A Performance Cost/Benefit Analysis of Adaptive Computing in the Tactical Edge

Alessandro Amato*, Harrie Bastiaansen†, Willem Datema†, Mattia Fogli‡,
Roberto Fronteddu*, Raffaele Galliera*, Johan van der Geest†, Thomas Kudla§,
Pablo Sanchez¶, Niranjan Suri*‖, Susan Watson**
* *Florida Institute for Human and Machine Cognition (IHMC)*, Pensacola, FL, USA
{aamato, rfronteddu, rgalliera, nsuri}@ihmc.org
† *The Netherlands Organisation for Applied Scientific Research (TNO)*, The Hague, The Netherlands
{harrie.bastiaansen, willem.datema, johan.vandergeest}@tno.nl
‡ *University of Ferrara*, Ferrara, Italy
mattia.fogli@unife.it
§ *Fraunhofer Institute for Communication, Information Processing and Ergonomics*, Wachtberg, Germany
thomas.kudla@fkie.fraunhofer.de
¶ *University of Cantabria*, Santander, Spain
sanchez@teisa.unican.es
‖ *US Army DEVCOM Army Research Laboratory (ARL)*, Adelphi, MD, USA
niranjan.suri.civ@army.mil
** *Defence Research and Development Canada*, Ottawa, Canada
susan.watson@forces.gc.ca

***Abstract*—Tactical Edge Computing is a promising solution to address the challenges of processing and managing large volumes of data collected by sensors deployed at the tactical edge. Tactical edge networks often lack sufficient bandwidth to transfer data at high rates. Much of the sensor raw data might be uninteresting and would waste networking resources to transmit. Edge computing solves these problems by staging the processing capabilities for sensor data at the edge, close to the data source. Local data processing reduces bandwidth needs in disadvantaged tactical networks. Furthermore, it could reduce the latency of data processing and avoid congestion in the tactical network. However, static solutions that deploy edge services could also be problematic because of the unpredictability of the tactical edge – the source / location of the data may not be known a priori, and there could be significant changes, such as sensors and nodes going offline. Therefore, tactical edge computing solution must be adaptive, where services are deployed on demand based on the sensors tasked and the mission requirements. This paper presents the work of the NATO IST-193 work on adaptive tactical edge computing and its analysis on the adaptivity benefits and overhead costs involved.**

***Index Terms*—Federated Cloud Architecture, Tactical Clouds, Tactical Networks, Tactical Edge Computing, Adaptive Orchestration, Resource Discovery, Service Deployment, Benefits, Performance Costs**

I. Introduction

Increased capabilities in sensing, processing, storage, and communication are ever more being deployed in military missions, facilitating data acquisition and processing [1]. However, these capabilities are often connected through disadvantaged tactical networks, which are typically characterized by limited bandwidth, intermittent connectivity, and variable latency. Consequently, there is growing relevance for federated and adaptive tactical edge and cloud architectures [2] for joint coalition missions. These architectures offer significant advantages for military and Information Technology (IT) operations in terms of data processing efficiency, survivability (as highlighted in [3]), and overall improvement in military mission effectiveness [4].

The North Atlantic Treaty Organization (NATO) Information Systems Technology (IST)-168 Research Task Group (RTG) *"Adaptive Information Processing and Distribution to Support Command and Control"* has analysed the performance of various cloud distributions for container workload orchestration in (simulated) Tactical Edge Environments (TEEs) [5]. The findings suggest that state-of-the-art, commercial off-the-shelf, Kubernetes-based orchestrators could be deployed in a federated and adaptive cloud architecture to facilitate data sharing and processing capabilities among mission partners over disadvantaged tactical networks.

Building on the outcomes of the IST-168 RTG, the NATO IST-193 RTG *"Edge Computing at the Tactical Edge"* has further developed an architecture for deploying federated cloud architectures in disadvantaged TEEs. Recognizing the limitations of static solutions for deploying edge services due to unpredictable availability of edge computing resources and network connectivity at the tactical edge, an adaptive tactical edge computing solution has been developed and analysed. This solution enables services (workloads) to be adaptively and dynamically deployed across the federations of edges in the TEE, taking into account the instantaneous TEE situation of workload requirements and edge processing and networking

resources. In this work, we present the results of the NATO IST-193 RTG on adaptive tactical edge computing.

This paper details the architectural implementation and provides an analysis of the benefits of such an adaptive solution, along with the associated overhead and performance costs. The structure of the paper includes a discussion of the broader perspective on adaptive tactical cloud computing in Section II, covering its objective, architecture, and performance parameters. Section III addresses an illustrative and representative scenario together with the experiment setup, while Section IV presents and assesses the experimental results. Finally, Section V provides the overarching discussion, conclusions and outlines future work.

