Coordination Protocol and Admission Control for Distributed Services in System-of-Systems With Real-Time Requirements

DHIAH EL DIEHN I. ABOU-TAIR\(^1\), ALA’ KHALIFEH\(^1\), (Member, IEEE), SAMEER AL-DAHIDI\(^2\), SAHEH ALOUNEH\(^{1,3}\), AND ROMAN OBERMAISSER\(^4\)

\(^1\)School of Electrical Engineering and Information Technology, German Jordanian University, Amman 11180, Jordan
\(^2\)School of Applied Technical Sciences, German Jordanian University, Amman 11180, Jordan
\(^3\)College of Engineering, Al Ain University, Abu Dhabi, United Arab Emirates
\(^4\)Chair for Embedded Systems, University of Siegen, 57076 Siegen, Germany

Corresponding author: Dhiah El Diehn I. Abou-Tair (dhiah.aboutair@gju.edu.jo)

This work was supported by the ECSEL Joint Undertaking (JU) through the European Union’s Horizon 2020 Research and Innovation Program from Spain, Italy, Austria, Germany, France, Finland, and Switzerland, under Grant 877056.

ABSTRACT  System-of-Systems (SoS) offer unprecedented potential for new types of emerging services, which significantly exceed the capabilities of the constituting systems. SoS in safety-critical domains (e.g., medical applications, smart grid, disaster recovery, defense) are prominent examples, but they have stringent real-time and reliability requirements. Therefore, a suitable temporal and spatial allocation of resources is required both within each constituent system and in the wide area networks between them. This paper introduces an algorithm for admission control and resources’ allocation, which considers these requirements and the autonomy of the constituent systems. To simulate a realistic admission control and resources’ allocation process of a typical SoS network, a simulated case study with eight constituent systems, six services, and twenty-five processes/requests is developed. The suggested admission control and resources’ allocation process’s performance is measured in terms of gain in the execution time and blockage probability. A sensitivity analysis is carried out to evaluate the influence of the number of constituent systems and the number of services sought by the received processes/requests on the efficacy of the proposed process. The results show that the proposed admission control and resources’ allocation process have very low blockage probability, high gain in the execution time, and high resources’ utilization.

INDEX TERMS  System-of-systems, admission control, resources’ allocation, distributed systems, real-time requirements.

I. INTRODUCTION  System-of-Systems (SoS) have recently gained popularity among the research community, especially with the advent of several emerging and disruptive technologies like the Internet of Things (IoT) [1], cloud computing [2], [3], big data [4], artificial intelligence (AI) [5], etc. Therefore, a new era of applications and services driven by a mixture of these technologies, has been introduced to cope up with our information age, thus research around SoS network architecture and paradigm becomes more in focus. Electronic health care services [6], remote security and monitoring [7], precision agriculture [8], aviation [9] and industrial automation [10] are some examples of these technology driven applications and services which can be effectively integrated and implemented in the context of SoS. However, to establish a reliable and effective SoS infrastructure for these services, the conventional information technology (IT) infrastructure that solely depends on the resources available at one single network, should be revamped in a way to support the diverse resources needed by these services. Thus, the SoS infrastructure requires the integration and collaboration

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Ali.
of several network service providers. For instance, several safety-relevant applications and scenarios often have stringent requirements such as reliability, security, high availability, and real-time constraints. Capitalizing the resources needed to meet the requirements of these applications and services in one network is very challenging and, in some case, impossible, especially when considering the cost, effectiveness, scalability, reliability, and efficiency aspects. Motivated by the above discussion, SoS architecture was proposed as an efficient and cost-effective solution to these challenges, which in turn will pave the way toward actualizing these services in an affordable manner, taking into account the application requirements and constraints, thus ensuring high-quality SoS services. The SoS initial system architecture was coined in Maier [11] sentimental paper, where the author defined the main taxonomies associated with SoS, along with the principle architecture. Maier emphasized that the SoS architecture is not physical but rather logical, where a set of standard components communicate with each other, to accomplish a certain functionality in a collaborative manner. The fundamental benefit of the SoS architecture lies in the fact that it utilizes the available resources for different systems, that work together as one complete system thus being capable of providing complex functions, which cannot be achieved by the independent systems.

This paper uses the definition of SoS as introduced in [12] and [13], dependent and which are networked together for a period of time to establish emerging services and satisfy requirements that cannot be met by any CS in isolation. Cyberphysical SoS are time-sensitive and include not only the computer system, but also a controlled object and humans [12]. An example of such a system is an intelligent transportation system with vehicle-to-vehicle communication and vehicle-to-infrastructure communication. Another example is a healthcare SoS where patient monitoring and treatment services are dynamically established with patients, medical doctors, hospitals, data centers, and care takers.

For a given application to be deployed in an SoS network architecture, the application is initiated at a CS, which is often named as the initiator CS as will be explained in the SoS section of this paper (section III). The initiator-CS is responsible for resources provision and coordination between the SoS CSs, which involves (1) identifying CSs that can provide the services, (2) optimizing the use of the services driven by extra-functional properties, (3) performing admission control for service-provision, (4) reserving resources for provided services and (5) recursively using services from subcontracted CSs to realize service-provisions [14].

This paper focuses on the admission control challenge in SoS network architecture, which is very crucial for effective and efficient service provisioning. Admission control in SoS ensures that the required resources demanded by the user request are fulfilled taking into consideration many factors such as: the service real-time provisioning, priority, reliability, cost, scalability, security, etc. In our previous works, [15] and [16], an SoS model and architecture were introduced. In [17] and [18], we focused on the security and network resource reservation between different CSs within the SoS utilizing the Multiprotocol Label Switching (MPLS) paradigm. In [19], the SoS network architecture taking into account the services’ provisioning concept for SoS have been described. Furthermore, a preliminary description of a proposed distributed admission control algorithm has been described in [20]. This paper expands our previous work in the field of SoS, mainly the one presented in [20] by expanding the proposed admission control algorithm, where a more detailed description is provided, in addition to presenting a detailed simulation and performance evaluation of the proposed algorithm for SoS network architecture, which addresses the problem of performing admission control for service provisioning utilizing the distributed SoS resources. Further, the paper provides a more holistic view on the network architecture, considering an SoS paradigm, where the resources must be first checked from availability point of view, then reserved and utilized by the requesting service. The proposed algorithm has been evaluated by simulation, which showed the effectiveness of the proposed algorithm. In summary, the original contributions of this work can be summarized as follows:

- The development of an admission control and resources’ allocation algorithm that considers the application under consideration requirements and the autonomy of the CSs.
- The verification of the proposed algorithm with respect to a simulated case study, properly designed to mimic a realistic admission control and resources’ allocation process of a typical SoS network.
- The evaluation of the proposed algorithm’s effectiveness by computing various performance metrics from the literature, such as the gain in the execution time and the blockage probability.
- The investigation of the influence of the number of CSs and the number of services sought by the received processes/requests on the efficacy of the proposed algorithm by a proper sensitivity analysis.

