RENOWNED: A Real-time Anomaly Detection and Mitigation Framework in Edge-Enabled IoV

Chandrajit Pal, Sangeet Saha, Xiaojun Zhai and Klaus McDonald-Maier

***Abstract*—The rapid adoption of smart vehicles and their interconnection through the Internet of Vehicles (IoV) has increased the use of Electronic Control Units (ECUs) in cars. These ECUs, while enabling advanced features, also present a larger target for cyberattacks, which can disrupt critical functions and jeopardize safety. The time-sensitive nature of automotive systems necessitates swift responses, making the protection of ECUs crucial. The imprecise computation (IC) task model can mitigate the risk of task completion failures by generating acceptable approximation results within deadlines when achieving absolute accuracy becomes difficult within fixed deadlines and energy budgets. This paper introduces RENOWNED, a solution that ensures the normal functioning of these Controller area networks (CAN) controlled ECUs even in the face of anomalies. It combines anomaly detection and mitigation through the HEALING module to maintain the desired performance. The anomaly detection module uses Graph Attention Networks (GAT) to identify unusual processor behaviour. If an anomaly is detected, the HEALING module takes over, reallocating tasks based on the available resources to guarantee that deadlines are met and energy constraints are not exceeded. Experiments have shown that RENOWNED delivers a Quality of Service (QoS) of 25% to 64% when system utilisation is varied in the range from 40% to 90%. It exhibits an excelling performance in detecting anomalies, achieving a 97.6% accuracy even when the magnitude mixed anomaly signals are very minute. Thus our proposed RENOWNED offers a robust way to enhance the reliability and energy efficiency of safety-critical automotive applications prevalent in IoV.**

***Index Terms*—Imprecise Computation (IC), energy-aware scheduling, quality of service (QoS), Precedence-constrained Task Graphs (PTGs), Normalised QoS (NQ), Hardware Performance Counters (HPCs), Graph Attention Networks (GAT), Internet of Vehicles (IoV), Electronic Control Units (ECUs).**

I. Introduction

The growing integration of smart vehicles into the Internet of Vehicles (IoV) has brought about a significant increase in the use of CAN controlled Electronic Control Units (ECUs) present within an automobile responsible for performing essential functions like V2X (Vehicle-to-Everything) communication, V2V (Vehicle-to-Vehicle) interaction, and a multitude of internal vehicular functionalities including sensing, signal processing, actuating and various infotainment [1], [2]. While this connectivity enhances vehicle capabilities, it also creates a larger attack surface for cyber threats, potentially leading to disruptions in vehicle operation and safety risks. The end goal of most cyberattacks on vehicles is to disrupt the normal functioning of the interconnected ECUs within a vehicle as illustrated in Figure 1. Hence, protecting these ECUs is crucial for continuing their normal functioning and meeting the hard real-time functioning nature. Critical task functions like sensing, braking and steering require immediate responses. Therefore, failing to meet task deadlines due to an external attack on them can be disastrous [3]. These ECUs are practical examples of hard real-time embedded devices that operate with limited resources and power, where partially accurate results obtained within a given deadline and energy budget are preferable to fully accurate results obtained after these constraints [4]. For instance, in V2V communication sensing video streams with lower-quality frames obtained before a deadline is preferable to entirely missing frames. An approximate location estimate within a deadline is preferable to an accurate location obtained too late in target tracking [5].

The applications running directly on these ECUs controlling the vehicular hardware are generally modelled as Precedence-constrained Task Graphs (PTGs) [6], [7] where, a node represents a task associated with the application, while an edge denotes interdependencies among tasks. Despite the battery-operated components' resource-constrained nature and restricted energy budgets, these interconnected ECUs must deliver superior performance and high-quality services [8]. Because of the fixed energy budget, an Imprecise Computation (IC) task model [9] can help reduce the risk of task completion after the deadline. This is achieved by generating an acceptable approximation result within a deadline if the system is unable to provide an accurate result on time. In this approach, a task can be decomposed into *necessary* and *optional* components [5]. To produce the minimum acceptable QoS, all necessary components must be executed before the deadline. Depending on resource availability, the optional component may then be partially or fully executed to increase the accuracy of the initial output obtained within the deadline—the more execution cycles used on the optional component, the higher the QoS.

Recent research focuses on the problem of energy-efficient real-time scheduling of IC tasks to enhance performance while adhering to the underlying system constraints. In [10], the authors proposed a scheduling mechanism which performs additional computations, when there is more energy available, and accepts imprecise results, when the energy budget is limited. However, their study is limited to independent tasks. Authors in [11], [12], [13] considered dependent IC tasks and proposed utilising dynamic voltage frequency scheduling DVFS for designing energy-efficient scheduling strategies. However, when DVFS decreases the supply voltage and frequency to save power, the system's soft error and transient fault

C.Pal, S.Saha, X. Zhai and Klaus McDonald-Maier are with the School of Computer Science and Electronics Engineering, University of Essex, CO4 3SQ Colchester, UK.

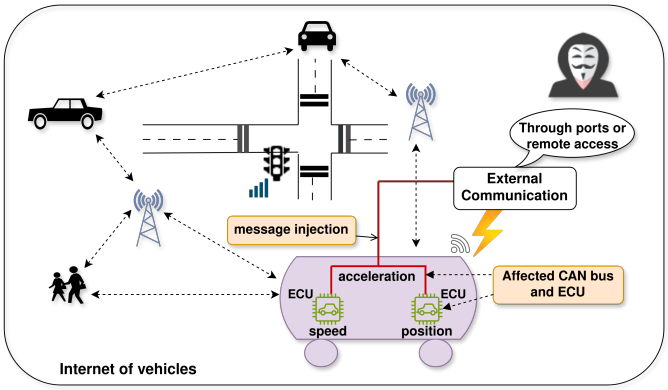


Fig. 1: A typical Internet of Vehicles scenario

rates increase dramatically, decreasing the system's reliability [14]. Wang et al. [15] proposed to determine an *optimal energy frequency* in multiprocessor platforms as a workaround.

Modern multi-ECU-based automotive systems are often subjected to faults resulting from run-time errors, preventing them from normal functioning [16]. Intrusions like hardware trojan attacks tend to slow down a processor's speed resulting in real-time applications missing its deadline [17], [18]. Malware can also result in unintentional power dissipation when the energy budget is fixed, which exhausts the system's energy budget and renders it unusable [19], [20]. As a result, real-time systems fail to accomplish their tasks within the deadline. This makes it crucial to implement error-resilient security modules capable of identifying these anomalies and implementing suitable measures for their mitigation to ensure a proper continuation of system execution within a vehicle. Hence, *given a task graph and a multi-ECU-based automotive application, the successful execution of all associated tasks within the given deadline by mitigating any runtime error while satisfying energy and precedence-related constraints is a challenging problem in IoV.*

In this manuscript, we propose RENOWNED, consisting of a reliability-aware scheduler that ensures the system maintains a desired QoS despite having any anomalies through intelligent detection and mitigation procedures. We considered a multi-ECU system and dependent real-time IC task graph simulating an automotive safety-critical application shown in Figure. 3. RENOWNED schedules the tasks by allocating them to appropriate processors at specific time instants, following a predefined scheduling strategy and QoS. At runtime, if additional power dissipation during task execution is observed, or if a task cannot be finished within the designated time resulting in a reduction of desired QoS, the anomaly detection module is enabled, which finds the presence of an anomaly in the system (through a detailed analysis based on the correlation among multiple observed Hardware Performance Counters (HPCs) being traced from the ECUs), if any and labels the corresponding processing core of an ECU as faulty. Thereafter a new schedule is generated based on the available energy budget, remaining execution time and the available idle processing cores as depicted in Figure. 2. The new schedule guarantees that every necessary and maximum possible optional portion will be completed within the energy budget to obtain the best possible QoS. RENOWNED's suitability is demonstrated and proven by the obtained high Normalised QoS (NQ) defined in Equation 8.

RENOWNED showcases several novelties compared to existing methodologies as it integrates anomaly detection and mitigation, ensuring continuous operation even in the middle of disruptions. It performs fine-grained anomaly detection at the hardware level, using GATs to analyze correlations among multiple HPCs and identify subtle anomalies with high accuracy. It efficiently schedules dependent IC tasks, intelligently allocating necessary and optional components based on energy and deadline constraints. Furthermore, it balances QoS and energy consumption under attacks, dynamically reallocating tasks and adjusting frequencies to maintain desired QoS while meeting energy constraints.