II. Approach: Objectives, Architecture and Performance Parameters

The objectives and the architectural approach of the NATO IST-168 and IST-193 RTGs for adaptive tactical edge computing have previously been described in [2], [5]. The following recapitulates this approach, with an emphasis on the capabilities to support adaptivity in TEEs.

A. Objective: Adaptive Workload Deployment in TEEs

Coalition tactical operations involve diverse mission partners with varying access control policies and security measures for the processing and networking resources available in their tactical clouds. Balancing the need for each mission partner to maintain sovereignty over its own resources with the challenges of jointly optimizing the use of those constrained resources for data processing is therefore a key objective in improving the information posture at the tactical edge.

The primary objective of the architecture for tactical edge computing is to optimize adaptivity in information processing and workload orchestration across a federation of tactical clouds, ensuring fine-grained resource control for mission partners while encouraging resource sharing.

B. Architecture: Federated Tactical Edges

In the IST-168 study [2], the architectural principles and artifacts were described for the implementation of a federated cloud architecture in coalition TEEs. Individual clouds share their internal resources with other clouds in the federation through well-defined Application Programming Interfaces (APIs). This enables standardized inter-cloud interactions and communications, prevents compromising each partner's sovereignty with respect to information about availability of services, resources, and ability to deploy workloads, and allows a vendor-agnostic approach ensuring interoperability and compatibility among partner clouds. Standards-based cloud container technology is to be used like the Open Container Initiative (OCI).

Within each individual cloud, Kubernetes-based orchestration can be used for *intra-cloud workload deployment orchestration* [5]. This paper focuses on the costs and benefits of adaptive orchestration of workloads between the clouds in the federation, i.e. adaptive *inter-cloud workload deployment orchestration*. Fig. 1 depicts the proposed high-level architecture [2].

This architecture relies on the already deployed Kubernetes infrastructure, with additional architectural services that are responsible for federation by exposing specific capabilities through APIs across all clouds in the federation. Communication between such services within a Kubernetes instance is handled through Custom Resource Definitions (CRDs). When one of these services updates a CRD, another will have a watcher for it and, therefore, be triggered to action. As the figure shows, four architectural services are deployed on each cloud:

- *Federated Resource Discovery (FRD)* - The FRD API provides information on both available services (available for deployment and already deployed) and resources (e.g., CPUs, RAM, storage). It queries its own cloud and other available clouds to collect and aggregate this information. Additionally, it exposes resource and service information about its own cloud through an external API after applying any policy constraints.
  Each instance of the FRD queries its own cloud and other clouds in the federation by invoking the API of the FRD in each cloud. The FRD utilizes the Kubernetes API to fetch locally available resources and running services. Furthermore, it communicates with Harbor [6], which is an open source registry for Kubernetes, to fetch the services available for deployment in each cloud. It also queries SENSEI [7] to fetch network link metrics between cloud instances. These calls result in updating, creating, or deleting various Custom Resources (CRs) based on the CRDs that were defined for the FRD. These CRs are Cloud Resources, Service Definitions, and Service Instances. A Cloud Resource contains information on CPUs, RAM, storage, and network links connecting to the other known federated clouds. Service Definitions, instead, provide information about what services are available for deployment in each cloud, while Service Instances provide information about the services deployed in each cloud. The CRDs are updated every n seconds, where n is a configurable parameter.
- *Federated Adaptive Orchestrator (FAO)* - The FAO is responsible for creating deployment plans across the federation. These plans are developed based on the information provided by the FRD. The FAO does not expose an external API.
  The FAO is triggered upon the creation of a Federated Workload, a CR that specifies a workload. This workload may include several services, outlining their requirements in terms of computational resources (e.g., CPU) and connectivity (e.g., at least 1 Mbit/s between Service A and Service B). Essentially, a Federated Workload tells the FAO the necessary services to accomplish a specific task, and whether to utilize services already deployed within the federation or to initiate new instances. The FAO then generates a Federated Deployment, which is

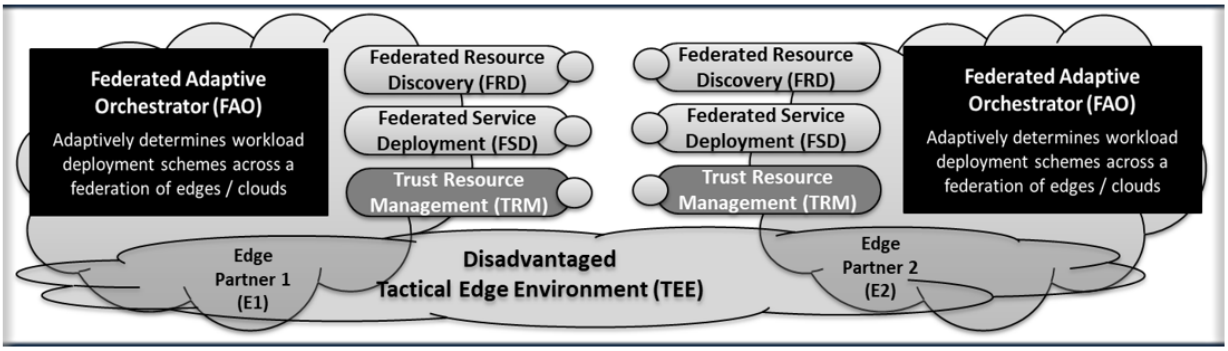

Fig. 1: High-level architecture for adaptive inter-cloud workload deployment orchestration.