Furthermore, most of the published works in the literature propose admission control algorithms for one system, while this work focuses on proposing an admission control algorithm for distributed services in SoS with real-time requirements.

The rest of the paper is organized as follows: Section II summarizes the state of the art papers. Section III describes the SoS network architecture. Section IV discusses the proposed admission control and resource allocation algorithm. The performance evaluation of the proposed algorithm is presented in Section V. Finally, the paper is concluded in Section VI and future recommendations are stated.

II. LITERATURE REVIEW

The SoS literature covers a wide range of applications, types, and research issues. We limit ourselves to the admission
control and scheduling part. There has been numerous works in the field SoS, however, there is a little work that focuses on and solve issues related to the coordinated and distributed admission control and scheduling for SoS networks.

For example, the authors in [21] tackled the issue of scheduling large size of dependent and independent task requests that reach cloud systems. The work proposed a heuristic scheduling mechanism to reduce the time of large requests scheduling. The results obtained presented an improved performance outcomes when compared to other current solutions in terms of speed-up, make-span, and length ratio scheduling. The dynamic request scheduling and resource allocation within consentient systems are considered in [22] and [23].

The work in [24] presented a data processing platform named Nephele that dynamically assigns resources in cloud systems for executions and scheduling tasks. The proposed platform does not consider quality of service (QoS) requirements and end-to-end deadlines in the scheduling process. Furthermore, reliability and the network inconsistency between the constituent systems are not considered.

A literature survey of resources’ scheduling algorithms for cloud computing systems was discussed and analyzed in [25]. The survey and analysis considered the grouping of resources, evolution of resources scheduling, and different available scheduling algorithms and the associated QoS constraints. The authors indicated that it is still not an easy task to discover the best planning and mapping of workloads capabilities and resources without having effective mechanisms for resource provisioning. Also, it can be observed from the conducted literature review and surveys that most of the existing works consider only scheduling requests and resource allocations on one and only CS, while the requirements for multiple CSs of diverse requests are not taken into consideration. Additionally, the vibrant nature of resources availability and requests are not addressed carefully.

The researchers in [26] provided new awareness into the energy sharing and low-carbon cost-effective scheduling between multiple energy systems from the systems of systems perspective. The relation between admission control and data analytic is also of main concern.

An aircraft fleet planning and architecture framework assessment for air mobility and distribution using a system-of-systems approach is discussed in [27]. The framework provides understanding of the SoS design space and successful deployment or optimization of UAM fleet.

The concept of emergence and its relation to the engineering of SoS and Smart energy grids environmental impact were studied in [28].

A recent work that proposes an algorithm for incremental scheduling with real time requirements for the SoS is discussed in [29]. The authors proposed a two-level cooperating heuristic method using a genetic algorithm (GA) to schedule real-time requests incrementally by adding requests to the time triggered SoS systems. The algorithm deploys and computes the schedule for every new SoS request after its arrival at run-time. Consequently, the computational resources and limited communication must be shared and communicated between different SoS requests and applications released and communicated over time. The blocking time thus is reduced for the shared resources; and as a result, the possible shortage of resources for future applications can be handled. Hence, the authors developed a new resource allocation algorithm for more balanced blocking times of shared resources and supports future resource requests.

The work in Roy et al. [30], proposed two integer linear programming (ILP)-based schemes, namely ILP with explicit time reduced (ILP-ETR) and ILP with non-overlapping constraints (ILP-NC), for optimally scheduling real-time precedence-constrained task graphs (PTGs) on platforms composed of heterogeneous processing elements interconnected through a set of heterogeneous shared buses in contrast to conventional schemes that deal with homogeneous elements and communication channels. The suggested schemes were shown to be realistically efficient when tested on an automotive cruise controller case study. In the same context of a heterogeneous distributed platform, the work of Roy et al. [31], proposed two low-overhead heuristic algorithms, namely global slack aware quality-level allocator (G-SLAQ) and total slack aware quality-level allocator (T-SLAQ) for optimally scheduling real-time directed-acyclic task graph (DTG) combined with multiple quality-level tasks, with convenient computational efforts. The proposed schemes were shown to be more effective than the traditional ILP scheme when tested in an automotive traction controller case study. Similarly, in the work of Roy et al. [32], the authors proposed a low-overhead heuristic algorithm, namely the contention cognizant task and message scheduler (CC-TMS) for optimally scheduling real-time DTG with convenient computational efforts. The proposed scheme was shown to be more effective than the traditional ILP scheme when tested in an automotive traction controller case study.

The authors in [33] illustrated the use of AI to sort space habitation sub-systems for NASA technological groups and to classify applicable sources of data for these sub-systems. The authors demonstrated how AI agents can support the recovery and retrieval of composite information needed to feed existing SoS analytic tools and discussed possible challenges and future steps.

The authors in [34] discussed the problem of managing volatile and unpredictable variations of SoS networks. The authors proposed a dynamic reconfiguration scheme that aims at enhancing the SoS agility to quickly respond from failures. The proposed scheme employs estimated dynamic programming technique to calculate the dynamic reconfiguration choices and decisions that can allow failed or degraded sub-systems to be detached and the allocation of new resources to be changed rapidly.

III. SYSTEM-OF-SYSTEMS ARCHITECTURE
As shown in Fig. 1, a typical SoS consist of several CSs where each CS can deliver certain services provided by its end
systems (ESs), which can be for example a computing device that runs a certain software application to provide services. The ESs are connected with each other via routers. Further, each CS contains a constituent system manager (CSM) that manages the CS resources in terms of admission control, resource allocation and scheduling.

