Metascheduling Using Discrete Particle Swarm Optimization for Fault Tolerance in Time-Triggered IoT-WSN

Haytham Baniabdelghany, Roman Obermaisser, Ala' Khalifeh, and Pascal Muoka

***Abstract*—In time-triggered (TT) systems, adaptation is performed by using metascheduling approaches to ensure temporal predictability. Wireless sensor networks (WSNs) are growing to be used as a reliable, an energy efficient, and a scalable network infrastructure for numerous Internet of Things (IoT) applications. Because of run-time changes because of failure events, a metascheduling system is required to address the changes by precomputing schedules for several context events at design time. Therefore, this article proposes a metascheduler that solves a new scheduling problem for each adaptation scenario in IoT-WSN using our offline algorithm named discrete particle swarm optimization for reliable task allocation (DPSO-TA), resulting in a multischedule graph (MSG) that combines the repeated schedules. In this work, a TT IoT-WSN metascheduler is proposed that takes into considerations the node or link failures that are common in WSN. Single- and two-failure events are considered, where a graph for all feasible schedules is formed at the network design time. Such a large number of schedules causes a state space explosion problem, which is managed using a reconvergence method. Different application model sizes and network topologies are used to assess the proposed metascheduler under various failure event scenarios. The results show a reduction in the size of the MSG by applying inputs with different deadline values and different number of tasks. Furthermore, the validity suggested by the proposed metascheduler, which is the ratio of valid to invalid schedules, is improved when compared to algorithms that schedule tasks to hosts with the shortest completion time or algorithms that select hosts for a set of sorted tasks based on energy consumed and task arrival time.**

***Index Terms*—Metascheduler, task scheduling algorithm, wireless sensor networks (WSNs).**

I. Introduction

WIRELESS sensor networks (WSNs) are made up of distributed wireless hosts that monitor and control the physical surroundings. The collected data are then transported and forwarded until they reach a location for further analysis and decision making. WSNs have been used in a range of industries and services due to their flexibility [1].

The Internet of Things (IoT) is an emerging technology that allows a collection of physical devices with built-in sensors and software to connect and exchange data with other systems over the Internet. Many researchers have recently explored the idea of integrating WSNs with IoT (IoT-WSN), as WSNs can be used as a low-cost, a reliable, and a scalable backbone network infrastructure if properly designed and optimized. As a result, designing WSNs is critical for many IoT applications, as well as ensuring the reliability and efficiency of IoT-based applications and services [2].

In the context of IoT-WSNs, our prior work discrete particle swarm optimization for reliable task allocation (DPSO-TA) [3] addressed the distribution of sensing tasks among hosts and scheduling transmitted messages to find a solution that ensures all tasks are completed on time. Because of the runtime changes that result from system resource failure events, the IoT-WSN requires a scheduling-based system that addresses run-time changes by precomputing schedules for several context events at design time, which is known as metascheduling. Therefore, this article proposes a metascheduler to provide a predictable adaptation mechanism, ensuring that in the event of run-time modifications, transitions to a previously verified schedule are made. Using the application, the architecture, and the context models, the proposed metascheduler precomputes schedules at design time and forms a multischedule graph (MSG). The MSG represents schedules and context events as vertices and edges. Its size grows exponentially as the number of context events increases. For systems with limited storage space, the state-space explosion poses a challenge.

The major contributions of this article can be summarized as follows.

1) DPSO-TA [3] targets to find a schedule in a failure-free system. The computed schedule by DPSO-TA considers the timeliness and saving energy. However, WSNs are prone to node or link failures at the runtime. Therefore, a time-triggered (TT) IoT-WSN metascheduler that takes failures into consideration is provided to find feasible schedules using DPSO-TA. The advantage of the metascheduler is considering all possible single and two-failure events and forming a graph for all feasible schedules at the time of the design. This leads to better results concerning schedulability.

Manuscript received 6 November 2022; revised 20 January 2023; accepted 28 February 2023. Date of publication 6 March 2023; date of current version 7 July 2023. This work was supported by the University of Siegen. *(Corresponding author: Haytham Baniabdelghany.)*

Haytham Baniabdelghany and Roman Obermaisser are with the Faculty of Science and Technology, University of Siegen, 57076 Siegen, Germany (e-mail: haytham.baniabdelghany@uni-siegen.de; roman.obermaisser@uni-siegen.de).

Ala' Khalifeh is with the Communication Engineering Department, German Jordanian University, Amman 11180, Jordan (e-mail: ala.khalifeh@gju.edu.jo).

Pascal Muoka is with the Electrical Engineering and Computer Science, University of Siegen, 57076 Siegen, Germany (e-mail: pascal.muoka@uni-siegen.de).

Digital Object Identifier 10.1109/JIOT.2023.3252820

2) The proposed metascheduler uses a reconvergence mechanism for repeated schedules to address the state-space explosion problem in the MSG.
3) Prior work such as [4] provided a metascheduling adaptation mechanism for network-on-chip-based wire-bound systems. But the metascheduling was not solved so far for IoT-WSNs, where nodes and links failure are probable. Therefore, we extend the scheduling problem by introducing decision variables of wireless networks, combining a wireless channel model with the metascheduling problem.
4) The proposed metascheduler assesses reconvergence using different input model sizes, network topologies, and task deadlines.
5) The DPSO-TA-based metascheduler is compared against other metaschedulers that rely on less efficient task allocation algorithms.

The remainder of this article is structured as follows. The related work is discussed in Section II, Section III demonstrates the optimization problem. The system model is described in Section IV. The DPSO-TA-based metascheduling technique that handles the repeated schedules is discussed in Section V. Section VI discusses the simulation setup and results. Finally, Section VII draws the conclusion and future work.

II. Related Work

Efficient metascheduling algorithms at design time are essential for generating precomputed schedules that satisfy the desired constraints of IoT-WSNs, such as task deadlines, energy savings, and signal interference avoidance, particularly in the face of unpredicted disturbances or unexpected failures. Muoka et al. [4] developed a metascheduling algorithm for design time to manage slack events that may occur at runtime. The algorithm repeatedly invokes a genetic algorithm (GA) to solve the scheduling problems. The work in [5] described the TT multicore architecture supports adaptation to multiple schedule graphs, while preserving key characteristics of TT architecture, such as error containment, implicit flow control, and freedom from overlapping shared resources. The mentioned research provides path reconvergence to overcome the problem of state-space explosion. But the research is limited to multicore architectures based on Networks-on-Chip (NoC), thus ignoring WSN restrictions, such as signal interference and several users transmitting at the same time.

