Federated Edge Computing for Disaster Management in Remote Smart Oil Fields

Razin Farhan Hussain[1], **Mohsen Amini Salehi**[1], **Anna Kovalenko**[1], **Yin Feng**[2] **and Omid Semiari**[3]
[1]{razinfarhan.hussain1, amini}@louisiana.edu
[1]High Performance Cloud Computing (HPCC) Laboratory, School of Computing and Informatics,
[2]Department of Petroleum Engineering,
University of Louisiana at Lafayette, Lafayette, LA, USA
[2]yin.feng@louisiana.edu
[3]osemiari@georgiasouthern.edu
[3]Department of Electrical and Computer Engineering, Georgia Southern University, Statesboro, GA, USA

***Abstract*—Oil & Gas (O&G) industry is extending the extraction operation to remote offshore sites. Cost-effective, efficient, and nature-friendly oil extraction is a challenging issue in these remote sites, due to the disaster-prone nature of oil extraction process and hurdles in accessing these sites. To overcome these difficulties, smart oil fields use numerous sensors (*e.g.,* pipeline pressure, gas leakage, temperature sensors) and can generate more than a terabyte of data per day. The data are transferred to cloud datacenters via high-latency and unstable satellite communication, which is not suitable for latency-intolerant (urgent) disaster-related tasks. Edge computing can be deployed in oil rigs to process the latency-intolerant tasks, however, processing capacity of an edge system falls short at the time of a disaster—when several coordinated activities must be processed within a short time. To address this shortage, we propose robust smart oil fields operating based on a federation of edge computing systems, provisioned from nearby/mobile micro datacenters. Our solution achieves robustness by capturing uncertainties exist both in communication and computation of the federated environment and allocating urgent tasks so that the likelihood of their on-time completion is maximized. Evaluation results reflect significant performance improvement (up to 27%) of the proposed solution when compared to conventional solutions for smart oil fields.**

***Index Terms*—Smart Oil Field, Edge Computing, Cloud Datacenter, Sensors, Offshore Oil Field.**

I. Introduction

A. Smart Oil Fields

Petroleum has been unarguably one of the most essential elements of world economic growth throughout the past decades. For nearly two centuries, petroleum has been exploited as the primary natural source for many industrial products such as gasoline, natural gas, diesel, oil, asphalt, and plastic. Nonetheless, oil and gas (O&G) industries currently face several challenges mainly due to scarcity of petroleum reservoirs, requiring the companies to extract O&G at remote and adverse locations (*e.g.,* Golf of Mexico, Persian Gulf, West Africa) [1] where giant reservoir exist, hence, multiple oil extraction sites are built within a short distance. Operating at such remote sites is costly and constrained with limited crew and equipment. In addition, petroleum extraction is a fault-intolerant process and requires high-reliability *e.g.,* for drilling and downhole monitoring. Disasters, such as the Deepwater Horizon oil spill in 2010 [2], occurred due to faults in the safety system of the extraction. As such, strict regulations are being enforced by governments (*e.g.,* U.S. environmental protection agency (EPA)) to prevent disasters and minimize the environmental impacts of O&G extraction.

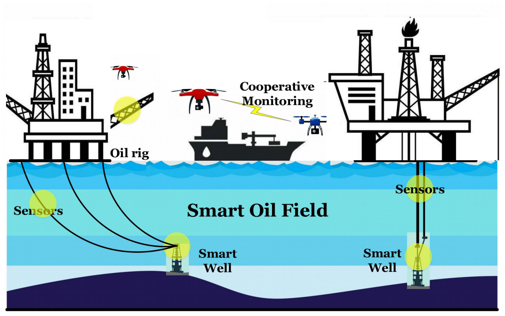

Fig. 1: A smart oil field scenario including different oil extraction and monitoring sensors.

To avoid potential flaws and disasters, oil fields are equipped with many cyber-physical devices (*e.g.,* sensors and actuators, as seen in Figure 1), and the concept of *smart oil field* has emerged. Smart oil fields leverage numerous sensors, including those for temperature, Hydrogen Sulphide (H2S) gas emission, pipeline pressure, air pollution, and flow monitoring. These sensors gather a large volume of data (up to two Terabytes per day [3]) that most need to be analyzed and used in a real-time manner. Various research works [3]–[7] have collectively emphasized the need for smart oil fields with following requirements:
(1) Real-time decision-making during the extraction process to manage the drilling operation, which is challenging when the operation is controlled remotely by the management team.
(2) Online monitoring of the site including rigs' structure, wells, and distribution lines to avoid any O&G leakage, corrosions identification, and future incidents prediction.
(3) Numerous sensors generate a large amount of data [8] that must be transferred to cloud datacenters for processing.

Services that utilize the sensors' data are categorized as either *latency tolerant* (aka non-urgent) or *latency intolerant* (aka urgent). Analyzing cost-efficiency of drilling, compressing and archiving captured surveillance videos, and generating weekly production reports are examples of non-urgent services. Such services are generally more compute intensive. In contrast, pipeline pressure alarm and gas (*e.g.,* H2S) leakage detection are instances of latency intolerant services. Such services have real-time constraints and must be processed within a short latency (deadline). For instance, preserving workers' safety in an oil field requires processing the data generated by H2S sensors in less than five seconds [9].

B. Challenges of the Current Smart Oil Field Solutions

Existing smart oil field solutions cannot meet the requirements of *remote* oil fields for the following two reasons:

1) Lack of reliable and fast communication infrastructure to onshore cloud datacenters for real-time processing of the extracted data.
2) Due to the harsh environment and shortage of manpower, more automation, and real-time processing is required to deal with abundant sensors and actuators.

