Building the Cloud Continuum with REAR

Building the Cloud Continuum with REAR

Stefano Galantino*, Elisa Albanese**, Nasir Asadov‡, Stefano Braghin§, Francesco Cappa*, Andrea Colli-Vignarelli*, Amjad Yousef Majid††, Eduard Marin‖, Jacopo Marino*, Lorenzo Moro†, Liubov Nedoshivina§, Fulvio Risso*, Domenico Siracusa¶, Antonio Skarmeta‡‡, and Luca Zuanazzi**
*Dept. of Control and Computer Engineering, Politecnico di Torino, Torino, Italy
†Consorzio TOP-IX, Torino, Italy
‡Technische Universität Berlin, Berlin, Germany
§IBM Research Europe, Dublin, Ireland
‖Telefonica Research, Barcelona, Spain
**RSE Ricerca sul Sistema Energetico, Milano, Italy
††Martel Innovate, Amsterdam, the Netherlands
¶Fondazione Bruno Kessler, Trento, Italy
‡‡Universidad de Murcia, Murcia, Spain

***Abstract*—The computing continuum combines computational resources and services from edge to cloud, promising enhanced efficiency and resilience with respect to the traditional siloed-based approach. This study presents the REAR (Resource Advertisement and Reservation) protocol, which tackles the complexities of managing resources within this continuum. REAR establishes standardized interfaces to enable interoperability, enhances resource allocation efficiency, and maintains security measures for workload execution. The paper details the protocol's design, key components, operational workflows, and potential uses, contributing to the optimization of resource use across the computing continuum.**

***Index Terms*—Computing Continuum, Resource Management**

I. Introduction

In today's rapidly evolving technological landscape, the concept of the computing continuum has emerged as a paradigm that seamlessly integrates a wide spectrum of computing resources, spanning from far-edge devices to centralized cloud infrastructure. The consolidation of computational resources in the continuum opens up unprecedented potential for enhancing both the efficiency of various applications through dedicated allocation policies and the resiliency of the infrastructure upon failures thanks to the geo-distribution of compute resources.

Within the context of the European project FLUIDOS [1], we argued that the effective utilization of resources within a continuum is contingent upon their recognition and exposure as a unified pool. The dynamicity of the continuum can therefore be fostered by allowing devices to advertise/purchase (possibly) any type of resource, ranging from traditional computing resources (e.g., VMs, slices of Kubernetes clusters), to sensors, actuators, and volumes of data. As a result, devices can dynamically join the continuum, advertising the locally available resources and purchasing remote resources when needed, having access to the entire catalog of possibilities.

This paper proposes REAR (REsource Advertisement and Reservation), a novel protocol designed to address the challenges associated with resource advertisement and reservation in the computing continuum. REAR aims to provide a flexible and scalable framework that enables entities within the continuum to advertise their resources and facilitate the reservation of those assets by consumers or applications.

REAR addresses three key objectives. (*i*) *Standardization*, defining common interfaces and messages for resource advertisement and reservation to promote interoperability and compatibility across heterogeneous computing environments. (*ii*) *Efficiency*, to optimize resource allocation and utilization by allowing devices to enrich the resource description including internal energy metrics, latency considerations, and cost models. (*iii*) *Security and Trust*, incorporating mechanisms for authentication, authorization, and secure communication to ensure the integrity and confidentiality of resource transactions.

Throughout this paper, we will delve into the design principles, architectural components, and operational workflows of our proposed protocol. Additionally, we will discuss the potential applications and benefits of adopting this protocol in various domains, including edge computing, the Internet of Things (IoT), and cloud computing. By presenting this protocol, we aim to contribute to the ongoing efforts aimed at realizing the full potential of the computing continuum by enabling efficient resource management and utilization.

The paper is structured as follows: Sec. II outlines the motivations for a flexible resource advertisement protocol. Sec. III reviews current protocols. Sec. IV introduces REAR. Sec., V and VI detail REAR's data models and workflows. Sec. VII evaluates REAR, and Sec. VIII concludes.