Some earlier research was focused on minimizing task completion time or energy consumption. For example, Gong et al. [6] proposed an algorithm that meets the deadline requirements by providing dynamic packet scheduling functions and reliable graph routing methods, whereas Yu et al. [7] improved task scheduling for WSNs by applying efficient clustering methods to conserve energy upon faults.

Based on GA, Jin et al. [8] proposed an adaptable intelligent task allocation scheme (ITAS). By distributing workloads across different hosts and ensuring application deadlines. A hybrid fitness function extends the lifespan of a WSN. Furthermore, by using a TDMA-based media access control (MAC) technique, communication interference is prevented. The proposed approach, on the other hand, is implemented in an error-free environment and the GA may become trapped at the local optimum.

Yang et al. [9] proposed a binary version of PSO for WSN tasks' scheduling. Although the suggested algorithm considers numerous criteria, such as task completion time, energy savings, and network life, it only works in error-free systems and does not include message scheduling or routing mechanisms to ensure that tasks are completed before deadlines. Similarly, to reduce task completion times, Izakian et al. [10] applied the DPSO technique for error-free task scheduling in network contexts. In contrast, Guo et al. [11] used a distinct PSO algorithm to support fault tolerance for WSNs. The goal of this research is to develop a solution for the event of a node failure by employing a primary/backup strategy to tolerate the failed nodes. The mentioned discrete PSO algorithm considers the amount of energy used and ensures that each task is completed before the deadline. However, the suggested algorithm ignores the precedence constraints, as well as the communication cost. Active replication is used in the work given in [12] to address fault tolerance and task deadlines.

III. Optimization Problem

The DPSO-TA-based metascheduler introduced in this article provides contributions that go beyond the mentioned state-of-the-art. In our prior work [3], DPSO-TA optimizes a schedule by assigning tasks to hosts and scheduling messages depending on the minimum global makespan, the total energy consumption, and the overall failure rates. DPSO-TA provides communication fault tolerance by sending multiple copies of messages across redundant paths [13]. Furthermore, the network load is also balanced by DPSO-TA utilizing a load-balancing method. The task and message scheduling model takes into account the following constraints.

1) *Resource Scheduling Constraint:* Task execution is restricted to a single host.
2) *Path-Attribute Constraint:* Each TT message is only permitted to transit via a relay node, once on its way to the receiving host in order to prevent routing loops.
3) *Physical Interference Constraint [14]:* Each TT message's designated time slot can only use a specific path provided that the interference caused by other messages does not fall below a predefined threshold.
   To illustrate, we assume that a message using the path (a, b) will be received correctly at node b if and only if the following condition is satisfied:

$$\frac{P_{rcv}(a,b)}{N+\sum_{p\in \text{Paths}} P_{rcv}(c,b)} \geq \beta \tag{1}$$

   where $P_{rcv}(a, b)$ represents the received power on b from a, $P_{rcv}(c, b)$ represents the interference on b by c. The ambient noise power is N, *Paths* is a subset of wireless paths used by other communication messages simultaneously with path (a, b) to the same receiver b. β is the SINR threshold value, which ensures that the message will be successfully received at b. Thus, path (a, b) is

inserted in a specific time slot with other *Paths* if the computed value on the left hand side of Inequality (1) is equal to or more than β. Otherwise, the message that uses path (a, b) uses the next time slot that satisfies this condition.

4) *Precedence Constraint:* Each task t is ready to be executed at time t_{rt} after the arrival of all messages M_t sent from its parent tasks. The ready time of task t is equal to the last arrival time. It is expressed as shown in

$$t_{rt} = \max_{m \in M_t}(m_{\text{arrival}}). \tag{2}$$

m_{arrival} is defined as the instant of time when a message m from a parent task arrives at task t. It is defined as

$$m_{\text{arrival}} = m_{\text{IT}} + m_{\text{e2eD}} \tag{3}$$

where m_{IT} and m_{e2eD} denote the injection time and the transmission time of the message m, respectively.

5) *Message Period Constraint:* DPSO-TA takes the period of the associated tasks into account. The period of the messages that use the same path is considered by the DPSO-TA when allocating time slots to transmit each message.

6) *Deadline Constraint:* The instant of time for the host to finish executing the task t after receiving all messages must be less than the deadline for that task

$$t_{rt} + t_{et} \leq t_{dl} \tag{4}$$

where, t_{et} and t_{dl} refer to the execution time and the deadline of a task t, respectively.

This article extends the DPSO-TA algorithm and integrates it into a metascheduler for IoT-WSNs. A metascheduler based on DPSO-TA is presented as a predictable adaptation mechanism to guarantee that transitions to a previously validated schedule are made in the event of run-time events. The proposed metascheduler computes schedules in advance at the time of design and produces an MSG as a result. A reconvergence strategy for repeated schedules is used to resolve the state-space explosion problem in the MSG.

IV. System Model

This section describes the application, the architecture, and the context models that are input to our metascheduler as shown in Fig. 1.

A. Architecture Model

The architecture model is represented by the tuple $<N, E_l>$, which is an undirected graph denoted G_{Arc}. As depicted in Fig. 1, each vertex $n \in N$ represents a network node, which is either a wireless host $h \in H$ or a router $r \in R$, where $N = H \cup R$. An access point (ap) or a wireless router (wr) is assigned to each router. Aside from operating as a gateway in wireless hosts, the access point functions as a wireless router. A bidirectional wireless link l between nodes is known as an edge $e_l \in E_l$, where E_l are all available wireless links in the architecture model.

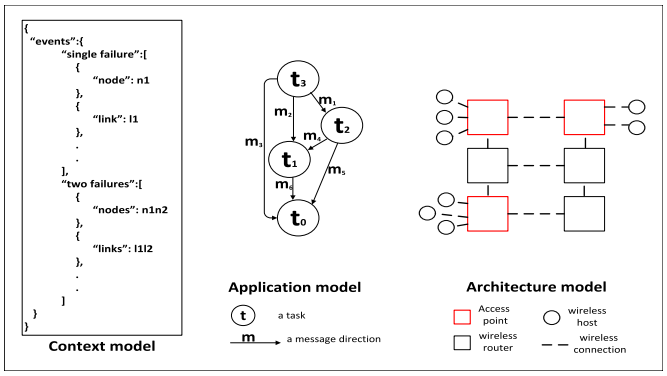

Fig. 1. Example of a system model.