also a CR. Given the dynamic nature of TEEs, the FAO continuously monitors the Federated Deployments it has established. If changes occur, such as a cloud going offline that affects a Federated Deployment, the FAO promptly revises its previous decisions and, if feasible, reallocates those services elsewhere within the federation.

- *Federated Service Deployment (FSD)* - The FSD is responsible for deploying services, e.g. a database, either on its own cloud or on other available clouds. A request for deployment can be initiated by either its own orchestrator or by another FSD. An instance of the FSD is deployed on each federated cloud. It also exposes an external API with policy constraints for communicating between all FSDs. On each cloud the FSD monitors—by using the Kubernetes API—the Federated Deployments generated by the FAO. A Federated Deployment contains information about which services should be deployed where. If a new service is to be deployed, there are two options. First, the service is to be deployed locally from the point of view of the FSD. If the service is available in the local registry, it is deployed using Helm [8]. Second, the service is to be deployed in a remote cloud. In this case, the FSD checks if the requested Service Definition is already available in the remote cloud. If so, it calls the remote FSD to do a deployment of that service. If the Service Definition is not available, it fetches the service Definition, Helm chart and OCI images from the local registry and send this data to the remote could, where it gets pushed into the remote repository. Then, it calls the remote FSD to do a deployment of that service on the remote cloud.
- *Trust Resource Management (TRM)* - The TRM is responsible for monitoring activities in a single federated cloud and updates its policies that other services, namely the FSD, FRD, FAO, must enforce. It does not expose an external API. The TRM collects security monitoring information, or metrics, such as from the cloud ingress controller, from Kubernetes, and from the other intra-cloud services; primarily the FRD and FSD, via the CRDs. The TRM uses these security metrics to compute trust values in the range $[0, 1]$ for all partner clouds in the federation. The trust values are dynamically updated as the TRM observes the outcomes of interactions and resource sharing with partner clouds. It then updates its own CRD, i.e., Federated Trust, with these trust values for use by other intra-cloud services, for example the FAO, to assist with decision-making. The TRM also manages access control policy, which may be either pre-configured or trust-based. Thus partner clouds in the federation may be granted differing levels of access to information about resources and to resources themselves. To compute trust values, the TRM must collect per-partner-cloud security metrics. This functionality is enabled by an authentication and identity management system that links each interaction and deployed service to its originator in the federation. In Fig. 1 the shading of the TRM service differs from that of the FRD and FSD service, indicating that the TRM service will be developed further in the IST-193 RTG project. It is out-of-scope in the remainder of this paper.

To enable cross-cloud service deployment in an adaptive federated cloud infrastructure, the two APIs exposed by the FRD and the FSD service must be identical across all clouds. While the internal implementation of each service may vary among clouds, the approach for implementing each service as outlined above should be regarded as a reference implementation.

C. Performance Parameters: Adaptivity Costs and Benefits

The drawbacks associated with static solutions for deploying edge services serve may be addressed by the dynamic deployment of services (workloads) across a federation of tactical clouds, referred to as adaptive tactical edge computing, enhancing the information posture at the tactical edge. The remainder of this paper considers the validity of this approach. In particular, experiments were conducted to assess the advantages and drawbacks of adaptivity in the tactical edge, focusing on a set of adapitivity benefits and cost performance metrics:

1) Adaptivity Benefit Metrics: The (main) metric expressing the benefits of adaptivity in a TEE is the *performance gain for concurrent service requests*. This metric expresses the number

of service requests that can concurrently be handled within the boundaries of acceptable and practically usable service response times. As such, it shows the advantages of using the (adaptive) tactical edge computing approach for end-users.

2) Adaptivity Cost Metrics: The (main) cost metrics to support adaptivity in a TEE are the bandwidth impact, the time efficiency of the FAO service and the performance cost of the FRD, FSD, and FAO services:

- *Bandwidth impact:* The bandwidth impact is due to the continuous interactions between the edges for querying the resources available and monitoring of the service status (by the FRD service) and for initiating deployment of services across clouds (by the FSD service). Moreover, the SENSEI tool requires continuous interactions to monitor the status of the connectivity infrastructure at the tactical edge.
- *Time efficiency of the FAO service:* This assesses the duration required by the FAO service to create a deployment plan (i.e., Federated Deployment) or update existing plans in response to changing conditions, such as a cloud going offline or changes in mission priorities.
- *Performance cost of the FRD, FSD, and FAO services:* This delves into the examination of the CPU and memory usage of the FRD, FSD, and FAO services and the network bandwidth usage of the FRD and FSD services (which are the only ones that require inter-cloud communications).

