Adaptive and interoperable federated data spaces: an implementation experience

Nikolaos Papadakis, Niemat Khoder, Daphne Tuncer, Kostas Magoutis, Georgios Bouloukakis

Adaptive and Interoperable Federated Data Spaces: An Implementation Experience

Nikolaos Papadakis*, Niemat Khoder*, Daphne Tuncer†, Kostas Magoutis‡§, Georgios Bouloukakis*
{nikolaos.papadakis, niemat.khoder}@telecom-sudparis.eu, daphne.tuncer@enpc.fr,magoutis@ics.forth.gr,
georgios.bouloukakis@telecom-sudparis.eu
*Télécom SudParis, Institut Polytechnique de Paris, France
†Ecole nationale des ponts et chaussees, Institut Polytechnique de Paris, France
‡University of Crete, Greece
§Institute of Computer Science (ICS), Foundation for Research and Technology - Hellas (FORTH), Greece

***Abstract*—As modern smart cities increasingly rely on federated IoT-enabled data spaces to support their operations, the dynamic and heterogeneous nature of these environments introduces significant challenges. Variations in data models, protocols, and policies hinder seamless federation and collaboration. To address these challenges, we present the prototype implementation of CCDUIT, a modular, scalable, and adaptive software overlay architecture for cross-federation collaboration. This architecture dynamically adapts to evolving data exchange requirements, including protocol variations, data model discrepancies, and policy constraints, ensuring continuity and efficiency in data discovery and exchange. The prototype uses context-aware mechanisms to autonomously adjust operations, dynamically reconfigure collaborations, enforce data exchange policies, and optimize data-sharing pathways. We validate the prototype using real-world data spaces operating to address barriers to climate change mitigation and energy transition. Experimental results highlight the ability of the architecture to adapt for diverse federation-specific needs with minimal operational overhead and respond to dynamic policy changes. We release our prototype as open source, enabling the community to explore, extend, and adapt it to their own needs.**

***Index Terms*—Smart Cities, IoT, Data Federation, Software Overlay, Prototype Implementation, Data Interoperability**

I. Introduction

As modern digital ecosystems become increasingly interconnected, federated data spaces [1] [2] have emerged as a key enabler for cross-domain collaboration and data sharing. These federations enable entities to share data while maintaining independence, addressing challenges in urban planning, healthcare, and sustainability. However, while federated systems within individual domains have demonstrated significant utility, enabling seamless collaboration across heterogeneous federations remains a challenge.

Existing solutions within federations typically rely on homogenizing technologies, such as semantic gateways or data converters [3]–[5], which help bridge internal heterogeneity. However, these approaches falter when applied at the inter-federation level, where each data space operates under unique technological (e.g., RESTful APIs vs. MQTT protocols), semantic (e.g., NGSI-LD vs. Brick data models), and policy paradigms (e.g., GDPR-compliant data handling). Furthermore, the static nature of most current solutions fails to account for the dynamic and evolving requirements of federated environments, such as changing data-sharing policies or fluctuating collaboration demands.

In our previous research we introduced a novel software overlay architecture designed to address these inter-federation challenges by enabling scalable, policy-compliant, and technology-agnostic collaboration across federations, the CCDUIT architecture [6]. This architecture lays the groundwork for what can be described as a "federation of federations," where individual federations can dynamically interact without sacrificing their autonomy or operational sovereignty. Although conceptually promising, the design's feasibility and effectiveness remained unvalidated in real-world contexts.

This paper focuses on bridging that gap by presenting a prototype implementation of this architecture, bringing the architecture to life. To validate the prototype, we apply it to a real-world use case involving federations that collaborate on climate change mitigation and energy transition initiatives, based on the Energy4Climate Interdisciplinary Center (E4C) [7]. This scenario, with its complex and evolving collaboration requirements, serves as an ideal testbed to demonstrate the capabilities of the system.

The primary contributions of our work include:

1) A comprehensive realization of the CCDUIT software overlay architecture.
2) Demonstration of the practical applicability of the prototype in a real-world scenario.
3) A detailed experimental analysis to evaluate the prototype's functionality, operational overhead and responsiveness to policy changes
4) The release of the prototype as an open source artifact [8], accompanied by detailed documentation to enable reproduction, extension, and further exploration by the research community.