The following summarises the contributions of RENOWNED:

1) RENOWNED schedules dependent IC tasks by executing the entire necessary components and appropriate optional components based on the available energy budget and deadline.
2) At runtime, our intelligent graph attention network-based runtime security mechanism can recognise abnormalities in processor performance by finding a correlation among multiple observed Hardware Performance Counters (HPCs) of multi-ECU cores at a fine granular hardware level. It excels at detecting anomalies, achieving a 97.6% accuracy even when the mixed anomaly signals are very subtle.
3) Given an IC task graph in a multi-ECU-based automotive application, RENOWNED maintains the Quality of Service of 25% to 64% when system utilisation is varied in the range from 40% to 90%.

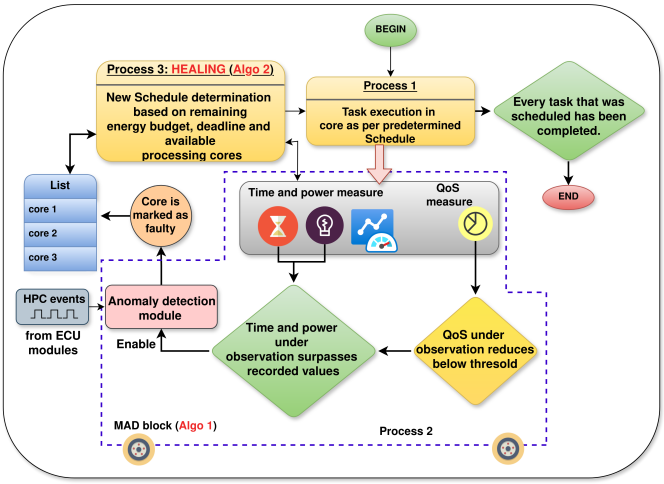


Fig. 2: RENOWNED FLOW CHART

II. System Modelling and Assumptions

A. Processing core Model

We consider an evaluation platform consisting of homogeneous multiprocessing ECU cores with p processing cores

represented as $C = \{c_1, c_2, ..., c_p\}$. Each core is characterised by discrete voltage, power and frequency levels. The next subsection will describe behavioural task modelling.

B. Task modelling

Considering a real-time vehicular application (A) that is represented by a PTG (Figure 3) which is directed and acyclic expressed as $G_r = (T_k, E_d)$, where T_k denotes a task set ($T_k = \{T_{ki} \mid 1 \leq i \leq |T_k|\}$) and E_d a list of directed edges ($E_d = \{\langle T_{ki}, T_{kj}\rangle \mid 1 \leq i, j \leq |T_k|; i \neq j\}$), illustrating the precedence relationships among different task pairs. An edge $\langle T_{ki}, T_{kj}\rangle$ denotes precedence such that task T_{kj} cannot initiate execution until T_{ki} completes.

Since this is a real-time application, all associated task nodes must be executed within the interval for the entire application (A) to meet its deadline. We decomposed every task T_{ki} ($1 \leq i \leq n$) into a necessary component expressed as N_i, together with an optional component OP_i. Executing the necessary components is required to achieve an acceptable result. Only after the necessary component N_i has been completed the optional component OP_i is executed either partially or entirely.

For every task T_{ki}, the execution length can be represented as:

$$L_i = N_i + \lambda \times OP_i \tag{1}$$

where λ consists of n discrete values and represents the portion of the executed optional component. Consequently, $\lambda = 1$ indicates the execution of entire OP_i units to produce the most accurate result possible. The total of the OP component cycles executed for each task determines the system-level result accuracy or the quality of service (QoS). It is also assumed that there are n_i distinct task versions T_{ki}; i.e. $T_{ki} = \{T^1_{ki}, T^2_{ki}, \ldots, T^{n_i}_{ki}\}$. For example as shown in Fig. 7 b. the optional portion of the first task T_k1 in the usual scenario, is 14. This means the complete range of λ i.e. 0 to 1 is divided into 14 parts and in the anomaly scenario a reduced range of the optional components is executed based on an anomalous situation to mitigate the harmful effects of the attack.

The energy consumption E_i of task T_{ki} having the length L_i is expressed as [21]:

$$E_i = L_i \times POW_i \tag{2}$$

where POW_i represents the power consumption of the processing core executing task T_{ki}. The equation denotes that the longer the task, the more energy it will consume to complete it. Conversely, more execution of the optional portion (which lengthens the task) will lead to more improvements in task accuracy, which will ultimately improve the system's QoS. The scheduling goal will be maximising the QoS by running a higher version of every task while minimising energy.

III. Threat Modelling

Threats typically arise from intentional or natural causes. Intentional manual threats in an Internet of Vehicles (IoV) are carried out by intruders, where adversaries introduce malware in firmware or inject hardware-based faults in working ECUs through network communication, causing delays, execution termination, and power draining etc. Below is a list of latent threats [23], [24]:

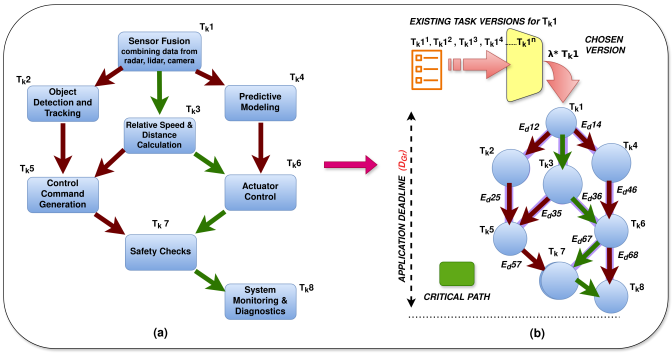


Fig. 3: (a) Dynamic in-vehicular control module [22], (b) Its corresponding Task Graph

A. Execution termination of a processing core

This could happen as a result of a runtime error or a hardware trojan-like malware that has been introduced, which prevents it from working as represented in Figure 4.b with the normal scenario shown in Figure 4.a.

B. Unexpected task execution delay

Malware or ageing processor hardware are the two possible causes of this. This prevents the real-time system's ability to complete within a deadline as shown in Figure 4.c.

C. Intentional draining of power

This generally occurs due to malware implanted by adversaries. Besides unwanted delay resulting in additional power draining, deliberate malware has its separate power-dissipating circuitry that executes simultaneously to the original circuit. They might not affect the timing but if they go unnoticed, they will lead to high power consumption and eventually exhaust the system's energy budget. The PTG's tasks closer to the sink won't have sufficient energy to accomplish their task completion if the energy budget is depleted in the early or middle phases. As a result, the system will be unable to finish the tasks before the deadline as illustrated in Figure4.d.

D. Reduced QoS

When the measured QoS reduces a predefined threshold it denotes an anomalous situation.

IV. Proposed Methodology

RENOWNED consists of two major design blocks namely the Monitoring and Anomaly Detection (MAD) block and the remedial scheduling block (HEALING) (Figure 2). The system RENOWNED initiates its execution based on a predefined normal schedule (process 1). During its execution, the MAD block (process 2) detects the threats through the following

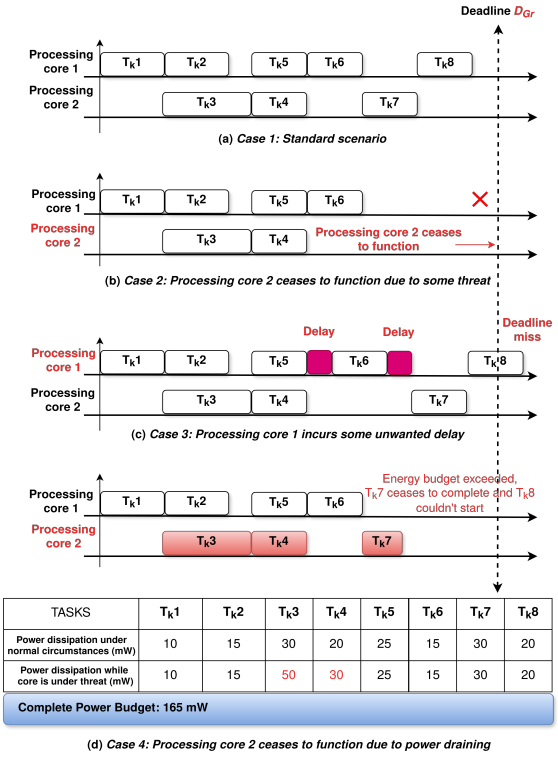


Fig. 4: Illustrating the threat Model

Algorithm 1: Enforcing Security Measure: Monitoring and Anomaly Detection (MAD)