II. Motivations

Resource discovery plays a pivotal role in distributed continuum infrastructures to prevent underutilized resources and inefficient allocation policies. In the following, we outline the optimizations enabled by the continuum's resource aggregation and highlight the requirements for REAR.

1) Support for intent-based allocation: Intelligent, data-driven applications can leverage the continuum to find the

[1] https://www.fluidos.eu/

optimal placement and satisfy user-specified service level requirements, often referred to as *intents*. Intent-driven orchestration is becoming a popular approach in several scenarios beyond workload management [1], for instance, to express user-level requirements for network configurations [2] or to model network security requirements [3]. Nevertheless, the integration within real-world scenarios is even more challenging due to the complexity of modern system architectures, whether programmable network switches or Kubernetes (K8S).

Such enhanced capabilities require approaches to gather information not only on existing (local) resources but also on those offered by other participants in the continuum. This enables the optimal utilization of available resources accounting for requirements beyond traditional computing (i.e., CPU, Memory) such as economics, latency, and security/compliance for the deployment and the lifecycle management of applications. The REAR protocol extends the boundaries of the local compute resources to implement intent-based allocation policies. Moreover, such dynamicity allows users to define the desired application architecture using intents, delegating to the infrastructure the task of retrieving and connecting services to the corresponding data sources.

2) Support for carbon-aware allocation: Increasing energy demand propelled by recent technological trends including the cloud computing [4], [5] underscore a critical dilemma. While the societal and environmental benefits of computing make its expansion both inevitable and desirable, the sustainability of this growth is questionable due to diminishing returns on hardware efficiency gains [6]. This scenario necessitates the exploration of new paradigms for carbon-aware computing.

A promising decarbonization strategy for a computing continuum advocates for workload scheduling that intelligently shifts computational jobs in space and time to capitalize on locations and periods of lower-carbon electricity availability [7]–[9]. This location-based approach ensures a genuine reduction in the carbon footprint of a computing continuum, bypassing the pitfalls of market-based strategies by focusing on the true carbon intensity of the grid mix at the time and place of usage. In addition, hardware-embodied emissions can be included in the picture as a way to provide a much more sustainable computing continuum [10], [11]. These are the emissions from the hardware manufacturing phase and they complement the operational emissions described above to provide a more holistic overview of the carbon footprint. Recent advancements in this field [12] demonstrate that it is both possible and essential to include embodied emissions in carbon-aware optimization efforts.

In light of these considerations, the motivation for integrating carbon-aware allocation mechanisms within REAR becomes even more compelling. By embedding such strategies into the fabric of the computing continuum, we not only aim to enhance operational efficiency and collaboration across heterogeneous computing resources but also to pioneer a sustainable approach in such a fluidified cloud-edge continuum.

3) Support for security features: The REAR protocol is seen as a fundamental component of the computing continuum, facilitating the dynamic (dis)aggregation of assets (resources, data, or capabilities) across various domains. This renders the traditional notion of a static security perimeter obsolete, adhering to the zero-trust approach, where neither asset, provider, or consumer is inherently trusted. Rather, continuous verification, authentication, authorization, and monitoring of all assets is required [13]. In this context, security is not only an integral component of REAR but is also communicated through it.

In particular, our vision entails mutual authentication and authorization for each party involved in the advertisement and reservation process, enabling authorized access to the offered assets. Given the dynamic nature of such a multi-party scenario, recent advancements in the field suggest a shift towards Decentralised Identifiers [14] and Verifiable Credentials [15] as a lightweight alternative to traditional centralized authentication. Some research endeavors also explore integrating this concept with DLT technology, which can serve as a registration authority and authorization scheme for distributed services [16].

