CCDUIT: a software overlay for cross-federation collaboration between data spaces

Nikolaos Papadakis, Georgios Bouloukakis, Kostas Magoutis

CCDUIT: A Software Overlay for Cross-Federation Collaboration between Data Spaces

Nikolaos Papadakis[†], Georgios Bouloukakis[†], Kostas Magoutis[*‡],
{nikolaos.papadakis, georgios.bouloukakis}@telecom-sudparis.eu, magoutis@ics.forth.gr
[†]Télécom SudParis, Institut Polytechnique de Paris, France
[*]Institute of Computer Science (ICS), Foundation for Research and Technology - Hellas (FORTH), Greece
[‡]Computer Science Department, University of Crete, Greece

***Abstract*—In contemporary urban environments, the federation of IoT-empowered data spaces is gaining ground as a concept, however a single unifying approach to federation is still elusive. As a result, the exchange of data *across* heterogeneous federated data spaces often encounters challenges, such as different data models and data exchange protocols, or stringent policies prohibiting data sharing across federations. This paper introduces CCDUIT, a software overlay architecture designed to address these issues, facilitating seamless cross-federation collaboration. As a comprehensive solution, CCDUIT offers modularity, scalability, and interoperability, enabling efficient and sovereignty-preserving data exchange across diverse federations. CCDUIT leverages rich property graph models for context modeling of federations, which are exchanged across federations via publish/subscribe topic schemes, with data sharing, and access control governed by policy mechanisms matching the topics. Our experimental results demonstrate that CCDUIT significantly reduces the complexity and effort involved in data management and sharing across federations, with a quantifiable decrease in operational overhead by approximately 40% to 60%. This streamlines collaboration while ensuring compliance with data sovereignty and sharing policies, providing a solution to a longstanding challenge in federation-based data ecosystems.**

***Index Terms*—Software Overlays, Interoperability, Smart Data Spaces, Distributed & Federated Information Systems**

I. INTRODUCTION

The evolving landscape of digital data ecosystems, along with emerging public policies such as the European Strategy for Data [17], are driving the realization of federated data spaces, networks of interconnected data repositories, often empowered by IoT technologies, that facilitate interoperability and collaboration across the boundaries of individual data spaces. The goal is to create common data spaces where seamless data exchange and integration are possible, enabling participants to work together more effectively. The existing body of work in federated data spaces is rich and varied, tackling numerous aspects of interoperability and data sharing [6], [7], [13], [16]. While the concept of federations has been instrumental in enabling interoperability among diverse communities [5], [14], [18], challenges are still posed by the distinct technologies, data models, and policy constraints inherent in these varied groups [19].

Bridging the heterogeneity of data formats across different technologies is often possible through data converters and semantic gateways [2], [11], [22] within a federation. However, a notable gap in this domain is the lack of solutions that enable different federated data spaces to collaborate without necessarily assimilating into a single federation. In other words, while current approaches focus on integrating diverse systems *within* a single federated framework, they do not extend to facilitating seamless collaboration *across* heterogeneous federations while allowing them to maintain their individual technological and policy characteristics. For instance, current approaches for inter-federation collaboration often entail converting different platforms into a common format [10] [18]. This approach, while effective to a degree, overlooks the potential and efficiency of allowing federations to interact in their native formats, keeping their existing setups, and thus the unique operational characteristics of each federation are not preserved. Another critical limitation that is often overlooked is the absence of robust inter-federation context-exchange mechanisms. Such mechanisms would enable federations to dynamically adapt to each other's policies, thereby preserving data sovereignty and policy compliance in multi-federational environments, and engage in policy-driven collaborations, which is essential for achieving true interoperability in practice. What is currently lacking is a system capable of creating and managing a *federation of federations*, addressing interoperability and scalability through the diverse technologies, data models, and policies prevalent in existing federated systems.

In response to these challenges, we introduce CCDUIT, a novel software overlay architecture designed to bridge the gaps in current federation-level collaborations of data spaces. CCDUIT offers a seamless integration platform for multiple federations, enabling them to collaborate effectively while preserving their individual sovereignty and operational characteristics. The primary contributions of our work include:

1) Design of the CCDUIT software overlay architecture to enable efficient cross-federation data exchange.
2) A publish/subscribe topic-based schema for context sharing that enables scalable "federation of federations".
3) Design of a component-based CCDUIT node architecture that facilitates seamless data sharing across diverse data spaces, while respecting individual data sovereignty and the need to use diverse technologies.
4) Evaluation, which includes an analysis of the scalability of CCDUIT and its comparison with scenarios devoid of

its application

The remainder of this paper is structured as follows: In §II, we outline the evolution and theoretical underpinnings of federated data spaces. §III presents a review of existing technologies for federation interoperability and identifies gaps that CCDUIT aims to fill. The complexities and needs of modern federations, illustrated through a practical scenario, are discussed in §IV. The CCDUIT software overlay architecture is presented in §V, where we discuss its architectural design and key features. A comparative evaluation of CCDUIT is covered in §VI. Finally, §VII concludes this article, reflecting on the impact of CCDUIT, its future potential, and avenues for further research in the field.

II. Background

Federated architectures and platforms [20] enable interconnectivity among heterogeneous systems and devices across various domains of information, including administrative, geographic, and contextual data. Federations are characterized by their unique topologies, data models, and data exchange protocols, each essential for effective data management within data ecosystems, particularly those driven by the Internet of Things (IoT).