Input: i. Observed Time and Power information obtained from sensors,
ii. Threshold QoS.
iii. Predefined Time and Power information,
iv. Measured QoS.
Output: Processor core labelled as normal/faulty

1 **for** *every Task* T_{ki} *running on Processing core* PC_i **do**
2 **if** *(Observed Power consumption* $>$ *Predefined Power consumption)* $\vee$
3 *(Observed Time* $>$ *Predefined Time)* $\vee$
4 *(Measured QoS* $<$ *Threshold QoS)* **then**
5 Call and enable anomaly detection module; /* Refer Figure 5 */
6 **if** *FAULTY==1* **then**
7 Processoring core CP_i is labelled faulty;
8 Call HEALING (i.e Algorithm 2) to schedule the remaining tasks on the non-faulty healthy processing cores (substituting the faulty core with a backup core executing at *high frequency*);
9 **else**
10 Go to step 12;
11 **else**
12 Reflects normal scenarios and follows the regular schedule of execution;

three observations with the monitored *time* of execution and *power* dissipation of individual processing cores. If this surpasses the predefined recorded values and the *monitored QoS* is reduced below a predefined threshold, the system enables the anomaly detection module in the MAD block, which if confirms the presence of an anomaly, marks the corresponding processing core as faulty. As a substitute it restores a backup core and increases its operating frequency and the HEALING module is called which generates a new schedule based on the remaining energy budget and application deadline and the available processing cores at that instant of time (process 3).

A. Anomaly detection and Mitigation Implementation

Our proposed security implementation covers two aspects, namely anomaly detection and mitigating its effects through our proposed HEALING module.

1) Alert for Anomaly: The threats discussed in Section III indicate anomalous situations. Three factors—power, timing and QoS variations are used as indicators of anomalous situations.

Timing based alert: Defects, ageing, and deliberate delays can all be found with detection based on the parameter timing. The security module of a processing core keeps track of the beginning and end times of a specific task. If the execution time that is observed is longer than a predefined threshold completion time, it indicates an anomalous situation and enables the anomaly detection module to begin functioning *(Algorithm 1) MAD*.

Power and QoS based alert: Besides unwanted delay resulting in additional power draining, deliberate malware has its separate power-dissipating processes that execute simultaneously to the original circuit that uses up the system's energy budget by dissipating excessive power. The security module determines if the observed power consumption exceeds a predefined threshold power consumption, it enables the anomaly detection module which detects the presence of faults if any, and labels the corresponding processing core as faulty. Following this, a standby core is restored and is made to function at an increased frequency, and the scheduler module is re-executed to schedule the remaining tasks on the unaffected processing cores within the available energy budget before the deadline. Similar actions are repeated for generating a fresh schedule if situations described in subsection III-D arise *(Algorithm 1)*.

In our current implementation, the operating frequency of the backup core is set to be higher than that of the faulty core to compensate for the potential delay in executing the remaining tasks before the deadline and available remaining energy. We employ a heuristic approach to determine the frequency, balancing the need for faster execution with energy constraints. This involves selecting the lowest frequency level among the available options that still ensures the timely completion of all remaining tasks within the energy budget.

2) Proposed Anomaly detection module: Our anomaly detection mechanism operates at a fine-grained hardware level, utilising the correlation among multiple HPCs. These HPCs are specialized registers within the processor that track various hardware events, such as cache misses, branch predictions, and instruction counts (referring to Fig. 5).

- HPC Data Collection: We begin by collecting time-series data from the HPCs while executing benchmark programs

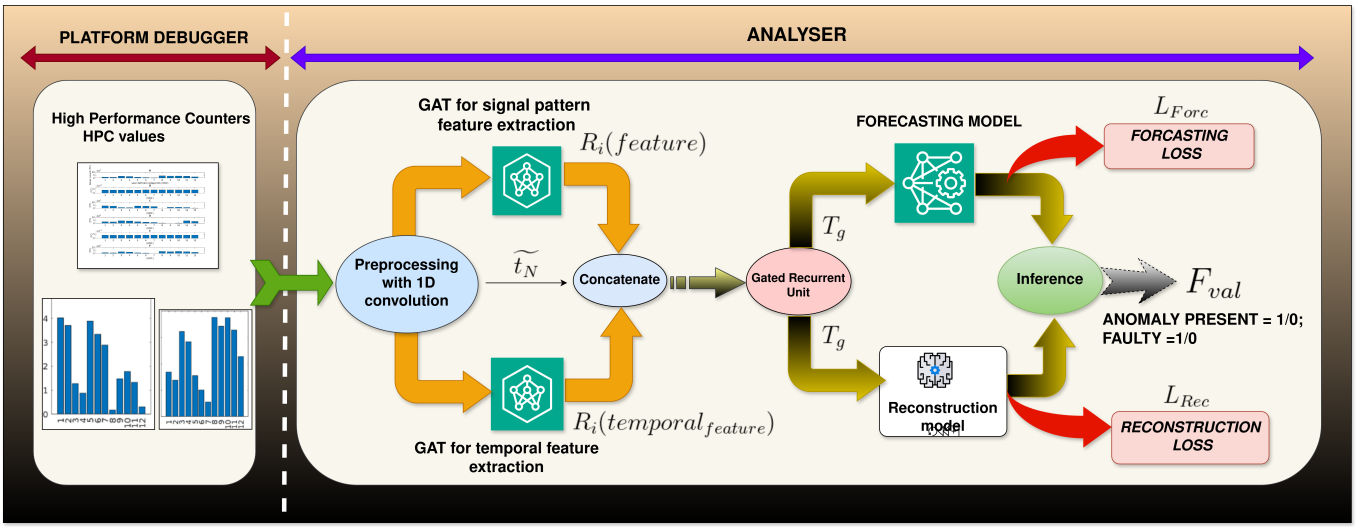


Fig. 5: Anomaly detection module (refer MAD block in Figure 2)

on the ECUs. This data reflects the normal behaviour of the processor under typical workloads.

- GAT-Based Correlation Modeling: The collected HPC data is then used to train two Graph Attention Networks (GATs). GATs are a type of neural network well-suited for capturing relationships and dependencies within graph-structured data. In our study, the Hardware Performance Counters (HPCs) are the node features, and the relationships (correlations) among these HPCs are the edge features. The HPCs are represented as nodes in a graph, and the GATs learn the intricate correlations and dependencies among them. Two GATs operate simultaneously: one for extracting signal pattern features and another for extracting temporal features.
  Node Features: The GATs learn node features (embeddings) by aggregating information from neighbouring nodes. Each GAT layer updates the representation of a node by attending to its neighbours' features. The attention mechanism allows the GAT to weigh the importance of different neighbours based on their feature values. **Edge Features:** The edge features are implicitly captured in the attention mechanism. The attention weights (r_{ij}) assigned to edges reflect the strength and importance of the relationship between connected nodes. **Temporal Features:** The temporal GAT specifically focuses on capturing temporal dependencies in the time-series data. It learns how the values of HPCs change over time and how these changes correlate with each other. **Embeddings:** The final output of the GATs is a set of learned embeddings (feature vectors) for each node (HPC timer). These embeddings capture the complex relationships and patterns within the HPC data, both spatially (between different HPCs) and temporally (over time).
- GRU for Temporal Dependencies: To further capture the temporal dynamics within the HPC data, we employ a Gated Recurrent Unit (GRU). GRUs are a type of recurrent neural network that excels at modelling sequential data and capturing long-term dependencies. The node embeddings from the two GATs are concatenated and fed into a Gated Recurrent Unit (GRU). The GRU further refines the feature representation by capturing long-term dependencies in the time-series data. It learns how the patterns and relationships between HPCs evolve over longer time intervals.
- Reconstruction and Forecasting: The output of the GRU is fed into two parallel models: a reconstruction model and a forecasting model. The reconstruction model, a variational autoencoder (VAE), learns the underlying data distribution of the HPC time series. The forecasting model focuses on predicting future HPC values based on past trends.
- Anomaly Inference: By jointly optimizing the reconstruction and forecasting models, we obtain an anomaly inference score. This score reflects the likelihood of a given HPC data point being anomalous, considering both its deviation from the expected distribution and its deviation from predicted values.

While executing benchmark programs, virtual and physical timer values are generated at four different privileges while we execute the benchmark programs in the processing cores, which we used to train our graph neural net-based attention model (Figure 5).