At the same time, the REAR protocol assumes a crucial role in conveying the security attributes delivered by the advertised assets. Let us consider a scenario where a cluster offers a pool of computing resources for others to expand. In this instance, the advertised pool might provide dynamic network policy control, restricting communication solely to services within the pool and refining the network perimeter in real time. Additionally, the advertised features may encompass hardware security capabilities, such as hardware-backed Trusted Execution Environments (TEEs) on compute nodes, to achieve a higher degree of workload confidentiality and integrity (which is particularly relevant for sensitive workloads), and to give tenants the ability to attest the environment (or enclaves) where their workloads run. Furthermore, assets could be equipped with proactive and reactive protection services, such as threat detection and mitigation systems or cyber deception tools, thus providing insights into attackers' tactics and techniques.

4) Use cases: The computing continuum is a viable solution in geographically distributed sensors connected to local edge computing devices, and especially relevant for beyond 5G scenarios. Specifically, the continuum allows applications to be executed near the data sources to reduce latency, enhance security through on-device data storage, and guarantee functionality even during temporary network isolation. However, edge devices lack the robust computational redundancy and resiliency typically offered by a cloud environment. Focusing on the resiliency of the infrastructure, REAR allows to dynamically acquire the necessary resources from the continuum. As a result, a failure of an edge device could be tackled by acquiring other resources and seamlessly moving the workload in the continuum. In addition, security features can be advertised allowing for the exploitation of resources owned by (possibly) different administrative domains, or either defining security and privacy constraints, enormously extending the possible allocation strategies.

The computing continuum has proven to be a viable solution also in the case of autonomous driving systems for vehicles,

fleets of drones, robots, or, in general, any scenario that requires coordination between moving units. In this context, the objectives for the task allocation can be extremely different, such as reducing the communication latency, reducing the power consumption by moving the computation from devices "in operation" toward the devices that are charging, or moving the computation closer to the data sources. Still, any optimization requires the resource negotiation protocol to provide support for intent-based allocation policies, and, as such, the protocol must be also flexible enough to represent resources in the continuum so that it can be customized for the specific use cases.

III. Related Work

Reservation protocols play a key role in multi-user applications, networking, and distributed systems, managing access to resources and ensuring efficient and fair resource allocation. The Resource reSerVation Protocol (RSVP) represents a foundational approach in computer networks, designed to manage resource reservations in Quality of Service (QoS) enabled networks for efficient data traffic delivery [17]. MRSVP [18] has extended the functionality to the context of which mobile devices perform reservations based on their current and future locations and RNAP [19], which integrates economic factors into the reservation process. Additionally, RSVP-TE (Resource Reservation Protocol-Traffic Engineering) introduces traffic engineering capabilities, enabling explicit path establishment for data traffic to optimize network utilization [20]. In our perspective, these protocols primarily address network-centric parameters, overlooking the nuanced multi-dimensionality of computing resources (e.g., CPU, RAM, etc.) and the heterogeneity inherent in modern computing platforms (aka the As-A-Service model).

In the realm of distributed systems, the Service Negotiation and Acquisition Protocol (SNAP) [21] and subsequent SLA negotiation mechanisms [22] present methodologies for establishing QoS agreements, emphasizing the importance of flexible, bilateral negotiation frameworks. Such advancements illustrate the effort to refine SLA negotiation, catering to the dynamic needs of clients and servers within distributed architectures. Furthermore, authors in [23] also describe a brokering architecture that can make advance resource reservations for SLAs, and also the need to consider security and privacy for managing the distributed services [16].

The advent of 5G technology introduces new dimensions to this landscape, offering telecommunication operators unprecedented opportunities to leverage their network and computing infrastructures [24], [25]. With 5G, the requirements for Edge infrastructure intensify [26], necessitating protocols that support seamless application deployment across diverse Telco providers and facilitate the federation of Operator Platforms [27]. Despite the potential, current proposals face limitations, including (*i*) a lack of resource price discovery mechanisms, (*ii*) limited support for dynamic environments, and (*iii*) a focus on containerized applications that may not fully capture the generality of offered resources/services.