Within this SoS architecture, the envisioned establishment process for dependable SoS-applications as depicted in Fig. 3 is as follows:

A. SoS-APPLICATION REQUEST AND TARGET-CS DISCOVERY
The starting point is a request for an SoS-application at a CS (henceforth called initiator-CS). As part of a graph of services depicted in Fig. 2, the initiator-CS will require services from other CSs (henceforth called target-CS). Therefore, the initiator-CS needs to find CS that provides these services by using a broker (e.g., [35]).

B. SERVICE-OFFER REQUEST
The initiator-CS will send requests for service-offers to the target-CS, which were named as potential providers of the services by the broker.

C. SHORT-TERM SERVICE ADMISSION BY TARGET-CS
Each target-CS will execute a schedulability test and in case of success it will reply with a service-offer and a short-term admission to the initiator-CS. The short-term admission will be associated with an expiration time. Each target-CS can recursively contact other CS if it needs to use other CS-services to provide its own services.

D. OPTIMIZED SELECTION OF TARGET-CS
The initiator-CS checks the service-offers from the target-CSs by considering their available resources and matches it with the requests’ constraints (e.g. end-to-end delay). The initiator-CS selects for each service the winner CS(s) in an optimized manner (e.g., cost, reliability, timing, network delays, network reliability). Confirmation messages will transform the short-term admission into a long-term admission. Short-term admissions without a confirmation from the initiator will expire at the target-CS.

E. LONG TERM ADMISSION AND EXECUTION OF SoS APPLICATION
The selected target-CS will execute a distributed algorithm for the incremental resource reservation of these CSs. Thereafter, the SoS-application is executed based on the allocated resources and the service contracts. Service revocations can occur in case of resource conflicts with SoS-applications of higher criticality.

IV. ADMISSION CONTROL AND RESOURCES ALLOCATION
To establish the process of dependable SoS applications, a coordination protocol needs to be defined that is responsible for resources’ scheduling among CSs in a distributed manner. In the proposed SoS network architecture, several requests...
submitted from different ESs asking for services with diverse requirements such as E2E delay, reliability, security, costs, etc. need to be processed by different CSs. These requests are generated from different CSs that may be issued for the same services at the same time. The coordination protocol must handle these requests dynamically and in a distributed manner such that each CS will handle the received/generated requests and allocate the needed resources in coordination with the other related CSs. This allocation process optimizes the available resources in all CSs to assure that all requests meet their requirements. The challenge in designing the coordination protocol results from the stringent requirements of the architecture and the applications of the SoS. A concurrent, incremental, and distributed scheduling and admission control process is required as described below.

1) Admission control: In SoS, different CSs may generate different requests with specific constraints. Admission control determines whether the requests can be admitted and scheduled according to the available resources, such that the requests’ requirements can be met. To achieve that, CSs must exchange their resource allocation and scheduling policies regularly in order to get a global picture of the available resources and determine if the new request can be admitted taking into account its constraints without jeopardizing the admitted requests.

2) Requests concurrency in mixed-critical systems: CSs can receive multiple requests with different priorities and criticalities at the same time competing for the current available resources. To handle these requests, a prioritization policy has to be implemented in all CSs such that it handles the requests in an optimal manner taking into account their criticalities and the resource availability. For example, some requests should be handled with stringent deadline requirements, since they are related to safety-critical services. These requests are considered more critical than other less stringent requests. This process may cause different challenges that need to be addressed such as deadlocks, request starvation and aging.

3) Revocation of previous admission based on criticality: High priority requests may arrive to CS without being admitted due to the lack of the available resources. To avoid such a case, resources revocation for lower priority admitted requests should take place, which will allow the admission of higher priority requests. This resource allocation and revocation process based on the priority should be done in coordination with all CSs to ensure optimal resource allocation with minimal resources wastage.

4) Handling requests with multiple constraints: Several requests with different constraints (e.g., real-time assurance, reliability, security, cost, etc.) should be handled properly in the resources’ optimization allocation process of different CSs. However, dealing with different constraints increases the admission process complexity. As such, heuristic resource allocation algorithms must be proposed in all CSs to deal with the problem efficiently.

To have an optimal distributed resource allocation between different CSs, a distributed resource allocation protocol is proposed. The protocol runs on all CSs of the SoS and implements the following tasks:

1) Resource Discovery. Each CS must have up-to-date knowledge about the available resources at its ESs with their current status. To achieve that, a resources allocation manager (RAM) is proposed, which runs a periodic resource discovery process to explore any new resources that were added and also to exclude vanished resources. While running the discovery process, a resources allocation table (RAT) should be established which has the following main entries: CS ID, resources ID (RID), resource status (RS), priority level (PL), and process constraints (PCons). In what follows, a brief description of the RAT fields is provided.

- Resources ID: each resource should have a unique ID within the SoS. This ID should indicate its main functionality and the ES that hosts the resource. Moreover, depending on the CS that is hosting the resource, each process will have different processing time. This can be represented using the resource processing time (RPT) entry.
- Resource Status: the RAM should be aware of the status of all resources, which is represented by a tuple of three fields; the Resource Status which can be either busy (B) or free (F), the Process ID that is occupying the resource, and the time slot on which the resource will be busy (busy slot (BS)). For example, if the resource $R_1$ in $CS_0$ is occupied by process ID ($P_{00}$), during the period from [0-75] ms, then the RS of the resource $R_1$ inside $CS_0$ RAT will be as follows: B:$P_{00}$:BS[0-75]. It is important to mention that to maximize the resources utilization, the entries of the RAT of each CS should be regularly updated whenever a new process is admitted or served.
- Priority Level: once a process utilizes resources, its priority level should be saved. A mechanism for determining the process priorities should be implemented.
- Process Constraints: all the process constraints such as E2E delay, fault tolerance, security, reliability, etc. are stored in this field. In addition, this field contains the set of the needed resources by the process at each CS.
- CSs Route: this entry shows the sequence of remaining CSs (including the one that is admitted to), that the process will follow to get all the needed resources and services.
Finally, it is important to emphasize that this table is frequently exchanged with the neighbouring CSs within the SoS, so all the CSs will have a global vision for all the available resources in different CSs. Once the resource discovery process is performed, the path of the CSs that have the needed resources can be identified, which will be used in the path reservation process utilizing the resource reservation protocol (RSVP).