A. Federation Elements

Topology. A federation's topology is defined by the arrangement of its participant nodes, ranging from simple sensors to complex processing units. Common configurations, such as tree, star, line, and mesh, influence data flow and network efficiency, where mesh topologies provide resilience at the cost of increased complexity, and star topologies offer simplicity with reliance on a central node [15]. **Data Model.** To tackle interoperability challenges, diverse data models are harmonized through standards like NGSI-LD, which leverages Linked Data semantics for data spaces—environments facilitating trusted data sharing [10], [1]. Specialized models like GTFS[1] and Brick [2] cater to specific domains such as public transportation and smart buildings, augmenting NGSI-LD for comprehensive data representation. **Protocol.** Protocols like MQTT and CoAP underpin data exchange by setting efficient communication rules for IoT devices, crucial for federation functionality [8], [21].

B. Federation Platforms

A variety of platforms and architectures have been developed to support federation in data spaces and beyond. TrustyFeer [13] utilizes federated cloud environments to enhance the quantity of services exchanged between cloud providers adhering to Service Level Agreements (SLAs). MARGOT [16] places a strong emphasis on ensuring High Availability Disaster Recovery (HADR) by making effective use of caching capabilities within its federated node infrastructure. ComDeX [18] adopts a distributed context-aware federation architecture, enabling effective widespread IoT applications through novel information dissemination techniques for collaborative data exchange between smart communities. Fogflow [6] presents a federated fog computing framework tailored for the design and execution of IoT applications at a metropolitan scale. CPaaS.io [7] established federation as a goal aimed at the integration of diverse private IoT systems, contributing to the evolution towards a worldwide IoT. The ALMANAC Smart City Platform [5] employs a federated deployment strategy in Turin to connect diverse ICT systems among different entities like the waste management company and the municipality, addressing both technological and governance challenges associated with smart city initiatives.

This confluence of federated architectures, data models, topologies, and protocols in IoT platforms exemplifies the complexity of these systems and highlights their potential to revolutionize the way we interact with and manage our interconnected world.

III. Related Work

The realm of federated data spaces ecosystems has seen significant advancements, as presented in the background section §II to foster interoperability among diverse stakeholder entities within these ecosystems. However, the challenges posed by the heterogeneity of technologies, data models, and policy constraints across different communities often extend beyond what federated architectures alone can resolve. This underscores the necessity for adoption of advanced tools, like data converters and semantic gateways, which are often used in combination with aforementioned federations [19].

Data converters play a pivotal role in mitigating the discrepancies in data formats prevalent across various communities. By transforming data from one format to another, these converters facilitate a smoother data flow between entities operating on different technological platforms. A notable example is FIWARE's IoT Agents [22], which enable the conversion of IoT data into the NGSI-LD format. IoT Agent applications and others that utilize similar approaches are data model-specific, thus necessitating different IoT Agents for distinct data structures. There are other, large-scale systems for this, like Apache Nifi [11] which offers a comprehensive suite of processors for versatile data transformation needs. Efforts have also been made to address specifically the challenges of handling various data exchange protocols between heterogeneous systems. Akasiadis et al. [2] address the challenges of handling various communication protocols in IoT systems by supporting multiple protocols, at once on the same platform, such as REST/HTTP, MQTT and AMQP. The platform utilizes open-source frameworks, via a microservices-based approach.

In addition to data converters, semantic gateways emerge as crucial elements in achieving interoperability by semantically aligning the data exchanged between heterogeneous systems. This involves mapping different data models and ontologies to a common understanding, thus preserving the meaning and context of data across various IoT platforms. The use of semantic gateways is particularly evident in solutions

[1]https://gtfs.org/

[2]https://brickschema.org/

that integrate disparate IoT systems while maintaining their semantic integrity. For instance, the Semantic IoT Gateway framework [12] offers a way to interlink different IoT systems by translating and aligning their semantic representations. This approach is vital in scenarios where maintaining the contextual meaning of data is as important as the data itself, especially in complex federated environments.

Beyond data transformation and technology alignment, there have been concerted efforts to foster agreements and smoother dynamic collaboration within federations. Initiatives like SOFIE [14] demonstrate the potential of federating existing IoT platforms in an open and secure manner, utilizing Distributed Ledger Technologies (DLTs) as a consensus mechanism. Additionally, the trust-based evolutionary game model presented by Yahaoui et al. [9] offers an approach to managing IoT federations by integrating trust scores based on direct experiences and feedback, rewarding or penalizing entities based on their behaviors.

Despite advancements in federated IoT ecosystems, a gap persists: no standardized method integrates diverse federated data spaces into a unified framework without altering native technologies. The concept of a *federation of federations* to handle the intricacies of varied technologies, data models, and policies across IoT communities is still unexplored. CCDUIT is a novel architecture designed to tackle these interoperability and scalability challenges, offering a cohesive model for managing and integrating disparate federations, thus revolutionizing the collaborative dynamics of federated ecosystems.

IV. Motivating Scenario

In this work we envision a network of smart data spaces, each operating using a federated platform in its own domain, yet potentially interdependent through future shared objectives and data dependencies. These federations, each focusing on environmental monitoring, healthcare, urban planning, transportation, and energy, are distinguished by unique data models, data exchange protocols, and governance policies. They operate autonomously but could potentially be a part of a larger ecosystem where the exchange and integration of data are critical for informed decision-making and effective management of urban environments.