The remainder of this paper is structured as follows: In §II we discuss our motivation along with the real-world challenges we address. The details of the prototype implementation, including the technologies used and the rationale behind them, are presented in §III. In §IV, we describe the experimental evaluation setup and provide a detailed analysis of the results, focusing on both performance and policy adaptation capabil-

ities. Related work is explored in §V, comparing existing solutions and highlighting the unique advantages of the selected approach. Finally, §VI concludes the paper with key findings and outlines potential directions for future research.

II. Motivation

In the face of increasing energy demands and climate change, achieving energy efficiency has become a priority for research institutions and industries alike [7]. Modern building networks with advanced energy management systems offer opportunities to address these challenges. However, even cutting-edge buildings often operate in isolation, and their systems are unable to collaborate effectively across a shared ecosystem.

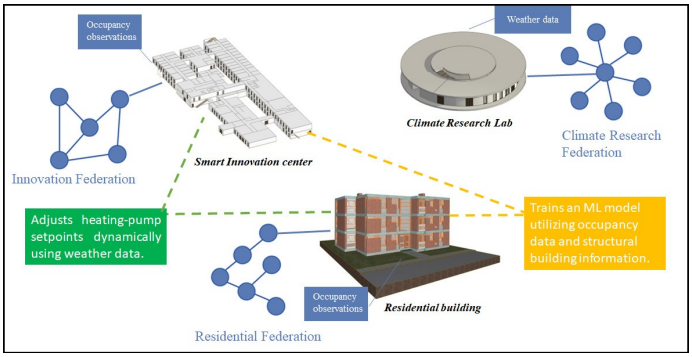

Fig. 1. Illustration of the Three Building Testbed for Collaborative Energy Management Solutions, Representing Distinct Federations

As a testbed for exploring collaborative energy management solutions, we use a network of three distinct buildings (Fig. 1): A smart innovation center, a residential facility, and a climate research laboratory, each embodying advanced technologies designed to promote sustainability and belonging to a different operational federation. The innovation center features rooftop solar panels, battery storage, and a Building Management System (BMS) that optimizes its HVAC (Heating, Ventilation, and Air Conditioning) operations. The residential facility incorporates electric vehicle chargers, smart thermostats, and solar energy systems. The climate lab provides meteorological data through a series of weather monitoring instruments.

Despite their individual capabilities, these buildings currently operate in silos. The innovation center and residential facility control their HVAC systems based on predefined schedules, without factoring in real-time occupancy patterns or localized weather conditions. This results in energy waste, either by heating and cooling unoccupied spaces or responding inefficiently to environmental changes. The hyperlocal weather data from the climate laboratory, which could significantly improve the energy management of other buildings, remains underutilized. Moreover, technical disparities between buildings create additional barriers to collaboration: while the innovation center and the climate lab use NGSI-LD [9], [10] data models accessed via RESTful APIs, the residential facility relies on the BRICK [11] schema and MQTT for communication.

The need for a unifying framework to enable seamless collaboration between such diverse systems is evident. Effective collaboration would allow these buildings to dynamically share and integrate data, optimizing HVAC operations in response to both real-time conditions and contextual information. For example, the climate lab could provide localized real-time weather updates to inform HVAC adjustments in the innovation center and residential facility, ensuring comfort while minimizing energy use. Similarly, Digital Twins occupancy and building data for residential and innovation buildings could guide the development of machine learning models for heating and cooling strategies, reducing unnecessary energy expenditure in unoccupied spaces.

However, achieving this vision requires overcoming significant technical and operational challenges. Interoperability between systems with differing data models and communication protocols must be addressed, all while maintaining data sovereignty (that is, the control over who can access, modify, and forward data). This is where the CCDUIT software overlay architecture plays a critical role. Designed to bridge heterogeneous systems, it provides a modular and scalable framework that enables seamless data exchange, policy compliance, and adaptive decision making across diverse environments.

III. Prototype Implementation

A. Overview

The core purpose of the CCDUIT [6] software overlay architecture is to enable seamless collaboration between different federations, each with its own data models, communication protocols, and operational constraints. By serving as an adaptable bridge, the system allows these federations to exchange data, share contextual information, and collaborate effectively while preserving their individual autonomy. As this work focuses on the prototype, the detailed rationale and design of the architectural mechanisms are only briefly outlined here. Readers are encouraged to consult the full paper for an in-depth explanation of the underlying architecture [6]. At a high level, Fig. 2, the system works by creating a dynamic network where federations act as nodes connected by **two types of interactions**: one for exchanging **actual data** and the other for sharing **contextual information**, such as policies and metadata. These interactions are continuously monitored and adjusted on the basis of the current needs and conditions of each federation. For example, if two federations use different data formats or protocols, the system adapts their communication in real time by transforming the data or aligning the protocols as needed. This ensures smooth data flow despite system differences.