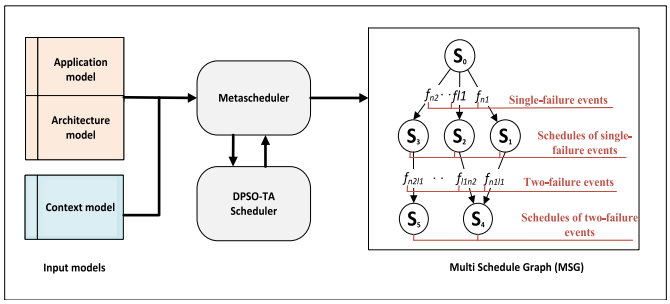

Fig. 2. Metascheduling overview.

B. Application Model

Fig. 1 shows an application model that is a directed graph denoted G_{App}, represented by the tuple $<T, E_m>$. A task is defined for each vertex $t \in T$, where T are all available tasks in the application model. E_m denotes a communication message that is communicated between two or more tasks. Each task t is represented by the tuple $<ET_t, D_t, P_t>$, where ET_t is the task's execution time, D_t is the task's execution deadline, and P_t is the task's period. A tuple $<S, R, RT_m, Tr_m>$ represents an edge $e_m \in E_m$, with S and R defining the sender and recipient tasks of a message, respectively. The message routing time is RT_m, and the time necessary to transmit the message is Tr_m.

C. Context Model

F is a set of single- and two-failure events. A permanent fault of a computing or communication resource is considered as a failure event $f \in F$ of node n or link l. All context events are given in the form of a set of events that contains all predefined possible failure events that may occur to network components (i.e., links and nodes).

V. Proposed Metascheduler

Fig. 2 shows that the application, the architecture, and the context models that are used as input models to the metascheduler to compute the MSG. The MSG is a directed acyclic graph of precalculated TT schedules that is created during the design time. The TT system is located in one of the states of the MSG at any given time while it is running. This state specifies how all computing and communication resources are distributed both temporally and spatially. The state is left when

an event from the context model occurs, and the system then moves to another state in the MSG.

The metascheduler frequently computes the MSG using the DPSO-TA scheduler that we introduced in [3]. DPSO-TA uses input models for the application (G_{App}) and architecture (G_{Arc}) to produce a TT schedule that complies with the scheduling specifications (e.g., avoiding collisions, deadlines, and precedence limitations). The decision variables include task distribution among hosts, task start times, message paths through routers, message injection timings. The preference for choosing a schedule depends on the minimum energy consumption, the failure rates, and the completion time when all tasks are finished.

Initially, assuming no failure event, a base schedule (S_0) is generated using an initial G_{Arc} and G_{App} models. The metascheduler then applies the first event from the context model that includes all potential failure events, resulting in a modified architecture model. The metascheduler calls the DPSO-TA, gets a new schedule (e.g., S_1) and adds it to the MSG if it is considered a valid schedule. The schedule is valid if all tasks are executed before their deadlines, otherwise, the schedule is considered invalid. In the MSG, the failure event f, for example, f_{n2} that causes S_3 is represented by the appropriate edge. The metascheduler is terminated after all events in the predefined context model have been processed, and the MSG is created. The metascheduling method and the adaption of the DPSO-TA algorithm to address TT scheduling problems are discussed in the sections.

A. DPSO-TA

A population-based stochastic method called DPSO [15], which is motivated by the cooperative behavior of a swarm of birds, is used for continuous optimization in discrete problems. Based on the performance of DPSO, DPSO-TA proposes a hybrid DPSO algorithm that schedules tasks and messages in IoT-WSN using the task assignment map, the message paths, and the quality metrics as a fitness value.

The DPSO-TA, designed for a TT wireless architecture, is described in Algorithm 1 (Fig. 3). The application and architecture models are used as inputs to the algorithm. To begin, a set of particles is created. For each particle (p_i), random position (X_i) and velocity (V_i) are initialized, and each particle moves in a direction to explore for a better position across iterations. Its location in space is determined by its position, while its movement and direction are determined by its velocity. During the iterations, each particle adjusts its position in space based on its velocity. In addition to the previously mentioned variables, each particle has two best values. P_{best_i} of a particle p_i, which is initialized as X_i at the beginning and presents the best local position value for that particle so far. The fitness value of the particle's position (fit(X_i)) is used to update P_{best_i}. The second-best value is G_{best}, which is the swarm's best global position so far. In every iteration, G_{best} and P_{best_i} are verified, the P_{best_i} value of any given particle is altered at iteration $k+1$ if fit(X_i) value is better than fit(P_{best_i}) value in the previous iteration k. Similarly, the G_{best} value is altered if $fit(X_i)$ value is better than the fitness value of G_{best} (fit(G_{best})).

Algorithm 1 DPSO-TA algorithm

Input: G_{Arc}, G_{App}
Output: the best task scheduling solution (G_{best})

```
for each particle p_i do
    Initialize position X_i randomly
    Initialize velocity V_i randomly
    Initialize P_best_i with a copy of X_i
end for
for each particle p_i do
    Evaluate fit(P_best_i)
    Initialize G_best with a copy of P_best_i with the best fitness
end for
while k < Max_k do
    for each particle p_i do
        Evaluate fit(X_i) using Eq. (5)
        if fit(X_i) < fit(P_best_i) then
            P_best_i ← X_i
        end if
        if fit(X_i) < fit(G_best) then
            G_best ← X_i
        end if
    end for
    for each particle p_i do
        Update V_i according to Eq. (6)
        Update X_i according to Eq. (7)
    end for
    Next iteration (k++)
end while
```

Fig. 3. DPSO-TA algorithm.

In DPSO-TA, the particle's position (X_i) represents a solution for scheduling tasks to random hosts. The fitness value of a position (fit(X_i)) is used to determine the particle's solution and is determined by

$$\text{fit}(X_i) = \theta * \text{makespan} + \gamma * \text{Ene}_T + \delta * FR_T \tag{5}$$

where *makespan* is the time when all tasks are finished, Ene_T and FR_T denote the consumed energy and the failure rates as a result of scheduling all tasks T, respectively. While θ, γ, and δ are weight parameters.

To do this, every task is scheduled to a host after receiving all messages delivered by all previous tasks using fault-tolerant routes [13]. Using the physical interference model [14], the time slot message scheduler is utilized to schedule message traffic along conflict-free time slots. As a result of the proposed approach and after all tasks are scheduled, DPSO-TA calculates the completion time, the energy consumption, and the failure rates that are used as inputs into (5) to calculate the fitness of the scheduling solution. After a certain number of iterations (Max_k), DPSO-TA calculates the best task-scheduling solution (G_{best}) based on the best (i.e., lowest) fitness value for all particles.