As an example of such a larger ecosystem, consider a **fictional smart community called EcoVille** which aims to enhance urban living conditions. Suppose that EcoVille aims to undertake a study to discover correlations between various urban factors and the health of its citizens. This necessitates the integration and analysis of data from the environmental, healthcare, transportation, and energy sectors. EcoVille wants to harness those diverse data for urban planning adaptations, targeted health interventions, and other policy changes. However, the process is not straightforward due to systemic challenges in data acquisition and integration across *five* federations, each with its own technological and policy frameworks: **Environmental Federation Ⓐ:** Utilizes environmental NGSI-LD models[3] and MQTT for sensor data exchange. **Urban Development Federation Ⓑ:** Employs the Brick Schema and RESTful APIs for smart building data integration, supported by analytics on platforms like SkySpark[4]. **Healthcare Federation Ⓒ:** Adopts HL7 FHIR and DICOM for healthcare information, integrating IoT standards for real-time patient data, with strict adherence to HIPAA[5] for privacy [3], [23]. **Transportation Federation Ⓓ:** Implements GTFS and AUTOSAR for public and vehicular transport data, using MQTT and Websockets for data exchange [4]. **Energy Federation Ⓔ:** Tracks energy data through CIM and IEC 61850 standards[6], employing AMQP and CoAP for communication.

These federations, each operating with unique data types and protocols, encounter significant interoperability challenges. Without a unified approach to address these issues, the lack of data integration hinders the seamless exchange of information across sources. This fragmentation leads to inefficiencies, potential data loss, and missed opportunities for synergy. Moreover, the diverse data sharing policies and privacy regulations across federations add layers of complexity to data governance. Non-compliance risks legal repercussions and undermines user trust. As the federations scale, neglecting interoperability can result in a cumbersome, error-prone system that fails to leverage the collective potential of federated data spaces, ultimately impeding the overarching goal of collaborative advancement CCDUIT comes to address exactly these challenges, through an architecture designed to facilitate efficient, seamless cross-federation data exchange.

V. CCduit Software Overlay

Here we describe in detail the architecture of CCDUIT. First, we present an overview of CCDUIT as a high level graph. Then, we introduce a mechanism for context-aware data exchange between federations. Finally, we describe the architecture of a CCDUIT node, along with a use case scenario.

A. Overview of CCDUIT

CCDUIT is a software overlay designed to seamlessly bridge the communication gaps between diverse federations, addressing interoperability and data sharing to facilitate collaboration and information exchange among smart communities. CCDUIT's architecture offers: (i) *modularity* by ensuring easy adaptability and integration with various existing systems; (ii) *scalability* by allowing to effortlessly manage the growing network of federations and their expanding data needs and (iii) *interoperability* by enabling the system to navigate through the diverse data models and data exchange protocols employed by different federations.

At a high level, CCDUIT can be visualized as a graph, $G = (N, E)$, as exhibited by Fig. 1, where each node $n \in N$, represents a distinct federated data space, termed a CCDUIT node. These nodes embody individual federations, encapsulating their unique characteristics and functionalities,

[3]https://github.com/smart-data-models/dataModel.Environment

[4]https://skyfoundry.com/skyspark/

[5]https://www.cdc.gov/phlp/publications/topic/hipaa.html

[6]https://webstore.iec.ch/publication/65191

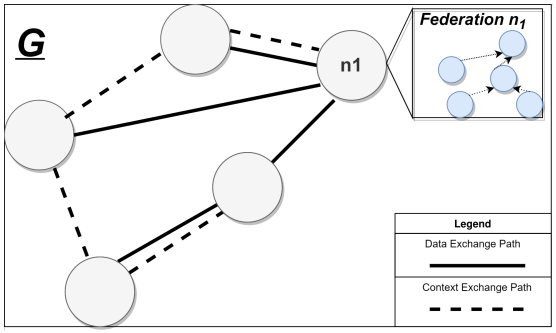

Fig. 1. High level graph representation of CCDUIT.

and facilitating the reception and dissemination of information, forming the backbone of the CCDUIT system.

In this graph, two types of edges $e \in E$ exist:

1) Data Exchange (DE) Edges, E_d, representing the actual data exchange interactions between federations. The labels of these edges, $l_d(e)$, include specific details of the data exchange, such as the nature of the data, format, and the technical protocol used for the exchange.
2) Context Exchange (CE) Edges, E_c, symbolizing the exchange of contextual information, crucial for enabling and guiding the data exchanges. The labels on these edges, $l_c(e)$, represent the *policies* that direct these context exchanges, encompassing aspects like data sharing rules and compliance requirements. Our system is designed to infer context through indirect paths by leveraging established policies and prior interactions, thus not all data exchanges require direct context exchange edges

Each CCDUIT node interacts with others through these edges, forming a network of contextual communication and data exchange. This dual interaction mechanism, represented by E_d and E_c edges, is fundamental to the operation of CCDUIT, ensuring that data flows efficiently and also in compliance with established policies and mutual understandings.

Policies play a crucial role in dictating what kind of contextual data is permitted to forward or exchange, and where it can be sent. For instance, a policy might specify that environmental data can only be shared with federations or communities that have agreed to certain environmental standards. This ensures that sensitive data is only exchanged with entities that adhere to the same level of commitment to environmental protection.

