trust count by the $\varsigma$ factor. The high level pseudocode for the anomaly detection process is expressed in Algorithm 2.

C. Anomalous node verification

The cluster member sensor node, msn, receives the anomaly agent, MA, and pushes the data brought by the MA into the stack memory, S_msn, a portion of the memory dedicated for the in situ verification process. The MA performs the in situ verification process by performing a match between the values of the SR, MS, and BS in the stack memories of the node (i.e., S_msn) and the mobile agent (i.e., S_agnt). If the values are matched, then the result is 0, i.e., confirmation of the normal status of the msn, is stored. In the case of any value not being matched, then result 1, indicating that the msn, is under the in situ attack or fault, is stored. The result is then transmitted to the chq, Note that if the EAADS is deployed only for the anomalies caused by the attacks, then the MA will insert a watermark in the result to make it secure from manipulation attack. The watermark embedded result, WR, then will be transmitted to the chq. On the other hand, if the EAADS is also deployed for the anomalies caused by the attacks, then the MA will insert a watermark in the result to make it secure from manipulation attack. The watermark embedded result, WR, then will be transmitted to the chq. In this case, however, the computation and transmission costs of the in situ verification process will increase. The watermarking overhead is analyzed in Section 8. The high level pseudocode of this process is shown in Algorithm 3.

D. Status update on the cluster head

The in situ verification process may yield one of these three cases: (i) the mobile agent is successfully executed on the suspicious msn, and it brings back the in situ verification result to the chq, (ii) the suspicious msn, may defend execution of the mobile agent and no result is transmitted to the chq, and (iii) migration of the mobile agent to the suspicious msn, is susceptible to the mobile agent execution manipulation attack. In Case (i), the mobile agent

Algorithm 2 Anomaly detection by the chq

Input: $F_q$, $Prf_q$, $TR$
Output: $SR, d^{al}_{aq}, d^{sr}_{aq}, MA, TR$

1: if $T_a^{lb} \leq T_{ar}(F_q) = T_a^{ub}$ then //check $F_q$ is received within the allocated timeslot
2: $CHK(F_q, Prf_q)$ //perform anomaly detection
3: if $CHK = TRUE$ then
4: $A_{an} \leftarrow SR$ //store sensor reading to the aggregation unit
5: $TRNSMT d^{sr}_{aq}$ to BS //transmit aggregated data to the BS after $T_a^{ub}$
6: $end$ if
7: else
8: $TRNSMT d^{sr}_{aq}$ to BS //trigger anomaly alarm to the BS
9: $TRNSMT MA$ to $msn$ //transmit mobile agent to the msn
10: if $TR > 0$ then //check the trust level
11: Decr $TR$ by $\varsigma$ //decrement the trust value
12: $end$ if
13: $end$ if
14: else
15: $TRNSMT d^{sr}_{aq}$ to BS //trigger anomaly alarm to the BS
16: $TRNSMT MA$ to $msn$ //transmit mobile agent to the msn
17: if $TR > 0$ then //check the trust level
18: Decr $TR$ by $\varsigma$ //decrement the trust value
19: $end$ if
20: $end$ if
Algorithm 3 Verification of the anomalies on msn_i

Input: MA, values of SR, BS, MS stored in S_msn_i
Output: WR
1: msn_i receives MA //the msn_i receives mobile agent
2: PUSHPMA in S_agnt_i //mobile agent stores in designated memory space
3: for (k = 1, i = 1 to S_msn_i[i] = ε) do
4: CMP (S_msn_i[i], S_agnt_i[k]) //compare previously stored values of features with that are brought by mobile agent
5: if (CMP (S_msn_i[i], S_agnt_i[k])) = TRUE then
6: i ++, k ++ //store “1” result, indicating no anomalies
7: R ← 0 //store “0” result, indicating no anomalies
8: else
9: R ← 1 //store “1” result, indicating anomalies
10: end if
11: end for
12: WMAP(R) := WR //mobile agent inserts watermark in the result
13: TRNSMT WR to ch_q //transmit the watermarked result to the ch_q

Algorithm 4 Status update on the ch_q

Input: WR, TR
Output: TR, d_i^a
1: if T_i^b \leq T_{wr}(WR) := T_i^b then //check verification result received within the allocated timeslot
2: Rem(WR) := R //remove watermark from result
3: if R = 1 ∧ TR > 0 then //check verification result and trust values
4: Decr TR by ε //decrement the trust value
5: TRNSMT d_i^a to BS //transmit alarm
6: else
7: break
8: end if
9: else
10: if TR > 0 then
11: Decr TR by ε //decrement the trust value
12: TRNSMT d_i^a to BS //transmit alarm
13: end if
14: end if

Algorithm 5 Status update on the BS

Input: d_i^ag, d_i^al \lor d_i^af
Output: update A_rep, update A_data
1: if BS receive d_i^ag \lor d_i^al then //received packet is alarm
2: update A_rep //update the application repository
3: break
4: else if BS receive d_i^af then //received packet has aggregated data
5: update A_data //update the application data
6: else
7: break
8: end if

E. Status update on the base station

The BS node receives either the aggregated data, d_i^ag, or the anomaly alarm, d_i^al or d_i^af, from the ch_q. For the anomaly report, the BS updates the repository to notify the system administrator about the anomalous status of the msn_i. Otherwise, it stores the aggregated sensor reading in the application data repository for analysis and further use by the system administrator. The high level pseudocode of this procedure is expressed in Algorithm 5.

V. THE UNIFIED MODEL FORMULATION

The functional specifications of the EAADS, described in the previous section, are first transformed into corresponding Petri net modules and then bottom up synthesis is carried out to formulate a unified model to formally characterize and analyze its behavior. This is initiated through a fine-grain mapping of functional specifications into net modules. During this process, both the in situ and in transit anomalous states of the nodes are also taken into account. The workflow states, such as sensor node wake up, threshold level, and so on, are modeled through places, denoted by p. On the other hand, transitions, represented by t, are employed to model different actions, for example, battery level check, sensor node activation, and so forth.

The first functional specification expresses the process of the F_0 collection by the msn_i. The formal definition of this process is given below.

Net module 1: (Features collection by the msn_i)
The features collection net module, \( P(N_1) \), is a 5-tuple net: \( P(N_1) = (P_1, T_1, F_1, W_1, (M_0_1)) \), where \( P_1 = \{p_1, p_2, p_3, p_4\} \) and \( T_1 = \{t_1, t_2, t_3, t_4, t_5\} \) are non-empty, finite, and disjoint sets of places and transitions, respectively. \( F_1 = \{(p_1, t_1), (t_1, p_2), (p_2, t_2), (p_2, t_3), (p_2, t_4), (t_2, p_3), (t_3, p_3), (t_4, p_4), (p_3, t_5), (t_5, p_5)\} \).

\[ W_1(p_1, t_1) = W_1(t_1, p_2) = W_1(p_2, t_2) = W_1(t_2, p_3) = W_1(t_3, p_3) = W_1(t_4, p_3) = 1, \]

\[ W_1(p_3, t_5) = W_1(t_5, p_4) = 3, \quad \text{and} \quad (M_0_1) = p_1. \]

The formal definition of the anomaly detection process is given below.