A critical aspect of this adaptability is the use of **policies** (see listing 1), which define rules for how and when data can be shared. They encapsulate constraints related to privacy, security, data sovereignty, and operational guidelines. For instance, a policy might specify that certain environmental data can only be shared with federations that adhere to specific sustainability standards, or that health-related data must comply with strict privacy regulations, or as in our

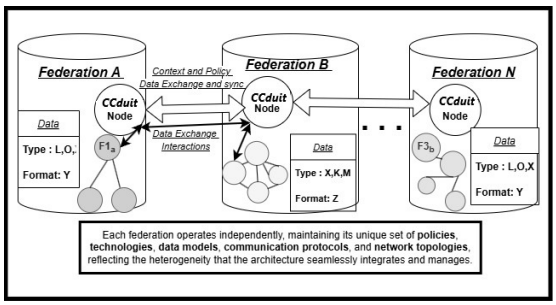

Fig. 2. Federations with distinct policies, technologies, and topologies, unified through the CCDUIT architecture.

experiments whether to allow specific federations by their id and not just their characteristics.

```
{ "id": "urn:ngsi-ld:ContextPolicy:Policy1",
  "type": "ContextPolicy",
  "name": "policy1",
  "description": "Simplified Example Policy",
  "providerFederation": "urn:ngsi-ld:Federation:Federation1",
  "permittedContextTypes": ["community", "federation", "policy", "function"],
  "ContextBrokerURL": "http://localhost:1051/ngsi-ld/v1/entities",
  "sharingRules": [
    {"urn:ngsi-ld:Federation:Federation2": {"canReceive": true, "canForward":
          false}},
    {"public": {"canReceive": false, "canForward": false}} ],
  "modificationPolicy": {
    "lastModified": "2024-12-05T14:05:09Z",
    "modifiedBy": "Admin_24"},
  "Geographic_Restrictions": null}
```

Listing 1. Example of an instance of a policy context entity in NGSI-LD key-values format. This Policy allows Federation 2 to receive information but to not share/forward information regarding Federation1

The system dynamically enforces these policies by continuously synchronizing them across federations and applying them to data exchanges in real time. When a federation requests data or initiates collaboration, the system checks the relevant policies, determines what can and cannot be shared, and adapts the interactions accordingly. In short, the system continuously monitors policy changes and automatically updates itself to reflect these modifications, ensuring that all exchanges remain fully compliant with the updated policies.

The architecture was designed to realize these capabilities through a **fully modular design**, as illustrated in Fig. 3 (A). Central to its operation is the **Interaction Engine**, a subsystem that facilitates data interoperability between federations. Supporting the Interaction Engine is the **Contextual Knowledge Base (KB)**, a graph-structured repository for storing essential information about federations, such as their data models, policies, and available resources. To simplify and manage interactions, the architecture provides a **suite of APIs**. **The Interaction API** allows federations to initiate and control data exchanges, query context entities, and trigger collaborative operations dynamically. Meanwhile, the **Administration API** empowers system administrators to handle tasks like system configuration, policy management, and real-time monitoring. Together, these APIs form a crucial interface layer. The system also includes a robust **policy management mechanism** to ensure that all inter-federation exchanges adhere to predefined rules. Policies are encoded and shared via the Context Exchange Mechanism, guaranteeing compliance with federation-specific governance frameworks. Additionally, a Publish/Subscribe Schema organizes all node communication into topics representing federation-specific entities, data types, and policies. To ensure adaptability in dynamic environments, the architecture integrates **Dynamic Adaptation Mechanisms**. These mechanisms enable the system to autonomously reconfigure interactions in response to changes in data exchange requirements, evolving policies, or federation configurations.

In the following subsection, we detail how these theoretical components have been translated into a functional prototype.

B. Implementation Details

Implementing CCDUIT brings the conceptual architecture to life through a working prototype that embodies the principles of modularity, scalability, and interoperability. To achieve this, we carefully selected technologies that support these goals, Fig. 3 (B), ensuring that the system can dynamically adapt to the challenges inherent in federated collaborations. The prototype is primarily developed in Python due to its simplicity, and extensive ecosystem of libraries for web development, data processing, and communication. The codebase, including the prototype and API endpoints, is available on our GitHub page GitHub repository and permanently archived in our Zenodo repository Zenodo repository [8].