B. Inter-Nodal Data Exchange

We now discuss how communities in different federations interact by exchanging contextual data. CCDUIT's main goal is to facilitate interactions between CCDUIT nodes, via inter-nodal data exchange for data, policy, and contextual information flow over a communication infrastructure that allows each node in the network, representing a federation (e.g., a smart community), to interact seamlessly with others.

In the CCDUIT architecture, policies are set by the federations and are enforced by the *Synergy Engine* (Fig. 2, more details in §V-C), which ensures that all data exchanges align with the federation's objectives and compliance requirements.

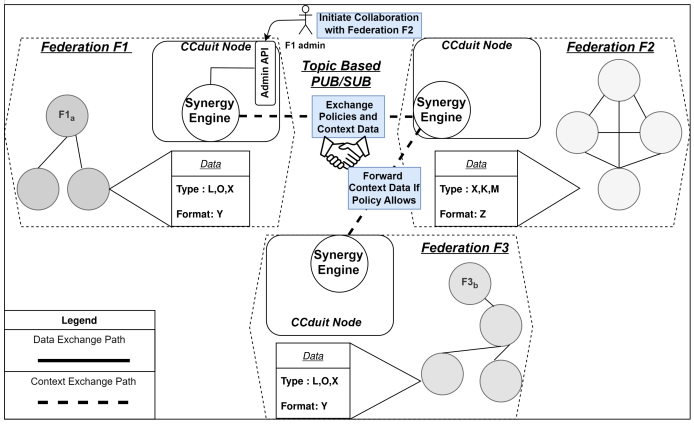

Fig. 2. Inter-Nodal Data Exchange within the CCDUIT Architecture.

This engine plays a crucial role in managing and synchronizing policies and contextual information across different nodes. It acts as a mediator, aligning the various data sharing and policies with the overarching objectives of the federations involved. The Synergy Engine also ensures that any updates in the context are promptly reflected across all connected nodes, maintaining a consistent and up-to-date network state. For policies specifically, this process involves automated protocols for minor adjustments, while significant changes could require human intervention for thorough discussion and agreement.

To enable efficient data exchange conforming to the defined policies (and thus context exchange), inter-nodal communication in CCDUIT is implemented via a topic-based publish/subscribe (pub/sub) schema, described next.

CE Schema: The schema of context exchange (topics) is structured as follows:

`Federation/Federation_ID/Policy_ID/Data_type`

with the following definitions for specific terms:

- *Federation ID:* uniquely represents each federation, ensuring that data and policies are correctly attributed.
- *Policy ID:* policies governing data sharing and usage are associated with specific identifiers, allowing for streamlined policy management and enforcement.
- *Data Type:* data being exchanged is categorized under specific data types (the data types present in the knowledge base to be elaborated later on), facilitating efficient data processing and transformation.

CCDUIT's schema enables federations to share and receive relevant data and information with precision and ease.

Fig. 2 demonstrates typical inter-nodal data exchange, highlighting the role of the *Synergy Engine* and the data flow enabled by the *CE schema*. Fig. 2 depicts a scenario involving three federations: F1, F2, F3. F1 and F3 use data model Y, while F2 operates with a different format, Z. When a community in Federation F1 needs data of type X, the process unfolds as follows: An administrator from F1 initiates collaboration with Federation F2 by defining a data exchange policy for F1's context and agreeing to exchange policies and context data with F2. This initiation involves the CCDUIT node of F2 subscribing to the `Federation/Federation_F1/`

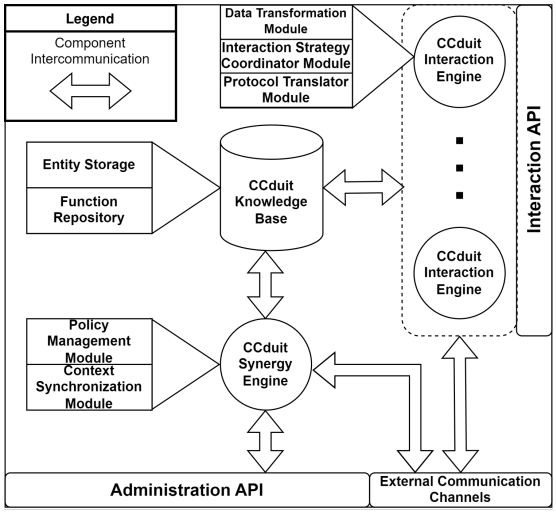

Fig. 3. The Structural Overview of a CCDUIT Node.

topic of F1's CCDUIT node, and vice versa for F1 to F2's topic. In instances where F2 requires only specific context elements from F1, such as policies, it could tailor its subscription to `Federation/Federation_F1/+/Policy`. Through this subscription, the community in F1 gains access to F2's contextual information, including the availability of the sought-after data type X and the data model used by F2. F1 then develops a custom function to facilitate this data exchange, which is also shared with F2 via the CE mechanism. If federation F3 also seeks to engage in data exchange with F2, it follows a similar collaboration pattern, subscribing to the relevant topics from F2. Importantly, if the policy governing F1's shared context information permits, F2 can forward this information to F3 through F3's subscription to the context topics. This action enables F3 to utilize the custom function created by F1, streamlining its data exchange with F2.