Currently, remote smart oil fields use satellite communication to cloud and monitoring centers that are located in the mainland. However, satellite communication is known to be unstable and imposing a significant propagation delay leading to the latency in the order of seconds, that is intolerable for many real-time services in smart oil fields [10]. Current smart oil field solutions [11] do not consider the latency exists in communication between the oil fields and cloud datacenters. As such, the goal of this study is to enable the idea of smart oil fields in remote offshore sites. In this research, we propose a robust edge computing system for remote smart oil fields with high latency connectivity to cloud datacenters. The edge computing system is defined as micro datacenter that aims at handling latency-sensitive (*i.e.,* urgent) tasks. At the time of a disaster (*e.g.,* oil spill or gas leakage), different urgent activities must occur in the orchestrated manner to manage the disaster. For instance, real-time simulations must be conducted to predict oil spill expansion; emergency teams must be notified, and Unmanned Aerial Vehicles (UAVs) must be scheduled and dispatched for finer inspection [12]. However, edge computing resources are generally insufficient to handle such surges in demands.

C. Contributions of this Research

To overcome resource constraints of a single edge system and make it robust against the surge in demand, we leverage the edge computing systems available in nearby oil rigs, drillships, or even mobile micro datacenters and propose a mechanism to engage them upon demand. Although federating edge systems can potentially mitigate the shortage of resources, such a federated environment involves new challenges that must be addressed to achieve the intended robustness.

For a given task, the federated edge system imposes the stochastic transmission latency to a neighboring edge and the stochastic execution latency on the destination edge system. These latency times collectively are called *end-to-end latency*. For a latency-intolerant task, the end-to-end latency and its implied uncertainties must be captured, so that the federated environment can be helpful. As such, the research problem we consider is *how to design a dynamic federated edge computing system that is robust against uncertainties exist in both communication and computation and can handle surges in demand for latency-intolerant tasks during a disaster?*

To address this problem, we design a load balancer for each edge computing system that provides robustness by maintaining the federation view. The load balancer is aware of both computation and communication uncertainties exist in the federated environment and uses a probabilistic model to capture them. The probabilistic model provides us with the likelihood of meeting the latency constraint of the arriving task (*i.e.,* task's deadline). Next, we leverage the probabilistic model and develop a resource allocation heuristic for the load balancer to utilize the federation such that the edge system becomes robust against surges in task arrival. As the edge computing system has a central role in the smart oil field, its robustness leads to the robustness of the smart oil field, and subsequently, a cleaner and more cost-efficient O&G industry. In summary, The **contributions** of this paper are as follows:

- Proposing a resource allocation model that dynamically federates edge computing systems to enable robust smart oil fields.
- Establishing a probabilistic model to capture end-to-end uncertainties exist in the federated edge environment and calculate the probability of success for tasks in this environment.
- Developing a federation- and QoS-aware resource allocation heuristic based on the probabilistic model.
- Analyzing the performance of the federated edge computing system under various workload conditions.

The rest of the paper is organized in the following manner. Section II represents the system model. Section III states the end-to-end latency in edge computing systems. Section IV demonstrates the robust resource allocation using federated edge computing system while section V provides the performance evaluation and experiments performed in our research. Section VI presents related works. Finally, section VII concludes the paper with some future avenues for exploration.

II. System Model

The system model, shown in Figure 2, includes two-tier of computing systems where edge systems are located in local or in the first tier, and cloud datacenters are in the second tier. An *edge system* is defined as a set of machines with limited computational power, storage, and communication capacity [13] that work together under the same management platform (*i.e.,* resource manager) to offer various services required at the oil field. In analogy with gigantic cloud datacenters, the edge systems are known as micro/mini datacenters [14]. The edge system can be located within an oil rig structure or mounted on a drill-ship near the rig [15]. In disaster-prone environments, the edge systems are protected with temperature and water resistive materials. As such, we assume that the edge system itself is safe from oil field disasters.

Each edge system is equipped with a load balancer module that can offload tasks to the central cloud. In addition, in this work, we propose to enable the load balancer to communicate with its peer edge systems using an underlying wireless network. The load balancer decides about assigning (offloading) arriving tasks to its peers or central cloud based

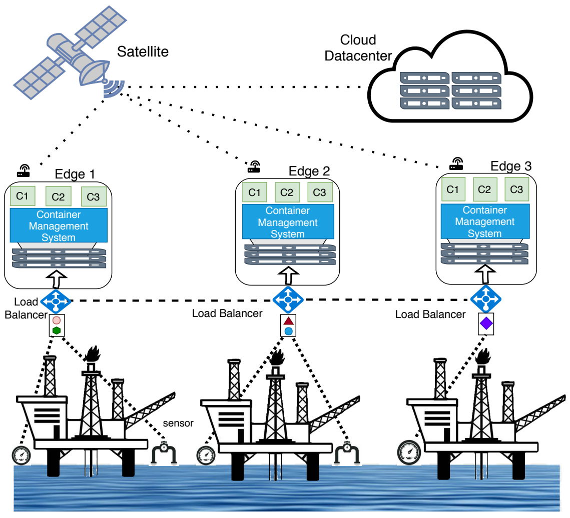

Fig. 2: Architecture of federated edge computing of smart oil field where sensor-generated tasks are sent to edge nodes for execution.

on the workload intensity or the type of arriving task requests. In particular, urgent tasks are suitable for processing at the edge or edge federation, whereas non-urgent tasks are suitable for processing on cloud datacenters. As such load balancer should assign urgent tasks to edge federation and non-urgent tasks are transferred to cloud datacenters. We consider heterogeneity across edge systems. That is, and some edge systems have more computational power (in terms of the number of processing cores and available memory) than others.