2) Resource Reservation. In order to serve multiple processes with different priorities and requirements, a resource reservation process will take place. To achieve that, a resource reservation protocol (RRP) is proposed. The RRP has the following main functionalities:

- **Process Pre-admission**: When a process arrives to the SoS, the RRP will admit it temporarily till it checks whether it can fulfill its constraints. The RRP will save the received requests in a temporary queue, extract its requirements and check with other SoS for resource availability.
- **Path Determination**: the RRP will consult the CSM of the CS in order to determine all possible paths toward the next CS. Different paths can be utilized to choose the one that can fulfill the process constraints.
- **Resources Reservation**: Once the path of the next CS is determined, the RRP will send a resource reservation request (RRR) with the following fields: The Process ID, which defines the ID of the process that is asking for a resource. The Source CS ID is the ID of the CS that the process belongs to. The Resource ID is the ID of the resource required by the process and the Priority Level defines the process priority level to assess its criticality. The constraints define the process constraints in terms of E2E delay, fault tolerance, security and reliability.
- **Process Admission**: The RRR will be sent to all the required CSs needed by the process. If all the needed CSs can fulfill the process requirements, then the needed resources in all CSs will be reserved, and the process is moved from the pre-admission queue in the receiving CS to the admitted queue. All the CSs will update their RAT to reflect the new admitted processes. However, if the reservation process was unable to fulfill the process requirements and constraints, then another admission process called priority-based admission (PBA) will take place, which is used as a mitigation procedure for the potential failure of the normal admission process. This may happen if the admission process was unable to admit a process since other processes are occupying the available resources. In this case, the process priority should be considered. The flow chart and the unified modeling language (UML) sequence diagram of the admission control algorithm are depicted in Fig. 4, and Fig. 1, respectively. Furthermore, the admission control computational complexity is $O(RsN)$, where $Rs$ is the maximum resources requested by the processes, and $N$ is the number of CSs. This complexity corresponds to the worst case scenario where the initiator CSM has to check with all CSs for all requested resources by the process. Finally, the spatial complexity which corresponds to the
storage needed to store the CSs RAT is $O(T_sN)$, where $T_s$ is the maximum possible RAT table size (which corresponds to the CS that has the most resources. It is apparent that both complexities are polynomial which makes the algorithm feasible from implementation point of view.

A. DEMONSTRATION EXAMPLE

The following example aims at elaborating the proposed process admission and resources reservation protocol, which also highlights the various challenges and requirements that need to be taken into consideration while developing the admission control and resource allocation algorithms.

Fig. 6 shows SoS that consist of four CSs ($CS_1$, $CS_2$, $CS_3$ and $CS_4$). Each CS has a RAT table that shows the current available resources with the CS, and the allocated processes. For example, in $CS_1$, the CS has four different resources ($R_1,R_2$) that are assigned to ES1, and $R_3,R_4$ that are assigned to ES2. The resources processing time for each resources is shown for each CS. For instance, in $CS_1$, the processing time needed for $R_2$ is equal to 100 ms, where it is equal to 50 ms in $CS_2$, which is expected since in $CS_2$, each ES is dedicated for one resource, which in turn will result in less processing time. The example below shows that the SoS is currently handling two processes, $P_0$, and $P_1$. The RAT tables show how the resources are allocated between different CSs. For example, as shown in the Constraints column of RAT1, $P_0$ had the following constraints: an E2E delay equals to 300 ms, and it needs the following resources in order: $R_1, R_2$, and $R_3$. Currently it is scheduled as follows:

It will be served first by $CS_1$: $R_1$, the busy time slot will be from 0 to 75 ms, and the CS-route is as follows: $CS_1, CS_2$, and then $CS_2$ again. One can notice that the needed resources for this process have been reserved in the respected CSs. For instance, after the process is served by $CS_1$: $R_1$, it will be routed to $CS_2$ and be served by $R_2$, the busy time slot for $CS_2$: $R_2$ will be from 75 to 125 ms. Notice that the processing time for $CS_2$: $R_2$ is equal to 50 ms according to the RPT entry. Further, to simplify the scheduling problem, we ignored the communication and transmission delay between CSs. Finally, the process will be delivered to $CS_2$ again, where it will occupy $R_3$ from 125-175 ms. Here the processing time for $CS_2$: $R_3$ is equal to 50 ms.

Once a resource reservation request arrives, the admission control manager at the receiving CS checks the resources utilization trees for all the resources in all CSs, and assigns the required resources to the available ones such that the assignment process will meet the process constraints. In this example, the admission control manager of $CS_1$ will attempt first to assign $P_1$ to $CS_1$, since $CS_1$: $R_3$ is free, then it will attempt to find the second needed resources ($R_1$) to a suitable CS, in this case, there is two available free resources for $R_1$, one in $CS_1$ with RTP equals to 75 ms, and another one in $CS_3$ with RTP equals to 100 ms. The admission control manager will chose the one with lowest RTP (i.e. $CS_3$), after that, the admission control manager will try to allocate the last resource to $P_1$ ($R_4$), which is allocated to $CS_2$. 

![FIGURE 5. The UML sequence diagram of the proposed admission control algorithm.](image-url)
After discussing the RAT tables and entries. We will discuss the process of admitting a new request \( P_2 \) which assumed to arrive at the same time of the previous two processes. The process \( P_2 \) request table is shown where the process was originated from \( CS_0 \), it is requesting the following resources \( R_1, R_2, \) and \( R_5 \), the process has a 250 ms E2E delay constraint. Finally, the process priority is equal to 1.

In order to decided whether the process can be admitted or not, each CS will prepare a Resources Utilization Tree (RUT) for all available resources in the SoSs. As shown in Fig. 7, a sample RUT is constructed for \( R_1 \). The RUT consists of a parent node (the resource ID) and child nodes (CSs that have the resource). On each child node, a resource utilization vector (RUV) in the form of a linked list is established. This RUV has the following entries: the first entry is the RPT of that resource in the respective CS. The second entry is a flag indicating whether the resource is free (F) or busy (B), if it is busy, then the reservation schedule as a function of the RPT is shown. For example, in Fig. 6, \( R_1 \) is reserved in the first time slot (from 0 to 75 ms, marked in red-color), while \( R_1 \) in \( CS_1 \) is busy in the second time slot. Finally, \( R_1 \) in \( CS_4 \) is free in all time slots. Note that the time slot duration is a function of the RPT of the respective CS. Thus it is 75 ms in both \( CS_1 \) and \( CS_3 \), while it is 100 ms in \( CS_4 \).