This dynamic exemplifies the capability of CCDUIT to orchestrate complex inter-federation data and context exchanges, via inter-nodal communication. We next delve into the internal structure of a CCDUIT node.

C. Detailed Breakdown of CCDUIT Node Components

Fig. 3 depicts the CCDUIT node, a key building block of the architecture that comprises the following components:

Interaction API. It is the primary interface for external entities to engage with the CCDUIT node. It is designed to be the central gateway for initiating and managing interactions such as data requests and context entity manipulations. It offers a front-end that caters to the diverse needs of various federations. Key functionalities include: (i) *Management of Context Entities:* users can define and add new or modify context entities to expand the knowledge base and interaction capabilities of the network; (ii) *Querying Context Entities:* enables users to retrieve specific information about context entities, facilitating informed decision-making and efficient data management; and (iii) *Lifecycle of Interactions:* initiating new interactions like data exchanges, triggering processes and collaborations within the network, providing real-time insights into ongoing interactions, allowing users to track progress and assess efficiency.

Interaction Engine. Comprising several modules, the Interaction Engine facilitates data interoperability. The *Data Transformation Module* adapts data models for seamless integration across federations using custom functions. The *Interaction Strategy Coordinator Module* identifies optimal data exchange pathways, considering context, policies, and specifications. The *Protocol Translator Module* aligns differing data exchange protocols to ensure cross-federation compatibility.

Knowledge Base (KB). Functions as a repository housing the following information crucial for facilitating interactions among federations: (i) *Federation Information:* profiles of each federation, including their structural characteristics and operational principles; (ii) *Community Context:* specific information about individual communities within federations; (iii) *Data Models:* specifications of the data models employed by the various federations, ensuring data compatibility and effective transformation; (iv) *Custom Functions:* a collection of specialized functions developed for unique data processing needs, promoting efficiency and reuse across federations; (v) *Policies:* comprehensive documentation of data sharing and usage policies, critical for maintaining legal and ethical compliance in data exchanges; (vi) *Interaction Context:* records of ongoing and past interactions, providing insights into the dynamics of federation relationships and aiding in decision-making for future exchanges. The KB utilizes a labeled property graph format to represent complex relationships within federations. For example, a "Federation" node, with properties such as name and topology, can be linked to a "Community" node via an "IncludesCommunity" edge, depicting their association [24]. This format is chosen for its ability to accurately mirror the complex interplay between federations and communities, offering an intuitive grasp of system-wide dependencies and connections. It supports efficient querying of intricate relationships, essential for quick data retrieval, and scales effectively with data growth, ensuring sustained performance.

Synergy Engine. It consists of the following modules: (i) *Policy Management Module:* tasked with enforcing data sharing policies. It ensures that all data exchanges comply with the established rules and regulations of the involved federations; and (ii) *Context Synchronization Module:* maintains up-to-date and consistent information across federations, this module synchronizes the contextual information, reflecting any changes or updates in real-time. It manages updates efficiently by tracking changes through the structured components of the CE schema – Federation ID, Policy ID, and Data Type. This approach ensures that all federations operate with the latest policies and other interchanged context data, maintaining the integrity and relevance of data exchanges within CCDUIT.

Administration API. It is responsible for managing the CCDUIT node. It serves as the backbone for administrators, offering an array of tools essential for maintaining the efficacy and integrity of the system. This API facilitates various administrative tasks, ensuring that the system remains adaptable and responsive to the evolving needs of each federation. Key

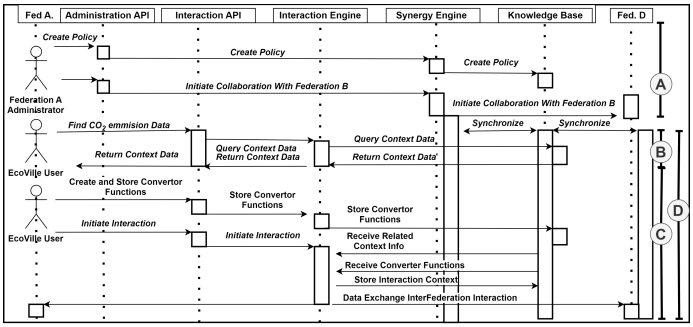

Fig. 4. Workflow of CCduit Node Operations within Federation A for Inter-federation Data Exchange with Federation D.

functionalities of the Administration API include: *Initiation of Federation Collaborations*, *System Configuration*, *Policy Management*, *Node Health Monitoring*, *User and Access Management* and *System Updates and Maintenance*.

D. Example Use Case of CCDUIT *Node's components*

We now explore the practical application of CCDUIT within the context of our motivating scenario (EcoVille, one of possibly many communities in Federation A) as described in §IV. EcoVille aims to acquire CO2 emission data from vehicles, which is vital for its goals. This data is available in a community belonging to Federation D. We focus on the interaction of Federation A's CCDUIT node and its internal components, as illustrated in Fig. 4.

Fig. 4 Ⓐ: Initiating Collaboration and Policy Exchange. EcoVille's first interaction with Federation D involves several preliminary steps. *Administration API Use:* an administrator from Federation A utilizes the Administration API of CCDUIT to initiate collaboration with Federation D. This is done by first defining a data exchange policy for EcoVille's context, using the Synergy Engine's Policy Management module. *Policy and Context Data Exchange:* following the policy setup, and collaboration agreement, Federation A exchanges its policies and context data with Federation D. This exchange leverages the CE topics, as detailed in §V-B.