Referring Figure 5 the problem definition is expressed as: The values from the multi-variable time series generated during program execution are $t_m \epsilon R_n^{T_L \times f_n}$, where T_L, f_n and R_n represent the length of the maximum timestamp, the no. of input features and real number respectively. Longer time series is handled by leveraging a fixed-length sliding window W_T as input. An anomaly in the i^{th} timestamp is indicated by $R_i \epsilon (0, 1)$, which is the multi-variable times series response $R \epsilon R_n^{W_T}$. Hence, we initiate by modelling the inter-feature correlations and temporal dependencies utilising two GATs operating simultaneously. Following this, a Gated Recurrent Unit (GRU) is placed to extract the long-term data

dependencies in the sequence. In the following sequence, we designed a tiny variational autoencoder-based reconstruction and forecasting model for jointly optimising an integrated objective function.

The process is initiated by feeding the input from various HPC registers as time series sequential data produced while executing the benchmark programs and it is then subjected to normalisation followed by single-dimension convolution (with kernel size set to 5) for extracting high-level features from the time series HPC variables [25]. The normalisation is performed considering the maximum and minimum values present in the labelled dataset and is expressed as:

$$\widetilde{t_N} = \frac{tm - min(tm_{label})}{max(tm_{label}) - min(tm_{label})} \tag{3}$$

Each timer variable (i.e High-Performance Counter HPC) tm_i is treated as a separate feature variable modelled as a node v_i of the graph shown in Figure 6, and the relationships r_{ij} among such nodes are the corresponding edges e_{ij}. The resultant attention score R is computed taking into account the contribution of adjacent nodes as the neighbouring HPC nodes equated as:

$$R_i = \sigma(\sum_{j=1}^{N} r_{ij} t_{Nj}) \tag{4}$$

where σ is the sigmoid activation function, the edges e_{ij} are represented by r_{ij} which measures the attention score representing the contribution of adjacent nodes(timers tm_j in our study) from i to j and N is the no. of neighbourhood HPC timers of j. Both feature- and time-oriented GATs are used to model the correlations between the multiple feature variables (i.e., timer registers) and the time series temporal dependencies simultaneously. These GATs are used to generate a learned relationship among the various HPCs. This allows for identifying any discrepancies that may arise when utilising an untrained routine arising out of threats.

Referring to Figure 5 the output obtained from the two GAT models is concatenated and fed into a 24-unit (GRU model size 492 KB) layer expressed as:

$$\begin{aligned} R_i(feature) + R_i(temporal_{feature}) + \widetilde{t_N} \rightarrow GRU \\ GRU \rightarrow Tg \end{aligned} \tag{5}$$

This is followed by a GRU layer to record time-series sequential patterns. The output of the GRU model is fed into both the *reconstruction* and *forecasting* models in parallel. The reconstruction model aims to learn the marginal data distribution by gaining knowledge of a latent representation of the entire time series, whereas the forecasting model concentrates on single-timestamp prediction. Both the model parameters are updated simultaneously during training.

Our proposed reconstruction model (size 1.5 MB) learns the marginal data distribution pattern acquired from the knowledge of the latent representation of the entire time series. The forecasting model focuses on single-timestamp prediction. During training, both model parameters are updated simultaneously, and the total loss L_f is computed by adding the losses from each model as follows:

$$L_f = L_{Rec} + L_{Forc} \tag{6}$$

where L_{Rec} and L_{forc} represents the reconstruction and forecasting module losses respectively. The forecasting module is designed using 2 fully connected layers (model size 2.6 MB approx) meant for computing the next predicted time stamp.

The reconstruction model aims to learn a minimal distribution of information over a latent representation. A tiny Variational Auto-Encoder (VAE) is leveraged to describe the phenomenon in the latent space in a probabilistic way [26]. By considering the time-series values as variables, the model can capture the data distribution of the entire time series. After taking into account the integrated optimisation target and computing the projected values of the forecasting and reconstruction models, the model inference is finally computed. For every feature, an inference value I_i is computed, and the final inference value F_{val} is the summation of all the individual features shown in Equation 7. We identify a predefined threshold [27], and label a timestamp as anomalous if its matching inference score exceeds it. The final inference is expressed as (ref. Figure 5):

$$F_{val}(L_{Rec}, L_{Forc}, \lambda) = \sum_{i=1}^{f} I_i = \sum_{i=1}^{f} \frac{(Tg_i - \hat{Tg}_i)^2 + \lambda \times (1 - p_i)}{1 + \lambda} \tag{7}$$

where $(\hat{Tg}_i - Tg_i)^2$ is the forecasting error computed as the absolute deviation between the predicted and present actual value, and $(1 - p_i)$ illustrates the likelihood of coming across an anomalous value by the reconstruction model. p_i represents the probability of the i^{th} feature being normal as estimated by the reconstruction model. It reflects this likelihood, where a higher value indicates a greater probability of the feature being normal. and λ a hyper-parameter which combines the reconstruction probability and the forecasting error, is chosen by a grid search on the testing dataset. The final inference determines the fate of a processor in ECU to be faulty.

3) Security Preservation: Following a processing core being labelled as faulty during execution, a new backup core is restored to substitute the faulty one and is made to work at a higher frequency compared to the faulty core to compensate for the delay in executing the rest of the tasks before the deadline. As shown in Figure 7 (a), Task 1 of length $L1$ is executed on core 1 bearing frequency $f1$ and consuming power $P1$. When the task is suspended due to a threat, it is restarted to execute (by opportunistically shredding some of the optional components) on a standby processing core 2 bearing frequency $f2$ and consuming power $P2$ where $f2 > f1$. This increase in processing frequency leads to an increase in energy consumption which is compensated by reducing the task length through opportunistically shredding some of the optional components of Task 1 to generate Task 1' (of length $L2$).

Subsequently, the scheduler module is re-executed to schedule the leftover tasks on the remaining processing cores within the available energy budget before the deadline. An example of Power and QoS-related alerts and threats are mitigated as shown in Figure 7b. As shown in Figure 7(b.1), excessive power draining in core 1, led to raising an alarm of suspicious

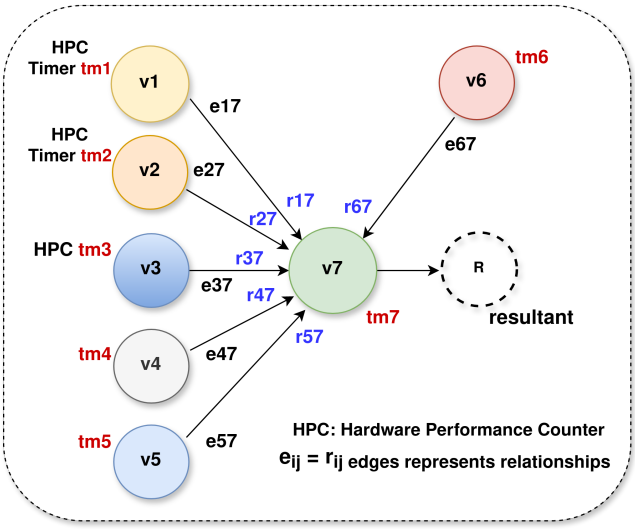


Fig. 6: Graph attention layer, $v_i \rightarrow tm_i$ and $e_{ij} \rightarrow r_{ij}$ denotes the vertices (used timers as HPC in our study) and edges (relationship among timing events), dashed node R is the resultant

events, enabling the anomaly detection block which after detecting an anomaly, labels the core 1 as faulty, resulting in Task 1 failing to complete within the deadline. A standby processing core is called for and RENOWNED calls for the HEALING module which generates a fresh schedule. This involves completing the necessary components of every task and relaxing the completion of optional portions, as shown to schedule the execution of the remaining tasks on the remaining processing cores within the available energy budget before the deadline *(Algorithm 2 HEALING)*. To compensate for the increased energy consumption by the high-frequency operating standby core, clock gating is applied in its idle states and the tasks are operated at reduced optional components. A similar strategy is applied as depicted in 7(b.2) when measured QoS goes below a threshold value.

The NQ is computed as the *ratio of the executed optional component to the total number of available optional components of all the tasks of a given application expressed as:*

$$NQ_A = \frac{\sum_{i=1}^{n} OP_e i}{\sum_{j=1}^{n} OP_{Lj}} \tag{8}$$

where OP_e is the executed optional component and the denominator is the total number of available optional components of all n tasks of application A. The ranges of the necessary N_i and optional OP_i components of a task are obtained by leveraging [13].

B. Proposed HEALING module

The HEALING module is a real-time adaptive task scheduler that is activated when an anomaly is detected. It is designed to ensure the continued execution of the application, while meeting deadlines and energy constraints, despite the presence of anomalies.