Different types of sensors (*e.g.,* temperature sensor, pressure sensor, gas sensor, camera) generate data that are consumed by heterogeneous tasks (defined as *task types*) to offer various services required in a smart oil field. The task types (*e.g.,* image processing for oil spill detection, toxic gas detection, weekly report generation) are assumed to be limited and known in advance. Also, each task is assumed to be independent of other tasks. The format [16] and size of generated data by some sensors can potentially vary, whereas, for some other sensors, they are constant. For example, images captured by cameras to detect oil spill can be of different sizes. This randomness serves as one primary reason for uncertainty in the execution time of the task type that processes images to detect an oil spill [17]. A contrary example is the data periodically generated by temperature sensors and is processed by a task type that identifies fire hazards in the oil field [18]. In the latter example, even though the data size does not vary, the task execution time can have uncertainty due to the workload of neighboring machines and multi-tenancy [19]. Apart from the execution time uncertainties exist within each edge system, due to heterogeneity, a task of a specific type can have different (*i.e.,* uncertain) execution times across different edge systems.

Upon arrival of a task of type i to an edge system j, an individual deadline (denoted δ_i) is assigned to the task based on its arrival time and the maximum latency the task can tolerate. It is noteworthy that the deadline of tasks depends on the service type they offer, and it varies from one service type to another. Tasks arrive at an edge system dynamically, and the arrival rate is not known in advance. Particularly, we concentrate on surge demands in the edge system that overload (*i.e.,* oversubscribed) the system. Thus, the arrival rate of tasks to the edge system is intense to the extent that it is not feasible to meet the deadlines of all the tasks. We assume that each arriving task is sequential and needs only one processing unit (*e.g.,* a processing core) for execution. In this system, a resource allocation method aims at maximizing the robustness of the edge system where robustness is defined as the number of tasks meeting their deadline constraint.

III. End-to-End Latency in Edge Computing Systems

Upon the arrival of a task request to the load balancer of an edge system, there are two types of latencies, namely communication and computational latencies, that together form the end-to-end latency. Several factors influence each one of these latencies and cause them to exhibit stochastic behavior. For these reasons, estimating end-to-end latency and capturing the stochastic nature of it is challenging in edge computing systems. In the following subsections, we elaborate on the influential factors in communication and computational latencies. Also, we provide a model to estimate the end-to-end latency while capturing its stochastic nature.

A. Estimating Communication Latency

Communication latency for a task request is the implication of transferring the task's data for processing and receiving the response. More specifically, communication latency is caused by *transmission latency* and *propagation latency*.

The transmission latency between any two points m and n (*e.g.,* two edge systems in the edge federation) for task t of type i, denoted $\Theta_i(m,n)$, is defined as the sum of uplink transmission latency, denoted $\tau_u(m,n,i)$, and downlink transmission latency, denoted $\tau_d(m,n,i)$. That is, we have $\Theta_i(m,n) = \tau_u(m,n,i) + \tau_d(m,n,i)$. Let $I_u(i)$ be the size of data payload (in bits), originally captured by a sensor, serving as input for task type i. Note that, for some sensors (*e.g.,* cameras), there can be randomness in the size of captured data, in every sensor reading. Also, let $R_u(m,n)$ represent the uplink bandwidth, through which the data is transmitted. T is the time required to transmit each data packet to the uplink channel (known as Transmission Time Intervals (TTI)). Then, the uplink latency is calculated based on Equation 1.

$$\tau_u(m,n,i) = \left\lceil \frac{I_u(i)}{R_u(m,n) \cdot T} \right\rceil \tag{1}$$

Similarly, the downlink latency is defined as Equation 2.

$$\tau_d(m,n,i) = \left\lceil \frac{I_d(i)}{R_d(m,n) \cdot T} \right\rceil \tag{2}$$

An orthogonal frequency-division multiplexing (OFDM) with total bandwidth W is divided equally into a set of k sub-channels (where $k \in K$) each with bandwidth w. Accordingly, the downlink bandwidth is defined based on Equation 3.

$$R_d(m,n) = w \cdot \sum_{k \in K} y_{mnk} \log_2(1 + \gamma_d(m,n,k)) \tag{3}$$

where $y_{mnk} = 1$, if sub-channel k is allocated, otherwise $y_{mnk} = 0$. As the wireless communication is prone to noise and interference from other edge systems in the federation, the value of $R_d(m,n)$ also depends on downlink *signal to* *noise plus interference ratio* (also known as *SINR* [20]). SINR is defined as the power of a particular signal divided by the sum of the interference power (from all the other interfering signals) along with the power of background noise. We note that, details of calculating uplink transmission latency ($\tau_u(m,n,i)$) is similar to those for downlink.

In edge federation, due to the vicinity, the propagation latency between edge systems is negligible. In contrast, the communication between edge systems and cloud datacenters is commonly achieved via satellite that introduces a substantial propagation latency [7]. The propagation latency, denoted τ_p, is calculated based on Equation 4.

$$\tau_p = 2 \cdot \frac{d(n,st)}{S_l} \tag{4}$$

In the Equation 4, $d(n,st)$ is the distance between edge n to satellite st and S_l is the propagation speed in medium or link. To calculate propagation latency in the round trip time, the fraction value should be doubled. Once we know propagation latency, the overall communication latency, denoted d_{comm}, to access cloud datacenter is calculated based on Equation 5.

$$d_{comm} = \Theta_i(m,n) + \tau_p \tag{5}$$

As we noticed, there are several factors that collectively form the communication latency with stochastic behavior. To capture this stochastic behavior, we treat communication latency as a random variable and model it using statistical distribution. That is, we represent the communication latency between any two points (*e.g.,* two edge systems in the federation) using a probability density function (PDF), built upon historical communication information [21]. Based on the central limit theorem, communication latency can be modeled using Normal distribution.

B. Estimating Computational Latency

Once the load balancer assigns arriving task request t to an edge system, the task has to wait in the scheduling queues of the edge system before its execution. For a given task t of type i, denoted t_i, its completion time (*i.e.,* computational latency) is influenced by the waiting time in the queue (queuing latency), plus the task's execution time (execution latency) on the machines of the assigned edge system. Importantly, both of these factors are stochastic, as a result, the task completion time exhibits a stochastic behavior.