Fig. 4 Ⓑ: Context Synchronization and Data Discovery. Once the initial exchange is complete, the process unfolds as follows. *Context Synchronization:* the Synergy Engine's Context Synchronization module ensures that the context data from both federations are aligned and up-to-date. *KB Interaction:* EcoVille then searches for the required CO2 emission data through the Interaction API, which accesses the KB's Entity Storage. This step allows EcoVille to understand where and in what format the data is available in Federation D.

Fig. 4 Ⓒ: Handling Heterogeneity and Interaction Creation. Upon discovering that the data format and exchange protocol of Federation D are incompatible with EcoVille's systems, the following steps are taken. *Function Creation for Conversion:* EcoVille creates custom functions to address the data format and protocol disparities. These functions are then stored in the KB's Function Repository via the Interaction API. *Data Interaction and Exchange Initiation:* With the custom functions in place, EcoVille initiates a new data interaction process through the Interaction API, which triggers the Interaction Engine. Then, the interaction Engine employs its various modules to facilitate effective data exchange. *Data Transformation module:* converts the data into a compatible format, from the AUTOSAR application compliant CO2 format to the NGSI-LD format that EcoVille uses. *Protocol Translator module:* translates the data exchange protocol as needed, here MQTT to HTTP. *Interaction Strategy Coordinator:* determines the most efficient pathways for data exchange. From the point interaction is created and on, data exchange happens without the need for any additional coordination.

Fig. 4 Ⓓ: Continuous Synchronization by Synergy Engine. Throughout this interaction, the Synergy Engine carries out the following tasks. *Policy and Contextual Updates:* it continuously synchronizes policy and contextual updates between the involved federations. *Adaptation to Changes:* any changes in policies or data models in either federation are promptly integrated into EcoVille's data requests and integration strategies, ensuring up-to-date data exchange.

This use case illustrates the comprehensive capabilities of CCDUIT 's components in facilitating intricate data exchange processes between different federations while maintaining data sovereignty, policy compliance and data integrity.

E. Extensibility and Customization Features

CCDUIT employs a modular design to facilitate the independent development, maintenance, and upgrade of components, streamlining complexity and easing the integration of new technologies. It is designed to be adaptive to address the varied technological and policy needs across federations, with custom functions tailored to the specific requirements of diverse federations.

Adaptability: The CCDUIT architecture empowers federations to craft custom functions, such as unique data transformation algorithms or policy enforcement mechanisms, which can be selectively shared with partner federations. This sharing is at the federation's discretion, ensuring confidential information remains protected. Shared functions enhance interoperability, reducing redundant development efforts across federations. Each federation can thus customize their CCDUIT node to meet their unique needs while maintaining the broader system's integrity and functionality.

VI. CCDUIT COMPARATIVE EVALUATION

In the following comparative evaluation, we provide an in-depth analysis of the advantages of CCDUIT when compared to scenarios where such a system is absent.

Experimental Setup. Our evaluation includes a targeted case study within a controlled environment across three federations, focusing on the exchange of transportation data. Federations Ⓐ and Ⓒ use NGSI-LD models and HTTP protocols, whereas Ⓑ employs GTFS models via MQTT. The goal is to enable federations Ⓐ and Ⓒ to access GTFS data from Ⓑ.

Development Approach: We explore 3 different methods:

1. The "Clean-Slate (CS)" CCDUIT *development process*: this method involves initially deploying CCDUIT within federations, a process that requires considerable setup effort,

TABLE I
COMPARATIVE ANALYSIS OF EFFORT IN LINES OF CODE (LOC)

Approach	Value LoC
Clean-Slate CCDUIT	1518
Pre-Configured CCDUIT	738
Non-CCDUIT	1420

as outlined in the first step subsequently presented in our methodology. Some effort, however, is partially offset when Federation Ⓐ shares its custom functions with Federation Ⓒ. To simplify understanding the total effort of this process using CCDUIT, we present it through a series of clear steps.

Steps in the CCDUIT Process:

1) Model Creation and Storage: Federations Ⓐ and Ⓑ define and store NGSI-LD entities representing contextual information in CCDUIT's knowledge base.
2) Initiation of Collaboration and Policy Exchange: Federation Ⓐ initiates a policy exchange with Federation Ⓑ, exchanging vital context information.
3) Transportation Data Discovery: Federation Ⓐ queries CCDUIT to discover and request transportation data types from Federation Ⓑ.
4) Custom Function Creation for Data Conversion: Federation Ⓐ develops a custom function for data conversion and uploads it to CCDUIT.
5) Initiation of Data Exchange: The data exchange is initiated between the communities of Federations Ⓐ and Ⓑ, utilizing stored custom functions and discovered API endpoints.
6) Extension to Federation Ⓒ: Federation Ⓒ repeats the process with Federation Ⓑ, leveraging shared context from Federation Ⓐ to reduce effort.

2. Pre-Configured (PC) Development: This approach utilizes pre-configured CCDUIT context data, eliminating initial setup stages like model creation and storage for all federations, termed as "PC CCDUIT " setup.