Equations (6) and (7), which are used to update the best values of each particle p_i and the best global position in the swarm are shown as follows:

$$V_i^{k+1} = w \cdot V_i^k + c_1 \cdot r_1 \cdot \left(P_{\text{best}_i}^k - X_i^k\right) + c_2 \cdot r_2 \cdot \left(G_{\text{best}}^k - X_i^k\right) \tag{6}$$

$$X_i^{k+1} = X_i^k + V_i^{k+1}. \tag{7}$$

Equation (6) computes a new velocity V_i^{k+1} for each particle at iteration $k+1$ according to its previous velocity V_i^k

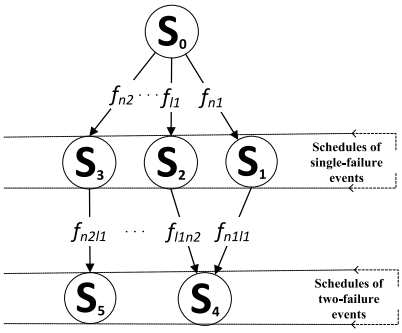

Fig. 4. Example of an MSG with reconvergence of repeated schedules.

at iteration k. The particle's position at which the best fitness value has been achieved for a particle so far is continuously updated in P_{best}. The global best position at which the best fitness value has been achieved so far in the swarm is continuously updated in G_{best}. Equation (7) updates the particle's position (i.e., X_i^{k+1}) at iteration $k+1$ by adding the new velocity resulting from (6) to its previous position at iteration k. r_1 and r_2 are random values, c_1 and c_2 are factors used to determine the acceleration and the pull for each particle in the swarm toward P_{best} and G_{best} values, w is an inertia weight, and it is decreased in every iteration.

B. Metascheduler With Reconvergence of Repeated Schedules

The suggested metascheduler provides schedule adaptation in response to each failure event $f \in F$. For example, as shown in Fig. 4, for the case that a single-failure (e.g., f_{n1}) or two-failure (e.g., f_{n1l1}) event occurs, a new schedule is generated to allow adapting to the failure event. The metascheduler calculates these new schedules, resulting in an MSG covering the occurrence of the specified failure events. The output MSG is used to determine the possibility of switching schedules in the event of a failure event during the operation of the TT IoT-WSN.

The metascheduler in Fig. 5 is used to build an MSG by continually invoking the DPSO-TA. To begin, the original system model (i.e., G_{Arc} and G_{App}) is utilized to create a base schedule S_0, which is then added as the MSG's root node. A set of events for failures is calculated using the formula in (8), where ${}^{C}Co_f$ is the number of the failed component combinations, f is the number of failed components, and C is the total number of components in the network, where a component represents a wireless node or link. For example, if the total number of components in an IoT-WSN is 20 (i.e., the total links and nodes are 20), and the number of failed components is 1 (i.e., $f = 1$), the number of failed component combinations is $[20!/(1!(20-1)!)] = 20$. In the same way, for $f = 2$, the number of failed component combinations is 190. Two sets (Set_1 and Set_2) are built in this work, one for single-failed component and the other for two-failed component combinations. The existing sets are merged into a single event set to index all combinations beginning with Set_1, which was created from S_0, where all events at Set_1 are defined from S_0

$$ {}^{C}Co_f = \frac{C!}{f!(C-f)!}. \tag{8}$$

The initial failure event is picked and deleted from the set in the metascheduling process, and it is used to change the original G_{Arc} to G'_{Arc} in the case of a failed link (f_l) or failed node event (f_n). The modified system model (m') is then sent to DPSO-TA to solve the new scheduling problem.

The metascheduler checks if the computed schedule (S') is valid. If all tasks are completed before their deadlines, the schedule is regarded to be valid. The metascheduler calls the next event in the event set if the schedule is invalid.

Each valid schedule is compared to the MSG's existing schedules. If the MSG contains repeated schedules, the schedules are converged, and a new transition (i.e., edge) is constructed from the previous schedule (i.e., previous state). If the schedule does not exist, a new transition is produced from the previous schedule, and it is added to the MSG. The metascheduler calls the next event, and the process is repeated. The metascheduler begins calling the first of the two-failure combination sets after processing all failures in the single-failure combination set. After calling all events, the MSG is completely formulated, and the number of valid, invalid, and stored schedules is calculated. Thus, convergence schedules that exist on multiple schedule tree routes merge the transitions and then reduce the size of the MSG.

VI. Simulation Setup and Results

We ran evaluation experiments on a ThinkPad laptop (Intel I5 CPU) with 24 GB of memory using the DPSO-TA-based metascheduler. The Stanford network analysis platform (SNAP) library [16] was used to generate random forest fire-directed graphs to obtain variable system models. These graphs feature a variety of interflow dependency patterns, and they were used for all experiments in separate runs.

Grid and ring topologies are used in our experimental setup, as shown in Fig. 6. Each topology has ten routers (i.e., wireless routers and access points). Each access point establishes a local network with one to four fixed wireless hosts. The weight parameters (θ, γ, and δ) of (5) are set to 0.1, 0.45, and 0.45, respectively, to give more importance to *makespan* when finding a schedule. The TT task deadline is 1500 ms. The communication message size is either 100 or 200 bytes and the wireless network data rate is set to 270 kB/s, so each message takes either 3 or 6 ms to transmit. The task execution time is between [200, 250] ms, and the message routing time is between [20, 25] ms. The initial value of w in (6) is set to be 0.9, c_1 and c_2 equal to 2. r_1 and r_2 are randomly selected from a range of [0, 1]. Max_k is set to be 30.