It is important to note that the actual instantiation of services and distribution of data sets in the adaptive federated cloud also depend significantly on the size of the containerized services and the volume of data to be distributed. However, since various use cases involve different service and data sizes, this aspect will not be the focal point of the performance assessment of adaptivity discussed in the remainder of this paper. Furthermore, it is assumed that the services are preconfigured and available at each cloud, eliminating the need to account for their distribution in the analysis of adaptivity within the federated cloud environment. An illustrative and representative video analysis scenario will be presented in the subsequent section.

III. Experiments: Scenario and Set-up

A. Scenario: Vehicle Recognition and Tracking

The illustrative and representative scenario for the assessment of the adaptivity performance parameters involves a joint NATO military mission in an urban area. The mission partners operate a federated and adaptive infrastructure as described in the previous section. The amount and type of processing resources and the trust level of the individual mission partner's edge instances may differ. A dismounted soldier sees a suspicious car speeding away from its location. The scenario is composed of two stages.

In the first stage, the soldier takes video footage of the car. The soldier issues a report of the event to the mission Head Quarters (HQ). The video is uploaded to a nearby vehicle of the soldier's nation, being part of the federated and adaptive tactical cloud. The report to the HQ doesn't include the license plate number of the car as it is not recognizable from the video. The HQ decides to conduct follow-up actions that require a processing-intensive License Plate Recognition (LPR) service on the video data. The LPR service is already deployed at the HQ but cannot be deployed in the cloud storing the video footage, due to insufficient resources in that cloud. Hence, the FRD service in the soldier's cloud is invoked and queries the FRD APIs running in the mission partner's clouds for getting information on which services and resources are available across the federation. Accordingly, the FAO formulates a deployment plan that may either necessitate a new deployment of the LPR service or allow for the reuse of the existing instance at the HQ, depending on the current conditions. Subsequently, the FSD behave accordingly. The FRD regularly provides status monitoring information to the FAO which may opt for re-deployment in case of failed execution, e.g. in case the status of the cloud or connectivity conditions in the TEE change. The results of the LPR service are returned to the soldier's cloud for further handling and distribution.

In the second stage, the car is identified as 'suspicious' and it is decided to track the car throughout the city by means of detecting the same license plate for video footage from camera's mounted on other mission partner's vehicles and therefore originating in other edges of the federated cloud infrastructure. As the various edges may have many vehicles with mounted cameras, multiple concurrent requests to the LPR service may be needed originating from the same edge. Alernatively, the car may be tracked by analyzing street camera video footage, similarly leading to multiple concurrent requests to the LPR service.

We assumed that the HQ is connected with the federated tactical clouds at the edge, henceforth defined as Edge Clouds (ECs), through satellite communications, while the ECs are interconnected with each other via military radios, which typically offer better connectivity than satellites. Table I lists the configured available bandwidth in both cases, i.e., EC-to-HQ and EC-to-EC, and describes the two videos used in the experiments. These videos have identical content but vary in video quality, which is responsible for the difference in their sizes. The video was recorded by a street camera capturing the back of a car speeding away in a Unity simulation. It is worth noting that the bandwidth is not shared. This means that each EC has a dedicated 256 Kbit/s link available to the HQ and each EC-to-EC pair has a dedicated 1 Mbit/s link.

TABLE I: Bandwidth and video size parameters

Parameters	Value
EC-to-HQ Available Bandwidth	256 Kbit/s
EC-to-EC Available Bandwidth	1 Mbit/s
Low-Resolution Video Size	0.5 MB
High-Resolution Video Size	13.0 MB

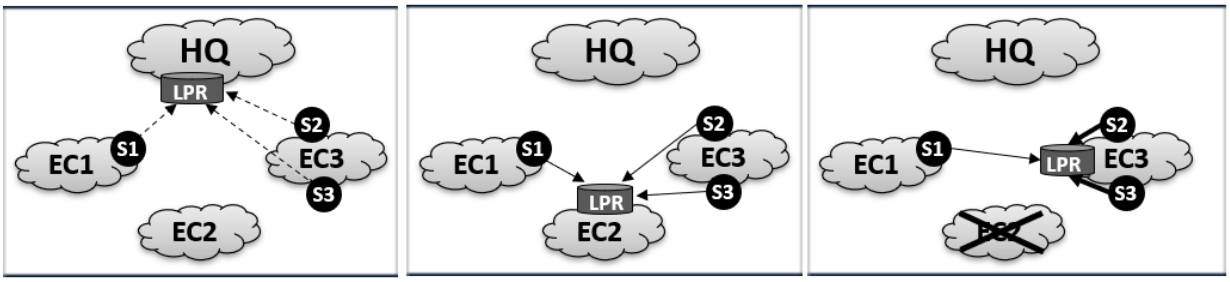

Fig. 2: Tactical edge cases: no tactical edge (l), static tactical edge (m) and adaptive tactical edge (r).