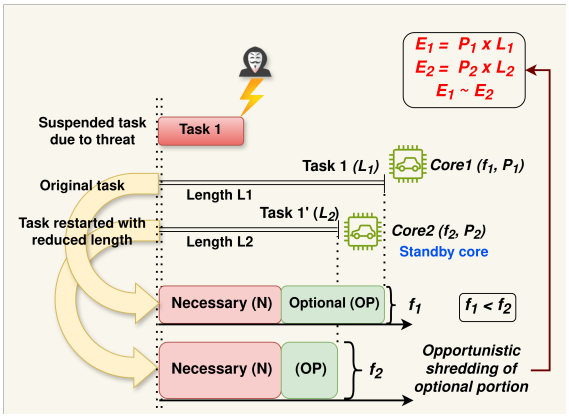
In this subsection, we will discuss the scheduling Algorithm 2 HEALING, which is responsible for assigning the tasks represented as nodes in the PTG (Figure 3). Suppose the parent nodes of tasks have already completed their execution. In that case, the task is allotted to the idle processing cores based on the energy budget at that instance, else it selects the task version with the lesser optional component OP (equation 1). Based on the task-to-core mapping, the allocation of the start time of a task will be the latest completion time of its predecessors. When a task has a single parent, the algorithm can start working on it as soon as the parent is finished. A task T_{kj} is assigned to a processing core c_i which executes it till it satisfies the condition. RT_i flag represents **R**emaining **T**ime necessary to complete the present task by processing core c_i, following which RT becomes zero. After a task is completed, it is added to the set C_T and removed from χ. This task allocation and execution operations mentioned above keep going iteratively until all of the tasks in χ have been completed, deadline D_{G_r} has been reached, and the allotted energy budget E_{budget} has been used up. Summarising the methodology of HEALING:

- Initialization: Identify available processing cores and initialize the task list.
- Task Mapping and Execution: Iterate through tasks until completion, deadline, or energy budget exhaustion.
- Task Selection and Allocation: Select a task whose predecessors are complete and allocate it to a free core.
- Energy-Aware Task Version Selection: Choose the appropriate task version based on the available energy budget.
- Task Execution and Monitoring: Execute task on the assigned core and monitor remaining time.
- Task Completion and Core Release: Release core after task completion, update completed task list, and adjust energy budget.
- Iteration and Termination: Repeat steps 2-6 until all tasks are complete, the deadline is reached, or the energy budget is depleted. Calculate the achieved Normalized QoS (NQ).

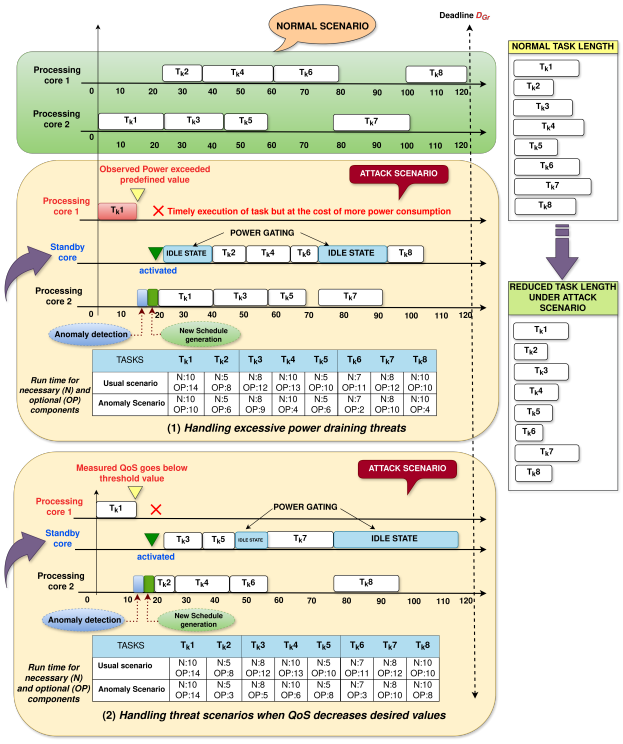
It prioritizes necessary task components for achieving minimum QoS while dynamically adjusting the optional tasks according to energy and deadline and leverages parallel execution for efficiency.

V. Experimentation and Results

An application under execution consists of a set of programs and every program (considered as PTG) consists of a set of dependent and independent functions which we consider as tasks of the PTG. Tasks have been carefully chosen which will not throw any exception but with a bit decrease in tolerable performance if partially executed as part of the optional component. The task graphs that we created were produced using the task graphs for free (TGFF) tool [28]. For our experiments, we used a homogeneous quad-core processing cluster environment similar to Neoverse-N1 processors to serve as the basis for the new, experimental, out-of-order CPU used in the ARM SoC prototype architecture [29]. Regarding Figure 8, the experimental setup comprises an ARM processor located in the hardware prototype platform connected to a host PC via a DSTREAM-PT trace device to probe the runtime



(a)



(b)

Fig. 7: (a) A threat suspending a task and restarting it on a standby processing core with high frequency and opportunistically shredding off an optional section of the task to accommodate for overall energy consumption (b) Overall scenario of threat mitigation: use of standby processing cores with high frequency to finish of task execution within deadline and use of clock gating and opportunistically shredding off optional components of tasks to meet deadlines and energy consumption whilst maintaining accuracy.

Algorithm 2: Proposed Task Scheduling based on Deadline and Energy Budget (HEALING)

Input:
i. $L_i^{v_i}$: Denotes the length of task T_{ki} of v_i^{th} version
ii. $E_i^{v_i}$: labelled energy dissipation for v_i^{th} version of task T_{ki}.
iii. D_{G_r}: denotes the PTG deadline (Figure3, 4).
iv. arranged list of tasks, χ.
v. E_{budget} of the present application.
Output: Assigning the remaining tasks to the available resources to meet the deadline and energy budget.

1 /*..... BEGIN.....*/
2 /* Considering PC representing the group of processing cores that are presently operative; */
3 A Task T_{kj} consists of Necessary N_j and optional components OP_j.
4 Assign $PC = C$; /* Set of processing cores */
5 $\forall\ c_i \in PC$, Set PA_i= 0; /* Initialise, a flag variable PA_i to 0 if the processing core is free for execution; else set to 1. In the beginning, every processing core is freely available */
6 Assign $\chi = T_{kj}$; /* Move tasks into χ variable.
7 OP_{Lj} a list of optional components of the task T_{kj}
8 /*.....TASK MAPPING & EXECUTION.....*/
9 **for** $t_m = 0$; $t_m \leq D_{G_r}$ & $(\chi$ & $E_{budget}) \neq VOID$; t_m++ **do**
10 **for** every single processing core available *in parallel* **do**;
11 **if** $\exists\ T_{kj} \in \chi$ | *Every ancestor of* T_{kj} *have completed their job &* $PC \neq VOID$ **then**
12 **if** $(E_{budget} \geq E_i^{v_i})$ **then**
13 Processing core(s) c_i with PA_i == 0 is selected;
14 Set PA_i = 1 /* Initialised c_i to *occupied*; */
15 *Task* T_{kj} *is mapped to processing core* c_i;
16 $T_{kj}B = t_m$ /* The present time t_m is initialised as the beginning of T_{kj} execution */
17 $RT_i = L_j^{v_j}$; /* Begin T_{kj} execution; RT_i: this flag holds the **R**emaining **T**ime necessary to complete the present task in c_i */
18 $PC = PC \setminus c_i$; /* c_i is removed from PC list */
19 **else**
20 The next lower version of OP_i is chosen from OP_{Lj} and move to step 12
21 **else**
22 $RT_i = RT_i$-1; /* Remaining Time is decremented */
23 **if** RT_i == *0* **then**
24 PC = PC $\cup \{c_i\}$; /* c_i is added to the list of freely available processing cores */
25 PA_i = 0; * c_i is set to *free*; */
26 $C_T = C_T \cup T_{kj}$ /* T_{kj} is added to the list of C_T of completed tasks */
27 $\chi = \chi \setminus T_{kj}$; /* Task T_{kj} is removed from list χ */ $E_{budget} = E_{budget} - E_j^{v_j}$; /* Decrease the amount of energy used from the remaining budget. */
28 Return and compute Equation 8.

data from the running programs on the platform. Our target platform has a DSTREAM-PT debug probe that can do high-performance debugging and tracing, is appropriate for quick downloads, and can adjust to JTAG clock rates. It also has several peripheral interfaces, including Ethernet connections, USB interfaces, and debug trace ports. Upto 300 MHz DDR (600 Mbit/s per pin) has been employed to capture traces that

are 32 bits wide. The trace device traces this runtime data from different program variables and functions via the trace port before sending it to the host PC for anomaly detection and fresh schedule generation.