PSO and GA are all examples of heuristic optimization techniques to solve complex optimization problems. Wihartiko et al. [17] showed that the PSO algorithm is better than GA in terms of the accuracy, the number of iterations, and the ease of the techniques employed to discover an optimized solution. Differences can be observed in large scale. PSO offers high computational accuracy and is simple to implement. Additionally, GA takes longer to execute than PSO which converges more quickly because PSO can be implemented without a lot of parameters and operators. While GA primarily

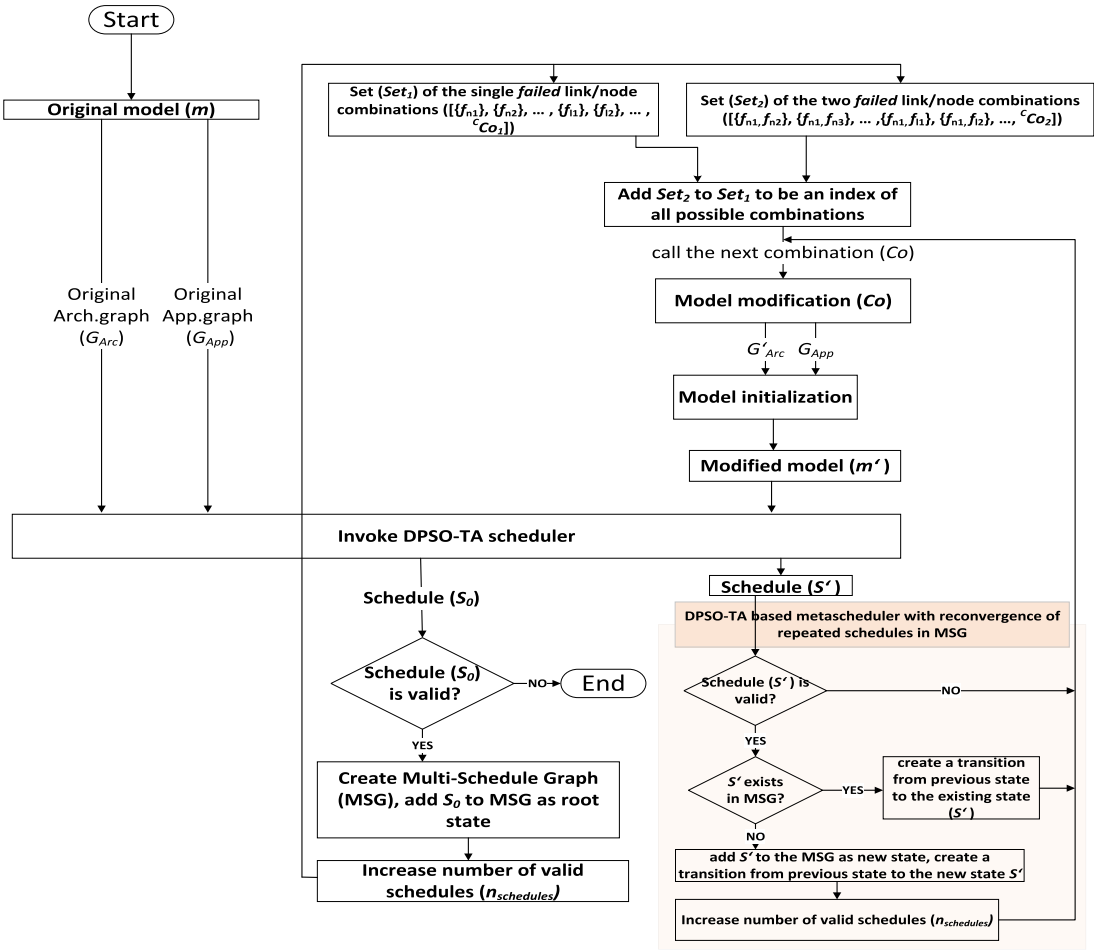

Fig. 5. Flowchart of the DPSO-TA-based metascheduler.

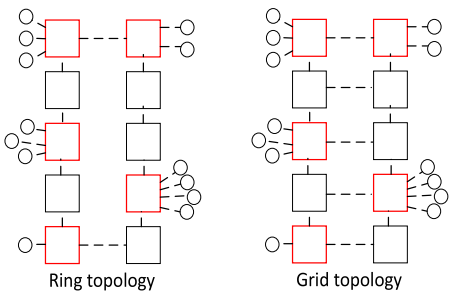

Fig. 6. Topologies used in the experimental tests.

focuses on global search, PSO simultaneously conducts local and global search.

Finding an effective solution in a setting prone to scaling up by the size of the input structure is addressed in this article. In addition to the scheduling solutions that grow significantly depending on the task, the messages input, and the number of hosts, the scheduling solutions become more challenging due to the high failure events encountered in WSN architecture. To obtain accurate solutions that generate a metascheduling table in the shortest period and with the least amount of convergence, the DPSO-TA algorithm is superior to GA, therefore, it is more suitable for this type of problem. Hence, DPSO-TA is used as a measurement reference for comparison. Reliable task allocation (RTA) [18], shortest task first allocation (STFA) [3], and min–min task allocation (MMTA) [3] algorithms were selected for the comparison to show how better the selected optimization technique (e.g., DPSO-TA) than other conventional algorithms in generating and designing metascheduling tables. The comparison is made based on the following considerations.

1) In RTA, the host's lowest fitness value is used to allocate a task from a sorted list. The value varies according to the makespan, the failure rate, and the energy consumed when finding a task-scheduling solution. The comparison is performed to show how DPSO can improve better solutions depending on the movement and intelligence of the swarms.
2) The MMTA scheduling algorithm is a straightforward algorithm that effectively manages the resources.
3) The STFA algorithm is one of the best scheduling strategies for reducing waiting times.

The primary distinction between DPSO-TA and RTA is that RTA does not support DPSO. When all the incoming messages, which are scheduled on time slots and sent to redundant routes, have been received, a host is selected for a listed task in RTA. Then RTA proceeds with the next task until all tasks have been scheduled, whereas DPSO-TA uses DPSO to compare preselected solutions until the best potential solution is

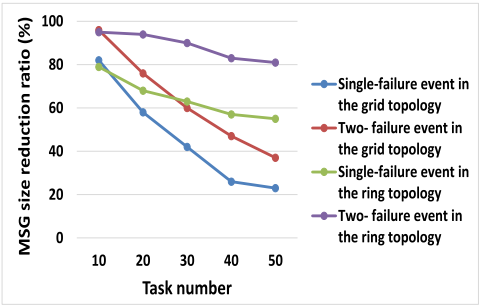

Fig. 7. MSG size reduction ratio in relation to increasing task numbers.

found. Meanwhile, to reveal the performance improvements gained by considering the communication cost as a result of time slot message scheduling, we apply MMTA and STFA-based metaschedulers that do not consider the cost due to the message transmissions. MMTA is a conventional heuristic method that computes the top-level of all tasks first, then it sorts them according to the task with the lowest top-level as the priority [19]. The algorithm then schedules the sorted tasks to the hosts with the shortest completion times, where completion time is defined as the time when the task execution time is added to the host's ready time. The technique is repeated for the next task until all tasks have been scheduled. STFA is a nonpreemptive algorithm that schedules tasks to hosts based on the shortest execution time, with the host who completes the task in the shortest amount of time receiving the highest priority [20]. For the comparison to be done properly, the physical interference model and fault-tolerant routes implemented in DPSO-TA are applied in all compared algorithms.