The queuing latency of task t_i is dependent on the number and execution times of tasks ahead of it in the edge system. The stochasticity in execution time can be due to different task types and characteristics of machines in different edge systems. Even the execution time of tasks from the same type on homogeneous machines of the same edge system is stochastic. This can be because of variations in the size of data to be processed and multi-tenancy of tasks in the edge system [19]. Other factors, such as machine failure, can also be reasons for stochastic task execution time. To capture the stochasticity in computational latency, we consider the task completion time of each task type on each edge system as a random variable. Then, we model the computational latency using statistical distribution. That is, the computational latency is modeled using PDF, built upon historical completion time information of each task type on each edge system. Based on the central limit theorem, the computational latency of each task type on each edge system can be modeled using Normal distribution.

C. Estimating End-to-End Latency

Once we estimate the communication and computational latencies, their compound latency forms the end-to-end latency. More specifically, the compound latency can be obtained by convolving the PDF of communication latency with the PDF of the computational latency. For an arriving task t_i to a load balancer, let N_i be PDF of its communication latency to another edge system in the federation. Also, let M_i be PDF of the computational latency of t_i on the other edge system. Then, the end-to-end latency for t_i, denoted E_i, is calculated as $E_i = N_i \circledast M_i$.

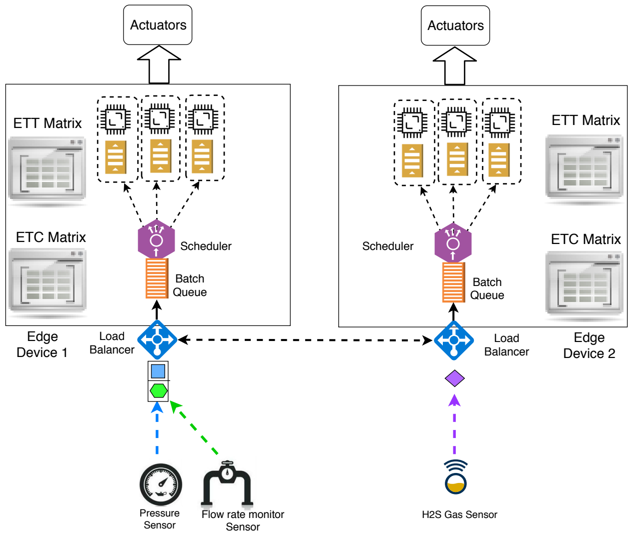

Fig. 3: An edge system with load balancer module that facilitates edge federation. Task requests generated by sensors are received by the load balancer module and are assigned to the edge system that maximizes the likelihood of success for the task.

IV. Robust Resource Allocation Using Federated Edge Computing System

The synopsis of the proposed resource allocation model in the federated edge computing system is demonstrated in Figure 3. The resource allocation model utilizes a *load balancer* module that is the main enabler of edge federation. Every edge system is equipped with a load balancer that, for each arriving task, it determines the appropriate edge system (either the receiving edge or to a neighboring one) where the task has the highest likelihood of completion before its deadline.

The functionality of load balancer is particularly prominent to cope with the uncertainty exists in task arrivals (*e.g.,* during disaster time) and make the edge system robust against it. The load balancer operates in *immediate mode* [22] and assigns arriving tasks to the appropriate edge system, immediately upon task arrival. The appropriateness is characterized based on the edge system that maximizes the probability of the task meeting its deadline (known as the *probability of success*). The probability of success for task t_i with deadline δ_i can be calculated for each neighboring edge system, by leveraging the end-to-end latency distribution of executing task t_i on that system. To avoid repetitive task reassignment and compound latency, we determine that once a task assignment decision is made, the task cannot be re-allocated.

The resource allocation of each edge system leverages the historical information of computational and communication latencies to build PDF of their distributions. For that purpose, each load balancer maintains two matrices, namely Estimated Task Completion (ETC) [23] and Estimated Task Transfer (ETT), to keep track of computational and communication latencies for each task type on each neighboring edge system. Entry $ETC(i,j)$ keeps the PDF of computational latency for task type i on edge system j. Similarly, entry $ETT(i,j)$ keeps the PDF of communication latency for task type i to reach edge system j. The entries of ETC and ETT matrices are periodically updated in an offline manner and they do not interfere with the real-time operation of the load balancer.

Upon arriving task t_i, load balancer of the receiving edge can calculate the end-to-end latency distribution of t_i on any neighboring edge j, using $ETC(i,j)$ and $ETT(i,j)$. The end-to-end distribution can be used to obtain the probability of completing t_i before its deadline, denoted $p_j(t_i)$, on any of those edge systems. We have: $p_j(t_i) = \mathbb{P}(E_i \leq \delta_i)$. We note that the probability calculation for task t_i on the receiving edge does not imply further communication latency. As such, for the receiving edge r we have: $p_r(t_i) = \mathbb{P}(M_i \leq \delta_i)$. In the next step, the edge system that provides the highest probability of success is chosen as a suitable destination to assign task t_i. This implies that task t_i is assigned to a neighboring edge system, only if even after considering the communication latency, the neighboring edge provides a higher probability of success.

It is noteworthy that the probability of success on a neighboring edge can be higher than the receiving edge by a non-significant amount. In practice, a task should be assigned to a neighboring edge, only if the neighboring edge system offers a substantially higher probability of success. To understand if the difference between the probabilities is substantial, we leverage confidence intervals (CI) of the underlying end-to-end distributions, from which the probability of success for receiving and remote edges are calculated. More specifically, we determine a neighboring edge offers a significantly higher probability of success for a given task, only if CI of end-to-end distribution of the neighboring edge does not overlap with the CI of end-to-end distribution of the receiving edge.