This evolving landscape underscores the imperative for innovative reservation protocols that can address the multi-faceted challenges of modern computing environments. Such protocols must not only accommodate the complex interplay of network and computing resources but also adapt to the rapidly changing dynamics introduced by advancements like 5G, thereby ensuring efficient, fair, and economically viable resource allocation in distributed systems.

IV. Architecture

REAR has been designed around the concept of *Node*, i.e., a unique computing environment, under the control of a single administrative entity. The node can be composed of one or more machines and modeled with a common, extensible set of primitives that hide the underlying details while maintaining the possibility to export the most significant distinctive features (e.g., the availability of specific services, peculiar HW capabilities). In addition, nodes belonging to the same administrative entities can be logically grouped in *Domains*, allowing for enhanced aggregation policies when advertising available resources (e.g., a given resource can be advertised only within the same domain).

Such a hierarchical infrastructure allows for two different interactions among nodes involved, generically referred to as *consumers* and *providers*, depending on the respective role: (*i*) a *Horizontal* interaction enables the creation of a resource exchange process among peers, which can share their resources and services, or part of them, based upon a set of policies. Horizontal interactions are carried out according to a peer-to-peer paradigm, hence without the need for any centralized entity that controls and supervises the entire process. (*ii*) a *Vertical* interaction introduces new concepts such as aggregation and hierarchical scaling into the picture. Third-party brokering entities can provide endpoints that customers and providers can browse and query to obtain aggregated views of the resources available in one (or more) domain. It is worth mentioning that such interaction can be recursively implemented to replicate multi-level hierarchical aggregations.

In the REAR protocol, each node has two different components: a *resource importer* and a *resource exporter*. The former is responsible for the discovery of available resources. Since the number (and type) of resources in the continuum infrastructure can be potentially huge, the component is also in charge of filtering the available options based on the data models presented in Sec. V. The latter advertises the node's available resources to the other members of the continuum. This approach is still compliant with the hierarchical model, as the resource exporter can perform the advertisement of the resources of the nodes for which it acts as a broker. In addition, a dedicated *contract manager* component keeps track of the purchased resources in the form of a contract and exposes an endpoint that enables further logic to be implemented to verify the Service Level Agreement (SLA) mentioned in the contract to ensure that it is, in fact, being upheld.

V. Data Model

The REAR data model outlines the various types of resources advertised in the continuum. This model includes the definition of two terms: *Flavor* and *FlavorType*. The *Flavor* encompasses the set of information shared among all possible resources, while the *FlavorType* is a pointer to another dedicated structure that specifies the unique characteristics of each resource. [2] The complexity of the data model requires a formal, semantic definition of the various components and their relationships. To enable such reasoning we created two ontologies[3] according to the standard defined by the W3C consortium, characterizing the relationships between K8S entities, the REAR data model, its relationship with K8S, and the various properties of the *FlavorTypes* described later.

A. Flavor

The *Flavor* data model provides a structured way to represent and manage different kinds of computing resources and it includes all the information that is independent from the chosen *FlavorType*, hence facilitating interoperability and standardization across various systems and platforms, enabling efficient resource advertisement and reservation within the computing continuum. The *Flavor* data model includes the following information:

- *FlavorID*: A unique identifier for the flavor.
- *ProviderID*: A unique identifier for the provider of the flavor, which can be different from the owner in case this flavor is being advertised by a broker.
- *Location*: Information about the location of the flavor described using the triplet <latitude, longitude, altitude>.
- *NetworkPropertyType*: The type of network property ensured by the provider (e.g., 5G, WiFi, Ethernet).
- *Price*: Information about the price of the flavor, including amount, currency, and billing time (e.g., daily, monthly).
- *Owner*: Information about the owner of the flavor, including domain, node ID, IP address.
- *FlavorType*: A reference to a specific FlavorType schema, allowing for defining details specific to each flavor type.

B. FlavorType

Currently, five *FlavorTypes* are defined in REAR, which provides flavor-specific data models that describe the computational resources to be purchased. New *FlavorTypes* data models can be added to support more use cases.

