Secure and Efficient Data Spaces over Named Data Networking

Yannis Thomas,* Nikos Fotiou,*† Iakovos Pittaras,* George Xylomenos*
* Mobile Multimedia Laboratory, Department of Informatics,
School of Information Sciences and Technology, Athens University of Economics and Business, Greece
† ExcID, Athens, Greece

***Abstract*—We propose the Secure and Efficient Data Spaces (SeEDS) architecture, which provides a content brokering service for decentralized data sharing. Our brokering service is built on top of Named-Data Networking (NDN), leveraging NDN's native request aggregation and caching. We support legacy HTTP-based content providers and consumers by building a SeEDS proxy that implements ETSI's NGSI-LD data spaces API. Our proxy receives API requests over HTTP(S) and translates them into the appropriate NDN messages. In order to fully support NGSI-LD operations, we design protocols that extend the core publish/subscribe communication pattern of NDN with advanced query processing. We present a prototype implementation of the SeEDS architecture and a preliminary evaluation and validation of its features. Our solution achieves significant performance gains, adding at the same time support for decentralization.**

***Index Terms*—NDN, NGSI-LD, Data Spaces.**

I. Introduction

It is often said that "data is the new oil" as it can enable new forms of innovation and shape digital transformation, as long as it is properly processed. Just like oil, data are not useful in their raw state; tools are required in order for value to be extracted. To achieve this goal, data spaces are emerging, a new form of digital platform that enables semantic interoperability of data, uniform data access methods, as well as increased data sovereignty and trust. Data spaces have already been investigated in various contexts, including data marketplaces [1], smart farming [2], personal data sharing [3], B2B services [4], Industrial IoT [5], space applications [6] and many others.

An integral part of a data space is the *data broker* (also known as the data intermediary), an entity which is responsible for facilitating data transfer from content providers to content consumers. In this paper, we explore the performance benefits of a Next Generation Internet architecture, when used as an underlay for implementing such a data brokering service. In particular, we leverage *Named Data Networking* (NDN), a content-centric architecture that supports a number of advanced data transport mechanisms [7].

NDN offers a content-centric alternative to IP that supports ubiquitous caching, native multicast transport and multisource content provision. For this reason it has been widely studied in content distribution systems (e.g., [8]–[11]). Nevertheless, although its publish/subscribe based API can be used for simple data transfers, it is not suitable for processing data and extracting "meaningful" information. Our *Secure and Efficient Data Spaces* (SeEDS) architecture adds support for the NGSI-LD Data Spaces API [12] to an NDN-based data brokering service. This HTTP-based API allows end-users to make sophisticated queries, instead of basic data pulling; these queries may involve data processing, such as aggregating data from multiple sources and filtering them based on their attributes, without the need for another layer of processing on top.

To ease the transition to SeEDS, the NDN-enabled network is combined with IP-enabled end hosts. The interfacing of the two architectures takes place at the edge of the NDN network through specialized SeEDS proxies that receive and translate NGSI-LD API requests to NDN operations, and vice versa. The goal of this paper is to describe the SeEDS architecture and explain how the NGSI-LD API is mapped to NDN operations, to allow IP-enabled content providers and consumers to interoperate with the NDN network. We also present our prototype implementation of SeEDS, along with initial performance results.

Our work contributes to both data spaces and the NDN architecture. Regarding data spaces, our solution introduces a decentralized and distributed brokering service, which marks a significant advancement over the centralized approach adopted by existing data space systems. This decentralization enhances the sovereignty of data providers by allowing them greater control over their data, while reducing reliance on a single central entity. Furthermore, our approach improves resilience by eliminating single points of failure, thereby increasing the robustness of the overall system. Additionally, we harness NDN's inherent caching and multicast capabilities to optimize performance, reducing redundant data transmissions and improving network efficiency. From

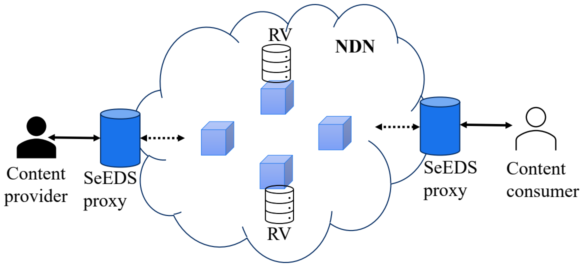

Fig. 1. The SeEDS Architecture.

an NDN perspective, we extend its API to support richer and more expressive data queries. While this extension is driven by the requirements of our data space use case, it is designed as a generic enhancement that can benefit a broader range of applications beyond data spaces.