B. Experiment Set-up: Primary Components

The adaptivity performance assessment requires an experiment set-up that resembles a realistic TEE with various tactical edge cases under disadvantaged network conditions, as depicted in Fig. 2.

The testbed is composed of four clouds, each owned by a different mission partner. These include three mission partner's clouds (designated as EC1, EC2, and EC3) and one cloud functioning as the HQ. In each cloud, there are three Virtual Machines (VMs): one serving as a Kubernetes control plane and two as workers. Every VM is equipped with 2 vCPUs, 4 GB of RAM, and 64 GB of storage. Within each cloud, there are no communication constraints among the VMs (i.e., between the control plane and the workers). The HQ differs from the mission partner clouds in that its worker VMs have enhanced resources, with each worker equipped with 4 vCPUs and 8 GB of RAM. The testbed was provisioned and configured in a reproducible manner using Terraform [9] and Ansible [10], respectively.

The following components have been deployed on the testbed:

- *Network Emulation (EMANE)*
  The emulation of the TEE is achieved using EMANE (Extensible Mobile Ad-hoc Network Emulator) [11]. EMANE is an open-source network emulator that can be used to apply a variety of communication effects. In this particular configuration, EMANE is used to control the characteristics of the network link between each partner cloud to the other partner clouds. To achieve network isolation, a unique VLAN ID is assigned to each cloud, ensuring that each operates within its isolated network segment. When communication is required, EMANE performs the packet forwarding from the sender VLAN to the receiver VLAN, enabling communication between the designated parties. In addition, communication effects that may be applied include capacity constraints for unicast and multicast traffic (bits/sec), latency (milliseconds), packet loss (0-100%), jitter (milliseconds), and ordered delivery. For this particular experiment, only the capacity constraints were specified. Specifically, EMANE was used to constrain EC-to-HQ and EC-to-EC communication links as specified in Table I.
- *Network Sensing (SENSEI)*
  SENSEI (for Smart Estimation of Network State Information) [7] is a microservice-based framework designed to perform network resource discovery and distribution over constrained tactical networks. By relying on traffic exchanged between SENSEI instances and by other applications, SENSEI can passively estimate network metrics that describe the link between groups of nodes, such as network latency, bandwidth, and throughput. SENSEI's passive approach to network discovery is particularly important in the context of resource-constrained links because it enables network awareness without saturating links that can already be congested. Moreover, SENSEI implements adaptive algorithms that control which information is exchanged between instances based on relevance and available resources. SENSEI's information is provided to applications designed to perform network-aware adaptation through a Representational State Transfer (REST) interface.
- *License Plate Recognition (LPR) service*
  The LPR service is a distributed Deep Learning (DL) application designed for recognizing license plates in video footage. It identifies and extracts the characters from each license plate detected, employing a distributed architecture integrating several components.
  The License Plate Recognition service (LPR) application comprises four distinct, containerized services: the License Plate Detector (LPD), the Optical Character Recognition (OCR) service, a Neural Autonomic Transport System (NATS) [12] server, and a MinIO [13] instance. The LPD and OCR services, which are the core components, utilize two separate DL models performing their respective duties. The NATS server enables a publish-subscribe messaging pattern to facilitate communication between LPD and OCR services. Meanwhile, the MinIO instance manages the storage and retrieval of video footage.
  Upon initialization, the LPD service loads a customized Ultralytics Yolo [14] model, specifically trained for license plate detection. The OCR service, instead, initializes the Tesseract Open Source OCR Engine [15] with the most recent and accurate Tesseract Long Short-Term Memory (LSTM) English language model available.
  The LPR service accepts processing requests through its API. These requests include the MinIO instance's

TABLE II: Response time [s] of LPR service: No Tactical Edge vs. Static Tactical Edge

Case	Video Size	No. of Concurrent Requests (CRs)		
		1	2	3
No Tactical Edge (HQ-to-EC communications, 256 Kbit/s)	0.5 MB	150.28 ± 1.63	179.61 ± 5.67	224.55 ± 3.44
	13 MB	616.15 ± 1.16	660.95 ± 1.75	1072.35 ± 11.63
Static Tactical Edge (EC-to-EC communications, 1 Mbit/s)	0.5 MB	150.94 ± 1.21	159.08 ± 2.90	208.31 ± 5.60
	13 MB	312.82 ± 2.89	316.21 ± 4.86	443.45 ± 4.12

address, along with the specific bucket and object name referencing the video. After retrieving the object, the LPR processes the video frame-by-frame. For each detected license plate, an asynchronous request is published on the "OCR subject" via the NATS system. This request contains an encoding of the grayscale-converted segment of the frame within the predicted bounding box. Upon receiving a request, the OCR service performs a series of morphological transformations on the detected area to diminish image noise and extract a clear Region of Interest (RoI). The Tesseract model then processes this refined image as a single line of text. The resulting text string is sent back in response to the request. After processing all frames, the LPR service sorts the received strings from the OCR based on their frequency of occurrence. The final response to the processing request comprises these ranked and sorted strings, completing the recognition process for the given video.