1) K8Slice: It identifies a Kubernetes cluster capturing both its hardware characteristics and various policies related to its deployment and usage within Kubernetes environments. When a K8Slice flavor is purchased, the remote resources can be seen as a logical extension of the local resources, to deploy general-purpose workloads (with Liqo.io). A K8Slice includes the following main information:

- *Characteristics*: Defines the hardware characteristics of the Kubernetes flavor, including CPU, GPU, Memory, Storage, and the number of pods that can be deployed.
- *Properties*: Specifies additional properties of the Kubernetes flavor, including the expected inter-cluster latency, the list of security standards supported (e.g., GDPR, ISO/IEC 27002), and the carbon footprint expressed in terms of embodied and operational emissions.
- *Policy*: Defines policies related to the Kubernetes flavor, including the fact that multiple instances of the same FlavorType can be aggregated into a single entity, or partitioned to obtain only a subset of the resources.

2) VM: The VM FlavorType provides an option to acquire computing resources in the form of Virtual Machines (VMs). The VM FlavorType shares most of the characteristics of the K8Slice previously described, including the possibility to describe the architecture of the node in which the VM is running as well as information on the operating system on the VM. Despite being similar, it is important to model both FlavorTypes separately as the usage of resources differs. For instance, a K8Slice needs to communicate with the K8S API server, while a VM requires an SSH connection.

3) Service: The *Service* FlavorType enables providers to advertise services, following the Software-as-a-Service model (SaaS). Since it is almost impossible to provide a generic description suitable for all possible services, the *Service* FlavorType data model follows the same pattern used for the Flavor and FlavorType data model: the *Service* FlavorType provides a high-level description of the Service, including all the specifications that are independent of the actual service, whereas the *ServiceType* details the service-specific characteristics. Specifically, the *Service* FlavorType describes:

- *Name* and *Description*: Respectively the name and the human-readable description of the service.
- *Tags*: A keyword list that summarizes a service's properties.
- *Plan*: The plan for the service (e.g., Enterprise, Trial).
- *Latency*: The expected latency with the consumer.
- *ServiceType*: The reference to the characteristics of the specific ServiceType. For instance, a *PostgreSQL* ServiceType can include the number of transactions per second or the number of tables that the user can create.

4) Sensor: The Sensor FlavorType allows for Sensors to be advertised and (possibly) shared among multiple consumers in the continuum. The Sensor FlavorType is characterised by:

- *SensorType*: The type of sensor (e.g., light, humidity).
- *SensorModel* and *SensorManifacturer*: Additional information on the sensor.
- *SamplingRate*: The frequency of the measurements.
- *Accuracy*: The expected accuracy of the measurements.
- *MeasurementUnit* and *SamplingRateUnit*: The unit of measure for both the measurements and the sampling rate.
- *AccessType*: How the measurements can be accessed from the consumer (e.g., HTTP, MQTT).

5) Data: The *Data* FlavorType allows datasets to be shared and advertised between participants of the continuum. This means that REAR enables a form of data sharing as recommended by various EU-funded initiatives such as GAIA-X

[2]Full specification: https://github.com/fluidos-project/REAR-data-models
[3]https://github.com/fluidos-project/fluidos-ontology

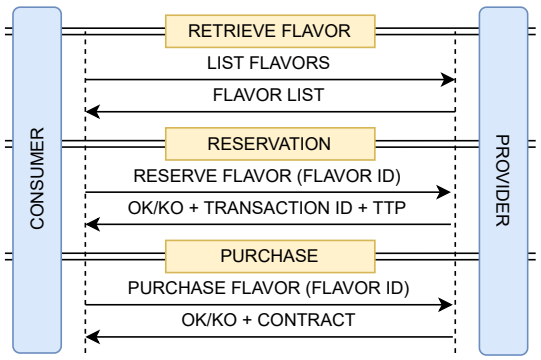

Fig. 1. REAR workflow overview, implementable via REST API calls.