Note that the assignment process may become more complicated, especially if the goal is to perform the resources allocation in an optimal manner. Another challenge appears more than one request arrive at the same time with different priorities. Then the admission process has to take into account not only the available resources, but also the process priorities. In some case, it may issue a resource revocation command to an assigned process to a specific resource with lower priority, to allow higher priority processes to be served. However, this revocation process should be done without jeopardizing the constraints of the lower priority process. For example, if a new request arrives \( P_3 \) as shown in Table 1, with high priority (PL = 1) and with a very strict E2E delay (150 ms), which requires two resources \( R_1 \) and \( R_2 \). Then if the admission control manager assigns the free available resources that exists on \( CS_4 \), \( CS_1 \), respectively, then the process will miss its E2E delay constraints, since \( R_1 \) and \( R_2 \) in these CSs require 200 ms processing time, which is higher than the 150 ms E2E delay constraints of \( P_3 \). However, according to the priority level of \( P_3 \), the admission control manager will give the priority to \( P_3 \) compared to the \( P_0 \) and \( P_1 \) (priority level is 2 and 3, respectively). Then, the resources requested by \( P_3 \) will have higher priority than \( P_0 \) and \( P_1 \). Thus, \( P_3 \) may get admitted and both \( P_0 \) and \( P_1 \) may get executed later (if they will still meet their E2E delays) or blocked.

V. SIMULATION AND PERFORMANCE EVALUATION

In this section, an SoS network architecture embedded with the proposed admission control process and resources’ allocation has been designed and evaluated. The SoS network architecture has been simulated to mimic an SoS senario.
TABLE 1. Process $P_3$ resources request.

| Process ID | Source SoS ID | Resource ID(s) | Priority level | Constraints |
|------------|---------------|----------------|---------------|-------------|
| $P_3$      | $C_{S0}$      | $R_1, R_2$     | 1             | E2E delay 150 |

![](image)

FIGURE 7. An example of the resource utilization tree of resource $R_1$.

...with a realistic admission control and resources’ allocation process. More specifically, the simulated SoS network architecture comprises $N$ CSs, each delivering some services out of $S$ possible services within a certain period of time based on the requirements of $P$ different processes/requests. Moreover, the simulated SoS network has been sufficiently designed to address the dynamic nature of the requests and the availability of the resources at any possible values of $N$, $S$, and $P$.

For illustration purposes, an SoS network architecture with $N = 8$ CSs has been considered: each can deliver a maximum of $S = 6$ services to fulfill $P = 25$ processes/requests within the time horizon. It is worth mentioning that the proposed admission control and resources’ allocation process has been performed with a Matlab based simulator that has been developed for this purpose. This simulator is capable of simulating various SoS network architectures with any possible random values of $N$, $S$, and $P$.

The generalizability of the designed SoS and its admission control and resources’ allocation process entails the following simulation aspects:

A. SERVICES PER CS

- The number of services to be offered by each CS is selected randomly from an arbitrary range that spans the interval $[5, 6]$, to ensure the complexity of the case study being developed, where 5 and 6 are the minimum, maximum number of services that a CS could deliver, respectively.
- The services to be offered by each CS are selected uniformly at random, without replacement, from the available $S = 6$ services.
- Each service is assumed to be delivered by each CS within a period of time that spans the interval $[15, 50]$ ms with a step size of 5 ms, where 15 and 50 are the minimum, maximum time duration, respectively by which a certain CS could deliver the service.

B. SERVICES PER PROCESS/REQUEST

- The number of services to be requested by each process/request is selected randomly from an arbitrary range of values that span the interval $[1, 6]$, where 1.6 are the minimum, maximum number of services that a process could request, respectively.
- The order of services to be requested by each process is randomly initiated.
- The priority level of each request is selected randomly from an arbitrary range of values that span the interval $[1, 3]$, where 1 and 3 are the minimum and the maximum priority level, respectively implied by each process/request.
- The E2E delay of each process/request is selected randomly from an arbitrary range of values that span the interval $[300, 450]$ ms, with a step size of 10 ms, where 300 and 450 are the minimum, maximum E2E delay constraint, respectively implied by each process/request.
- The number of processes/requests to arrive at a time is selected randomly from an arbitrary range of values that span the interval $[1, 2]$, where 1 and 2 are the minimum, maximum number of processes/requests, respectively that could be arrived at a time instant.
- The processes/requests arrive according at Poisson distribution with a parameter ($\lambda$) equals to 20.

For clarification purposes, Table 2 and Table 3 show the randomly generated CSs and processes/requests, respectively. For instance, looking at Table 2, one can notice that the first CS ($C_{S1}$) can deliver the six considered services (i.e., $R_1 - R_6$), each with a particular processing time. For example, the processing time needed for $R_1$ by $C_{S1}$ is equal to 25 ms, whereas the last CS ($C_{S6}$) can only deliver five services (i.e., all except $R_5$), each with a particular processing time. For example, the processing time needed for $R_1$ by $C_{S6}$ is equal to 40 ms.

Furthermore, Table 3 reports the simulated random processes/requests received at the SoS under consideration, their requested constraints, i.e., execution order, priority, and the E2E delay. For instance, the second service ($P_2$) to be received either individually or together with the previous or the other subsequent process (e.g., $P_2$ or $P_2$ in this case, respectively) requires the execution of the following services in a chronological order as follows: $R_1, R_3, R_6$, then $R_4$. It necessitates the execution of the above-mentioned services.
TABLE 2. The simulated CSs and their available services and delivery time.

| Constituent Systems (CSs) | C|S|1 | C|S|2 | C|S|3 | C|S|4 | C|S|5 | C|S|6 | C|S|7 | C|S|8 |
|---------------------------|---|---|---|---|---|---|---|---|---|---|---|---|
| R|1 | 25 | 40 | 30 | NA | 45 | 30 | 45 | 40 |
| R|2 | 30 | 45 | 20 | 35 | 40 | 20 | 45 | 30 |
| R|3 | 25 | 15 | 35 | 15 | 35 | 25 | 25 | 15 |
| R|4 | 15 | 20 | 40 | 25 | 25 | 25 | 15 | 35 |
| R|5 | 45 | 50 | 25 | 40 | 30 | 45 | 30 | NA |
| R|6 | 50 | 30 | 15 | 30 | 15 | 30 | 50 | 15 |