**Net module 2:** (Anomaly detection by the \( ch_q \).) The anomaly detection net module, \( P(N_2) \), is a 5-tuple net: \( P(N_2) = (P_2, T_2, F_2, W_2, (M_0_2)) \), where \( P_2 = \{p_5, p_6, p_7, p_8, p_9, p_{10}\} \) and \( T_2 = \{t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}\} \) are non-empty, finite, and disjoint sets of places and transitions, respectively. \( F_2 = \{(p_5, t_6), (t_6, p_9), (p_6, t_7), (p_6, t_9), (t_7, p_7), (t_7, p_8), (t_8, p_7), (t_8, p_9), (t_9, p_7), (t_9, p_{10}), (p_9, t_{11}), (t_{10}, p_{10}), (t_{11}, p_{10}), (p_{10}, t_{13})\} \).

\[ W_2(p_5, t_6) = W_2(p_6, t_9) = W_2(t_9, p_10) = W_2(p_{10}, t_{13}) = 3, \]

\[ W_2(p_6, t_7) = W_2(p_6, t_9) = W_2(t_7, p_7) = W_2(t_7, p_8) = W_2(t_8, p_7) = W_2(t_9, p_7) = W_2(t_9, t_{10}) = W_2(t_{10}, p_9) = W_2(p_9, t_{12}) = 1, \quad \text{and} \quad (M_0_2) = 3p_5. \]

The in situ verification process performed on the \( msn_q \) is formally defined below.

**Net module 3:** (In situ verification of the \( msn_q \).) The in situ verification net module 3, \( P(N_3) \), is a 5-tuple net: \( P(N_3) = (P_3, T_3, F_3, W_3, (M_0_3)) \), where \( P_3 = \{p_{11}, p_{12}, p_{13}\} \) and \( T_3 = \{t_{14}, t_{15}, t_{16}, t_{17}, t_{18}\} \) are non-empty, finite, and disjoint sets of places and transitions, respectively. \( F_3 = \{(p_{11}, t_{14}), (t_{14}, p_{12}), (p_{12}, t_{15}), (p_{12}, t_{16}), (p_{12}, t_{17}), (t_{15}, p_{13}), (t_{16}, p_{13}), (t_{17}, p_{13}), (t_{18}, p_{13})\} \).

\[ W_3(p_{11}, t_{14}) = W_3(p_{12}, t_{15}) = W_3(p_{12}, t_{16}) = W_3(p_{12}, t_{17}) = W_3(t_{15}, p_{13}) = W_3(t_{16}, p_{13}) = W_3(t_{17}, p_{13}) = W_3(t_{18}, p_{13}) = 1, \quad \text{and} \quad (M_0_3) = 3p_{11}. \]

The fourth functional process specifies the \( msn_q \) status update on the \( ch_q \) after the in situ verification process. The formal definition of the status update on the \( ch_q \) procedure is given below.

**Net module 4:** (Status update on the \( ch_q \).) The status update on the cluster head net module, \( P(N_4) \), is a 5-tuple net: \( P(N_4) = (P_4, T_4, F_4, W_4, (M_0_4)) \), where \( P_4 = \{p_{14}, p_{15}, p_{16}, p_{17}\} \) and \( T_4 = \{t_{19}, t_{20}\} \) are non-empty, finite, and disjoint sets of places and transitions, respectively. \( F_4 = \{(p_{14}, t_{19}), (t_{19}, p_{15}), (p_{15}, t_{20}), (t_{20}, p_{16}), (t_{20}, p_{17})\} \).

\[ W_4(p_{14}, t_{19}) = W_4(t_{19}, p_{15}) = W_4(p_{15}, t_{20}) = W_4(t_{20}, p_{16}) = W_4(t_{20}, p_{17}) = 1, \quad \text{and} \quad (M_0_4) = p_{14}. \]

Finally, the fifth functional specification, which expresses the procedure of the in situ verification result and aggregated data handling by the BS, is formally defined below.

**Net module 5:** (Status update on the BS). The status update on the base station net module, \( P(N_5) \), is a 5-tuple net: \( P(N_5) = (P_5, T_5, F_5, W_5, (M_0_5)) \), where \( P_5 = \{p_{18}, p_{19}, p_{20}\} \) and \( T_5 = \{t_{21}, t_{22}, t_{23}\} \) are non-empty, finite, and disjoint sets of places and transitions, respectively. \( F_5 = \{(p_{18}, t_{22}), (t_{21}, p_{19}), (t_{22}, p_{19}), (p_{19}, t_{23}), (t_{23}, p_{20})\} \).

\[ W_5(p_{18}, t_{22}) = W_5(t_{21}, p_{19}) = W_5(t_{22}, p_{19}) = W_5(p_{19}, t_{23}) = W_5(t_{23}, p_{20}) = 1, \quad \text{and} \quad (M_0_5) = p_{18}. \]

After formally defining the individual net modules, a next step is to methodically construct a unified Petri net model to depict the overall workflow of the system. The unified Petri net model should be able to satisfy all the functional specifications of the system. This phase is critical due to the fact that the individual net modules may project a consistent and correct behavior but grafting of terminal places or transitions of one net module may not be consistent with root places or root transitions of a next module. Therefore, in order to keep the unified Petri net model consistent, the pre and post conditions of each individual net module are satisfied during the construction of the unified Petri net model. Otherwise, the missing places or transitions may take the system into the halt state or the net modules may remain disjoint.

In order to construct the unified Petri net model, all of the net modules are combined in a place-transition or a transition-place joining manner. Additional arcs from transitions to places and places to transitions are introduced as \( F_6 = \{(t_5, p_6), (t_12, p_{18}), (t_{13}, p_{11}), (t_{18}, p_{14}), (p_{16}, t_{21})\} \). Corresponding weights are then defined as, \( W_6(t_5, p_6) = W_6(t_{13}, p_{11}) = 3 \) and \( W_6(t_{12}, p_{18}) = W_6(t_{18}, p_{14}) = W_6(p_{16}, t_{21}) = 1 \). This shows that in the unified Petri net model, a basic weight of an every arc is either 1 or 3 except those states where anomaly detection processes are modeled and their weights are determined through executions of the conditions \( g_1, g_2, g_3, g_4, \) and \( g_5 \). Note that these five conditions are part of the anomaly detection method presented in Section 4.