The pseudo-code provided in Algorithm 1 expresses the robust task assignment heuristic that load balancer utilizes to take advantage of federated edge system and increase the robustness of the system. The heuristic is called *Maximum Robustness (MR)* and invoked upon arrival of a new task t_i to the load balancer of an edge system. Based on the deadline of the arriving task (δ_i), the algorithm first calculates the probability of success for t_i on the receiving edge and on its neighboring edge systems (Step 1-7 in Algorithm 1). Then, in Step 8, the calculated probabilities are sorted in the descending order. If the probability of success on the receiving edge is higher, then the task is allocated to the receiving edge system (Step 9). Otherwise, CI of the end-to-end latency distribution for the neighbor with the highest probability of success is compared against receiving edge CI. If the CIs do not overlap, then task t_i is assigned to the neighboring edge (Step 12). Otherwise, the same procedure is performed for the rest of the neighbors of the receiving edge system. If there is no no-overlap neighbor found then, task t_i is assigned to the receiving edge system (default assignment in Step 9).

Algorithm 1: Task assignment algorithm for load balancer.

```
Input  : Task $t_i$; $ETC$ and $ETT$ matrices; $G$ (set of
         neighboring edge systems)
Output: Chosen edge $j \in G$ to assign $t_i$
1  $p_r(t_i) \leftarrow$ Probability of success on receiving edge $r$
2  foreach edge system $j \in G$ do
3      $p_j(t_i) \leftarrow$ Probability of success on neighbor edge $j$
4      if $p_j(t_i) > p_r(t_i)$ then
5          Add $p_j(t_i)$ to $P$, as a potential edge for assignment
6      end
7  end
8  Sort elements of set $P$ in descending order
9  Consider receiving edge $r$ as default assignment for $t_i$
10 foreach $p_j \in P$ do
11     if CI of $E_j$ does not overlap with CI of $N_r$ then
12         Choose edge $j$ as destination and assign $t_i$ to it
13         Exit the loop
14     end
15 end
```

V. Performance Evaluation

A. Experimental Setup

We have used EdgeCloudSim [24], which is a discrete event simulator for performance evaluation. We simulate five edge systems (micro-datacenters) each one with eight cores and [1500, 2500] Million Instructions Per seconds (MIPs) computational capacity. Cores of each edge system are homogeneous: however, different edge systems have different MIPs that represents the heterogeneity across the edge systems. We also consider a cloud datacenter with 40,000 MIPs to process non-urgent tasks. Task within each edge is mapped in the first come first serve manner. The bandwidth to access cloud is based on satellite communication and set to 200 Mbps, and the propagation delay is 0.57 seconds [17].

In each workload trial, generated to simulate load of a smart oil field, we consider half of the tasks represent urgent and the other half represent non-urgent tasks. Each task is of a certain type that represents its service type. In each workload trial, urgent tasks are instantiated from two different task types and non-urgent tasks are instantiated from two other task types. The execution time of each task instantiated from a certain type is sampled from a normal distribution, representing that particular task type. Each task is considered to be sequential (requires one core) and its execution time is simulated in the form of MIPs. Poisson distribution (with different means for different task types) is used to generate the inter-arrival rate of the tasks and simulate task arrival during oversubscription periods. The number of tasks in each workload trial is varied to represent different oversubscription levels.

Deadline for task i in a workload trial is generated as: $\delta_i = arr_i + \beta \cdot avg^i_{comp} + \alpha \cdot avg^i_{comm} + \varepsilon$, where arr_i is the task arrival time, avg^i_{comp} is average computational latency of the task type across edge systems, and avg^i_{comm} is average communication latency. β and α are coefficients, respectively, represent computation and communication uncertainties, and ε is the slack of other uncertainties exist in the system. We consider maintaining ETC and ETT matrices in every edge system and update them in every 10% of the workload execution. The entries of these matrices are considered as normal distribution as mentioned in the system model. For accuracy, each experiment was conducted 30 times and the mean and 95% confidence interval of the results are reported.

B. Baseline Task Assignment Heuristics for Load Balancer

Minimum Expected Completion Time (MECT): This heuristic [21] uses the ETC matrix to calculate the average expected completion time for the arriving task on each edge system and selects the edge system with the minimum expected completion time.

Maximum Computation Certainty (MCC): This heuristic (used in [25]) utilizes ETC matrix to calculate the difference between the task's deadline and average completion time (called certainty). Then, the task is assigned to the edge that offers the highest certainty.

Edge Cloud (EC): This heuristic operates based on conventional edge computing model where no federation is recognized. Specifically, urgent tasks are assigned to the receiving edge and non-urgent tasks are assigned to the cloud datacenter.

C. Experimental Results

1) Analyzing the Impact of Oversubscription: The main metric to measure the robustness of an oversubscribed edge system in a smart oil field is the deadline miss rate of tasks. In this experiment, we study the performance of our system by increasing the number of tasks sensors generate (*i.e.,* oversubscription level). Figure 4 shows the results of varying the number of arriving tasks (from 1,500 to 7,500 in the horizontal axis) on deadline miss rate (vertical axis) when different task assignment heuristics is applied.

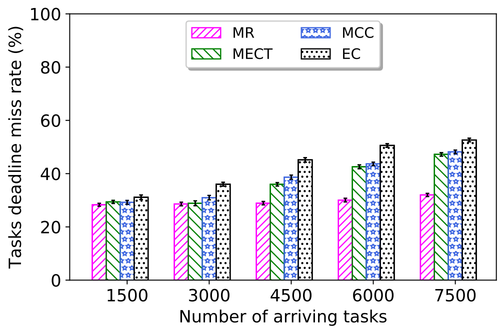

Fig. 4: The impact of increasing oversubscription level (number of arriving tasks) on deadline miss rate using different task assignment heuristics in the load balancer.