TABLE 3. The simulated processes/requests, execution order, priority, and E2E.

| Process/Request | Requested services and Constraints |
|-----------------|-----------------------------------|
| R|1 | P|1 | 0 | 2 | 4 | 3 | 3 | 1 | 410 |
| R|2 | P|1 | 0 | 2 | 4 | 3 | 3 | 1 | 450 |
| R|3 | P|1 | 1 | 3 | 2 | 0 | 1 | 420 |
| R|4 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|5 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|6 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|7 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|8 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|9 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|10 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|11 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|12 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|13 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|14 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|15 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|16 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|17 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|18 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|19 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|20 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|21 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|22 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|23 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|24 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |
| R|25 | P|1 | 0 | 0 | 0 | 2 | 4 | 3 | 360 |

Gain in Time [ms] = E2E – Actual Execution Time  
Blocking Probability [%] = Number of Blocked Processes * 100%  

FIGURE 8. The gain in time and its characteristics of the executed processes/requests as well as the blocked processes/requests.

TABLE 4. The proposed SoS performance metric.

| Metric | Value |
|--------|-------|
| Minimum gain in time | 28 ms |
| Maximum gain in time | 381 ms |
| Average gain in time | 175.958 ms |
| Blocking probability | 4% |

The admission control process and resources’ allocation continues until the execution (or blocking) of the considered simulated processes/requests (i.e., P = 25). The former (execution) entails that if/there exist(s) CS/CSs that is/are capable of handling the services required by the received process(es) respecting the E2E delay constraint, whereas the latter (blocking) entails that there are no CSs capable of managing the required services while respecting the E2E delay or there are no CSs capable of delivering the requested services.

Once the whole processes/requests are received and managed, the performance of the proposed admission control process and resources’ allocation is evaluated. Fig. 8 shows the executed processes/requests (depicted as bars in the figure) and the blocked processes/requests (depicted as x in the Figure). The order of the received processes/requests in this illustrative example is shown on the x-axis. The Figure also highlights the gain in time of those processes/requests executed by utilizing admission control algorithm. The figure also depicts: the average, minimum and maximum gain in time which are illustrated by a dashed line, a yellow highlighted bar and a red highlighted bar, respectively.

In summary, Table 4 reports the above-mentioned SoS performance metrics used in the simulation which shows a considerable average gain in time (for illustration, the average gain in time is computed as per Eq. (1)) and a low blocking probability (for illustration, the average gain in time is computed as per Eq.(2)), thus illustrating the effectiveness of the proposed admission control algorithm.

Looking at Fig. 9, one can recognize the following:
The time necessities to execute/fulfil the received processes/requests increases with time due to the fact that the resources offered by the available CSs become occupied with time. This is clear while looking at Fig. 9a, i.e., the actual execution time approaches the E2E of the executed/blocked processes/requests, and, consequently at Fig. 9b, i.e., the gain in execution time decreases as long as more processes/requests received and executed/blocked with time.

The processes/request \( P_{24} \) is blocked due to the fact the CSs available to fulfil the process’ services (i.e., six services as per Table 3) require more time (i.e., 411 ms) than the required E2E of the process/request \( P_{24} \) (i.e., 400 ms). Thus, \( P_{24} \) is blocked.

Fig. 10 shows the dynamic nature of the received processes/requests and the availability of the resources. The Figure shows the \( P = 25 \) processes/requests, their requested services (shown in different colors), and their admission control process and resource allocation over the available CSs (i.e., indicated as a number inside each service). The blocked processes/requests is (are) shown in the Figure as well. The remaining processes/requests have continued to be received and executed/blocked on timely basis following a Poisson distribution function depending on the CSs resources availability and their ability to fulfil the processes/request E2E delay constrains. Last, it is worth mentioning that no time slots have been reserved for the blocked \( P_{24} \) process/request.

To further study the performance of the proposed admission control process and resource allocation, a sensitivity test has been carried out. Specifically, the influence of two parameters of the SoS network architecture on the gain in the execution time (in ms) and the blockage probability has been investigated. The two parameters are:

1) The number of CSs. The possible number of CSs is assumed to cover the interval \([1, 10]\), where the lower and upper bounds are the minimum and the maximum number of CSs that could be available in the SoS network architecture, respectively.
2) The number of resources required by each received process (s)/request (s) (RPP). The possible number of resources is assumed to cover the interval [1, 6], where the lower and upper bounds are the minimum and the maximum number of resources that could be requested by each received process (s)/request (s), respectively.

To this aim, the admission control and resource allocation process is simulated 100 times for each possible combination of the two above-mentioned parameters, considering \( P = 10 \) processes/requests to be received adaptively with the time.

Once the 100 simulation trials are completed, the ultimate gain in execution time and the blockage probability are calculated by averaging their values across the 100 simulation results.

Fig. 11 shows the gain in execution time (in ms) versus the possible number of CSs available while varying the number of resources required by each received process (s)/request (s), RPP = 1 to RPP = 6. Looking at Fig. 11, one can recognize the following:
VI. CONCLUSION AND FUTURE WORK

This work studied the admission control and resources’ allocation process in SoS paradigm, which comprise various systems (constitute systems) working independently from each other to fulfill any complex task received. The coordination between these systems is crucial to fulfilling the received task requirements and constraints. To this aim, this work proposes an admission control and resources’ allocation process to effectively manage any complex task received by distributing it among the proper working systems that can fulfill its constraints with fewer efforts. The proposed approach has been verified concerning a simulated case study that was appropriately developed to mimic a realistic SoS paradigm embedded with realistic admission control and resources’ allocation. The performance of the proposed algorithm has been investigated in terms of gain in the execution time and blockage probability. The former entails highlighting the time gained in executing the received task by the available systems of the SoS, whereas the latter entails stating the services that have been requested but blocked due to various reasons, like, for example, the non-availability of the systems that could fulfill the requested services. Further, a sensitivity analysis has been carried out to study the effect of having different possible numbers of available systems and the number of services requested on the overall efficacy of the proposed process. As a future work, we are investigating more complex cases where requests may have multiple constraints that should be fulfilled jointly while maximizing the resources’ utilization and minimizing the blockage probability.