The first condition, \( g_1 \), checks the current battery level, \( B^{l}_{ec} \), of the sensor node, \( msn_q \). The \( msn_q \) can only initiate its working if its \( B^{l}_{ec} \) is equal or greater than the predefined threshold level, \( B^{l}_{o} \). If the \( B^{l}_{ec} \) of the \( msn_q \) is below the \( B^{l}_{o} \), then the \( msn_q \) is considered as faulty or dead. The first condition is defined below.

\[ g_1 = f(B^{l}_{ec}, B^{l}_{o}) = \begin{cases} 1, & B^{l}_{ec} \geq B^{l}_{o} \\ 0, & \text{Otherwise} \end{cases} \quad (1) \]

The second condition, \( g_2 \), verifies that a data packet, containing values of an \( F_q \), is received within an allocated timeslot, \( T^{l}_{t_1} \), or not. This condition is defined as

\[ g_2 = f(T^{l}_{t_1}, T^{l}_{t_2}, T^{l}_{t_2}) = \begin{cases} k, & T^{l}_{t_1} \leq T^{l}_{t_2} \leq T^{l}_{t_2} \\ 0, & \text{Otherwise} \end{cases} \quad (2) \]
The notation $T_{s}^{rg}$ represents a time of the arrival of a data packet containing values of features. The notations $T_{s}^{bg}$ and $T_{s}^{lb}$ denote a start and finish times of a timeslot dedicated to receive an $F_{q}$, respectively. The symbol $k$ represents a number of features of the normal profile of the $msn_{q}$, where $k = 3$ in this case. If a data packet is received within an allocated timeslot, then $k = 3$ features are processed for an anomaly detection.

Next, an anomaly detection process is carried out according to a nature of features. A corresponding condition from the $g_{3}$ and $g_{4}$ is executed for bounded and fixed value features, respectively. A fixed value features, such as the memory status, of the $msn_{q}$ are discrete in nature and they take discrete values such as 90%, 85%, etc. On the other hand, bounded features, such as sensor reading and battery status, are continuous random variables and take real number values. This type of features have upper and lower bound values to define a normal behavior of the $msn_{q}$.

$$g_{3} = f(P_{rg}^{lb}, m_{rg}^{f}, P_{rg}^{ub}) = \begin{cases} 1, & P_{rg}^{lb} \leq m_{rg}^{f} \leq P_{rg}^{ub} \text{ Otherwise.} \\ 0, & \end{cases} \tag{3}$$

$$g_{4} = f(m_{fx}^{f}, P_{fx}^{ub}) = \begin{cases} 1, & m_{fx}^{f} = P_{fx}^{f} \text{ Otherwise.} \\ 0, & \end{cases} \tag{4}$$

The notation $m_{rg}^{f}$ denotes a value of a bounded feature received from the $msn_{q}$. Similarly, the notations $P_{rg}^{lb}$ and $P_{rg}^{ub}$ represents minimum and maximum bounds to define a normal behavior of the $msn_{q}$ with respect to an $m_{rg}^{f}$. The notation $m_{fx}^{f}$ shows a received value of a fixed value feature and $P_{fx}^{f}$ represents a corresponding normal profile value. Finally, the last condition, $g_{5}$, checks a timely arrival of an in situ verification result.

$$g_{5} = f(T_{j}^{lb}, T_{j}^{WR}, T_{j}^{ub}) = \begin{cases} 1, & T_{j}^{lb} \leq T_{j}^{WR} \leq T_{j}^{ub} \text{ Otherwise.} \\ 0, & \end{cases} \tag{5}$$

The notation $T_{j}^{WR}$ denotes a time of the arrival of a watermarked in situ verification result, $WR$. The symbols $T_{j}^{lb}$ and $T_{j}^{ub}$ represent start and finish times of a timeslot allocated to receive an in situ verification result, respectively.

The definitions of the places and transitions of the unified Petri net model are given in TABLE I and the model itself is illustrated in Figure 2. States are represented by white circles in Figure 2, whereas transitions are denoted by black rectangles. A small black circle (namely token) in the place $p_{1}$, denotes an initial state of the system. Firing of a token from one place to another place denotes a change of a state in the system. The unified Petri net model is formally defined below.

**Unified Petri net model** The unified model, PN, is a 5-tuple net: $PN = (P, T, F, W, M_{0})$, where $P = \bigcup_{i=1}^{5} P_{i}$, $T = \bigcup_{i=1}^{5} T_{i}$, $F = \bigcup_{i=1}^{5} F_{i}$, $W = 1 \forall$ arcs except $W(p_{3}, t_{5}) = W(t_{5}, p_{4}) = W(t_{5}, p_{5}) = W(t_{6}, p_{6}) = W(p_{8}, t_{11}) = W(t_{11}, p_{10}) = W(p_{10}, t_{13}) = W(t_{13}, t_{11}) = W(p_{11}, t_{14}) = W(t_{14}, p_{12}) = 3, W(p_{1}, t_{9}) = 1, W(p_{5}, t_{6}) = 2, W(p_{6}, t_{7}) = 3, W(p_{6}, t_{8}) = 3, W(p_{6}, t_{9}) = 4, W(p_{14}, t_{19}) = g_{5}$, and $M_{0} = p_{1}$.

The place $p_{1}$ holds a single token at the time of the initiation of the first work cycle of the functionality of the system. The number of tokens from the place $p_{3}$ and onwards in the net module 1 must be equal to the number of features in the normal profile, i.e., three in this case. The flow of these tokens will then denote the current state of the system. The second functional specification expresses an anomaly detection process that is performed by the $ch_{q}$. In this process, the $ch_{q}$ receives values of a $F_{q}$ from the $msn_{q}$ and performs an anomaly detection process. Three tokens in the place $p_{5}$, in the net module 2, are resultant of the firing of the transition $t_{5}$, denoting the transmission of the accumulated values of $F_{q}$. The tokens are merged at place $p_{7}$, denoting an aggregation of a sensor reading at the $ch_{q}$. On the other hand, the tokens remain three at the place $p_{8}$ and onwards, each representing a value of a single feature. Next, the $msn_{q}$ receives an $MA$ which performs an in situ verification process.

The firing of the transition $t_{12}$, representing a transmission of an aggregated data, inputs a single token into the place $p_{11}$, which is the first place of the net module 5 (i.e., the status update on the BS process). On the other hand, the firing of the transition $t_{13}$, denoting the transmission of an $MA$, yields the three tokens into the place $p_{11}$. The three tokens, each denoting values of a single feature, are merged at place $p_{13}$, showing the state of the ready to transmit $MA$ to the $ch_{q}$. The firing of the transition $t_{18}$ yields a single token in the place $p_{14}$, representing the state of ready to receive the $MA$ from the $msn_{q}$.

The token received by the place $p_{15}$ is resultant of the firing of the transition $t_{12}$, denoting a transmission of an aggregated data to the BS. The aggregated data are then ready to be retrieved and analyzed by the system administrator. Alternatively, a firing of the transition $t_{21}$ yields a token in the place $p_{19}$ to retrieve and analyze a received alarm by the system administrator.