*3. Non-*CCDUIT *Development:* We also evaluated a manual strategy without CCDUIT, involving data sharing policy establishment and script development for data handling tasks. No contextual data exchange occurs between Federations Ⓐ and Ⓒ, while also ensuring a fair comparison by using the same codebase for both CCDUIT and non-CCDUIT scenarios.

Metrics: The primary measure is the effort, quantified by lines of code (LoC), required for achieving federation collaboration with and without using CCDUIT.

Results. The data presented in Table I indicates a clear trend in the effort required across different setup approaches. Clean-Slate CCDUIT, while initially labor-intensive, establishes a robust approach for data exchange. The high initial effort of 1518 LoC can be attributed to the comprehensive setup required, including model creation, policy exchange, and custom function development. This foundational work, however, paves the way for more efficient subsequent interactions.

In contrast, the Pre-Configured CCDUIT setup shows a significant reduction in effort, requiring only 738 LoC. This reduction is primarily due to the reuse of existing context data and pre-developed custom functions, demonstrating the benefits of shared resources and knowledge within the CCDUIT ecosystem. The Non-CCDUIT approach, with 1420 LoC, underscores the inefficiencies of traditional methods. The lack of a shared context and the need to independently develop protocols and data models result in a nearly equivalent effort to the Clean-Slate approach but without the long-term benefits of an established CCDUIT infrastructure.

A. Scalability Evaluation

We expand our analysis by examining CCDUIT's scalability across a progressively increasing number of simulated federation networks. This broader analysis is crucial for assessing the potential of CCDUIT in large-scale applications.

Experimental Setup. We conducted an experiment focusing on simulations of an expanding array of federation networks using CCDUIT.

Simulation Approach. We systematically varied three key parameters to reflect realistic and challenging scenarios:

Data Type Diversity (DT%): This metric varies from 0% to 100%, assessing the range of data models and protocols across federations. A 0% DT indicates uniform data models, reducing the need for conversion and lessening context exchange benefits. Conversely, a 100% DT signifies complete diversity, hindering the use of shared context due to a lack of commonality. A 50% DT, representing a balance between uniformity and diversity, optimizes the utility of context exchange.

Context Exchange Percentage (CE%): CE% evaluates context exchange frequency among federations. Extreme DT% values (0% and 100%) minimize context exchange's relevance—unnecessary at 0% DT and ineffective at 100% DT due to incompatible data models.

Pre-configured Nodes Percentage (PC%): PC% examines how the initial setup of CCDUIT nodes affects integration effort. A higher PC% lowers the initial effort, but the benefit diminishes once a node completes its first data exchange, highlighting the limited long-term significance of initial node conditions.

These parameters were selected to reflect the complexity and adaptability of CCDUIT in diverse federation setups, simulating networks from homogeneous (low DT%) to highly diverse and interconnected (high DT% and CE%). The simulation employs a script to generate nodes and interactions, creating varied network configurations.

Metrics: The scalability of CCDUIT is assessed using the number of lines of code (LoC) needed for federation interactions under different scenarios.

Results. Results, depicted in multi-panel plots in Fig. 5, show LoC efforts across network sizes, with annotations indicating the number of interactions per node. Notably, context exchange (CE) is most beneficial at 50% data type diversity (DT), enhancing data exchange efficiency. At 0% and 100% DT, CE's effect on effort reduction is minimal.

Findings reveal CCDUIT's scalability and suggest that effort depends on network initial conditions and interaction complexity. Strategic configurations and managing of context exchanges can optimize CCDUIT's large-scale deployment.

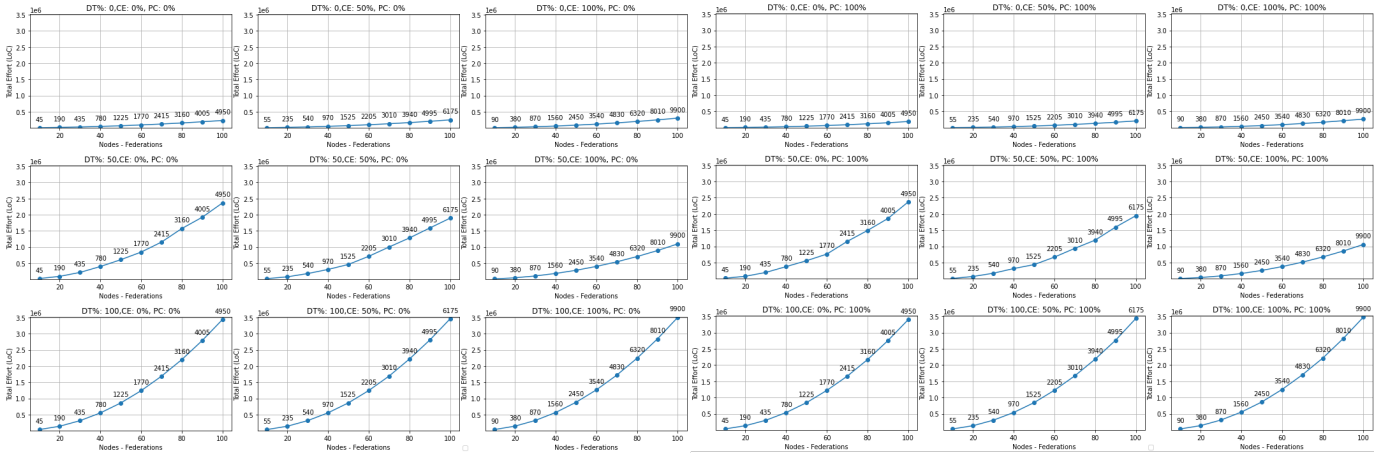

Fig. 5. Scalability Analysis: Tracking Code Volume Growth in CCDUIT across Simulated Federated Data Spaces

Initial setup efforts, even in a Clean-Slate mode, are balanced by the efficiency of subsequent exchanges.