REFERENCES

[1] F. C. Delicato, P. F. Fires, T. Batista, E. Cavalcante, B. Costa, and T. Barros, “Towards an IoT ecosystem,” in Proc. 1st Int. Workshop Softw. Eng. Syst. Syst. (SESoS), 2013, pp. 25–28.

[2] R. M. Tanash, A. F. Khalifeh, and K. A. Darabkh, “Communication over cloud computing: A security survey,” in Proc. 42nd Int. Conf. Inf. Commun. Technol., Electron. Microelectron. (MIPRO), May 2019, pp. 496–501.

[3] T. Kiblawi and A. Khalifeh, “Disruptive innovations in cloud computing and their impact on business and technology,” in Proc. 4th Int. Conf. Rel., Infocom Technol. Optim. (ICRITO), (Trends Future Directions), Sep. 2015, pp. 1–4.

[4] B. K. Tannahill and M. Jamshidi, “System of systems and big data analytics—Bridging the gap,” Comput. Electr. Eng., vol. 40, no. 1, pp. 2–15, Jan. 2014.

[5] E. Alonso, N. Karcanias, and A. G. Hessami, “Multi-agent systems: A new paradigm for systems of systems,” in Proc. 8th Int. Conf. Syst. (ICONS), Seville, Spain, Jan./Feb. 2013.

[6] A. Khalifeh, R. O. Obermaisser, M. Abuteir, and D. E. I. Abou-Tair, “Systems-of-systems framework for providing real-time patient monitoring and care,” in Proc. 8th Int. Conf. Pervasive Comput. Technol. Healthcare, 2014, pp. 426–429.

[7] A. Khalifeh, Z. Sakkijha, H. Qaderi, O. Ghatasheh, A. Issa, A. Shatat, H. Jaber, R. Alwardat, and S. Alhaj-Ali, “An energy efficient WSN implementation for monitoring and critical event detection,” in Proc. 2nd IEEE Middle East North Afr. Commun. Conf. (MENAComm), Nov. 2019, pp. 5–6.

[8] J. Mahalakshmi, K. Kuppusamy, C. Kaleeswari, and P. Maheswari, “IoT sensor-based smart agricultural system,” in Emerging Technologies for Agriculture and Environment. Cham, Switzerland: Springer, 2020, pp. 39–52.

[9] H. Liu, Y. Tian, Y. Gao, J. Bai, and J. Zheng, “System of systems oriented flight vehicle conceptual design: Perspectives and progresses,” Chin. J. Aeronaut., vol. 28, no. 3, pp. 617–635, Jun. 2015.

[10] A. W. Colombo, S. Karnouskos, and T. Bangemann, “A system of systems view on collaborative industrial automation,” in Proc. IEEE Int. Conf. Ind. Technol. (ICIT), Feb. 2013, pp. 1968–1975.

[11] M. W. Maier, “Architecting principles for systems-of-systems,” Syst. Eng., vol. 1, no. 4, pp. 267–284, 1998.

[12] A. Bondavalli, S. Bouchenak, and H. Kopetz, Cyber-Physical Systems of Systems, Foundations—A Conceptual Model and Some Derivations: The AMADEOS Legacy. Cham, Switzerland: Springer, 2016.

[13] M. Jamshidi, Systems of Systems Engineering-Innovations for the 21st Century. Hoboken, NJ, USA: Wiley, 2009.

[14] R. Obermaisser, M. Abuteir, A. Khalifeh, and I. Dhiab el Diehn, “Systems-of-systems framework for providing real-time patient monitoring and care: Challenges and solutions,” in ITCs for Improving Patients Rehabilitation Research Techniques. Cham, Switzerland: Springer, 2014, pp. 129–142.

[15] R. Obermaisser, M. Abuteir, A. Khalifeh, and D. e. D. I. Abou-Tair, Systems-of-System framework for Providing Real-Time Patient Monitoring and Care: Challenges and Solutions. Berlin, Germany: Springer, 2015, pp. 129–142, doi: 10.1007/978-3-662-48645-0_12.

[16] A. Khalifeh, R. O. Obermaisser, M. Abuteir, and D. E. D. I. Abou-Tair, “Systems-of-systems framework for providing real-time patient monitoring and care,” in Proc. 8th Int. Conf. Pervasive Comput. Technol. Healthcare, 2014, pp. 426–429, doi: 10.4108/icst.pervasivehealth.2014.255324.

[17] D. Abou-Tair, S. Alouneh, A. Khalifeh, and R. Obermaisser, “A security framework for systems-of-systems,” in Advances in Computer Science and Ubiquitous Computing, J. J. Park, V. Loia, G. Yi, and Y. Sung, Eds. Singapore: Springer Singapore, 2018, pp. 427–432.

[18] S. Alouneh, D. e. D. I. Abou-Tair, A. Khalifeh, and R. Obermaisser, “Service identification framework for systems of systems based on MPLS technology,” in Advances in Computer Science and Ubiquitous Computing, J. J. Park, V. Loia, G. Yi, and Y. Sung, Eds. Singapore: Springer Singapore, 2018, pp. 359–364.

[19] D. E. D. I. Abou-Tair, A. Khalifeh, S. Alouneh, and R. Obermaisser, “Incremental, distributed, and concurrent service coordination for reliable and deterministic systems-of-systems,” IEEE Syst. J., vol. 15, no. 2, pp. 2470–2481, Jun. 2021.

[20] A. Khalifeh, D. E. D. I. Abou-Tair, S. Alouneh, R. Obermaisser, and M. Schmidt, “Admission control and resource allocation for distributed services in system-of-systems: Challenges and potential solutions,” in Proc. IEEE 13th Int. Symp. Ind. Embedded Syst. (IIES), Jun. 2018, pp. 1–4.

[21] S. Sharma and P. Kuila, “Design of dependable task scheduling algorithm in cloud environment,” in Proc. 3rd Int. Symp. Women Comput. Inform. (WCI), New York, NY, USA: ACM, 2015, pp. 516–521, doi: 10.1145/2791405.2791502.

[22] J.-J. Hwang, Y.-C. Chow, F. D. Anger, and C.-Y. Lee, “Scheduling precedence graphs in systems with interprocessor communication times,” SIAM J. Comput., vol. 18, no. 3, pp. 244–257, 1989, doi: 10.1137/0218016.
[23] G. C. Sih and E. A. Lee, “A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures,” IEEE Trans. Parallel Distrib. Syst., vol. 4, no. 2, pp. 175–187, Feb. 1993, doi: 10.1109/71.207593.