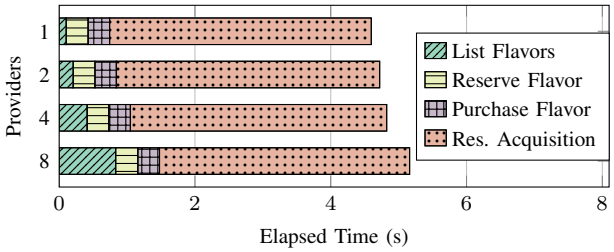

Fig. 2. REAR stages' timing vs. time to ready purchased resources.

Data Spaces[4] and IDSA Reference architecture [28]. The Data FlavorType is characterized by:

- *Name* and *Description*: Respectively the name and the human-readable description of the dataset.
- *Tags*: keyword list summarizing the dataset properties.
- *License*: The license(s) regulating the utilisation of data.
- *Plan*: Dataset usage plan (e.g., Free, Enterprise, Trial).
- *Format*: The format of the dataset (e.g., CSV, TSV, JSON).

VI. REAR Workflow

REAR defines several messages, which can be classified as either *required* or *optional* (an example is depicted in Figure 1).

A. List Flavor (required)

This message is sent by the consumer to probe the available flavors offered by a given provider. Since different FlavorTypes can be offered by a single provider, the consumer can filter out possible resources by specifying the desired characteristics in the *List Flavor* message, using the FlavorType data model described earlier. For example, if the consumer wants to purchase VMs, it can retrieve the list of possible VMs offered by a provider by specifying characteristics of the VM FlavorType, such as 2 CPUs and 4GB of RAM. The provider will then reply with a list of VMs that match the requirements, if any.

The *List Flavor* message thus contains the requested FlavorType with the desired characteristics and some form of identification for the consumer with the tuple $<$*ConsumerID*, *Region*$>$ to allow the provider to generate a (possibly) customized offer for the consumer.

B. Reserve Flavor (required)

Once the consumer knows the Flavors and their IDs, the *Flavor* reservation process is performed through the *Reserve Flavor* message, sent by the consumer to inform the provider about its willingness to reserve a specific flavor. Specifically, the consumer/provider interaction can be summarized as follows. After the client has collected the list of available *Flavors* offered by the provider, it notifies the intention of reserving a specific flavor by sending the *Reserve Flavor* message, specifying the ID of the *Flavor* to be reserved. To verify the consumer identity, the *Reserve Flavor* message must also include an authentication token that will be then validated by a Trusted third-party authentication and authorization service. Once received, the provider checks if the flavor is still available and if so it replies with a summary of the reservation process including the *TransactionID* and the *Time To Purchase* (TTP), i.e., the time by which the Flavor must be purchased. This allows reserved Flavors to be released in case either the consumer becomes unreachable, or the subsequent purchase process exceeds a predefined threshold. If the *Flavor* is not available a 404 error message is sent to the consumer.

C. Purchase Flavor (required)

The *Purchase Flavor* message is sent by the consumer upon receipt of the provider's response during the reservation phase to complete the purchase of an offered flavor. To do so, the consumer sends the *Purchase Flavor* message including the *TransactionID* (obtained with the *Reserve Flavor*) and the identification token to the provider. If authorized, the consumer will then be prompted to a payment service (either external or managed by the provider) and, if successful, a copy of the *Contract* is returned to the consumer, detailing the purchase and the information required to access the purchased resource (e.g., IP address, API endpoint).

D. Subscribe / Refresh / Withdraw Flavor (optional)

Given that each offered flavor may not be always available, the consumer can notify the intention to receive continuous updates on a specific set of *Flavors* using the *Subscribe Flavor* message. This internally triggers the creation of a stateful channel between the consumer and the provider, which asynchronously sends back updates for any change in the specified *Flavor* using the *Refresh Flavor* message.