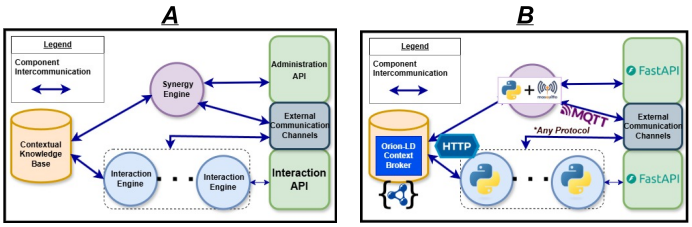

Fig. 3. (A) High level overview of architecture components of a CCDUIT node, (B) Prototype node implementation details.

At the core of context management is the Contextual Knowledge Base (KB), implemented using the Orion-LD context broker. Orion-LD[1] adheres to the NGSI-LD standard, representing context data in a structured JSON-LD format and uses MongoDB [12] as its back-end storage. This choice allows for a flexible graph-based approach to the rest of the context data. Orion-LD stores and serves queries to context data, but is not designed for fast real-time updates that would allow for dynamic synchronization between federations. To address this, we integrated Mosquitto [13] MQTT brokers for asynchronous communication. MQTT (Message Queuing Telemetry Transport) is a lightweight publish/subscribe protocol ideal for low-latency communication. When a federation updates a policy, the Policy Management Service stores it in Orion-LD and simultaneously publishes it to a Mosquitto topic. The Policy Monitoring Service listens to these topics, ensuring federations receive and apply policy changes in real time. This dual approach leverages the strengths of both technologies: Orion-LD provides a reliable and queryable context store, while Mosquitto ensures rapid dissemination of critical updates.

[1]https://github.com/FIWARE/context.Orion-LD

The prototype's functionality is exposed through a set of RESTful APIs built using FastAPI[2]. By keeping API endpoints lightweight and delegating business logic to dedicated service layers, we ensure the code remains maintainable and adaptable for future enhancements. The implemented interaction engine handles the complexities of data exchange between federations. When an exchange is initiated, the Interaction Engine retrieves the necessary conversion functions from the Knowledge Base (Orion-LD) to adapt data models and translate communication protocols. Our implementation currently supports the HTTP and MQTT protocols, providing a proof of concept for the system's adaptability. The modular design of the Interaction Engine means that additional protocols can be integrated with minimal effort in the future. Collaboration between federations follows a structured workflow. When a federation wants to collaborate, the service sends a request to the target federation. The target federation evaluates the request against its policies and responds accordingly. Once approved, the system automatically establishes context and policy synchronization channels, ensuring that data exchanges adhere to the rules of each federation.

Our implementation demonstrates the core functionalities of CCDUIT, but there is room for improvement. For example, fully automating the context exchange process and supporting additional communication protocols would further enhance flexibility. However, our prototype provides a solid foundation for validating the architecture in real-world scenarios.

IV. The CCDUIT prototype in practice

The experimental setup, Fig. 4 for evaluating our prototype, mirrors the real-world collaboration scenario described in the motivation section II. It involves three distinct buildings that represent different federations: a smart innovation center, a residential facility, and a climate research lab. Each building belongs to a separate federation (Federations 1,2 and 3), each employing unique data models and communication protocols. The data used in these experiments consist of synthetic occupancy observations and historical weather data from 2019, replayed in real-time to simulate realistic conditions.

Each federation is equipped with a CCDUIT node, implemented in Python, and deployed using Docker containers. All nodes run on the same machine, featuring a 12th Gen Intel Core i7-12700H processor (20 cores, 2.3 GHz base frequency) and 16 GB RAM. Context entities and policies are represented using the NGSI-LD format, and interactions between federations are facilitated through Swagger-defined RESTful APIs provided by the prototype.

To evaluate the prototype, we conducted two types of experiments: one measuring latency overhead for real-time data exchanges across different data models and protocols, and another assessing the system's responsiveness to policy updates, such as terminating interactions or propagating new policies across federations. The evaluation focuses on latency and responsiveness as critical factors for real-time cross-federation collaboration, with scalability already explored in prior work [6] and robustness left for future exploration, as the current goal is to validate the prototype's core adaptation mechanisms. All experiments are designed to be reproducible, with detailed instructions provided in the artifact repositories [8] to facilitate validation and further exploration.