IV. Results: Adaptive Tactical Edge Computing

In Section II-C, the metrics for both the benefits and costs of the proposed adaptive edge computing architecture have been described. The results of the benefits and cost analysis on these metrics are addressed in the subsequent paragraphs of this section.

A. Adaptivity Benefits

The benefits of the adaptive edge architecture is expressed by means of the *performance gain in concurrent service requests* in terms of service response times. To this end, Table II shows the response times for the requested license plate result as returned to the requesting Edge Cloud (EC), i.e. EC1 as depicted in Fig. 2 which the soldier taking part in. The results presented in the table are averaged over 10 experiment runs. The measured response times consist of the transfer times of the video files and the processing time for the LPR service. The measured transfer times of the video files correspond well to the values as also theoretically calculated, the measured processing time for the LPR service is approximately 2 minutes and 30 seconds for both the low-resolution (i.e., 0.5 MB) and high-resolution (i.e., 13.0 MB) video sizes.

The table includes two cases. For the first case, only the HQ hosts an LPR service instance, i.e. no tactical edge is used as depicted in the left image in Fig. 2. It has to serve both the LPR service requests coming from EC1 for the first stage of initial vehicle recognition and the concurrent LPR service requests stemming from EC3 for the second stage of the vehicle recognition and tracking scenario as described in Section III-A. For the second case, the LPR service instances in the tactical edge can be used as depicted in the middle image in Fig. 2. This can be done for both stages of the vehicle recognition and tracking scenario.

As the table shows, a significant performance gain for using a tactical edge is already achieved when simulaneously executing a limited number of concurrent service requests. This is due to the required times for data transfer of the video files from the edge to the HQ and the LPR service processing time. Note that in the case of multiple concurrent service requests, the entire video is processed by different LPR service instances. The rationale is to simulate a scenario where different videos, potentially recording the target car, are available at various locations. Adaptivity in the edge architecture may take care of dynamic transtion from the first to the second case when the response times for the LPR service running at the centralized HQ become too large to be of practical use.

Similarly, the performance gain of adaptivity in the edge architecture arises when a tactical edge running the LPR service instances becomes completely unavailable, for example, due to adversarial activities or moving out of range, resulting in a loss of connectivity to the rest of the network. Consequently, in such cases the service becomes inaccessible, underscoring the importance of adaptive orchestration in such a dynamic environment. This adaptation is crucial for leveraging the benefits of edge computing while ensuring service availability over time. This situation is depicted in the right image in Fig. 2.

B. Adaptivity Costs

The cost implications of the adaptive edge architecture are quantified by evaluating several factors. These include the bandwidth impact of the FRD and FSD services, and SENSEI, the time efficiency of the FAO service, as well as the performance cost, specifically CPU and memory utilization, of the FRD, FSD, and FAO services. These services were configured to expose the necessary metrics, which were subsequently collected using Prometheus [16]. Prometheus is a monitoring system and time series database designed to periodically scrape metrics from Kubernetes and services deployed through Kubernetes. The data for this analysis were collected over a 10-minute time window. EMANE was used to make EC2 go offline.

The *bandwidth impact* is due to the inter-cloud interactions of the FRD service, the FSD service, and SENSEI. The amount of incoming and outgoing data depends on several

TABLE III: Inter-cloud bandwidth impact of the FRD, FSD, and SENSEI services

Service	Traffic In [Bytes/s]	Traffic Out [Bytes/s]
FRD	246.75	21.62
FSD	0	0.33
SENSEI	1545	1545

factors, including the number of federated clouds in the TEE, the frequency of the interactions, and the amount of data being transferred per interaction. Both the FRD service and SENSEI interactions are bilateral (peer-to-peer) between the federated clouds, therefore the bandwidth impact will be linearly dependant on both the number of clouds in the federation and the interaction frequency. For the FRD service, the amount of data being transferred per interaction depends on the number of resources being exchanged in the format of CRs (see Section II-B) and the protocol used, with a non-zero lower boundary in case of no CRs being included. The FSD service only needs interactions when a new service instance is being deployed. Table III details the inter-cloud bandwidth impact of the FRD, FSD, and SENSEI services.