VI. FORMAL CHARACTERIZATION AND ANALYSIS

This section first formally characterizes and verifies the behavioral properties of the EAADS, namely, boundedness and liveness. Boundedness expresses the maximum number of tokens in the system, that is, the maximum number of processes a system can have at any given state. On the other hand, the liveness property shows that the system is deadlock free. The workflow of the EAADS and the reachability of the
TABLE I: Definitions of places and transitions

<table>
<thead>
<tr>
<th>Place</th>
<th>Definition</th>
<th>Transition</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>The msn_q wakes up</td>
<td>t1</td>
<td>The msn_q checks the battery level</td>
</tr>
<tr>
<td>p2</td>
<td>The msn_q is ready to collect the features</td>
<td>t2</td>
<td>The msn_q collects the value of SR</td>
</tr>
<tr>
<td>p3</td>
<td>The accumulated values of the features are saved</td>
<td>t3</td>
<td>The msn_q collects the value of MS</td>
</tr>
<tr>
<td>p4</td>
<td>The msn_q goes into the sleep mode</td>
<td>t4</td>
<td>The msn_q collects the value of BS</td>
</tr>
<tr>
<td>p5</td>
<td>The ch_q is ready to receive the F_q</td>
<td>t5</td>
<td>The msn_q transmits the accumulated F_q values</td>
</tr>
<tr>
<td>p6</td>
<td>The ch_q is ready to perform the anomaly detection</td>
<td>t6</td>
<td>The ch_q waits for $T^{ub}_j$ to receive the F_q</td>
</tr>
<tr>
<td>p7</td>
<td>The ch_q is ready to aggregate data</td>
<td>t7</td>
<td>The ch_q checks SR value for $(SR^{ub}, SR^{vb})$</td>
</tr>
<tr>
<td>p8</td>
<td>The ch_q is ready to decrement the trust count</td>
<td>t8</td>
<td>The ch_q checks MS value for $(MS^{ub}, MS^{vb})$</td>
</tr>
<tr>
<td>p9</td>
<td>The ch_q is ready to transmit the aggregated data</td>
<td>t9</td>
<td>The ch_q checks BS value for $(BS^{ub}, BS^{vb})$</td>
</tr>
<tr>
<td>p10</td>
<td>The ch_q is ready to transmit the MA</td>
<td>t10</td>
<td>The ch_q aggregates the sensed data</td>
</tr>
<tr>
<td>p11</td>
<td>The msn_q is ready to receive the MA</td>
<td>t11</td>
<td>The ch_q decrements the trust count</td>
</tr>
<tr>
<td>p12</td>
<td>The MA is ready to compare the data for the in situ verification</td>
<td>t12</td>
<td>The ch_q transmits the aggregated data</td>
</tr>
<tr>
<td>p13</td>
<td>The MA is ready to transmit the in situ verification result after insertion of the watermark</td>
<td>t13</td>
<td>The ch_q transmits the MA to the msn_q</td>
</tr>
<tr>
<td>p14</td>
<td>The ch_q is ready to receive the in situ verification result</td>
<td>t14</td>
<td>The msn_q receives the MA</td>
</tr>
<tr>
<td>p15</td>
<td>The ch_q is ready to check the in situ verification result</td>
<td>t15</td>
<td>The MA compares the SR with the previously stored values</td>
</tr>
<tr>
<td>p16</td>
<td>An alarm is generated to the BS by the ch_q</td>
<td>t16</td>
<td>The MA compares the MS with the previously stored values</td>
</tr>
<tr>
<td>p17</td>
<td>Status result saved by the ch_q</td>
<td>t17</td>
<td>The MA compares the BS with the previously stored values</td>
</tr>
<tr>
<td>p18</td>
<td>The BS is ready to receive the aggregated data</td>
<td>t18</td>
<td>The MA transmits the watermarked in situ verifica-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>tion result to the ch_q</td>
</tr>
<tr>
<td>p19</td>
<td>The BS is ready to analyze the data</td>
<td>t19</td>
<td>The ch_q waits for $T^{ub}_j$ time to receive the in situ verifica-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>tion result</td>
</tr>
<tr>
<td>p20</td>
<td>The BS is ready to retrieve the application data</td>
<td>t20</td>
<td>The ch_q checks the result</td>
</tr>
<tr>
<td></td>
<td></td>
<td>t21</td>
<td>The BS receives an alarm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>t22</td>
<td>The BS receives the aggregated data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>t23</td>
<td>The BS updates the repository</td>
</tr>
</tbody>
</table>

Figure 2: The unified Petri net model

Theorem 1. The unified Petri net model, PN, is 3-bounded.

Proof. $\forall p \in P, M(p) = 1$ except $P_c = \{p_4, p_5, p_6, p_8, p_{10}, p_{11}, p_{12}\}$, which has 3 tokens. The corresponding anomalous states in the unified Petri net model are also analyzed in this section. The reachability tree, as depicted in Figure 3, characterizes the workflow of the EAADS.
transitions are \( T_e = \{ t_5, t_6, t_7, t_8, t_9, t_{11}, t_{13}, t_{14} \} \), \( P_e \subseteq P \), and \( T_e \subset T \). This allows the independent treatment of the each feature for the anomaly detection process. There is a maximum number of 3 features which float over the specific places of the system; thus, the PN is 3-bounded.

**Theorem 2.** All transitions in the PN are level 4 live except the \( T_t \) transitions which are conditionally level 0 live.

**Proof.** \( \forall t \in T \), the level of liveness is 4 except \( T_t \), which is 0-live if and only if the output of the conditions \( g_1, g_3, g_4, g_9 \neq 1 \) and \( g_2 \neq k \), where \( T_t \subset T \) and \( T_t = \{ t_1, t_6, t_7, t_8, t_9, t_{19} \} \). This theorem holds as long as there exists the condition-based weighted arcs that input the transitions which model the anomaly detection process. The weighted arc conditions and the corresponding transitions are \( (g_1, t_1), (g_2, t_6), (g_3, t_7), (g_2, t_6), (g_3, t_7), (g_4, t_9), \) and \( (g_5, t_{19}) \).

**Theorem 3.** The net modules, \( \bigcup_{i=1}^{4} PN_i \), are sequentially executable, whereas the net module, \( PN_5 \), is sequential to both \( PN_2 \) and \( PN_4 \) to constitute the workflow of the EAADS.

**Proof.** The net module, \( PN_1 \), executes the \( F_e \) collection process through the reachable markings \( M_0[t_1] M_1[t_2] M_2[t_3] M_3[t_4] M_4[t_5] \), \( M_0[t_1] M_1[t_2] M_2[t_3] M_3[t_4] M_4[t_5] M_5[t_6] \), \( M_0[t_1] M_1[t_2] M_2[t_3] M_3[t_4] M_4[t_5] M_5[t_6] \), \( M_0[t_1] M_1[t_2] M_2[t_3] M_3[t_4] M_4[t_5] M_5[t_6] M_6[t_7] \), and \( M_0[t_1] M_1[t_2] M_2[t_3] M_3[t_4] M_4[t_5] M_5[t_6] M_6[t_7] \). \( M_0[t_1] M_1[t_2] M_2[t_3] M_3[t_4] M_4[t_5] M_5[t_6] M_6[t_7] \), as depicted in Figure 3, where \( M_{(\cdot)} \) and \( M^{(\cdot)} \) denote the terminal and nonterminal states, respectively. The terminal states represent the sleep mode of the node in question, i.e., \( m_{sn_q} \), following the transmission of the \( F_q \) after their collection. Next, without the loss of generality, only the case for the sensor reading which is straightforward to extend to rest of the features in the feature set is discussed. The \( F_e \) collection process is followed by the anomaly detection process through the following reachable marking sequences: \( M_0[t_6] M_1[t_7] M_1[t_{10}] M_2[t_{12}] M_2[t_6] M_0[t_6] M_1[t_7] M_1[t_{11}] M_2[t_{13}] M_2[t_8] \), where former marking denotes the transmission of the aggregated data, \( d^9 \), whereas the latter represents the transmission of the MA to the \( m_{sn_q} \). Similarly, the in situ verification process is carried out through the marking \( M_{27}[t_{14}] M_{33}[t_{15}] M_{39}[t_{18}] M_{44}, \) which is followed by the markings \( M_{44}[t_{19}] M_{47}[t_{20}] M_{50} \) and \( M_{44}[t_{19}] M_{47}[t_{20}] M_{51} \) for the storage of the in situ verification result, \( R \), on the \( ch_q \) and the transmission of the watermarked verification result, \( WR \), or aggregated data, \( d^9 \), to the BS, respectively. The former branch is terminated, whereas the latter is followed by the marking \( M_{51}[t_{21}] M_{56}[t_{23}] M_{59}, \) This yields that the net modules, \( \bigcup_{i=1}^{4} PN_i \), are sequentially executable. The unique reachable marking \( M_{39}[t_{22}] M_{32}[t_{23}] M_{38} \) is grafted after the marking \( M_0[t_6] M_1[t_7] M_1[t_{10}] M_2[t_{12}] \), a reachable marking of the \( PN_2 \). Thus, the net module \( PN_5 \) is sequential to both the \( PN_2 \) and \( PN_4 \), as depicted in the reachability tree shown in Figure 3.