[2]https://fastapi.tiangolo.com/

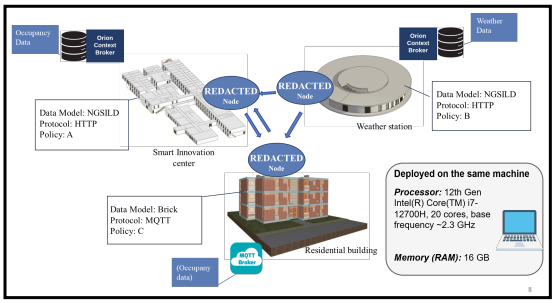

Fig. 4. Experimental Setup

A. CCDUIT Prototype Performance Evaluation

Evaluation of CCDUIT performance focuses on two key aspects of interaction latency: startup latency and ongoing interaction latency. We consider these two distinct phases because they capture different operational behaviors of the prototype. Startup latency, or cold start overhead, measures the delay from the moment a user initiates a data-exchange interaction to the reception of the first converted data at the destination endpoint. This phase includes the time required for the prototype to search for the relevant context in the KB, apply the necessary data transformations, and align communication protocols before initiating the interaction. Once the interaction is established, subsequent data exchanges occur at regular intervals. The latency for each of these exchanges, termed interaction latency overhead, measures the time from when the interaction handling service requests data until it arrives at the destination.

We evaluated latencies by testing scenarios with varying data models, protocols, and with and without using our prototype. In the "without" parts, buildings exchanged data directly using native protocols and APIs, such as an NGSI-LD HTTP endpoint communicating with another NGSI-LD HTTP endpoint or an MQTT broker connecting directly to another MQTT broker. In these cases, no dynamic adaptation or policy enforcement was applied.

First the cold start latency, representing the delay associated with initializing data exchanges, was evaluated across all scenarios. In each case, the cold start experiment was repeated 20 times to ensure consistency, and the results are presented in the corresponding Fig. 5. The 'cold start' latency for the non CCDUIT solutions, is due to the time that it takes to connect to each broker. As expected, scenarios that required complex transformations and protocol adaptations exhibited higher cold start latencies. However, the prototype's performance remained within reasonable limits. In addition to measuring cold start latency, we evaluated the ongoing interaction latency. The results of these experiments, presented in the accompanying Fig. 6,

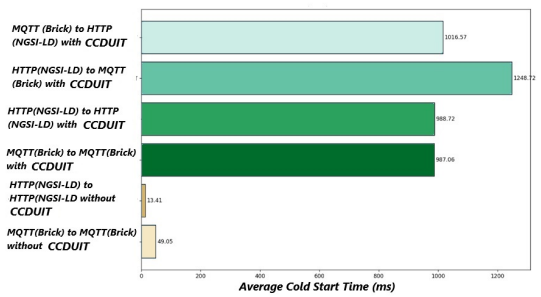

Fig. 5. Cold start latency measurements across different scenarios.

illustrate the latency incurred under different configurations and help us assess the prototype's efficiency and adaptability. Each scenario consisted of 100 data exchanges at an average interval of 0.8 seconds, repeated over five runs each. In the first

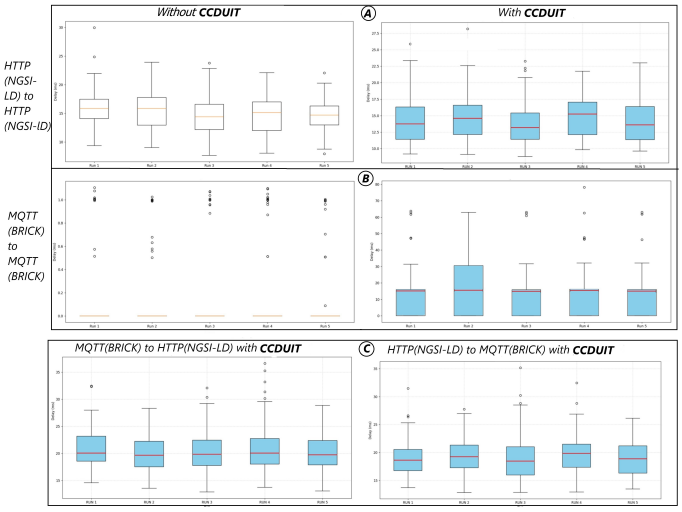

Fig. 6. Ongoing interaction latency for various data exchange scenarios ⒶⒷⒸ, comparing the overhead introduced by CCDUITunder different configurations.

scenario (scenario Ⓐ), we measured the interaction latency for exchanges where both federations used the NGSI-LD data model and communicated over HTTP, without CCDUIT. The results consistently showed low latency, since no additional processing or transformation was required. We then introduced CCDUIT to the same setup, measuring the interaction latency when the system was actively managing the exchanges. Although the prototype ensured context-sensitive validation and policy enforcement, the latency increased only slightly compared to the baseline. This increase is attributable to the prototype's context-checking, policy-checking, and transformation mechanisms.