A. Evaluation of the MSG Size Reduction in Relation to Increasing Task Numbers

This section shows the efficiency of the proposed DPSO-TA-based metascheduler in the reconvergence of repeated schedules in the MSG. In the ring and grid topologies with 1500 ms deadline, Fig. 7 demonstrates the ratio of reducing the size of the MSG when increasing the number of tasks. To show how MSG removes repeated schedules, the reduction ratio is defined in

$$\text{MSG size reduction ratio} = 1 - \frac{\text{MSG size}}{\text{number of valid schedules}} \times 100\%. \quad (9)$$

When two-failure events are evaluated, the decrease ratio is larger than when single-failure events are considered for both topologies. In the case of two-failure events, the ratio jumped to 46% compared to single-failure events in the grid topology. Similarly, in the ring topology, the ratio increased to 38%. The reason for this observation is that as the number of failed component combinations grows, so does the number of similar schedules. While the reduction ratio in the ring topology is higher than in the grid topology, this is due to the structure of the rings and the limitation of available paths compared to the grids, which results in the similarity of many valid schedules, even when there are failures.

TABLE I
MSG SIZE REDUCTION RATIO IN RELATION TO INCREASING DEADLINE VALUES

Deadline in ms	1,100	1,300	1,500	1,700	1,900
Single-failures in grid	50%	50%	70%	67%	53%
Two-failures in grid	59%	65%	73%	72%	75%
Single-failures in ring	94%	84%	46%	65%	74%
Two-failures in ring	98%	98%	93%	94%	96%

Fig. 7 demonstrates that when the number of tasks increases, the MSG size reduction ratio decreases. The reason for this observation is that as the number of tasks to be scheduled grows, it becomes increasingly difficult to find symmetrical schedules.

B. Evaluation of the MSG Size Reduction in Relation to Increasing Deadline Values

For 30 tasks distributed in the ring and grid topologies, the MSG size reduction ratio is shown in Table I against the task deadlines in milliseconds. All tasks of the application model have the same deadline, which varies between 1100 and 1900 ms. The size of the MSG is greatly reduced when the ring and grid topologies are reconverged for single- and two-failure events. The MSG size is reduced while keeping a valid schedule.

When the task deadline is fixed, the lowered ratio for two-failure events is found to be larger than for single-failure events. Similarly, ring topologies have a larger ratio than grid topologies. It is assumed that as deadlines increase, valid schedules increase as well, raising the MSG size reduction ratio. However, if an increase in deadlines leads to a considerable increase in MSG size compared to an increase in valid schedules, the MSG size reduction ratio will be smaller.

C. Effect of Task Load on Validity

We ran a series of tests to see how task load and network topology affect the metaschedueler, which is based on DPSO-TA, GA, RTA, MMTA, and STFA. In the grid and ring topologies in Fig. 8, increasing the number of tasks has an impact on the validity, where the validity formula is defined in (10). The number of tasks has been increased from 10 to 50, with a step size of 10 and a deadline of 1500 ms

$$\text{Validity} = \frac{\text{number of valid schedules}}{\text{number of valid and invalid schedules}}. \quad (10)$$

Due to limited network resources and host capability, increasing the number of tasks reduces the task's capacity to be completed before its deadline, resulting in an increase in the number of invalid schedules and a decrease in the validity of all compared algorithms.

The task scheduling process to hosts is complex and depends on the mechanism and the number of messages flowing from the parent tasks. Moreover, differences in the structure of application models may ignore communication

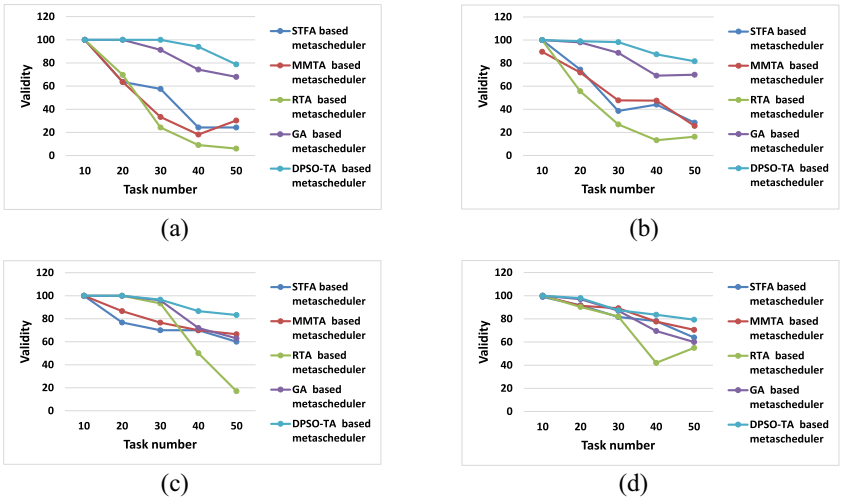

Fig. 8. Effect of increasing task numbers on validity. (a) Single-failure/Gird topology. (b) Two-failure/Gird topology. (c) Single-failure/Ring topology. (d) Two-failure/Ring topology.

costs by scheduling more than one task on the same host, which may lead to higher validity with a larger number of tasks. The MMTA in Fig. 8(a), for example, is observed to schedule 50 tasks with higher validity than 40 tasks. In the same way, RTA in Fig. 8(d).

When single-failure events are present, a DPSO-TA-based metascheduler outperforms those based on GA, STFA, MMTA, and RTA by 9%, 44%, 49%, and 57%, respectively, in the grid topology. With two-failure events, it also demonstrates better validity than GA, STFA, MMTA, and RTA by 7%, 40%, 41%, and 56%, respectively.

In the ring topology when single-failure events are present, a DPSO-TA-based metascheduler outperforms those based on GA, STFA, MMTA, and RTA by 8.3%, 32%, 14%, and 24%, respectively. It also outperforms GA, STFA, MMTA, and RTA in terms of validity, by 8%, 7%, 4%, and 18%, respectively, with two-failure events.

Because it continuously updates the solution by updating the best local position achieved by each particle and the best global position gained by all particles, the metascheduler based on DPSO-TA produces better results. A similar fitness algorithm is utilized in RTA, but it selects an appropriate host until all the tasks are completed. RTA does not assess and evaluate different solutions in the same way that DPSO-TA does. MMTA and STFA do not profit from message routing during the message scheduling process and cannot provide the individual advantage of nodes.

As mentioned, DPSO-TA has a greater ability than GA to delay convergence with local minima and thus the solutions provided are of higher quality.