The table illustrates that the bytes sent and received by the system are minimal, measured in the order of bytes. Specifically, an average FRD instance sends approximately 21.66 bytes/s to gather information from other clouds, and in return, receives about 246.75 bytes/s. In more complex scenarios, the bandwidth impact could be more significant. However, it's important to note that these exchanges could be optimized further. One approach could be to implement strategies that reduce the frequency of updates, such as not transmitting an update unless there's a substantial change from the previous version. Another option could be to explore different applications and serialization protocols. The current version of FRD utilizes HTTP request/response mechanisms with JSON serialization for the body content. The impact of the FSD on bandwidth is around 0 because it generates traffic only when the FAO service creates or updates a Federated Deployment. In this experiment, there was a single update, i.e., when EC3 went offline. The overhead caused by SENSEI is more significant and in the order of a few Kilobytes. Each instance sends an average of 1545 bytes per second to each neighbor. This data is used to convey network statistics and estimates about the link with the involved neighbor. Most of the footprint is caused by the fact that there is no significant other traffic that SENSEI can leverage to perform packet pair estimation (this traffic would have to be generated by a middleware capable of sending packet pairs, for example using the Mockets [17] transport protocol), thus forcing the service to periodically inject probes to estimate bandwidth. This overhead could be further reduced by limiting updates to network changes or increasing the time between estimations. SENSEI additionally offers two other operational modes: fully passive and on-demand. In the fully passive mode, SENSEI generates reports only when it accumulates sufficient information, potentially resulting in operating with outdated network knowledge. On the other hand, the on-demand mode prioritizes obtaining a current view of the network status, albeit with the downside of significant delays in the inference process. These modes cater to various scenarios; for instance, the fully passive mode may be suitable for emission control tasks where occasional delays are acceptable. However, it's important to note that while these modes are integral to SENSEI's functionality, they were not the primary focus of this work.

TABLE IV: Time efficiency [s] of the FAO service

Percentile				
50th	75th	90th	99th	99.9th
0.65	0.77	0.94	2.13	2.46

Table IV showcases the *time efficiency of the FAO service*. On average, the FAO service takes about 661 ms to update its decision plans. 99.9 % of the updates took at most 2.46 s. These data are related to over 10000 updates carried out by the FAO service. In this case, the update was the reallocation of the LPR service from EC3 to EC2, triggered by EC2 going offline. This duration accounts for the time taken by the FAO service from the moment the FRD service updates the resources, to the point where the FAO modifies its existing Federated Deployments. The resource allocation algorithm in its current iteration is implemented in Python with the OR-Tools CP-SAT solver [18]. If the solver finds multiple solutions, indicating that several deployment plans are available, the current implementation of the FAO prioritizes maximizing trust. In these experiments, each cloud was assigned a static trust level, i.e., a number between 0 and 1. In scenarios with a larger number of clouds, there is a possibility that this implementation might not scale effectively. For these more complex scenarios, Machine Learning (ML) algorithms could offer a more feasible solution to create and update deployment plans within a reasonable timeframe.

The *performance cost of the FRD, FSD, and FAO services* includes both CPU and memory utilization. In this regard, Table V provides the CPU and memory utilization for each of these services. As shown, such services are very frugal. More complex scenarios may be more challenging, especially from the FAO perspective, which would be needed to make more difficult and/or frequent orchestration decisions.

V. Discussion, Conclusions, and Future work

This paper has presented the overall architecture and described the major components that are part of the adaptive and federated cloud computing framework developed by the IST-193 Research Task Group. A specific exemplar problem was introduced and used as the basis for analysing the performance benefits as well as the overhead involved in the adaptivity mechanisms for the architecture. Based on the results as presented in this paper, it may be observed that:

- The communications links at the tactical edge are often constrained and become the limiting factor for handling large volumes of sensor data generated at the edge. Tactical edge computing helps to address this problem

TABLE V: CPU and memory performance cost of the FRD, FSD, and FAO services

Service	CPU [milliCPU]	Memory [MB]
FRD	230	106.05
FSD	20	241.61
FAO	30	119.43

by processing the data at or close to the point of generation, thereby improving response times and overall performance. This can be observed by the reduction in the response time as the number of concurrent requests increases.

- Static solutions that do not adapt dynamically to changing circumstances will fail due to the high variability that typically occurs at the tactical edge. Node mobility and adversarial actions may cause network disconnections as well as node failures, which can cause statically deployed solutions to fail. Hence, a dynamic approach is necessary that can re-orchestrate deployed workflows when circumstances change.
- The bandwidth impact and performance cost of the federated FRD, FSD and FAO services for introducing adaptivity in the federated tactical edge is relatively low. Hence, adaptivity may yield high benefits at low performance cost.

Note that the experimental network used in this paper was fairly simple, with just a handful of nodes. Hence, the time-efficiency of the FAO service for the configurations explored in this paper is low. Further analysis is needed, taking into account more advanced FAO algorithms based on ML concepts.