In the second scenario (scenario Ⓑ), we explored data exchanges where both federations used the Brick data model and communicated over MQTT, first without CCDUIT. As expected, this setup also demonstrated low latency due to the straightforward communication process. When we introduced CCDUIT to manage these MQTT-to-MQTT exchanges, we observed a modest latency increase, consistent with the additional processing.

The third more complex and more relevant scenario (scenario Ⓒ), involved federations with different protocols. We evaluated exchanges where one federation used the Brick data model with MQTT and the other used NGSI-LD with HTTP. The transformation process introduced a higher latency compared to homogeneous scenarios, reflecting the computational effort required for real-time adaptation. However, the results showed that CCDUIT successfully managed these exchanges, maintaining a consistent latency pattern. In all experiments,the results consistently indicated that CCDUIT introduced manageable latency overhead, with initial latencies typically below 1.5 seconds and ongoing interaction latencies below 80 milliseconds for most scenarios. Given that real-time HVAC adjustments or occupancy updates in smart buildings typically operate on timescales of several seconds, these latency values are acceptable for practical deployment. The slight increase in latency is justified by the added benefits of dynamic adaptation, interoperability, and data sovereignty enforcement.

B. CCDUIT Prototype Policy Adaptation Evaluation

To further assess the adaptability of CCDUIT, we conducted a series of experiments designed to measure the system's responsiveness to policy updates.

In the first set of experiments, we measured how quickly CCDUIT responds to a policy update that requires the termination of an active data interaction. This scenario reflects a critical use case in which an ongoing data exchange must be stopped because a policy change no longer allows the interaction. The experiment involved initiating a data exchange between two federations, updating the policy to restrict this exchange, and then recording the latency from the moment the policy was updated until the interaction was terminated. This test was repeated 100 times to ensure the reliability of the results. The findings, illustrated in the accompanying graph, Fig. 7, demonstrate that CCDUIT efficiently detects policy changes and terminates interactions within a short latency window. In addition to evaluating termination latency, we

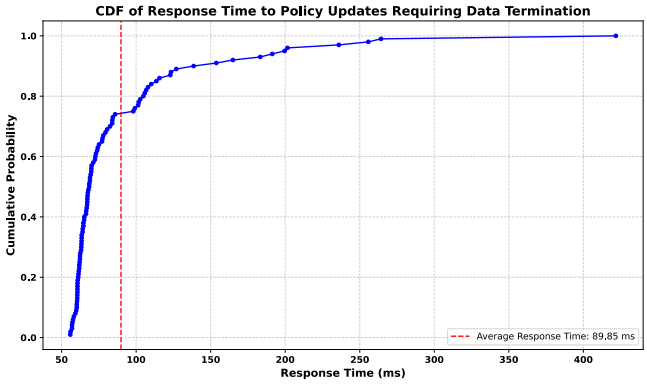

Fig. 7. Response time of CCDUIT to policy updates requiring the termination of active data interactions.

tested the ability of CCDUIT to propagate policy updates across a linear federation network. In this setup, we considered

the three federations of the motivating scenario connected sequentially. Federation 1 collaborates with Federation 2, and Federation 2 collaborates with Federation 3. The goal was to measure how quickly a policy update in Fed. 1 is forwarded to Fed. 2 and subsequently to Fed. 3.

Two key latency metrics were recorded during this experiment. The first metric, latency between consecutive federations, measured the time taken for a policy update to travel from Fed. 1 to Fed. 2 and then from Fed. 2 to Fed. 3. The second metric, end-to-end network latency, captured the total time from the initial policy update in Fed. 1 to its final reception in Fed. 3. These tests were carried out over 100 runs to account for variability and ensure accurate measurements. The results in Fig. 8 demonstrate that CCDUIT efficiently propagates policy updates, exhibiting a low, linearly increasing latency (and as expected 1→3 latency equals the sum of 1→2 and 2→3 latencies).