The remainder of this paper is structured as follows. Section II introduces the design of SeEDS, while Section III explains the mapping between the NGSI-LD API and NDN. Section IV describes our prototype implementation and its evaluation, while Section V discusses related work. Finally, Section VI presents our conclusions and outlines future work.

II. SeEDS Design

An overview of the SeEDS architecture is presented in Fig. 1, where IP-based end-users, i.e., *content consumers* and *content providers*, exchange content via an NDN-based network. The interfacing of the two networks takes place over the SeEDS *proxies*, which are located at the edge of the NDN network. These proxies receive, translate, and forward requests from the IP end-users, in the form of NGSI-LD API calls encapsulated in HTTP(s) messages, to the NDN network and vice versa. Additionally, we introduce in-network *Rendezvous* (RV) nodes that are used for implementing queries over collections of content items.

A. Named-Data Networking architecture

In the NDN architecture, everything revolves around data items. Content *Subscribers*[1] issue Interest messages to request content items, which content *Publishers* return using Data messages; all messages carry the name of a content item. NDN names are hierarchical, but not necessarily human-readable. Content availability is advertised to the network via Announcement messages,[2] which contain the *prefixes* of the names of the content items that a content provider serves.

SeEDS leverages several key properties of NDN:

- **Longest-prefix match:** In NDN, content is retrieved based on hierarchical, structured names. Suppose a publisher P creates an Announcement for a content name $/A$. When a subscriber issues an Interest for content name $/A/B$ NDN's forwarding plane will forward the Interest toward P based on the longest-prefix match routing rule.
- **Request aggregation:** NDN inherently supports Interest aggregation to reduce redundant traffic. For instance, if two subscribers (almost) simultaneously send an Interest for the same content name, these Interests will be forwarded through the network toward the publisher. If their paths converge at an intermediate NDN node R, R will aggregate these Interests, forwarding only a single Interest upstream toward the publisher. When the corresponding Data packet returns from the publisher, R will duplicate and forward it downstream to each requesting subscriber.
- **In-network caching:** Every NDN node acts as a cache for the Data packets it forwards. Because each data item is uniquely identified by its name, matching an incoming Interest with a cached Data packet becomes a simple name-based lookup.

B. The NGSI-LD API

The NGSI-LD API is a standardized interface designed for managing and exchanging context information within data spaces, as specified by the *European Telecommunications Standards Institute* (ETSI). It enables seamless interaction between content providers, content consumers, and data brokering services by providing a common HTTP-based framework for creating, updating, querying content items, as well as subscribing to content item related events. In the NGSI-LD API, each content item is encoded in a structured file format, typically, JSON-LD, which consists of a unique id field, a type field and multiple optional attribute fields. An example of a JSON-LD encoded content item is illustrated in Fig. 2.

The NGSI-LD API offers two content query options: by id and by type. Using the former approach, an HTTP GET request explicitly specifies the id of a *unique* content item, while in the latter approach, an HTTP GET request indicates a type. Consequently, the response to a successful request by id is a single content item, whereas the response to a successful request by type is a set of content items. In addition, two types of filters can be specified: the *selective* filter, which indicates which attributes of the content item(s) should be returned, and the *conditional* filter, which indicates a set of conditions over the attributes that should be met by the returned content item(s). Multiple filters (of both types) can be combined in a single request, thus enabling quite complex queries.

[1] Since NDN entities share the same names as data space entities, in this paper we change NDN's entity names in order to avoid confusion.

[2] Technically, these messages are part of a routing protocol, like the Named Data Link State Routing (NLSR) protocol.

```
{
    "id": "urn:seeds:Car:001",
    "type": "Car",
    "brand": {
        "type": "Property",
        "value": "BWM"
    },
    "dateVehicleFirstRegistered": {
        "type": "Property",
        "value": "2012"
    },
    "emissionsCO2": {
        "type": "Property",
        "value": "22"
    },
    "@context": [
        "https://example.com/context.jsonld"
    ]
}
```

Fig. 2. A JSON-LD encoded content item.

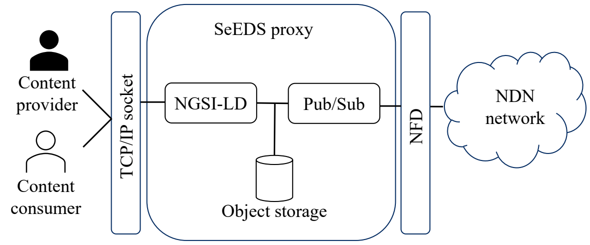

Fig. 3. The SeEDS Proxy.