Due to its efficacy in regulating parameters and quickness in discovering solutions, the metascheduler based on DPSO-TA provides high validity stability with single- or two-failure events in the ring or grid topologies besides its overall effectiveness. Although the metascheduler based on DPSO-TA in Fig. 8 has a smaller benefit in the ring topology than in the grid topology, this is because there are few valid solutions, especially when there are several failures.

D. Effect of Increasing Deadline Values on Validity

In the grid and ring topologies, increasing the value of deadlines in Fig. 9 has an impact on the validity. The number of tasks has been increased to 30, and the deadline has been extended from 1100 to 1900 ms with a 200-ms step size.

When single-failure events are present, the DPSO-TA-based metascheduler outperforms those based on GA, STFA, MMTA, and RTA by 13%, 76%, 62%, and 62%, respectively, in the grid topology. With two-failure incidents, it also demonstrates 19%, 70%, 47%, and 65% better validity than GA, STFA, MMTA, and RTA, respectively.

In the ring topology when single-failure events are present, a DPSO-TA-based metascheduler has better validity than those based on STFA, MMTA, and RTA by 38%, 22%, and 1%, respectively. With two-failure events, it also outperforms STFA and RTA by 6% and 4%, respectively, in terms of validity. When the deadline starts at 1500 ms, DPSO-TA shows better validity than MMTA.

In the metascheduler based on DPSO-TA, if a task in a solution misses its deadline, that solution is considered an unsuccessful task assignment and then is skipped when the global best solution is constructed. The missing deadline clause is adopted by the RTA, although it applies to each task separately. This means that scheduling a task to a host in RTA may lengthen the time required for outgoing communications to reach the following child task, causing the next task to miss its deadline. The MMTA and STFA work without regard for the requirement of a deadline. DPSO-TA is able to find solutions for different values of deadlines.

We note that in ring topologies, DPSO-TA does not outperform GA, particularly when there are multiple failures. The rationale is that as the number of network failures rise, there are fewer and more constrained feasible solutions that may be found in ring topologies employing both algorithms.

VII. Conclusion and Future Work

A metascheduler that computes an MSG is provided in this article. This metascheduler solves scheduling problems for TT

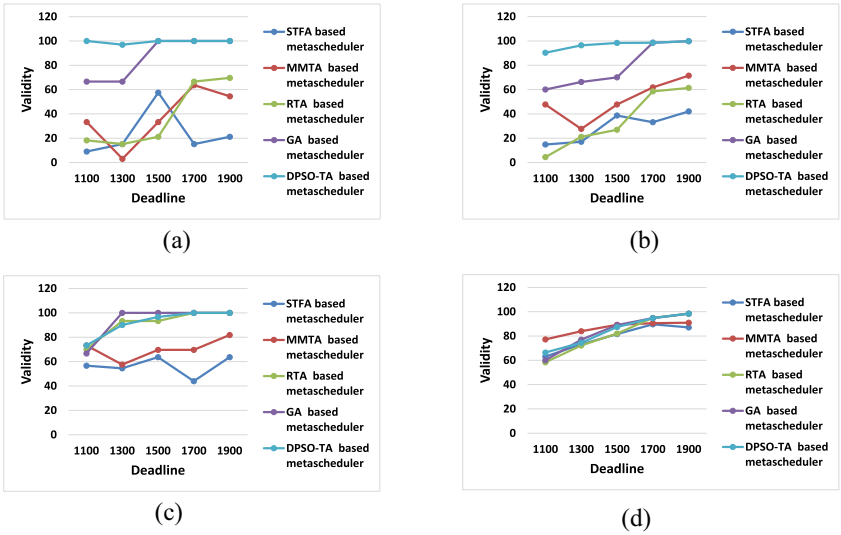

Fig. 9. Effect of increasing deadline values on validity. (a) Single-failure/Gird topology. (b) Two-failure/Gird topology. (c) Single-failure/Ring topology. (d) Two-failure/Ring topology.

IoT-WSNs by using the DPSO-TA algorithm. The proposed metascheduler controls the state-space explosion of the resulting multischedule diagram because of rising failure events by merging repeated schedules.

The ratio of MSG size decrease to task growth is shown in the results. In a grid topology, for example, two-failure events lower the MSG size by 46%, and in a ring topology, the MSG size is reduced by 38%. The decrease ratio versus deadline rise is higher for two-failure events than for single-failure events, according to the observations. Similarly, ring topologies have a larger ratio than grid topologies.

GA, STFA, MMTA, and RTA algorithms are used for a comparison of metaschedulers. The suggested metascheduler has up to 57% improved validity against increasing task numbers with single-failure events in a grid topology, and up to 56% with two-failure events. In a ring topology, the suggested metascheduler improves validity by up to 32% for single-failure events and up to 18% for two-failure events. Furthermore, in a grid topology, the suggested metascheduler has up to 76% greater validity against increasing deadline values with single-failure events, and up to 70% with two-failure events, according to the results. In a ring topology, the suggested metascheduler improves validity by up to 38% for single-failure events and by up to 6% for two-failure events.

We intend to adopt a primary/backup approach in future work to provide fault-tolerant task scheduling, which is used to run tasks on backup systems in the event that the primary hosts fail. It is also required to include different traffic types [such as TT, audio/video (AV), and best effort (BE)] and several fault categories (intermittent and transient faults) at the same network medium to study the impact of scheduling all traffic and to address the fault categories on the validity and size of the MSG.

REFERENCES

[1] A. Khalifeh et al., "Wireless sensor networks for smart cities: Network design, implementation and performance evaluation," *Electronics*, vol. 10, no. 2, p. 218, 2021.

[2] K. Bajaj, B. Sharma, and R. Singh, "Integration of wsn with IoT applications: A vision, architecture, and future challenges," in *Integration of WSN and IoT for Smart Cities*. Cham, Switzerland: Springer, 2020, pp. 79–102.

[3] H. Baniabdelghany, R. Obermaisser, and A. Khalifeh, "Reliable task allocation for time-triggered IoT-WSN using discrete particle swarm optimization," *IEEE Internet Things J.*, vol. 9, no. 14, pp. 11974–11992, Jul. 2022.

[4] P. Muoka, D. Onwuchekwa, and R. Obermaisser, "Adaptive scheduling for time-triggered network-on-chip-based multi-core architecture using genetic algorithm," *Electronics*, vol. 11, no. 1, p. 49, 2022.