**Theorem 4.** The anomaly detection process for every
individual feature is parallel executable.


The reachability of the $M' \in RM(M_0)$ states which corresponds to the anomaly detection process in the EAADS is verified next.

Theorem 5. The anomaly detection states, i.e., $M_j[t_i] M'_j$, are only reachable when system is anomaly free, where $M_j = M_0$, $M'_j = (M_1, M_11, M_12, M_13, M_14, M_15, M_16, M_17, M_18, M_19, M_47, M_48, M_49)$, and $t_i = t_1, t_6, t_7, t_8, t_9, t_{19}$ for the corresponding reachable markings.

Proof. The marking $M_1$ is reachable if and only if $W(p_{t_1}, t_1) = 1$ in the unified PN model. If the execution of the condition $g_1$ yields a value other than 1, then the node in question, $msn_{q_4}$, is treated as anomalous either due to a fault or dead status. Thus, $M_1$ is not reachable in this case. Similarly, the markings $M_{11}, M_{12},$ and $M_{13}$ are reachable through marking sequences $M_0[t_1] M_1[t_2] M_2[t_5] M_6[t_6] M_{11}, M_0[t_1] M_1[t_3] M_3[t_5] M_8[t_6] M_{12},$ and $M_0[t_1] M_1[t_4] M_4[t_5] M_{10}[t_6] M_{13}$, respectively, if and only if $W(p_{t_6}, t_6) = k$, which is determined through the condition $g_2$. Otherwise, if $W(p_{t_6}, t_6) \neq k$ then these states are not reachable, denoting the anomalous behavior of the $msn_{q_4}$. This is followed by the firing of the transitions $t_7, t_8,$ and $t_9$ to reach the markings $(M_14, M_15), (M_16, M_17)$ and $(M_18, M_19)$, respectively. The condition $g_3$ defines the arc weights of the transitions $t_7$ and $t_8$ for the markings $(M_14, M_15)$ and $(M_16, M_17)$, respectively. Similarly, the condition $g_4$ defines the arc weight of the transition $t_9$ for the markings $(M_18, M_19)$. The outcomes other than 1 of these conditions make the states unreachable, representing the anomalous values of the normal profile features of the $msn_{q_4}$. Next, the result of the execution of the condition, $g_5 \neq 1$, makes the $M_47, M_48,$ and $M_49$ states unreachable due to the inconsistency between the weights of the arcs and the tokens in the input place of the transition-place arcs.

The above proof advocates that the anomalous states are unreachable in the unified Petri net model and only those states which exhibit the normal behavior of the $msn_{q_4}$ are reachable. Thus, the unified Petri net model is capable of detecting different kinds of anomalies caused by the erroneous values of the features and also the temporal anomalies which occurred due to the delayed arrivals of the observations and in the situ verification results.

An important implication of the formulation of the unified PN model is the integration of the minor but important specifications in the algorithmic specifications of the initial design of the proposed system. The updated features collection process (i.e., Algorithm 1) puts the sensor node into the sleep mode after completion of its designated task. This adjustment, in the original functional specification, saves unnecessary energy consumption. Another key update is made in the anomaly detection process (i.e., Algorithm 2) where the $Ch_q$ waits for the allocated timeslot to receive the $F_q$. This empowers the AADS to detect the temporal anomalies caused due to the delayed or non-arrival of the $F_q$ at the $ch_q$.

VII. The unified GSPN model

The appropriate temporal behavior of an anomaly detection system is imperative in order to detect and verify the source of the anomalies in a timely manner. There are two key processes in the EAADS which require the temporal behavior analysis: (i) the combined process of the tasks of the collection of the values of the features on the $msn_{q_4}$ and their arrival on the $ch_q$, which will be called as $\alpha$ process in the sequel and (ii) the combined process of the tasks of the anomaly detection, mobile agent transmission, in situ verification, and verification result arrival on the $ch_q$ are named as $\beta$ process in the sequel. In order to analyze the temporal behavior of the EAADS in the nondeterministic communication environment of WSNs, which is usually caused by traffic characteristics, channel errors, and environmental influences [27], the unified Petri net model is extended into the corresponding high class unified GSPN model.

The policy semantics of the unified GSPN model are based on the transition types, rates, server semantics, and memory policy. All transitions are considered as immediate except those which are involved in the computation of the $\alpha$ and $\beta$ processes. The weight of the immediate transitions is kept as 1, whereas the rate for the timed transitions is variable according to the processing performed by them. The rates for the timed transitions are based on the resource capability of the MICAz mote [26].

The atomic firing policy is employed for the timed transitions in which a token remains in the input place until the timer expires. The functionality of
the EAADS is sequential. Therefore, single server semantics are chosen for all transitions, except the transitions \((t_2, t_3, t_4), (t_7, t_8, t_9), \) and \((t_{15}, t_{16}, t_{17})\), which have \(k\)-server semantics due to their parallel processing in the unified GSPN model. It is easy to observe that the value of \(k\) is 3, i.e., the number of features for the anomaly detection in the EAADS. Furthermore, the age memory policy is considered for all of the transitions due to the continuous nature of the operations involved in the EAADS. Based on the above policy semantics, the formal definition of the unified GSPN model is given below.