In Figure 4, it is visible that as the number of tasks increases, the deadline miss rate grows for all of the heuristics. Under low oversubscription level (1,500 tasks), MR, MECT, and MCC perform similarly. However, as the system gets more oversubscribed (4,500 tasks) the difference becomes substantial. With 7,500 tasks, MR offers around 16% lower deadline miss rate than MECT and MCC and approximately 21% better than EC. The reason is that MR captures end-to-end latency and proactively utilizes federation, only if it has a remarkable impact on the probability of success. Nonetheless, EC does not consider federation, and other baseline heuristics only consider the computational latency. We can conclude that considering end-to-end latency and capturing its underlying uncertainties can remarkably improve the robustness, particularly, when the system is oversubscribed (*e.g.,* at a disaster time).

2) Analyzing the Impact of Urgent Tasks Ratio: In this experiment, while the system is oversubscribed with 8,000 tasks, we vary the percentage of urgent tasks in the workload from 10% to 90% (horizontal axis in Figure 5) and in each case we measure the deadline miss rate (vertical axis in Figure 5).

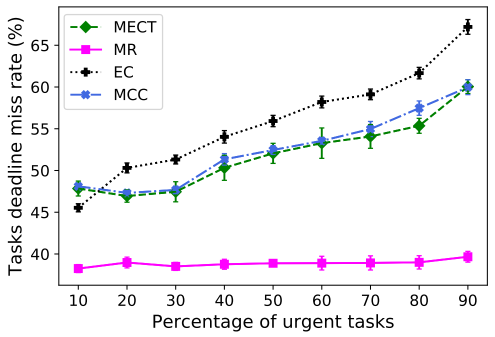

Fig. 5: Impact of increasing urgent tasks on deadline miss rate.

As we can see in Figure 5, EC provides lower deadline miss rate than MECT and MCC at 10% the urgent tasks (*i.e.,* 90% no-urgent tasks). The reason is that EC redirects non-urgent tasks to the cloud datacenter and the remaining urgent tasks can complete on-time on the receiving edge system. Although MECT and MCC use the cloud for non-urgent tasks too, they utilize federation for some of the remaining urgent tasks without considering end-to-end latency. Hence, their performance is degraded. However, for more than 20% urgent tasks, EC performs worse than MECT and MCC, because it cannot utilize the federation. In all cases, we observe that MR outperforms other heuristics due to consideration of end-to-end latency and its underlying uncertainties.

3) Analyzing Communication Overhead of Edge Federation: Although we showed in the previous experiment that using federation improves system robustness, we are unaware of the communication overhead of task assignment in the federated environment. Therefore, in this experiment, we evaluate the communication latency imposed as a result of applying different task assignment heuristics. Specifically, we measure the mean communication latency overhead (vertical axis in Figure 6) induced to each task, for the various number of arriving tasks (horizontal axis in Figure 6).

Figure 6 shows that MECT and MCC cause higher average communication latency. The reason is that these heuristics do not consider the communication latency and aggressively redirect tasks to the same edge system, making that particular network link (between receiving edge and redirected edge sys-

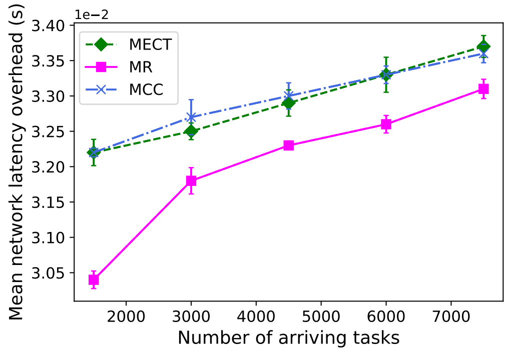

Fig. 6: Mean communication latency overhead introduced to each task in edge federation by different heuristics.

tem) congested. In contrast, MR that considers communication latency and redirect tasks more conservatively, only if the improvement in the probability of success is substantial.

4) Analyzing Average Makespan of Tasks: Different task assignment heuristics cause various computational latencies for the tasks. To understand the computational latency, we measure the average makespan of tasks, resulted by applying various task assignment heuristics.

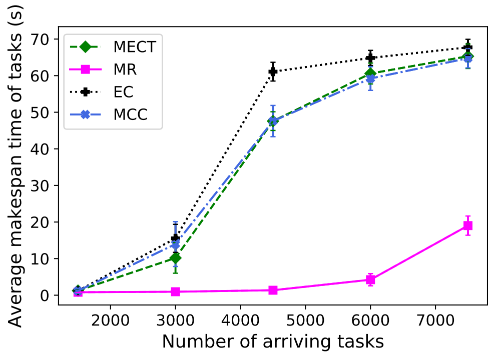

Fig. 7: Average makespan of tasks using various task assignment heuristics.

Figure 7 demonstrates that EC leads to the maximum average makespan time. The reason is that EC does not utilize federation, making the receiving edge system highly oversubscribed while other neighboring edge systems are underutilized. Hence, average makespan time rapidly rises after the receiving edge is saturated with 3,000 tasks. MECT and MCC do not consider the stochastic nature of task completion time; hence, they can potentially assign arriving tasks to one edge and oversubscribe that. As a result, the average makespan of tasks rises. In contrast, MR considers stochastic nature of end-to-end latency and calculates the probability of success on neighboring edge systems. Besides, it assigns tasks to a neighboring edge system, only if it offers a sufficiently higher probability of success. Hence, MR offers the lowest average makespan time than other heuristics.

VI. Related Works

To improve the response time of latency intolerant services, edge computing systems have been exploited in the literature from the network latency perspective. Lorenzo *et al.* [26] proposed a resource allocation model for wireless edge systems that harvest unused resources of mobile devices to mitigate network congestion. The proposed model utilizes solutions at physical, access, networking, application, and business layers to reinforce the network robustness. This work solely considers networking latency and not end-to-end latency. In [27], Chang *et al.* proposed an optimized resource migration scheme from mobile IoT devices to heterogeneous Cloud-Fog-Edge computing environment that is aware of the resource-constrained nature of edge devices. It focuses on the performance gain of process migration and assigns tasks based on their run time expectations on the participating systems. The problem of heterogeneous data acquisition from sensors in different sectors (e.g., upstream, midstream, downstream) of smart oil fields are addressed in [18] where khan *et al.* proposed an IoT based architecture to enable data acquisition process more simple, secure, robust, reliable and quick. There are several other works (*e.g.,* [28]–[30]) that either do not consider emergency situation (oversubscription) or ignore the uncertainties exists in federated edge environments. In our previous work [31], the main focus was on optimizing the wireless network while no resource allocation performed at the edge system.