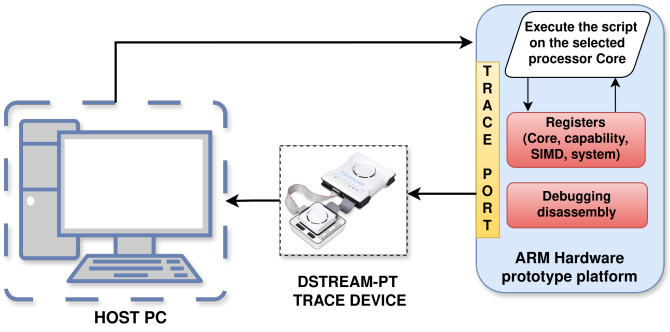


Fig. 8: RENOWNED experimental setup

This study made use of seven benchmark algorithms from the well-known automobile package of the EEMBC benchmark suite [30]. This benchmark simulates an actuator being driven by a PWM signal proportionate to an input. PWM signals provide proportionate velocity control, while phase signals control the motor's direction. Although having different complexities and characteristics, they are suitable to serve as test subjects for the proposed experiments since they share similar sub-routines being executed on the SPMD datasets [31]. As shown in Table I, we have conducted an initial experiment by training the first five benchmark programs and kept the last two namely the $idctrn$ and $ttsprk$ for testing on our trained models as unknown routines. This has yielded a result of anomaly detection with accuracies as shown in Table I. The unknown programs are being detected as anomalous programs even though their signature is similar to that of known trained programs $a2time$ and $tblook$ respectively.

The SPMD dataset, which is massive and extensively used in many fields of linked autonomous vehicle technology, records the driving data of over 2500 automobiles over two years. The present investigation involved the extraction of comprehensive high-frequency data from a single vehicle in the SPMD dataset. This data included the experimental vehicle's in-vehicle velocity (sensor 1), GPS velocity (sensor 2), and in-vehicle acceleration (sensor 3). It is important to note that although the acceleration sensor records the vehicle's acceleration value based on speed, the in-vehicle speed sensor and the GPS speed sensor record distinct types of speed values. Four potential genuine anomalies with a 5% anomaly rate were created and added to the SPMD data because anomalies were not included in the original SPMD dataset. The anomalies are namely instantaneous, constant, progressive drift, and deviation anomalies [31] describes an analogous experimental procedure, as described in brief below:

- *Instantaneous*: denotes a sudden, drastic change between two readings in a row;
- *Constant*: transient, ongoing change unrelated to "normal" sensor reading.
- *Progressive drift*: A minor but enduring change that takes place during a certain time;

TABLE I: EEMBC Autobench benchmarks performance

Benchmarks	Accuracy	Precision	Recall	F1 SCORE
$bitmnp$	97	99	98.2	99.09
$a2time$	96.6	98	98.1	98.04
$puwmod$	97.3	99	97.4	98.19
$rspeed$	98	99.05	98.04	98.54
$tblook$	96	98.3	97.6	97.94
$idctrn$	98.4	0	0	0
$ttsprk$	98.6	0	0	0
General performance	98.46	99	98	98.36

- *Deviation/Bias anomalies*: a consistent offset within a certain time frame.

Results comparison under single and mixed anomaly types: Table II compares the performance evaluation of the SOTA methodologies of various magnitudes of instant anomalies. The model performance has been *compared at the lower range of* 10,25,50,100 and 200 *amplitudes* respectively. It is very challenging to detect anomalies at the smaller magnitude of the instant anomaly that is consistent with reality. Since at a smaller magnitude, the difference between normal and abnormal data is less making it difficult to distinguish anomalous situations, which motivated us to design models which work better in lower range amplitude suitable for anomaly detection. This is evident from Table II our proposed RENOWNED performs better than the SOTA models when evaluated on Accuracy, Precision and Sensitivity scores respectively at lower magnitudes while compared to [31], [32] and [33]. All the compared models perform better at higher magnitudes, however, our model still achieves 97.6% classification accuracy when the anomaly amplitude is as low as base value + 10 ×N(0, 0.01) below 25 reflecting RENOWNED's robustness. Here, N(0, 0.01) represents a Gaussian distribution with a mean of 0 and a standard deviation of 0.01. This signifies that the anomalies are very subtle and challenging to detect, yet our model maintains high accuracy. The anomaly signal's magnitude is measured using the amplitude of the injected anomalies relative to the base value of the time series data.

For a better evaluation, the performance of RENOWNED is evaluated under mixed anomaly types with other SOTA models in Table III where it is evident that the performance of RENOWNED has qualitatively improved. It is seen that for constant anomaly detection RENOWNED achieved the maximum F1 score of 84.93%, 87.43 % while detecting linear anomalies and 88.12% for bias-based anomalies.

A. Results

1) Related to anomaly detection: The generated HPC values resulting from executing the complete dataset on the benchmark programs are used for training our AI models. During validation, the various anomalies are injected and prediction is done accordingly. Figure 9(a), (b) shows the snapshot of HPC values obtained while executing a set of tasks of a benchmark program on a processing core. (a) represents the system's response to an injected instant anomaly. The top part shows the raw HPC values, with a clear, sudden spike indicating the anomaly. The bottom part presents the corresponding loss curves for the forecasting and reconstruction

TABLE II: Illustrating and comparing detection performance of *instant anomaly type* among SOTA models with RENOWNED on SPMD dataset [31]

	CNN-KF (%)[31]			MSALSTM-CNN(%)[32]			WKN-OC(%)[33]			RENOWNED (%)(Ours)		
Anomaly Magnitude	Acc	Sens	Prec	Acc	Sens	Prec	Acc	Sens	Prec	Acc	Sens	Prec
base value+10*N(0,0.01)	74.2	48	96.4	76.4	51	97.2	94.7	60.2	94.8	**96.2**	**62.4**	**97.6**
base value+25*N(0,0.01)	80	51.5	97.6	84.1	54.6	98.1	96.5	**65.8**	96.7	**96.8**	65	**97.2**
base value+50*N(0,0.01)	88.1	72	97.2	90.2	75.9	97.9	98	73.5	98.4	**98.4**	**75.7**	**98.7**
base value+100*N(0,0.01)	93.5	86.2	97.9	95.8	89.6	98.4	**99.0**	90.7	98.8	**99.0**	**91.8**	**99.3**
base value+200*N(0,0.01)	94	90.3	98	96.02	93.5	98.3	**99.4**	94.6	99.2	99.3	**96.3**	**99.6**

TABLE III: Illustrating and comparing detection performance of RENOWNED with *mixed anomaly types* on SPMD dataset

		CNN-KF [31]		MSALSTM-CNN [32]		WKN-OC [33]		RENOWNED (Ours)	
Anomaly Magnitude	Sensors	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Instant, 25*N(0,0.01)	1	82.6	70.1	85	72.3	92.81	74.03	**96.79**	**81.62**
	2	89	66.24	91.4	79.65	95.64	81.37	**95.87**	**83.76**
	3	88	73.54	87.49	76.91	**96.5**	72.9	96.4	**76.21**
Constant, U(0,1), d=3	1	85.32	74.95	92.54	81.67	**91.57**	82.01	90.46	**84.93**
	2	91.82	80.97	93.84	78.97	93.87	**90.39**	**95.82**	88.36
	3	88.64	75.19	89.21	80.61	90.92	82.57	**94.32**	**84.09**
GD, linespace(0, 1), d=1	1	88.32	80.39	92.69	83.12	94.5	85.24	**96.37**	**87.43**
	2	90.82	79.58	93.21	80.78	92.65	83.17	**94.32**	**86.77**
	3	92.14	**85.36**	89.08	80.37	93.14	79.97	**97.39**	82.53
Bias, U(0, 2), d=5	1	93.24	86.12	95.32	84.21	92.34	86.32	**96.93**	**88.12**
	2	88.36	74.36	96.12	82.36	95.12	89.81	**97.65**	**90.37**
	3	92.81	86.27	95.36	90.47	94.35	86.57	**97.56**	**89.21**