**Unified GSPN model:** The Generalized Stochastic Petri net-based unified model, GSPN, is an 8-tuple net: \(GSPN = (\mathcal{P}, \mathcal{T}, \Pi(\cdot), I^-(\cdot), O^+(\cdot), H(\cdot), W(\cdot), M_0)\), where \(\mathcal{P} = \bigcup_{i=1}^{I} P_i, \mathcal{T} = \bigcup_{i=1}^{I} T_i, I^-(\cdot) \cup O^+(\cdot) = \mathcal{F} = \bigcup_{i=1}^{I} F_i\). The workflow of the underlying PN is sequential and there exists no inhibition arc. Therefore, \(\Pi(\cdot) = \emptyset\) and \(H(\cdot) = \emptyset\). The weight for the time transitions \(W(\cdot) = t_1 = t_2 = t_3 = t_4 = 0.25\text{ms}, t_5 = 6\text{ms}, t_7 = t_8 = t_9 = t_{11} = t_{15} = t_{16} = t_{17} = 1\text{ms}, t_{13} = t_{14} = 15\text{ms}, t_{18} = 1\text{ms}.\) Finally, the initial marking \(M_0 = p_1\).

The temporal behavior of the \(\alpha\) and \(\beta\) processes is analyzed below by employing the above defined unified GSPN model. The time taken by the \(\alpha\) process can be computed as

\[
\alpha = \alpha_1^t + \alpha_2^t + \alpha_3^t \tag{6}
\]

The notation \(\alpha_1^t\) denotes the time taken by the \(F_q\) collection process (represented by the transitions \(t_1\) to \(t_4\) in the unified GSPN model). The symbol \(\alpha_2^t\) represents the time taken by the \(F_q\) to reach the \(ch_q\) from the \(msn_q\) (modeled by the transition \(t_5\)). Finally, the \(\alpha_3^t\) shows the delay of the arrival of the \(F_q\) on the \(ch_q\) due to channel errors, environmental factors, and traffic characteristics. The delay occurs on the communication link which is modeled by the arc \((t_5, p_3)\).

The time taken by the \(\beta\) process can be derived from the following formula.

\[
\beta = \sum_{i=1}^{6} \beta_i^t \tag{7}
\]

The notation \(\beta_i^t\) denotes the overall time taken by the task of the anomaly detection performed by the \(ch_q\) (represented by the \(t_7, t_8, t_9, \) and \(t_{11}\) transitions in the unified GSPN model), \(\beta_0^t\) represents the transmission time taken by the \(MA\) for traveling from the \(ch_q\) to the \(msn_q\) (denoted by the transition \(t_{13}\)), \(\beta_1^t\) shows the time taken by the \(MA\) to perform the task of the in situ verification on the \(msn_q\) (modeled by the transitions \(t_{14}\) to \(t_{17}\)), and \(\beta_2^t\) indicates the time taken by the in situ result arrival on the \(ch_q\) (denoted by the transition \(t_{18}\)). The notations \(\beta_5^t\) and \(\beta_6^t\) denote the delay factors for the arrival of the \(MA\) on the \(msn_q\) and the in situ verification result on the \(ch_q\), respectively. The delay factors \(\beta_3^t\) and \(\beta_4^t\) occur at the links which are modeled by the arcs \((t_{13}, p_{11})\) and \((t_{18}, p_{14})\), respectively.

The delay factors \(\alpha_1^t, \beta_0^t, \) and \(\beta_5^t\) are considered in the closed interval \([0,1]\) for the analysis, where \(0\) represents no delay (i.e., representing the low error-prone communication channel) and \(1\) denotes the 100% delay (i.e., representing the high error-prone communication channel). Furthermore, the delay factors have the exponential distributions, which is consistent with the modeling requirements of the Petri net theory for the stochastic processes [21]. The sensor nodes, i.e., \(msn_q\) and \(ch_q\), are assumed to be 10 meters apart from each other for the theoretical analysis and corresponding experiments.

First, a sensitivity analysis over the delay factors \(\alpha_3^t, \beta_0^t, \) and \(\beta_5^t\) was conducted. In order to accomplish this task, the \(\alpha_3\) was varied over the closed interval \([0,1]\). This yielded the result that the time taken by the \(\alpha\) process in the low-error prone communication channel was as low as 6.99 ms. On the other hand, a single \(\alpha\) process took as long as 8.00 ms time in the communication channel, which delays the communication among the \(msn_q\) and \(ch_q\) by as much as the 100%. The unified GSPN model-based results of the temporal behavior of the \(\alpha\) process are shown in the upper panel of Figure 4. The corresponding statistics that include mean (\(\mu\)), standard deviation (\(\sigma\)), minimum (\(Min\)), and maximum (\(Max\)) of the results are given in TABLE II.

![Figure 4: The temporal behavior of the \(\alpha\) process](image)

<table>
<thead>
<tr>
<th>Table II.</th>
<th>The (\alpha) process statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>(\mu)</td>
</tr>
<tr>
<td>Unified GSPN model</td>
<td>7.35</td>
</tr>
<tr>
<td>Implementation</td>
<td>7.86</td>
</tr>
</tbody>
</table>

© 2015 ACADEMY PUBLISHER
38.96 ms and 69.92 ms for the low and high-error prone communication channels, respectively. These unified GSPN model-based results are shown in the upper panel of Figure 5 and the corresponding statistics are presented in TABLE III.

VIII. Temporal behavior validation

The temporal behavior of the EAADS is validated by the implementation of the proposed functional specifications on the TinyOS, running on the MICAz motes, which have the Atmel ATmega128L low power microcontroller [26]. The relatively high data rate radio of the MicaZ mote is capable of transmitting the data at the rate of 250 kbps. The MICAz mote has the 128 and 4 kb of Program Flash Memory and Configuration Electrically Erasable Programmable Read-Only Memory (EEPROM), respectively. It can also be equipped with an additional 512 kb of Measurement (Serial) Flash, which can store more than 100,000 measurements.

The network topology, based on the two available micaZ motes, was deployed to perform the experiments. This minimal network topology is sufficient for the validation of the temporal behavior of the α and β processes due to the fact that these processes are performed through the processing and communication between the $ch_q$ and $msn_q$ nodes. Therefore, one available node was deployed as the $ch_q$, whereas the other was configured as the $msn_q$. It is pertinent to mention that the results obtained from this topology may be generalized to a large scale single hop cluster-based WSNs by considering the communication link between the $msn_q$ and $ch_q$ as a unit component of the network-wide communication.

The $MA$ was developed which had a size of 762 bytes including both the code and data. The Zigbee/802.15.4-compliant devices (i.e., MicaZ motes in this case) can only transmit a maximum of 127 bytes (i.e., header = 25 bytes, payload = 102 bytes) in a single packet [31]. Therefore, the $MA$ was segmented into the 8 packets on the transmitter side and reassembled into $MA$ on the receiver side in order to carry out its designated task. The header size for all of the data packets was set as 25 bytes, whereas the payload size for the first 7 packets was 102 bytes and for the last packet was 48 bytes. Similarly, the size of the each data packet which contained the values of the $F_q$ was set as 31 bytes (i.e., header = 25 bytes and payload = 6 bytes). The size of the in situ verification result data packet was 27 bytes (i.e., header = 25 bytes and payload = 2 bytes). Furthermore, in the experiments, the delay factor was induced using the exponential distribution in order to emulate the nondeterministic delay caused in the WSNs.