There has been limited work conducted by researchers from academia and industry to model networking characteristics of remote smart oil fields [3], [4], [6]. The key challenges for remote sites that have not been addressed by previous works are as follows: First, the communication medium between the remote sites and management centers depends on satellite communication [17] which is not suitable for real-time decision making during a disaster or oversubscription time. Second, current works (*e.g.,* [32]) consider wireless support from cellular Base Station at a nearby location that is not the case for remote offshore oil fields.

Although there are several research works (*e.g.,* [33]–[35]) on smart oil fields, there is no rigorous study on resource provisioning for disaster management applications using edge computing and by considering low-connectivity to the back-end cloud datacenters. Instead, major efforts have concentrated on big data analytics and machine learning applications for smart oil fields. Parapuram *et al.* [36] studied the use of artificial intelligence and data mining model to predict geomechanical properties of future oil wells. To reduce exploration and drilling costs, in [37], machine learning methods have been developed by Cameron *et al.* .

VII. Conclusions

In this paper, our goal was to provide a smart oil field that is robust against disasters and surges in real-time service requests. To achieve that, we presented dynamic federation of edge computing systems, exist in nearby oil fields. Within the federated environment, we captured two sources of uncertainty, namely communication and computation, that are otherwise detrimental to the real-time services. The federation is achieved by a load-balancer module in each edge system that is aware of the end-to-end latency between edge systems and can capture the stochasticity in it. The load balancer leverages this awareness to find the edge system that can substantially improve the probability of success for each arriving task.

Experimental results demonstrate that our proposed federated system can enhance the robustness of edge computing systems against uncertainties in arrival time, communication, and computational latencies. We concluded that the load balancer could be particularly useful (by up to 27%) for higher levels of oversubscription. Even for naïve load balancing methods (MCC and MECT) in the federation, the performance improvement is approximately 13%. In future we plan to explore the use of Markov Chain model in load balancer to predict the success rate and to improve the overall performance of the system. We also plan to explore heterogeneity within each edge system, in addition to heterogeneity across edge systems.

Acknowledgments

We thank anonymous reviewers of the paper. Portions of this research were conducted with high performance computational resources provided by Louisiana Optical Network Infrastructure [38]. This research was supported by the Louisiana Board of Regents, under grant number LEQSF(2016-19)-RD-A-25 and grant number LEQSF(2017-20)-RD-B-06.

References

[1] D. Mathieson *et al.*, "Forces that will shape intelligent-wells development," *Journal of Petroleum Technology*, vol. 59, pp. 14–16, Aug. 2007.

[2] H. K. White, P.-Y. Hsing, W. Cho, T. M. Shank, E. E. Cordes, A. M. Quattrini, R. K. Nelson, R. Camilli, A. W. J. Demopoulos, C. R. German, J. M. Brooks, H. H. Roberts, W. Shedd, C. M. Reddy, and C. R. Fisher, "Impact of the deepwater horizon oil spill on a deep-water coral community in the gulf of mexico," *Proceedings of the National Academy of Sciences*, vol. 109, pp. 20303–20308, Feb. 2012.

[3] Cisco, "A New Reality for Oil & Gas: Data Management and Analytics," White paper, April 2015.

[4] S. Prabhu, E. Gajendran, and N. Balakumar, "Smart oil field management using wireless communication techniques," *International Journal of Inventions in Engineering & Science Technology*, pp. 2454–9584, Jan. 2016.

[5] N. G. Franconi, A. P. Bunger, E. Sejdi, and M. H. Mickle, "Wireless communication in oil and gas wells," *Energy Technology*, vol. 2, pp. 996–1005, Oct. 2014.

[6] WoodMackenzie, "Why are some deepwater plays still attractive?," White paper, Sep. 2017.

[7] ABB, "Field Area Communication Networks for Digital Oil and Gas Fields," White paper, October 2014.

[8] M. A. Hayes and M. A. M. Capretz, "Contextual anomaly detection in big sensor data," in *Proceedings of IEEE International Congress on Big Data*, pp. 64–71, June 2014.

[9] David Riddle, "Danger and detection of hydrogen sulphide gas in oil and gas exploration and production," White paper, April 2009.

[10] A. A. Bisu, A. Purvis, K. Brigham, and H. Sun, "A framework for end-to-end latency measurements in a satellite network environment," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2018.

[11] J. Pickering, S. Sengupta, M. Pfitzinger, *et al.*, "Adopting cloud technology to enhance the digital oilfield," in *Proceedings of International Petroleum Technology Conference*, Dec. 2015.

[12] J. Cho, G. Lim, T. Biobaku, S. Kim, and H. Parsaei, "Safety and security management with unmanned aerial vehicle (uav) in oil and gas industry," *Journal of Procedia Manufacturing*, vol. 3, pp. 1343–1349, Jul. 2015.

[13] B. I. Ismail, E. M. Goortani, M. B. Ab Karim, W. M. Tat, S. Setapa, J. Y. Luke, and O. H. Hoe, "Evaluation of docker as edge computing platform," in *Proceedings of the IEEE Confernece on Open Systems (ICOS)*, pp. 130–135, Aug. 2015.

[14] W. Hu, Y. Gao, K. Ha, J. Wang, B. Amos, Z. Chen, P. Pillai, and M. Satyanarayanan, "Quantifying the impact of edge computing on mobile applications," in *Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems*, APSys '16, pp. 5:1–5:8, 2016.