Moreover, as future work, we intend to explore additional TEE scenarios that could benefit from adaptive and federated tactical edge computing. One area to explore is training or retraining ML models at the edge with data that is collected at the edge. We also intend to further integrate the Trust Resource Management (TRM) capability, which will keep track of the trust level of different federated clouds and influences the FAO in the decision-making process.

Finally, we intend to perform additional experimentation within the Anglova scenario [19], which will provide a larger and dynamic TEE, requiring a more continuous adaptivity mechanism. The Anglova scenario will allow us to conduct a comprehensive analysis of the proposed architecture's adaptivity costs, scalability, and robustness, factoring in a realistic number of nodes and the effects of node mobility. We also plan to experiment with other resource allocation algorithms, for example, that leverage Multi-agent Reinforcement Learning (MARL) concepts.

ACKNOWLEDGMENT

The work presented in this article is being done as part of the IST-193 RTG *"Edge Computing at the Tactical Edge"*. We would like to thank the NATO IST-Panel for providing us the opportunity to do this highly relevant and interesting research and the individual participating partners for providing valuable inputs within a stimulating and cooperative setting.

REFERENCES

[1] M. Tortonesi, A. Morelli, M. Govoni, J. Michaelis, N. Suri, C. Stefanelli, and S. Russell, "Leveraging internet of things within the military network environment — challenges and solutions," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016, pp. 111–116.

[2] H. Bastiaansen, J. v. d. Geest, C. v. d. Broek, T. Kudla, A. Isenor, S. Webb, N. Suri, M. Fogli, B. Canessa, A. Masini, R. Goniacz, and J. Sliwa, "Federated control of distributed multi-partner cloud resources for adaptive c2 in disadvantaged networks," *IEEE Communications Magazine*, vol. 58, no. 8, pp. 21–27, 2020.

[3] K. Zaerens, "Enabling the benefits of cloud computing in a military context," in *2011 IEEE Asia-Pacific Services Computing Conference*, 2011, pp. 166–173.

[4] W. Smith, G. Kuperman, M. Chan, E. Morgan, H. Nguyen, N. Schear, B. Vu, A. Weinert, M. Weyant, and D. Whisman, "Cloud computing in tactical environments," in *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, 2017, pp. 882–887.

[5] T. Kudla, M. Fogli, S. Webb, G. Pingen, N. Suri, and H. Bastiaansen, "Quantifying the performance of cloud-oriented container orchestrators on emulated tactical networks," *IEEE Communications Magazine*, vol. 60, no. 5, pp. 74–80, 2022.

[6] "Harbor," https://goharbor.io/, 2023, [Online; accessed 20-12-2023].

[7] R. Fronteddu, A. Morelli, M. Mantovani, B. Ordway, L. Campioni, N. Suri, and K. M. Marcus, "State estimation for tactical networks: Challenges and approaches," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, 2018, pp. 1042–1048.

[8] "helm - The package manager for Kubernetes," https://https://helm.sh/, 2024, [Online; accessed 04-01-2024].

[9] "Terraform - it enables you to safely and predictably create, change, and improve infrastructure." https://github.com/hashicorp/terraform, [Online; accessed 2024-01-05].

[10] "Ansible - a radically simple it automation system," https://github.com/ansible/ansible, [Online; accessed 2024-01-05].

[11] "EMANE- Extendable Mobile Ad-Hoc Network Emulator," https://github.com/adjacentlink/emane, 2023, [Online; accessed 21-12-2023].

[12] "Nats - the cloud native messaging system," https://github.com/nats-io, [Online; accessed 2024-01-05].

[13] "Minio - the object store for ai data infrastructure," https://github.com/minio/minio, [Online; accessed 2024-01-05].

[14] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[15] A. Kay, "Tesseract: An open-source optical character recognition engine," *Linux J.*, vol. 2007, no. 159, p. 2, jul 2007.

[16] "Prometheus - a monitoring system and time series database," https://github.com/prometheus/prometheus, [Online; accessed 2024-01-05].

[17] N. Suri, M. Tortonesi, M. Arguedas, M. Breedy, M. Carvalho, and R. Winkler, "Mockets: a comprehensive application-level communications library," in *MILCOM 2005 - 2005 IEEE Military Communications Conference*, 2005, pp. 970–976 Vol. 2.

[18] "Google or tools - cp-sat solver," https://developers.google.com/optimization/cp/cp_solver, [Online; accessed 2024-01-05].

[19] N. Suri, J. Nilsson, A. Hansson, U. Sterner, K. Marcus, L. Misirlioglu, M. Hauge, M. Peuhkuri, B. Buchin, R. in't Velt, and M. Breedy, "The anglova tactical military scenario and experimentation environment," in *2018 International Conference on Military Communications and Information Systems (ICMCIS)*, 2018, pp. 1–8.