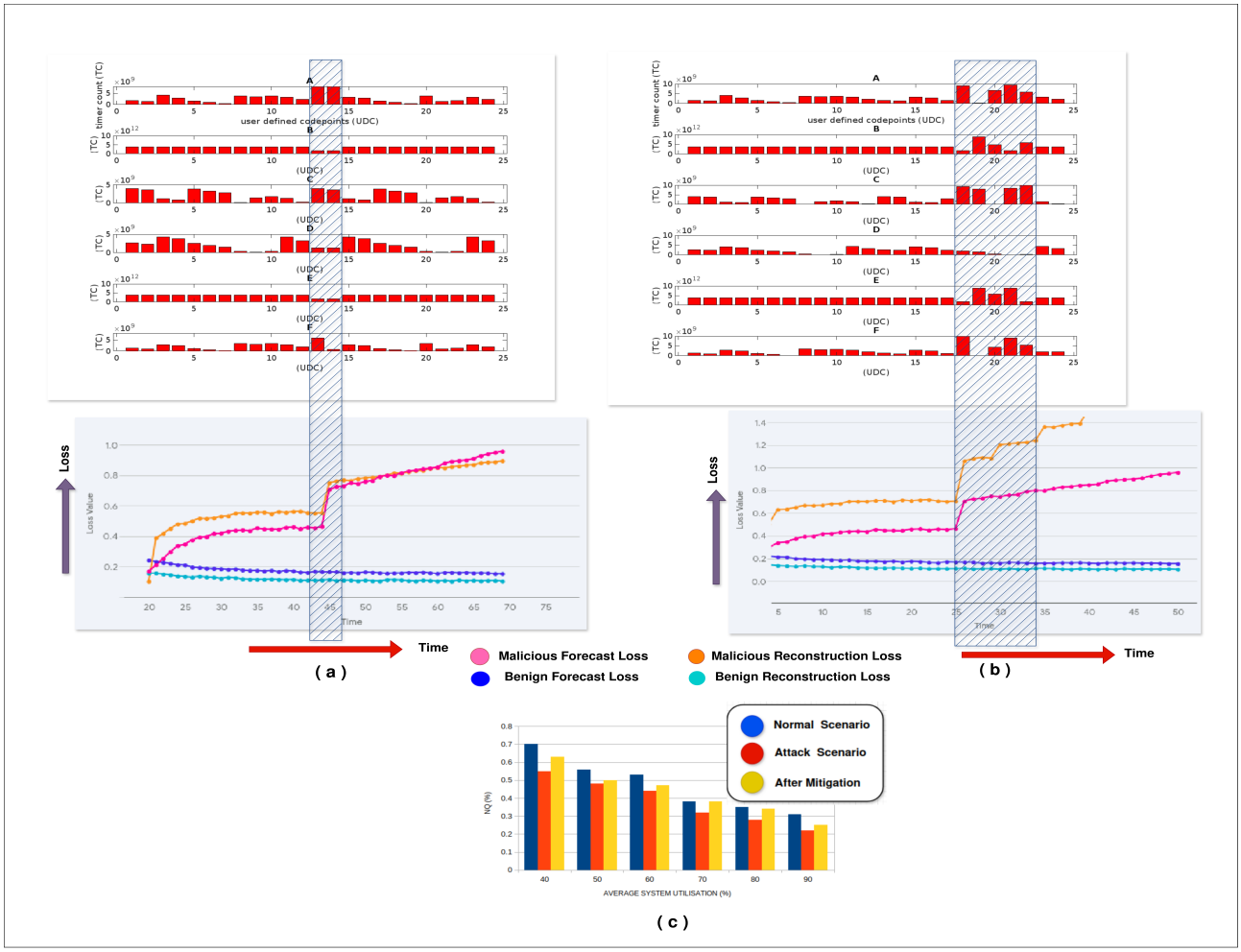


Fig. 9: (a) sudden increase in the loss curve for the reconstruction and forecasting models upon detecting an anomaly (instant anomaly type) (b) sudden increase in the loss curve for the reconstruction and forecasting models upon detecting an anomaly (constant anomaly type); displays the variation with time for forecasting and reconstruction module distinguishing between unknown benign and malicious programs. (c) Normalised QoS (NQ) variation in normal, attack and after mitigation scenarios against system utilisation percentages.

models. A dramatic increase in the loss is observed at the point of the anomaly. This happens because the AI models, trained on normal data, encounter an unexpected pattern, signifying an anomaly.

(b) represents the case of a constant anomaly injection as depicted by a sudden and consistent change in the HPC value and its corresponding change as reflected in the loss curve of the forecast and reconstruction model. This sudden change in the loss occurred since the AI models did not encounter this pattern of data from the HPCs (physical and virtual timer counts) during training, resulting from a deviation from its learned representation predicting this as an anomaly. Alternatively, the blue loss curves represent benign programs without any attack scenario.

(c) demonstrates the impact of anomalies on system QoS (measured as Normalized QoS - NQ) and the effectiveness of the mitigation strategy. The graph plots NQ against system utilization percentages. Under normal scenarios (blue bars), NQ gradually decreases with higher utilization. This is expected as higher utilization implies more tasks, leading to a reduction in the execution of optional components to meet deadlines and energy constraints. When an attack occurs (red bars), NQ further degrades due to disruptions caused by the anomaly. However, after mitigation through RENOWNED's rescheduling approach (yellow bars), NQ recovers significantly, demonstrating the system's ability to restore performance and maintain QoS even in the presence of anomalies.

2) Related to NQ and utilisation: To create task graphs, we have utilised task graphs for free (TGFF) [28]. The NQ is already computed in Equation 8 for a PTG. Again the summation of each subtask's execution times represents a PTG's total execution time expressed as T_{PTG}, and DL_{PTG} denotes the maximum allowable execution time of a given PTG. Here, we compute an average utilisation, which is the ratio of the T_{PTG}, and DL_{PTG} .i.e $U_t = T_{PTG}/DL_{PTG}$. In our experimentation, we have assumed to have a fixed given deadline DL_{PTG}. Therefore increasing the utilisation U_t increases the number of tasks and to accommodate an increased number of tasks (i.e. nodes in the PTG) the optional portion of every task has to be reduced, which decreases the QoS (and the NQ) and vice versa. While there is an attack, the QoS decreases and mitigating it by RENOWNED's new scheduling approach again increases the QoS as depicted in Fig. 9 (c). We have computed and analysed the NQ with increasing utilisation percentiles and it varies as shown. The maximum NQ reached is 64% and the least is 25% when utilisation in the x-axis is varied in the range from 40 to 90% (Fig. 9 (c)).

3) Limitations: A Few identified limitations regarding deployment, program selection, operating frequencies, threat and vehicle variety have been discussed. Porting RENOWNED to a new platform may require adjustments in the anomaly detection module to accommodate the specific characteristics of the available HPCs. Regarding benchmark program selection, further research is needed to explore systematic approaches and evaluate the impact of different benchmark suites on anomaly detection accuracy. More sophisticated dynamic frequency scaling techniques, such as DVFS, could be explored to further optimize energy consumption and performance. Expanding the threat model to encompass a wider range of attacks, such as network intrusion or data manipulation, would enhance the system's robustness. And finally expanding the evaluation to encompass a larger and more diverse set of vehicles would strengthen the generalisability of the results.

VI. Conclusion

Our proposed design RENOWNED tackles the challenge of ensuring the dependable and energy-efficient operation of Electronic Control Units (ECUs) in the Internet of Vehicles (IoV). RENOWNED's novel approach combines anomaly detection and task scheduling, allowing it to maintain the desired level of performance even when faced with unexpected disruptions. The AI-powered anomaly detection module, constituting the graph attention networks and variational autoencoders, demonstrates a remarkable ability to identify abnormal processor behaviour, achieving a notable accuracy rate of 97.6% in the fine granular hardware level. The system's scheduling mechanism further enhances its robustness by dynamically reallocating tasks in response to detected anomalies. The experimental results highlight RENOWNED's effectiveness, which still maintains an equivalence in energy consumption even after an attack, while still maintaining a Quality of Service (QoS) of 64% and the least is 25% when system utilisation is varied in the range from 40 to 90%. The system's ability to balance energy efficiency and real-time performance makes it a promising solution for safety-critical automotive applications in the IoV. We are exploring federated learning approaches for anomaly detection and creating a multi-layered defence strategy that could lead to even higher accuracy and faster response times. We believe the continued development and refinement of solutions like RENOWNED will be crucial in ensuring the safety and reliability of the increasingly connected and autonomous vehicles of the future.

Acknowledgment

This work is supported by the UK Engineering and Physical Sciences Research Council through grants, EP/X015955/1, EP/X019160/1 and EP/V000462/1, EP/V034111/1. For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) licence to any arising Author Accepted Manuscript version.

References

[1] C. Chen, Z. Liu, S. Wan, J. Luan, and Q. Pei, "Traffic flow prediction based on deep learning in internet of vehicles," *IEEE transactions on intelligent transportation systems*, vol. 22, no. 6, pp. 3776–3789, 2020.