C. The SeEDS Proxy

The architecture of the SeEDS proxy, the cornerstone of SeEDS, is depicted in Fig. 3. The service provides both an HTTP interface, over which NGSI-LD API messages are exchanged with IP-based nodes, and a `netlink` socket to the *NDN Forwarding Daemon* (NFD) [13], over which NDN specific messages are exchanged with NDN nodes. The SeEDS proxy consists of the following modules:

- The NGSI-LD module receives NGSI-LD API HTTP(S) calls from IP-based content providers or consumers.
- The Pub/Sub module receives and sends ANNOUNCEMENT, INTEREST, and DATA messages from and to the NDN network.
- The Object storage module is responsible for storing content items serialized using JSON-LD.

From a high level perspective these modules interact as follows:

- When a content provider invokes the NGSI-LD API to create a new content item, the NGSI-LD module stores this item to the Object storage module and through the Pub/Sub module *advertises* it to the NDN network.
- When a content consumer invokes the NGSI-LD API to query for a new content item, the NGSI-LD module sends a corresponding INTEREST message to the NDN network through the Pub/Sub module. Note that if the requested item is already stored in the Object storage module the Pub/Sub module will immediately return it.
- When the Pub/Sub module receives an INTEREST message from the NDN network it retrieves the content from the Object storage module and responds with the corresponding DATA message.

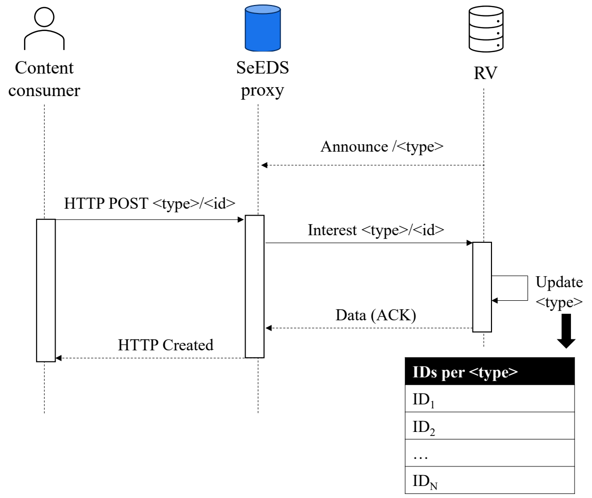

Fig. 4. Registering a new content ID to a TYPE RV.

III. SEEDS OPERATIONS

The NDN architecture can straightforwardly support the NGSI-LD API queries based on ID, as the (unique) ID of a content item can be used as the content name included in the corresponding NDN operations. On the other hand, it does not directly support the NGSI-LD API queries based on TYPE, which requires an INTEREST for the same content name (i.e., the TYPE) to stimulate multiple DATA responses, each of which includes a different content item. Therefore, in order to support TYPE-based queries an NDN node should be able to resolve a TYPE to the corresponding IDs.

In order to achieve this functionality, SeEDS introduces a novel approach that requires no changes to the NDN architecture. Specifically, it introduces in-network *Rendezvous* (RV) nodes each of which is responsible for managing a different TYPE value. An RV node *announces* a content name prefix equal to the corresponding TYPE, allowing SeEDS proxies to *register* and *resolve* IDs per TYPE. This process is described in more detail in the following section.

A. ID registration to an RV

The process of registering a new ID for a TYPE in a SeEDS RV is illustrated in Fig. 4. This process

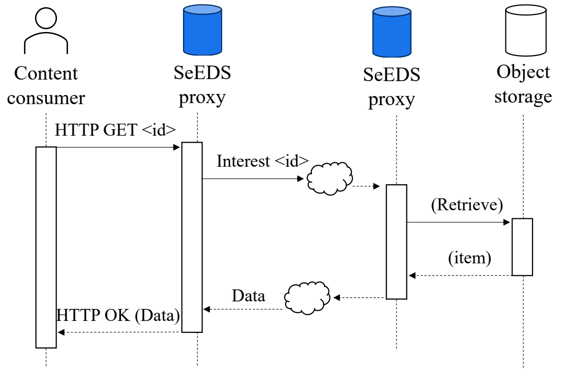

Fig. 5. Requesting content by ID
.

assumes that each RV node has advertised a content name that corresponds to the TYPE. When a content provider invokes the NGSI-LD API to create a new content item, the SeEDS proxy sends an INTEREST using as a content name the ID of the new item prefixed by its TYPE (i.e., `<type>/<id>`). NDN will route this INTEREST to the corresponding RV using the longest-prefix match routing approach. The RV will update an internal data structure that includes a list of IDs per TYPE and will send a DATA message ACKnownledging the registration of the ID. Finally, the SeEDS proxy will send the appropriate response to the content provider. In Section III-C we discuss our approach for protecting the RV against fake advertisements.