The five cases were implemented in order to validate and analyze the temporal behavior of the EAADS. The first two cases were implemented for the validation of temporal behavior results of the $\alpha$ and $\beta$ processes obtained through the formal modeling and analysis carried out in Section 7. The third case was implemented to study the impact of the overhead on the temporal behavior of the EAADS which was caused by securing the $MA$ with watermark and also empowering the $MA$ to watermark the in situ verification results. The last two cases were implemented to further analyze the temporal behavior of the EAADS with respect to the distance between the nodes and the data rate of two different sensor network applications.

Case 1: The implementation of the $\alpha$ process was performed in order to validate the results obtained through the $\alpha$ process modeled in Section 7. The experimental results revealed that in contrast with the unified GSPN model, the time taken by the $\alpha$ process remained slightly higher, i.e., between 7.50 ms and
TABLE IV.: The statistics of the $\beta$ process with the normal and secure $MA$

<table>
<thead>
<tr>
<th>Model</th>
<th>$\mu$</th>
<th>$\sigma$</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta$ process with secure $MA$</td>
<td>80.81</td>
<td>13.69</td>
<td>61.71</td>
<td>113.83</td>
</tr>
<tr>
<td>$\beta$ process with normal $MA$</td>
<td>51.24</td>
<td>8.54</td>
<td>39.76</td>
<td>71.35</td>
</tr>
</tbody>
</table>

8.50 ms for the low error-prone and high error-prone communication channels, respectively. These results are shown in the lower panel of Figure 4 and the corresponding statistics are presented in TABLE II. It can be observed from TABLE II that the lower and upper bound differences between the Unified GSPN model and implementation results for the $\alpha$ process are just 0.51 ms and 0.50 ms, respectively, which are not significant. This difference may have occurred due to environmental influences.

**Case 2:** The implementation of the $\beta$ process was carried out in order to validate the results obtained through the $\beta$ process modeled in Section 7. In this case, the in situ verification results were transmitted to the $ch_q$ without the insertion of the watermark. In practice, this case is applicable when the EAADS is deployed for the detection and verification of the anomalies caused by only the in transit or in situ errors or faults. The time taken by the $\beta$ process in the experiments was between 39.76 ms and 71.35 ms, in contrast with the unified GSPN model where it oscillated between 38.96 ms and 69.92 ms, as shown in Figure 5. The corresponding statistics are given in TABLE IV. The time consumption by the $\beta$ process is very efficient in order to perform the tasks of the anomaly detection, in situ verification, and in situ verification result arrival on the $ch_q$ which is 10 meters away from the $msn_q$ in a highly nondeterministic communication environment.

**Case 3:** Next, in the $\beta$ process case, the $MA$ was programmed to insert the watermarks in the in situ verification results which were transmitted to the $ch_q$ by the $MA$ from the $msn_q$. In this case, the watermark was inserted in the code and data of the $MA$ itself using Radix-$k$ encoding [25], in order to protect it from the in transit or in situ attacks. This case, in practice, is applicable when the EAADS is capable of detecting and verifying the anomalies also caused by the in transit or in situ attacks. In this case, the size of the $MA$ was increased from 762 bytes to 977 bytes due to its additional capability to insert the watermark in the in situ verification result. The insertion of the watermark in the $MA$ itself caused an overhead of around 25%. Thus, the total size of the $MA$ became 1220 bytes. In compliance with the Zigbee/802.15.4 standard, the $MA$ was segmented into 12 packets for transmission. The size of the each of the first 11 packets was 127 bytes including both the header and payload. The size of the last packet was, however, 123 bytes. In this case, the size of the in situ verification result was 127 bytes including the watermark.

The experimental results, in this case, indicate that the time consumption for the scenario where $MA$ was secured with the watermark and was capable of inserting the watermark in the in situ verification result was 61.71 ms and 113.83 for the low and high error-prone communication channels. On the other hand, in the case of the normal $MA$, the time consumption was 39.76 ms and 71.35 ms for the low and high-error prone communication channels, respectively. This indicates that the overhead in terms of the time consumption was 62.68% and 64.37% as compared to that of the $\beta$ process with the normal $MA$ (i.e., case 2). The results are shown in Figure 6 and the corresponding statistics are given in TABLE IV.

**Case 4:** In this case, the distance between the $msn_q$ and $ch_q$ was varied from 3 meters to 15 meters and the other experiment configurations of the case 3 were kept the same. The distance was varied in order to study the affect of the distance on the in situ verification process performed by the $MA$. The results show that the time consumption was between 61.20 ms and 61.94 ms when the distance was between 3 meters and 15 meters, respectively, for the low error-prone communication channel. Similarly, the time consumption was between 112.50 ms and 115.04 ms for the high error-prone communication channel when the distance was between 3 meters and 15 meters, respectively. These results indicate that the increase in the distance between the $msn_q$ and $ch_q$ does not greatly influence the temporal behavior of the $\beta$ process. These results are given in Figure 7 and the corresponding statistics are presented in TABLE V.

**Case 5:** The $\beta$ process was implemented and its
The temporal behavior was analyzed for both the periodically as well as the continuously sensed data transmitting applications. In the case of the periodically sensed data transmitting applications, the behavior of the built infrastructure monitoring sensor network in which the msnq periodically transmits the sensed data to the respective chq [28] was modeled. On the other hand, the behavior of the fire tracking sensor network was modeled for the continuously sensed data transmitting application, where the msnq continuously transmits the sensed data to the respective chq after the detection of an event [29]. For the continuous data transmitting application, the serial flash was used to store the values of the Fq which were employed by the MA to perform the task of the in situ verification on the msnq.

During the experiments, it was observed that the application which continuously transmits the sensed data to the chq inevitably stores more observations before the arrival of the MA to perform the task of the in situ verification on the msnq. Therefore, the MA was required to spend more time to perform the task of the in situ verification for such applications. This increases the overall time of the \( \beta \) process. The experimental results indicate that the time taken by the \( \beta \) process remained between 73.29 ms and 104.25 ms for the continuous application as compared to 39.76 ms and 71.35 ms for the periodic application. It was also observed that the msnq stored the values of 10 additional observations for the task of the in situ verification before the arrival of the MA. This increased the time taken by the task of the in situ verification from 3.79 ms to 37.9 ms. This result implies that the in situ verification process is more suited to periodically sensed data transmitting applications rather than continuously sensed data transmitting applications. The results are shown in Figure 8 and the corresponding summary of the statistics is provided in Table VI.

![Figure 8: Continuous vs periodic application](image)

### Table VI: Statistics of the applications

<table>
<thead>
<tr>
<th>Model</th>
<th>( \mu )</th>
<th>( \sigma )</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous application</td>
<td>83.63</td>
<td>7.88</td>
<td>73.29</td>
<td>104.25</td>
</tr>
<tr>
<td>Periodic application</td>
<td>51.24</td>
<td>8.54</td>
<td>39.76</td>
<td>71.35</td>
</tr>
</tbody>
</table>

IX. Comparison and Discussion

The EAADS is compared with the recent related studies [7]–[9]. The direct experimental comparison with these recent works may not be possible due to the fact that these are architecture focused studies and no experimental analysis is performed. The comparison with our previous work, namely, Agent-enabled Anomaly Detection System (AADS) [10], [11], is also made in order to distinguish the characteristics of the work presented previously and in this paper. The comparison is made in terms of the following aspects: (i) detection complexity, (ii) types of detected anomalies, (iii) number of mobile agents per node, and (iv) the temporal behavior suitability of the schemes.