[2] Z. Yu et al., "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Tran. on Intell. Trans. Sys.*, vol. 22, no. 8, pp. 5341–5351, 2020.

[3] M. Hasan, S. Mohan, T. Shimizu, and H. Lu, "Securing vehicle-to-everything (v2x) communication platforms," *IEEE Trans. on Intel. Vehicles*, vol. 5, no. 4, pp. 693–713, 2020.

[4] D. Cao et al., "Future directions of intelligent vehicles: Potentials, possibilities, and perspectives," *IEEE Trans. on Intelligent Vehicles*, vol. 7, no. 1, pp. 7–10, 2022.

[5] S. Saha et al., "Delicious: Deadline-aware approximate computing in cache-conscious multicore," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 2, pp. 718–733, 2022.

[6] M. Qamhieh, "Scheduling of parallel real-time dag tasks on multiprocessor systems," Ph.D. dissertation, Université Paris-Est, 2015.

[7] B. Hu et al., "Safety-guaranteed and development cost-minimized scheduling of dag functionality in an automotive system," *IEEE Trans. on Int. Trans. Sys.*, vol. 23, no. 4, pp. 3074–3086, 2020.

[8] B. Hu, Z. Cao, and M. Zhou, "Energy-minimized scheduling of real-time parallel workflows on heterogeneous distributed computing systems," *IEEE Trans. on Services Comp.*, vol. 15, no. 5, pp. 2766–2779, 2021.

[9] N. Irtija et al., "Energy efficient edge computing enabled by satisfaction games and approximate computing," *IEEE Trans. on Green Commun. and Net.*, vol. 6, no. 1, pp. 281–294, 2021.

[10] K. Cao et al., "Qos-adaptive approximate real-time computation for mobility-aware iot lifetime optimization," *IEEE Tran. on CAD of Int. Cir. and Sys.*, vol. 38, no. 10, pp. 1799–1810, 2018.

[11] X. Li, L. Mo, A. Kritikakou, and O. Sentieys, "Approximation-aware task deployment on heterogeneous multi-core platforms with dvfs," *IEEE Tran. on CAD of Integrated Circuits and Systems*, 2022.

[12] L. Mo, A. Kritikakou, and O. Sentieys, "Energy-quality-time optimized task mapping on dvfs-enabled multicores," *IEEE Tran. on CAD of Int. Circuits and Sys.*, vol. 37, no. 11, pp. 2428–2439, 2018.

[13] ——, "Approximation-aware task deployment on asymmetric multicore processors," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1513–1518.

[14] M. A. Haque, H. Aydin, and D. Zhu, "On reliability management of energy-aware real-time systems through task replication," *IEEE Tran. on Parallel and Dist. Sys.*, vol. 28, no. 3, pp. 813–825, 2016.

[15] Y. Wang, M. Karimi, Y. Xiang, and H. Kim, "Balancing energy efficiency and real-time performance in gpu scheduling," in *2021 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2021, pp. 110–122.

[16] S. Mahmood, H. N. Nguyen, and S. A. Shaikh, "Systematic threat assessment and security testing of automotive over-the-air (ota) updates," *Vehicular Communications*, vol. 35, p. 100468, 2022.

[17] T. Zhang, M. Tehranipoor, and F. Farahmandi, "Trustguard: Standalone fpga-based security monitoring through power side-channel," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2023.

[18] F. Ayaz et al., "A proof-of-quality-factor (poqf)-based blockchain and edge computing for vehicular message dissemination," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2468–2482, 2020.

[19] M. Abdelrehim et al., "Bic: Blind identification countermeasure for malicious thermal sensor attacks in mobile socs," in *2022 23rd Int. Symp. on Quality Electronic Design (ISQED)*. IEEE, 2022, pp. 1–6.

[20] T. Zhang, H. Antunes, and S. Aggarwal, "Defending connected vehicles against malware: Challenges and a solution framework," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 10–21, 2014.

[21] B. Salami et al., "Fairness-aware energy efficient scheduling on heterogeneous multi-core processors," *IEEE Tran. on Comp.*, vol. 70, no. 1, pp. 72–82, 2020.

[22] D. Senapati et al., "Hmds: A makespan minimizing dag scheduler for heterogeneous distributed systems," *ACM Tran. on Embedded Computing Systems (TECS)*, vol. 20, no. 5s, pp. 1–26, 2021.

[23] S. Rajapaksha, H. Kalutarage, M. O. Al-Kadri, G. Madzudzo, and A. V. Petrovski, "Keep the moving vehicle secure: Context-aware intrusion detection system for in-vehicle can bus security," in *2022 14th International Conference on Cyber Conflict: Keep Moving!(CyCon)*, vol. 700. IEEE, 2022, pp. 309–330.

[24] M. Almehdhar, A. Albaseer, M. A. Khan, M. Abdallah, H. Menouar, S. Al-Kuwari, and A. Al-Fuqaha, "Deep learning in the fast lane: A survey on advanced intrusion detection systems for intelligent vehicle networks," *IEEE Open Journal of Vehicular Technology*, 2024.

[25] S. Yang et al., "Robust and efficient star identification algorithm based on 1-d convolutional neural network," *IEEE Trans. on Aeros. and Elec. Sys.*, vol. 58, no. 5, pp. 4156–4167, 2022.

[26] H. Gao, B. Qiu, R. J. D. Barroso, W. Hussain, Y. Xu, and X. Wang, "Tsmae: a novel anomaly detection approach for internet of things time series data using memory-augmented autoencoder," *IEEE Trans. on network science and Eng.*, vol. 10, no. 5, pp. 2978–2990, 2022.

[27] A. Siffer et al., "Anomaly detection in streams with extreme value theory," in *Proceedings of the 23rd ACM SIGKDD Int. conf. on knowledge discovery and data mining*, 2017, pp. 1067–1075.

[28] K. Vallerio, "Task graphs for free (tgff v3. 0)," *Official version released in April*, vol. 15, 2008.

[29] T. Bauereiss et al., "Verified security for the morello capability-enhanced prototype arm architecture," in *European Symposium on Programming*. Springer International Publishing Cham, 2022.

[30] E. M. B. Consortium *et al.*, "The embedded microprocessor benchmark consortum," 2008.

[31] F. Van Wyk et al., "Real-time sensor anomaly detection and identification in automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1264–1276, 2019.

[32] A. R. Javed et al., "Anomaly detection in automated vehicles using multistage attention-based convolutional neural network," *IEEE Trans. on Int. Transp. Sys.*, vol. 22, no. 7, pp. 4291–4300, 2020.

[33] Z. He, Y. Chen, H. Zhang, and D. Zhang, "Wkn-oc: a new deep learning method for anomaly detection in intelligent vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 3, pp. 2162–2172, 2023.

Chandrajit Pal is currently associated with the Embedded and Intelligent Systems (EIS) Research Group, University of Essex, UK as a Senior Research Officer. Prior to that, he worked as a National post-doctoral fellow at IIT Hyderabad, India and as an AI Research Engineer at Ceremorphic Inc. His research interests mainly include computer vision and signal processing algorithms, custom computing using FPGAs, embedded systems and HW/SW co-design.

Sangeet Saha is currently associated with the Embedded and Intelligent Systems (EIS) Research Group, University of Essex, UK as a Lecturer. Prior to that, he worked as a lecturer at the University of Huddersfield, UK, and Senior research officer (Postdoctoral scholar) at the University of Essex, UK. His current research interests include real-time scheduling, scheduling for reconfigurable computers, real-time and fault-tolerant embedded systems, and cloud computing. He published several of his research contributions in conferences like CODES+ISSS, ISCAS, Euromicro DSD, and in journals like ACM TECS, IEEE TCAD, IEEE TSMC.

Xiaojun Zhai (SM'21) is currently a Reader in the Embedded Intelligent Systems Laboratory at the University of Essex. He has authored/co-authored over 140 scientific papers in international journals and conference proceedings. His research interests mainly include the design and implementation of digital image and signal processing algorithms, custom computing using FPGAs, embedded systems and hardware/software co-design.

Klaus McDonald-Maier is currently the head of the Embedded and Intelligent Systems Laboratory and director of research at the University of Essex, Colchester, U.K. He is also the founder of UltraSoC Technologies Ltd., the CEO of Metrarc Ltd., and a visiting professor at the University of Kent. His current research interests include embedded systems and SoC design, security, development support and technology, parallel and energy-efficient architectures, computer vision, data analytics, and the application of soft computing and image processing techniques for real-world problems. He is a member of VDE and a fellow of the BCS and IET.