[24] D. Warneke and O. Kao, “Exploiting dynamic resource allocation for efficient parallel data processing in the cloud,” IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 6, pp. 985–997, Jun. 2011, doi: 10.1109/TPDS.2011.65.

[25] S. Singh and I. Chana, “A survey on resource scheduling in cloud computing: Issues and challenges,” J. Grid Comput., vol. 14, no. 2, pp. 217–264, Jun. 2016, doi: 10.1007/s10723-015-9359-2.

[26] Y. Huang, Y. Wang, and N. Liu, “Low-carbon economic dispatch and energy sharing method of multiple integrated energy systems from the perspective of system of systems,” Energy, vol. 244, Apr. 2022, Art. no. 122717. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544221029662

[27] P. S. Prakash, N. Naem, P. Ratel, and B. Nagel, “Aircraft architecture and fleet assessment framework for urban air mobility using a system of systems approach,” Aerosp. Sci. Technol., vol. 125, Jun. 2022, Art. no. 107072. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1270963821005824

[28] M. Mennenga, L. Büth, F. Cerdas, and C. Herrmann, “Synthetic emergence as a functional unit for the environmental assessment of a system of systems,” Proc. CIRP, vol. 90, pp. 393–398, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212827120301451

[29] S. Majidi and R. Obermaisser, “Evolutionary algorithm for incremental scheduling in systems of systems with real-time requirements,” in Proc. IECON 47th Annu. Conf. IEEE Ind. Electron. Soc., Oct. 2021, pp. 1–6.

[30] S. K. Roy, R. Devaraj, A. Sarkar, K. Maji, and S. Sinha, “Contention-aware optimal scheduling of real-time precedence-constrained task graphs on heterogeneous distributed systems,” J. Syst. Archit., vol. 105, May 2020, Art. no. 101706. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1270963820301583

[31] S. K. Roy, R. Devaraj, A. Sarkar, and D. Senapati, “SLAQA: Quality-level aware scheduling of task graphs on heterogeneous distributed systems,” ACM Trans. Embedded Comput. Syst., vol. 20, no. 5, pp. 1–31, Sep. 2021, doi: 10.1145/3462776.

[32] S. K. Roy, R. Devaraj, and A. Sarkar, “Contention cognizant scheduling of task graphs on shared bus-based heterogeneous platforms,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 41, no. 2, pp. 281–293, Feb. 2022.

[33] C. Guariniello, T. B. Marsh, R. Porter, and C. Crumbly, “Artificial intelligence agents to support data mining for SoS modeling of space systems design,” in Proc. IEEE Aeros. Conf., Mar. 2020, pp. 1–11.

[34] Z. Fang, J. Liao, and X. Zhou, “Improving system-of-systems agility through dynamic reconfiguration,” in Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC), Oct. 2020, pp. 4466–4472.

[35] P. Mazzetti, M. A. Latre, M. Bauer, R. Brumana, S. Braumann, and S. Nativi, “Energetic-OD virtual hubs: A brokered architecture for facilitating open data sharing and use,” in Proc. eChallenges e-Conf., Nov. 2015, pp. 1–11.

**DHIAH EL DIEHN I. ABOU-TAIR** received the Doctoral degree in computer science from the University of Siegen, Germany. He is currently an Associate Professor with the School of Electrical Engineering and Information Technology, German Jordanian University (GIU). He has been involved in several EU and German-funded research and capacity building projects. His research focus is on privacy laws and regulations adoption in information systems through an ontology-based approach. His research interests include privacy-enhancing technologies, security, system-of-systems, and the Internet of Things.

**AL’ KHALIFEH** (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of California at Irvine, Irvine, CA, USA, in 2010. He is currently the Vice Dean of the Deanship of Innovation and Technology Transfer (DT-TECH) and an Associate Professor with the Department of Electrical Engineering, German Jordanian University. His research interests include communication technology and networking with a particular emphasis on the Internet of Things, wireless sensor networks, and cloud computing. He was a recipient of the Fulbright Scholarship, from 2005 to 2007. He received numerous academic and technical awards/fellowships, such as The Leaders in Innovation Fellowships (LIF) 2021 administrated by the Royal Academy of Engineering and funded by the U.K. Government’s Department of Business, Energy and Industrial Strategy Newton Fund Program, 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. He is the IEEE Jordan Section Chair.

**SAMEER AL-DAHIDI** received the Ph.D. degree in energy and nuclear science and Technology from the Politecnico di Milano, Italy, in 2016. He is currently an Associate Professor with the Department of Mechanical and Maintenance Engineering, German Jordanian University. He is the author of several journal articles and conference publications. His current research interests include the development of analytics and models for PHM, operation, maintenance, RAMS analysis of engineering systems, the development of AI-based methods for renewable energy production prediction, renewable energy systems, and HVAC.

**SAHEL ALOUNEH** received the B.Sc. degree in electrical and computer engineering from the Jordan University of Science and Technology (JUST), Jordan, in 2000, and the M.Sc. and Ph.D. degrees from Concordia University, Canada, in 2004 and 2008, respectively. He is currently a Full Professor of electrical and computer engineering. He is also the Program Director of the Cyber Security Program, College of Engineering, Al Ain University, Abu Dhabi Campus, United Arab Emirates. He is also on sabbatical leave from the German Jordanian University. His research interests include computer and communication networks, big data security, cloud computing, software security, MPLS security and recovery, wireless networks security, software testing, and computer design and architecture.

**ROMAN OBERMAISSER** received the master’s degree in computer science from the Vienna University of Technology, in 2001, and the Ph.D. degree in computer science from the Vienna University of Technology, with Prof. Hermann Kopetz, in 2004. He was a Research Advisor with the Vienna University of Technology. He is currently a Full Professor at the Division for Embedded Systems, University of Siegen. He has also participated in numerous EU research projects (e.g., SAFEPOWER, universAAL, DECODES, and NextITTA) and was the co-ordinator of the European research projects, such as DREAMS, GENESYS, and ACROSS. He wrote a book on an integrated time-triggered architecture published by Springer-Verlag, USA. He is the author of several journal articles and conference publications. His research interest includes systems architectures for distributed embedded real-time systems. He received the Habilitation (“Venia Docendi”) Certificate for Technical Computer Science, in 2009.