In a study, Khanum et al [7] presented an architecture of the mobile agents based hierarchical Intrusion Detection System for WSNs which employs three agents for the anomaly detection, namely, Analyzer Agent, Management Agent, and Coordinating Agent. The anomalies are detected at two levels: node and network. The detection complexity of the scheme is \( O(n) \). Nevertheless, no attempt has been made to quantify the performance of the architecture through an experimental study or numerical analysis. Eludiora and colleagues [8] presented a scheme which can detect anomalies caused by the Denial of Service, DoS, attack. The detection complexity of the scheme is \( O(n^2) \). In another study, Ketel [9] suggested the use of the several static and mobile agents for distributed anomaly detection. Although the increased use of the mobile agents enhances performance of the anomaly detection, it also increases the computation complexity, transmission cost, and memory overhead.

During the experiments, it was observed that the application which continuously transmits the sensed data to the chq inevitably stores more observations before the arrival of the MA to perform the task of the in situ verification on the msnq. Therefore, the MA was required to spend more time to perform the task of the in situ verification for such applications. This increases the overall time of the \( \beta \) process. The experimental results indicate that the time taken by the \( \beta \) process remained between 73.29 ms and 104.25 ms for the continuous application as compared to 39.76 ms and 71.35 ms for the periodic application. It was also observed that the msnq stored the values of 10 additional observations for the task of the in situ verification before the arrival of the MA. This increased the time taken by the task of the in situ verification from 3.79 ms to 37.9 ms. This result implies that the in situe verification process is more suited to periodically sensed data transmitting applications rather than continuously sensed data transmitting applications. The results are shown in Figure 8 and the corresponding summary of the statistics is provided in Table VI.

The EAADS is compared with the recent related studies [7]–[9]. The direct experimental comparison with these recent works may not be possible due to the fact that these are architecture focused studies and no experimental analysis is performed. The comparison with our previous work, namely, Agent-enabled Anomaly Detection System (AADS) [10], [11], is also made in order to distinguish the characteristics of the work presented previously and in this paper. The comparison is made in terms of the following aspects: (i) detection complexity, (ii) types of detected anomalies, (iii) number of mobile agents per node, and (iv) the temporal behavior suitability of the schemes.

In a study, Khanum et al [7] presented an architecture of the mobile agent based hierarchical Intrusion Detection System for WSNs which employs three agents for the anomaly detection, namely, Analyzer Agent, Management Agent, and Coordinating Agent. The anomalies are detected at two levels: node and network. The detection complexity of the scheme is \( O(n) \). Nevertheless, no attempt has been made to quantify the performance of the architecture through an experimental study or numerical analysis. Eludiora and colleagues [8] presented a scheme which can detect anomalies caused by the Denial of Service, DoS, attack. The detection complexity of the scheme is \( O(n^2) \). In another study, Ketel [9] suggested the use of the several static and mobile agents for distributed anomaly detection. Although the increased use of the mobile agents enhances performance of the anomaly detection, it also increases the computation complexity, transmission cost, and memory overhead.

During the experiments, it was observed that the application which continuously transmits the sensed data to the chq inevitably stores more observations before the arrival of the MA to perform the task of the in situ verification on the msnq. Therefore, the MA was required to spend more time to perform the task of the in situ verification for such applications. This increases the overall time of the \( \beta \) process. The experimental results indicate that the time taken by the \( \beta \) process remained between 73.29 ms and 104.25 ms for the continuous application as compared to 39.76 ms and 71.35 ms for the periodic application. It was also observed that the msnq stored the values of 10 additional observations for the task of the in situ verification before the arrival of the MA. This increased the time taken by the task of the in situ verification from 3.79 ms to 37.9 ms. This result implies that the in situ verification process is more suited to periodically sensed data transmitting applications rather than continuously sensed data transmitting applications. The results are shown in Figure 8 and the corresponding summary of the statistics is provided in Table VI.
aptness of the EAADS in such environments. The summary of the above discussion, given in Table VII, shows the competitiveness of the EAADS in a number of aspects with the several related schemes.

The key findings of the theoretical analyses, experimental results, and comparison of the EAADS with the previous studies as follows.

- The design of the EAADS is complete and correct.
- The EAADS is capable of detecting the temporal anomalies in addition to perform the in situ verification and detection of anomalies in the received data from the cluster member sensor nodes.
- The EAADS has a suitable temporal behavior in a highly nondeterministic communication environment.
- The EAADS takes 64.37% more time in order to perform the task of the in situ verification when it is also deployed for the detection and verification of the anomalies caused by the attacks, unlike when it is deployed only for the anomalies caused by the faults or errors.
- The variation in the close range distance has a minimal impact on the temporal behavior of the EAADS. This implies that the EAADS is highly suitable for the WSN applications such as the smart home sensor network and built infrastructure monitoring applications where nodes are generally deployed in a relatively close proximity to each other.
- The EAADS is more suited for periodically sensed data transmitting applications as compared to continuously sensed data transmitting applications.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection complexity</td>
<td>$O(n)$</td>
<td>$O(n^2)$</td>
<td>Not applicable</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Detected Anomalies</td>
<td>Sensor reading anomalies</td>
<td>DoS attack anomalies</td>
<td>Node anomalies by neighbor monitoring</td>
<td>Denial of sleep attack, resource exhaustion attack, and node faults</td>
<td>Temporal, sensor reading, battery status, and memory status anomalies, and also node faults</td>
</tr>
<tr>
<td>Number of mobile agents per node</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Temporal behavior Aptness</td>
<td>Not considered</td>
<td>Not considered</td>
<td>Not considered</td>
<td>Not considered</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**X. Conclusion**

This study has presented an Enhanced mobile Agent-Enabled Anomaly Detection (EAADS) for WSNs. The formal specifications of the algorithmic functionalities are addressed by employing a rich formalism offered by the Petri net theory. The behavioral properties of the EAADS are then formally characterized. The analysis shows that the system is deadlock free and unlike previous studies it can detect different types of anomalies caused by the erroneous values of the sensor readings, battery status, and memory status of the sensor nodes. The EAADS is also capable of detecting temporal anomalies which are caused by the delayed arrival of the values of the features and also the in situ verification results. Furthermore, a non-estimation of the temporal behavior of a system may jeopardize its effectiveness after deployment. Therefore, in this view, the unified Petri net model is extended into a high class Petri net model, namely, a unified GSPN model, in order to quantify the temporal behavior of the EAADS. The unified GSPN model-based temporal behavior results are then validated by the implementation of the proposed functional specifications on a real testbed. The experimental results also establish the suitability of the EAADS for periodically sensed data transmitting applications in contrast with continuously sensed data transmitting applications.

In further work, we aim to formalize the extended feature set to exploit the statistical association to detect complex natures of anomalies. We also intend to extend and evaluate the proposed anomaly detection system and corresponding Petri net models to more advanced scenarios such as multi-node anomaly detection system in the multipath routing sensor networks.

**References**