If the *Flavor* is no longer available, the provider can tear down the communication channel related to the specific *Flavor* and notify the consumer that the *Flavor* can no longer be purchased using the *Withdraw Flavor* message.

VII. Experimental Evaluation

We here report the empirical evaluation of REAR by means of the testbed created for FLUIDOS[5]. In the testbed, we collect metrics from the three different stages of the REAR

[4]https://gaia-x.eu/what-is-gaia-x/deliverables/

[5]https://github.com/fluidos-project/node

protocol, namely, *List Flavors*, *Reserve Flavor*, and *Purchase Flavor* to provide a breakdown of the overhead, varying the number of providers offering flavors of type K8Slice. The *Resource Acquisition* phase represents instead the reference time required to extend the pool of locally available resources in Kubernetes using the Liqo framework.

Figure 2 details the elapsed time for a generic consumer before the purchased resources are available and ready to use. As we can see, the *List Flavor* message is influenced by the number of available providers, leading to higher latencies when the consumer has an extensive list of available providers. In the worst-case scenario, the consumer must iterate the entire providers' list before identifying a suitable resource, resulting in a linear time complexity $\theta(n)$ w.r.t. the provider list size n. However, the *Reserve Flavor* and *Purchase Flavor* messages maintain stability through the different scenarios. It is worth noticing that despite any fluctuations, the REAR protocol's overhead remains minimal (approximately 30% of the total time in the worst case), and can be further reduced by aggregating resources through third-party brokering and aggregation services.

VIII. Conclusions

This study presents REAR, a novel protocol designed to address resource advertisement and reservation in the computing continuum. REAR provides a standardized interface for messages and data models to promote interoperability and compatibility across heterogeneous computing environments, fosters enhanced allocation policies allowing devices to enrich the resource description and provides security support to ensure the integrity and confidentiality of resource transactions.

Acknowledgment

This work was supported by European Union's Horizon Europe research and innovation programme under grant agreement No 101070473, project FLUIDOS (Flexible, scaLable, secUre, and decentralIseD Operating System).

References

[1] A. Leivadeas and M. Falkner, "A survey on intent-based networking," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, 2022.

[2] B. Martini, M. Gharbaoui, and P. Castoldi, "Intent-based network slicing for sdn vertical services with assurance: Context, design and preliminary experiments," *Future Generation Computer Systems*, vol. 142, 2023.

[3] A. Chowdhary, A. Sabur, N. Vadnere, and D. Huang, "Intent-driven security policy management for software-defined systems," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, 2022.

[4] T. Kaur and I. Chana, "Energy Efficiency Techniques in Cloud Computing: A Survey and Taxonomy," *ACM Computing Surveys*, vol. 48, no. 2, Oct. 2015.

[5] M. Sedlacko, A. Martinuzzi, and K. Dobernig, "A Systems Thinking View on Cloud Computing and Energy Consumption." Atlantis Press, Aug. 2014, iSSN: 2352-538X.

[6] J. Koomey, S. Berard, M. Sanchez, and H. Wong, "Implications of Historical Trends in the Electrical Efficiency of Computing," *IEEE Annals of the History of Computing*, vol. 33, no. 3, Mar. 2011.

[7] A. Radovanovic, R. Koningstein, I. Schneider, B. Chen, A. Duarte, B. Roy, D. Xiao, M. Haridasan, P. Hung, N. Care, S. Talukdar, E. Mullen, K. Smith, M. Cottman, and W. Cirne, "Carbon-Aware Computing for Datacenters," Jun. 2021, arXiv:2106.11750 [cs, eess].

[8] Y. G. Kim, U. Gupta, A. McCrabb, Y. Son, V. Bertacco, D. Brooks, and C.-J. Wu, "GreenScale: Carbon-Aware Systems for Edge Computing," Apr. 2023, arXiv:2304.00404 [cs].

[9] T. Sukprasert, A. Souza, N. Bashir, D. Irwin, and P. Shenoy, "Quantifying the Benefits of Carbon-Aware Temporal and Spatial Workload Shifting in the Cloud," Jun. 2023, arXiv:2306.06502 [cs, eess].