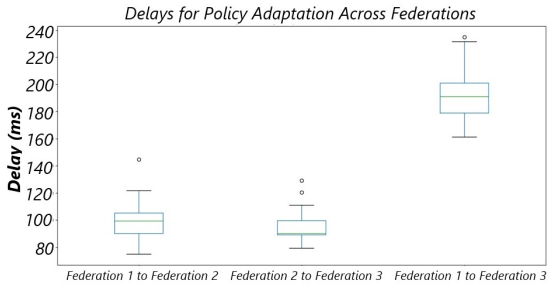

Fig. 8. Latency metrics for propagating policy updates across multiple federations. The graph illustrates both the latency between consecutive federations and the end-to-end network latency from Federation 1 to Federation 3

In general, the evaluation shows that CCDUIT introduces manageable latency overhead while providing critical functionalities such as context-aware data transformation, protocol adaptation, and policy enforcement.

V. Related Work

Federated data spaces have evolved significantly [14] [15] [16] [17] [18]. Although these advances have contributed valuable solutions to isolated aspects of the problem, they lack the ability to offer a comprehensive, adaptive framework for seamless, cross-federation collaboration. Existing solutions focus on specific challenges within federated ecosystems. For example, data converters such as FIWARE's IoT Agents [4] and Apache NiFi [3] address data format discrepancies by transforming data into standardized models such as NGSI-LD. Although effective within their respective scopes, these tools often require manual model-specific configurations, making them less adaptable to dynamic, heterogeneous environments. Similarly, platforms such as the one proposed by Akasiadis et al. [5] support multiple communication protocols, such as REST/HTTP, MQTT and AMQP, through microservices. However, these approaches lack the ability to dynamically adapt to evolving federation requirements and policy changes. Semantic gateways, such as the Semantic IoT Gateway framework [19], facilitate semantic alignment by mapping different data models and ontologies to a shared understanding. Although these gateways preserve contextual integrity during data exchanges, maintaining such mappings in large, dynamic federations is resource intensive and difficult to scale. Furthermore, initiatives like SOFIE [20] leverage Distributed Ledger Technologies (DLTs) to enable secure federation of IoT platforms, while trust-based models [21] manage collaboration by rewarding or penalizing entities based on trust scores. Although these solutions improve security and reliability, they do not fully address the technical complexities of interoperability between federations.

Despite these advancements, existing solutions often require compromising the native technologies of federations by homogenizing data models or embedding one system within another. These compromises reduce flexibility, limit scalability, and impede the autonomy of individual federations. The CCDUIT architecture addresses these limitations. CCDUIT provides a unified framework that dynamically manages data models, communication protocols, and policy constraints without requiring federations to alter their underlying technologies.

VI. Conclusions and Future Work

In this work, we present a prototype implementation of our previous work, the CCDUIT architecture, designed to enable seamless and adaptive collaboration between federated data spaces with heterogeneous data models, protocols, and policies. The modular, scalable, and self-adaptive design of the prototype allows federations to maintain operational autonomy while dynamically managing data exchanges and policy enforcement.

Our evaluation demonstrated that the prototype introduces minimal latency for both initial and ongoing data interactions, even when handling complex transformations and protocol adaptations. In addition, the system efficiently enforces data sovereignty by promptly responding to policy updates and propagating changes across federations. Future work includes extending support for additional communication protocols, automating more context exchange processes, and optimizing data transformation to further reduce latency. Integrating ML techniques for predictive adaptation and improving system resilience for larger federation networks are also promising directions. Real-world deployments in diverse domains, such as smart cities and green energy (e.g. hydro power plant clusters), will help refine the capabilities of the system.

CCDUIT sets the foundation for scalable, policy-compliant and interoperable federations. By open-sourcing the prototype, we invite the community to explore, extend, and contribute to the advancement of federated data ecosystems.

Acknowledgements

This work is partially supported by the Horizon Europe project DI-Hydro under grant agreement number 101122311 and the Energy4Climate Interdisciplinary Center (E4C), which is supported by 3rd Programme d'Investissements d'Avenir [ANR-18-EUR-0006-02].

References

[1] P. DS4SSCC. (2024) Data spaces for smart and sustainable cities and communities (ds4sscc). Accessed: 2024-06-11. [Online]. Available: https://www.ds4sscc.eu/

[2] B. Otto, "A federated infrastructure for european data spaces," *Commun. ACM*, vol. 65, no. 4, p. 44–45, mar 2022. [Online]. Available: https://doi.org/10.1145/3512341

[3] S.-S. Kim, W.-R. Lee, and J.-H. Go, "A study on utilization of spatial information in heterogeneous system based on apache nifi," in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, 2019, pp. 1117–1119.