[15] H.-W. Lee and M.-I. Roh, "Review of the multibody dynamics in the applications of ships and offshore structures," *Journal of Ocean Engineering*, vol. 167, pp. 65–76, 2018.

[16] R. B. Rodrigo Escobar, David Akopian, "A sensor data format incorporating battery charge information for smartphone-based mhealth applications," vol. 9411, Mar. 2015.

[17] M. Skedsmo, R. Ayasse, N. Soleng, M. Indregard, *et al.*, "Oil spill detection and response using satellite imagery, insight to technology and regulatory context," in *Proceedings of the Society of Petroleum Engineers (SPE) International Conference and Exhibition on Health, Safety, Security, Environment, and Social Responsibility*, Apr. 2016.

[18] W. Z. Khan, M. Y. Aalsalem, M. K. Khan, M. S. Hossain, and M. Atiquzzaman, "A reliable internet of things based architecture for oil and gas industry," in *Proceedings of 19th International Conference on Advanced Communication Technology*, pp. 705–710, Feb. 2017.

[19] X. Li, M. A. Salehi, M. Bayoumi, and R. Buyya, "Cvss: A cost-efficient and qos-aware video streaming using cloud services," in *Proceedings of 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, CCGRID '16, pp. 106–115, May 2016.

[20] S. Mukherjee, "Distribution of downlink sinr in heterogeneous cellular networks," *Journal of Selected Areas in Communications*, vol. 30, pp. 575–585, Apr. 2012.

[21] M. A. Salehi, J. Smith, A. A. Maciejewski, H. J. Siegel, E. K. Chong, J. Apodaca, L. D. Briceño, T. Renner, V. Shestak, J. Ladd, *et al.*, "Stochastic-based robust dynamic resource allocation for independent tasks in a heterogeneous computing system," *Journal of Parallel and Distributed Computing (JPDC)*, vol. 97, pp. 96–111, Jun. 2016.

[22] C. Denninnart, J. Gentry, and M. A. Salehi, "Improving robustness of heterogeneous serverless computing systems via probabilistic task pruning," in *Proceedings of the 33rd IEEE International Parallel & Distributed Processing Symposium Workshops*, May. 2019.

[23] C. O. Diaz, M. Guzek, J. E. Pecero, G. Danoy, P. Bouvry, and S. U. Khan, "Energy-aware fast scheduling heuristics in heterogeneous computing systems," in *Proceedings of International Conference on High Performance Computing & Simulation*, pp. 478–484, Jul. 2011.

[24] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," in *Proceedings of the 2nd International Conference on Fog and Mobile Edge Computing*, FMEC '17, pp. 39–44, May 2017.

[25] R. F. Hussain, M. A. Salehi, A. Kovalenko, S. Salehi, and O. Semiari, "Robust resource allocation using edge computing for smart oil fields," in *Proceedings of the 24th International Conference on Parallel and Distributed Processing Techniques & Applications*, Aug. 2018.

[26] B. Lorenzo, J. Garcia-Rois, X. Li, J. Gonzalez-Castano, and Y. Fang, "A robust dynamic edge network architecture for the internet-of-things," *Journal of IEEE Network*, vol. 32, pp. 8–15, Jan. 2017.

[27] C. Chang, A. Hadachi, and S. Srirama, "Adaptive edge process migration for iot in heterogeneous cloud-fog-edge computing environment," *arXiv preprint arXiv:1811.10939*, Nov. 2018.

[28] S. Wunderlich, J. A. Cabrera, F. H. Fitzek, and M. Reisslein, "Network coding in heterogeneous multicore iot nodes with dag scheduling of parallel matrix block operations," *Journal of IEEE Internet of Things*, vol. 4, pp. 917–933, May 2017.

[29] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *Proceedings of the 7th IEEE International Symposium on Service Oriented System Engineering (SOSE)*, pp. 494–502, Mar. 2013.

[30] Z. Hu, Y. Wei, X. Wang, and M. Song, "Green relay station assisted cell zooming scheme for cellular networks," in *Proceedings of 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, pp. 2030–2035, Aug. 2016.

[31] M. M. K. Tareq, O. Semiari, M. A. Salehi, and W. Saad, "Ultra reliable, low latency vehicle-to-infrastructure wireless communications with edge computing," *arXiv preprint arXiv:1808.06015*, 2018.

[32] B. Prabhu, E. Gajendran, and N. Balakumar, "Smart oil field management using wireless communication techniques," Jan. 2017.

[33] G. H. Aggrey, D. R. Davies, A. A. Ajayi, M. R. Konopczynski, *et al.*, "Data richness and reliability in smart-field management - is there value?," in *Proceedings of SPE Annual Technical Conference and Exhibition*, Sep. 2006.

[34] A. Ekebafe, Abraham; Ogan, "Smart well technology application in deepwater field development," in *Proceedings of Nigeria Annual International Conference and Exhibition*, Aug. 2012.

[35] F. G. Van den Berg, "Smart fields - optimising existing fields," in *Proceedings of Digital Energy Conference and Exhibition*, Aug. 2007.

[36] G. K. Parapuram, M. Mokhtari, J. B. Hmida, *et al.*, "Prediction and analysis of geomechanical properties of the upper bakken shale utilizing artificial intelligence and data mining," in *Proceedings of the SPE/AAPG/SEG Unconventional Resources Technology Conference, Austin, TX, USA*, pp. 24–26, Jul. 2017.

[37] D. Cameron *et al.*, "Big data in exploration and production: Silicon snake-oil, magic bullet, or useful tool?," in *Proceedings of SPE Intelligent Energy Conference & Exhibition*, Apr. 2014.

[38] "Louisiana Optical Network Infrastructure (LONI) Resources QB2." http://hpc.loni.org/resources/hpc/system.php?system=QB2, accessed Aug 10, 2018.