[10] N. Bashir, D. Irwin, and P. Shenoy, "On the Promise and Pitfalls of Optimizing Embodied Carbon," in *Proceedings of the 2nd Workshop on Sustainable Computer Systems*, ser. HotCarbon '23. New York, NY, USA: ACM, Aug. 2023.

[11] J. Wang, U. Gupta, and A. Sriraman, "Peeling Back the Carbon Curtain: Carbon Optimization Challenges in Cloud Computing," in *Proceedings of the 2nd Workshop on Sustainable Computer Systems*, ser. HotCarbon '23. New York, NY, USA: ACM, Aug. 2023.

[12] U. Gupta, M. Elgamal, G. Hills, G.-Y. Wei, H.-H. S. Lee, D. Brooks, and C.-J. Wu, "ACT: designing sustainable computer systems with an architectural carbon modeling tool," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, Jun. 2022.

[13] C. Buck, C. Olenberger, A. Schweizer, F. Völter, and T. Eymann, "Never trust, always verify: a multivocal literature review on current knowledge and research gaps of zero-trust," *Computers & Security*, vol. 110, 2021.

[14] World Wide Web Consortium (W3C), "Decentralized Identifiers (DIDs) v1.0: Core architecture, data model, and representations," 2022, accessed on April, 2024.

[15] W3C, "Verifiable Credentials Data Model v1.1," 2022, accessed on April, 2024. [Online]. Available: https://www.w3.org/TR/vc-data-model/

[16] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, "A survey on essential components of a self-sovereign identity," *Computer Science Review*, vol. 30, 2018.

[17] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "Rsvp: A new resource reservation protocol," *IEEE network*, vol. 7, no. 5, 1993.

[18] A. K. Talukdar, B. Badrinath, and A. Acharya, "Mrsvp: A resource reservation protocol for an integrated services network with mobile hosts," *Wireless Networks*, vol. 7, 2001.

[19] X. Wang and H. Schulzrinne, "Rnap: A resource negotiation and pricing protocol," *Transit*, vol. 6, no. B7, 1999.

[20] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "Rfc3209: Rsvp-te: Extensions to rsvp for lsp tunnels," USA, 2001.

[21] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, and S. Tuecke, "Snap: A protocol for negotiating service level agreements and coordinating resource management in distributed systems," in *Job Scheduling Strategies for Parallel Processing: 8th International Workshop, JSSPP 2002 Edinburgh, Scotland, UK, July 24, 2002*. Springer, 2002.

[22] S. Venugopal, X. Chu, and R. Buyya, "A negotiation mechanism for advance resource reservations using the alternate offers protocol," in *2008 16th Interntional Workshop on Quality of Service*. IEEE, 2008.

[23] E. Elmroth and J. Tordsson, "A grid resource broker supporting advance reservations and benchmark-based resource selection," in *International Workshop on Applied Parallel Computing*. Springer, 2004.

[24] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[25] R. Dangi, P. Lalwani, G. Choudhary, I. You, and G. Pau, "Study and investigation on 5g technology: A systematic review," *Sensors*, vol. 22, no. 1, p. 26, 2021.

[26] GSMA, "Gsma operator platform telco edge requirements 2022," 2022, accessed on March, 2024. [Online]. Available: https://www.gsma.com/futurenetworks/resources/gsma-operator-platform-telco-edge-requirements-2022/

[27] GSMA, "Gsma operator platform group – east-westbound interface apis," 2023, accessed on March, 2024. [Online]. Available: https://www.gsma.com/futurenetworks/resources/east-westbound-interface-apis/

[28] I. Otto, S. Steinbuß, A. Teuscher, I. Lohmann, A. Auer, S. Bader, H. Bastiaansen, H. Bauer, I. Birnstil, and M. Böhmer, "International data spaces association—reference architecture model—version 3.0," 2019.