[4] I. Zyrianoff, A. Heideker, L. Sciullo, C. Kamienski, and M. Di Felice, "Interoperability in open iot platforms: Wot-fiware comparison and integration," in *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2021, pp. 169–174.

[5] C. Akasiadis, V. Pitsilis, and C. D. Spyropoulos, "A multi-protocol iot platform based on open-source frameworks," *Sensors*, vol. 19, no. 19, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/19/4217

[6] N. Papadakis, G. Bouloukakis, and K. Magoutis, "Ccduit: A software overlay for cross-federation collaboration between data spaces," in *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*, 2024, pp. 143–150.

[7] J. Badosa, P. Crifo, D. Dalmazzone, P. Drobinski, C. Guivarch, N. Girard, M. Marot, G. Memmi, and D. Suchet, Eds., *Energy4Climate (E4C) Interdisciplinary Center White Book - Energy and Climate: Research Perspectives*. Palaiseau, France: Institut Polytechnique de Paris and Ecole des Ponts, 2020.

[8] N. Khoder and N. Papadakis, "Ccduit prototype software overlay artifact," https://zenodo.org/records/14753342, https://github.com/satrai-lab/ccduit, 2024, accessed: Dec 15, 2024.

[9] S. Jeong, S. Kim, and J. Kim, "City data hub: Implementation of standard-based smart city data platform for interoperability," *Sensors*, vol. 20, no. 23, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/23/7000

[10] "Context information management (cim) ngsi-ld api v1.4.2," 04 2021. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.07.01_60/gs_cim009v010701p.pdf

[11] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, M. Bergés, D. Culler, R. K. Gupta, M. B. Kjærgaard, M. Srivastava, and K. Whitehouse, "Brick : Metadata schema for portable smart building applications," *Applied Energy*, vol. 226, pp. 1273–1292, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261918302162

[12] H. Krishnan, M. Elayidom, and T. Santhanakrishnan, "Mongodb – a comparison with nosql databases," *International Journal of Scientific and Engineering Research*, vol. 7, pp. 1035–1037, 05 2016.

[13] R. Light, "Mosquitto: server and client implementation of the mqtt protocol," *The Journal of Open Source Software*, vol. 2, 05 2017.

[14] N. Papadakis, G. Bouloukakis, and K. Magoutis, "Comdex: A context-aware federated platform for iot-enhanced communities," in *Proceedings of the 17th ACM International Conference on Distributed and Event-Based Systems*, ser. DEBS '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 37–48. [Online]. Available: https://doi.org/10.1145/3583678.3596890

[15] B. Cheng, G. Solmaz, F. Cirillo, E. Kovacs, K. Terasawa, and A. Kitazawa, "Fogflow: Easy programming of iot services over cloud and edges for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 696–707, 2018.

[16] F. Cirillo, G. Solmaz, E. L. Berz, M. Bauer, B. Cheng, and E. Kovacs, "A standard-based open source iot platform: Fiware," *IEEE Internet of Things Magazine*, vol. 2, no. 3, p. 12–18, Sep 2019.

[17] J. Carvajal Soto, O. Werner-Kytölä, M. Jahn, P. J., D. Bonino, C. Pastrone, and M. Spirito, "Towards a federation of smart city services." 11 2015.

[18] A. Morelli, L. Campioni, N. Fontana, N. Suri, and M. Tortonesi, "A federated platform to support iot discovery in smart cities and hadr scenarios," 09 2020, pp. 511–519.

[19] K. Kotis and A. Katasonov, "Semantic interoperability on the web of things: The semantic smart gateway framework," in *2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems*, 2012, pp. 630–635.

[20] D. Lagutin, F. Bellesini, T. Bragatto, A. Cavadenti, V. Croce, Y. Kortesniemi, H. C. Leligou, Y. Oikonomidis, G. C. Polyzos, G. Raveduto, F. Santori, P. Trakadas, and M. Verber, "Secure open federation of iot platforms through interledger technologies - the sofie approach," in *2019 European Conference on Networks and Communications (EuCNC)*, 2019, pp. 518–522.

[21] H. Yahyaoui, Z. Maamar, M. Al-Khafajiy, and H. Al-Hamadi, "Trust-based management in iot federations," *Future Generation Computer Systems*, vol. 136, pp. 182–192, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X22002023