This analysis underscores CCDUIT's flexibility and the importance of a deployment strategy tailored to the federation network's data model landscape. Effective context exchange and the selective benefit of pre-configuration significantly reduce integration efforts in dynamic, moderately diverse environments.

VII. Conclusion and Future Work

CCDUIT contributes an architecture for tackling the challenges of data exchange between federated IoT-enhanced data spaces. State-of-the-art solutions focus on enabling data exchange within such data spaces using federated software architectures, while ignoring cross-federation data exchange. CCDUIT introduces a software overlay that handles diverse data models, protocols, and data sharing policies, ensuring smooth and effective cross-federation collaboration. Its architecture, characterized by modularity, scalability, and interoperability, is ideally suited for contemporary federations. CCDUIT's use of graph models for contextualizing data and specialized Context Exchange (CE) publish/subscribe schema for data exchange is particularly effective in managing data sovereignty and complying with policies defined by federations. Empirical studies validate CCDUIT's ability to streamline data sharing processes, enhancing collaboration while upholding data sovereignty. However, comprehensive experimental studies, particularly on latency and performance aspects, are still required.

Looking ahead, CCDUIT offers scope for further development and refinement. Future initiatives could focus on incorporating enhancements such as advanced analytics, AI-driven decision-making processes, and stronger security measures. The potential of CCDUIT to adapt to new technologies and evolving needs of federations highlights its versatility, making it an invaluable asset for smart city ecosystems.

Acknowledgements

This work is partially supported by the Horizon Europe project DI-Hydro under grant agreement number 101122311

References

[1] Data Space Business Alliance - DSBA. Technical convergence discussion document version 2.0, 2023. Accessed on: November 2023.
[2] Akasiadis et al. A multi-protocol iot platform based on open-source frameworks. *Sensors*, 2019.
[3] Bender et al. HL7 FHIR: An Agile and RESTful approach to healthcare information exchange. In *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, 2013.
[4] Bo et al. Basic Concepts on AUTOSAR Development. In *ICICTA*, 2010.
[5] Carvajal et al. Towards a Federation of Smart City Services. 2015.
[6] Cheng et al. FogFlow: Easy Programming of IoT Services Over Cloud and Edges for Smart Cities. *IEEE Internet of Things Journal*, 2018.
[7] Cirillo et al. A Standard-Based Open Source IoT Platform: FIWARE. *IEEE Internet of Things Magazine*, 2019.
[8] Fysarakis et al. Which IoT Protocol? Comparing Standardized Approaches over a Common M2M Application. In *GLOBECOM*, 2016.
[9] Hamdi Yahyaoui et al. Trust-based management in IoT federations. *Future Generation Computer Systems*, 2022.
[10] Jeong et al. City Data Hub: Implementation of Standard-Based Smart City Data Platform for Interoperability. *Sensors*.
[11] Kim et al. A Study on Utilization of Spatial Information in Heterogeneous System Based on Apache NiFi. In *ICTC*, 2019.
[12] Kotis et al. Semantic Interoperability on the Web of Things: The Semantic Smart Gateway Framework. In *CISIS*, 2012.
[13] Kurdi et al. TrustyFeer: A Subjective Logic Trust Model for Smart City Peer-to-Peer Federated Clouds. *Wireless Communications and Mobile Computing*, 2018.
[14] Lagutin et al. Secure Open Federation of IoT Platforms Through Interledger Technologies - The SOFIE Approach. In *EuCNC*, 2019.
[15] Mamat et al. Network Topology Comparison for Internet Communication and IoT Connectivity. In *ICOS*, 2019.
[16] Morelli et al. A Federated Platform to Support IoT Discovery in Smart Cities and HADR Scenarios. 2020.
[17] Otto et al. A Federated Infrastructure for European Data Spaces. *Commun. ACM*, 2022.
[18] Papadakis et al. ComDeX: A Context-Aware Federated Platform for IoT-Enhanced Communities. DEBS. ACM, 2023.
[19] Pradhan et al. Toward an Architecture and Data Model to Enable Interoperability between Federated Mission Networks and IoT-Enabled Smart City Environments. *IEEE Communications Magazine*, 2018.
[20] Saad Liaquat Kiani et al. Federated broker system for pervasive context provisioning. *Journal of Systems and Software*, 2013. SI : Software Engineering in Brazil: Retrospective and Prospective Views.
[21] Zach Shelby et al. The Constrained Application Protocol (CoAP). RFC 7252.
[22] Zyrianoff et al. Interoperability in Open IoT Platforms: WoT-FIWARE Comparison and Integration. In *SMARTCOMP*, 2021.
[23] O.S. Pianykh. *Digital imaging and communications in medicine (DICOM): A practical introduction and survival guide*. 2008.
[24] Marko A. Rodriguez and Peter Neubauer. The Graph Traversal Pattern. *CoRR*, 2010.