[5] R. Obermaisser, H. Ahmadian, A. Maleki, Y. Bebawy, A. Lenz, and B. Sorkhpour, "Adaptive time-triggered multi-core architecture," *Designs*, vol. 3, no. 1, p. 7, 2019.

[6] T. Gong, T. Zhang, X. S. Hu, Q. Deng, M. Lemmon, and S. Han, "Reliable dynamic packet scheduling over lossy real-time wireless networks," in *Proc. 31st Euromicro Conf. Real-Time Syst. (ECRTS)*, 2019, p. 11.

[7] W. Yu, Y. Huang, and A. Garcia-Ortiz, "Distributed optimal on-line task allocation algorithm for wireless sensor networks," *IEEE Sensors J.*, vol. 18, no. 1, pp. 446–458, Jan. 2018.

[8] Y. Jin, J. Jin, A. Gluhak, K. Moessner, and M. Palaniswami, "An intelligent task allocation scheme for multihop wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 3, pp. 444–451, Mar. 2012.

[9] J. Yang, H. Zhang, Y. Ling, C. Pan, and W. Sun, "Task allocation for wireless sensor network using modified binary particle swarm optimization," *IEEE Sensors J.*, vol. 14, no. 3, pp. 882–892, Mar. 2014.

[10] H. Izakian, B. T. Ladani, A. Abraham, and V. Snasel, "A discrete particle swarm optimization approach for grid job scheduling," *Int. J. Innov. Comput. Inf. Control*, vol. 6, no. 9, pp. 1–15, 2010.

[11] W. Guo, J. Li, G. Chen, Y. Niu, and C. Chen, "A pso-optimized real-time fault-tolerant task allocation algorithm in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3236–3249, Dec. 2015.

[12] F. F. Marshall, M. Mu'azu, I. Umoh, A. Salawudeen, B. Sadiq, and D. Ikpe, "A modified real-time fault-tolerant task allocation scheme for wireless sensor networks," *Kinetik: Game Technol. Inf. Syst. Comput. Netw. Comput. Electron. Control*, vol. 4, no. 1, pp. 45–54, 2018.

[13] "Institute of electrical and electronics engineers, inc. 802.1CB-frame replication and elimination for reliability." Accessed: Apr. 11, 2021. [Online]. Available: http://www.ieee802.org/1/files/private/cb-drafts/d2/802–1CB-D2–9.pdf

[14] P.-J. Wan, O. Frieder, X. Jia, F. Yao, X. Xu, and S. Tang, "Wireless link scheduling under physical interference model," in *Proc. INFOCOM*, 2011, pp. 838–845.

[15] A. R. Guner and M. Sevkli, "A discrete particle swarm optimization algorithm for uncapacitated facility location problem," *J. Artif. Evol. Appl.*, vol. 2008, Apr. 2008, Art. no. 861512.

[16] "SNAP library 4.0, user reference documentation." 2017. [Online]. Available: https://snap.stanford.edu/snap/doc/snapuser-ref/index.html

[17] F. Wihartiko, H. Wijayanti, and F. Virgantari, "Performance comparison of genetic algorithms and particle swarm optimization for model integer programming bus timetabling problem," *IOP Conf. Series: Mater. Sci. Eng.*, vol. 332, no. 1, 2018, Art. no. 12020.
[18] H. Baniabdelghany, R. Obermaisser, and A. Khalifeh, "A reliable job allocation scheduler for time-triggered wireless networks," in *Proc. IEEE 24th Int. Symp. Real-Time Distrib. Comput. (ISORC)*, 2021, pp. 1–10.
[19] T. Kokilavani and D. G. Amalarethinam, "Load balanced min-min algorithm for static meta-task scheduling in grid computing," *Int. J. Comput. Appl.*, vol. 20, no. 2, pp. 43–49, 2011.
[20] T. Aladwani, "Types of task scheduling algorithms in cloud computing environment," *Scheduling Problems-New Applications and Trends*. London, U.K.: IntechOpen, 2020.

Haytham Baniabdelghany received the master's degree in computer engineering and networks from the University of Jordan, Amman, Jordan, in 2016, and the Ph.D. degree in computer engineering from the Department of Embedded Systems, University of Siegen, Siegen, Germany, in 2023.

He worked as a Computer Engineer from 2005 to 2018. His research interests are in the areas of clock synchronization, message scheduling, and fault-tolerant task allocation in real-time wireless systems.

Dr. Baniabdelghany was a DAAD Scholarship Holder.

Roman Obermaisser received the master's degree in computer sciences from Vienna University of Technology, Vienna, Austria, in 2001.

He is a Full Professor with the Division for Embedded Systems, University of Siegen, Siegen, Germany. In 2004, he was a Doctoral Research Advisor of Computer Science with Prof. Hermann Kopetz with Vienna University of Technology. He wrote a book on an Integrated Time-Triggered Architecture (Springer - Verlag, USA). He is the author of several journal papers and conference publications. In 2009, he has received the habilitation "Venia docendi" certificate for Technical Computer Science. He has also participated in numerous EU research projects such as SAFEPOWER, universAAL, DECOS, and NextTTA, and was the Coordinator of the European research projects DREAMS, GENESYS, and ACROSS. His research work focuses on system architectures for distributed embedded real-time systems.

Ala' Khalifeh received the Ph.D. degree in electrical and computer engineering from the University of California, Irvine, Irvine, CA, USA, in 2010.

He is currently an Associate Professor with the Communication Engineering Department, German Jordanian University, Amman, Jordan. His research is in communication technology and networking with particular emphasis on Internet of Things and wireless sensor networks.

Dr. Khalifeh is the recipient of the Fulbright scholarship from 2005 to 2007). He has received numerous academic and technical awards/fellowships, such as The Leaders in Innovation Fellowships (LIF - 2021) run by the Royal Academy of Engineering and funded by the U.K. Government's Department of Business, Energy and Industrial Strategy Newton Fund programme. The Young AFCEA 40 under 40 in 2017, the German Jordanian University Award for the industrial relationship in 2016, and the German Jordanian University Excellence in Research Award in 2015.

Pascal Muoka received the M.Sc. degree in microelectronic and communications engineering from the University of Northumbria Newcastle, Newcastle upon Tyne, U.K., in 2017.

He is currently a Research Assistant with the Department of Electrical Engineering and Computer Science, University of Siegen, Siegen, Germany. In the FRACTAL project, he developed a hierarchical adaptive time-triggered multicore architecture and hierarchical meta-scheduling algorithms for the computation of time-triggered system schedules used for runtime adaptation. His research interests are hierarchical and distributed systems, time-triggered adaptive architectures and scheduling, and optimization algorithms.