B. Content Queries

1) Queries based on ID*:* The process of resolving a content query that follows the ID matching approach can be straightforwadly implemented using NDN's native functionality (see also Fig. 5). Particularly, an IP-based content consumer sends an NGSI-LD API request for an ID over HTTP(S), which is received by a SeEDS proxy. The SeEDS proxy translates this request to an INTEREST message that uses ID as a content name. The INTEREST message is delivered by the NDN network to the SeEDS proxy at the content provider's end, which returns the content item in a DATA message. The SeEDS proxy at the content consumer's end translates the response to an NGSI-LD message, encapsulates it into an HTTP(S) response, and returns it to the consumer. The DATA message may be cached by the NDN core network, allowing subsequent requests to be served by the cache.

2) Queries based on TYPE*:* Queries following the TYPE matching approach utilize the RV, as shown in Fig. 6. As in queries based on ID, an IP-based content consumer sends an NGSI-LD API request for a TYPE over HTTP(S), which is received by a SeEDS proxy. The proxy sends an INTEREST message for a content name of the form `<type>/IDs`. This is a special-purpose reserved content name, indicating interest for "all identifiers that belong to a specific type".

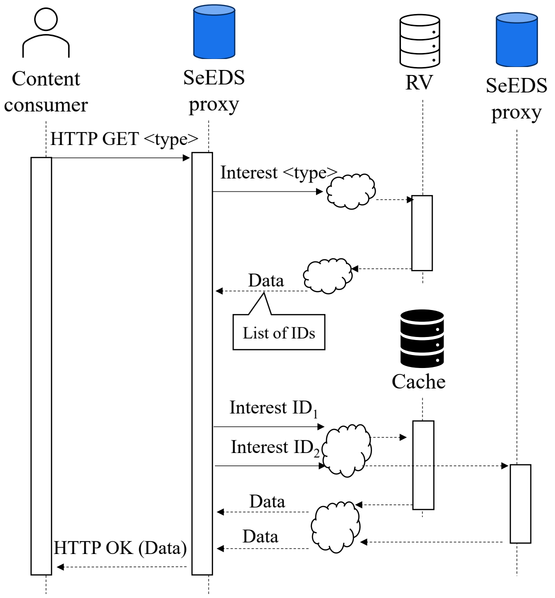

Fig. 6. Requesting content by TYPE.

Using longest-prefix match, the NDN network routes this message towards the RV of `<type>`. The RV responds with a list of IDs corresponding to this type. Then, the SeEDS proxy sends, in parallel, requests for each of these IDs. In contrast to existing data space approaches, each such request may be responded to by a different SeEDS proxy (or cache). In other words, SeEDS allow different content items of the same TYPE to be stored in different network locations.

3) Queries based on attributes: A content consumer may provide a list of attributes in a content item query indicating which attributes of a content item should be returned. Supporting these types of queries may be implemented using two approaches: either by mapping the list of attributes into an NDN content name, or by requesting the content items as a whole and performing attribute-based filtering at the consumer's SeEDS proxy.

Using the first approach, an NGSI-LD request that includes attribute filtering can be mapped to an NDN content name of the form `<id>/<attr1>/../<attrN>`. Using longest prefix match, an NDN INTEREST message for this content name will be forwarded to the SeEDS proxy of the corresponding content provider. The latter proxy will send a DATA packet that includes only the corresponding attributes. The SeEDS proxy at the consumer's side will then simply encapsulate the received data in the corresponding NGSI-LD API response.

Using the latter approach, an NGSI-LD request that includes attributes filtering can be mapped to an NDN content name of the form `<id>`. Similarly, an NDN

INTEREST message for this content name will be forwarded to the SeEDS proxy of the corresponding content provider. The latter proxy will send a DATA packet that includes the whole content item. The SeEDS proxy at the consumer's side will then extract the requested attributes and send an appropriate response to the consumer.

Both approaches have trade-offs and selecting an appropriate approach depends on the use case. The former approach may result in caches being filled at a slower pace; also, leveraging caching requires a form of name canonicalization, as from the NDN perspective the name `<id>/<attr1>/<attr2>` and the name `<id>/<attr2>/<attr1>` do not refer to same content item. On the other hand, having a per-attribute cache extirpation time may result in improved cache usage, as there might be attributes of content items that can be cached for longer times (e.g., "the brand of a car"). Finally, the former approach can be beneficial in cases where attribute sizes and request rates are different. Consider for example, a content item for a surveillance camera having an attribute that corresponds to the "digest of latest captured video" and another that corresponds to the "latest captured video". Typically, a content consumer will request the first attribute and if there is a change compared to the last known value, it will request the second.

The latter approach results in caches being filled fast, therefore subsequent requests for other attributes of the same content item can be responded by a cache. Furthermore, it results in simpler proxy implementations. On the other hand, a content item may be cached for a shorter period, e.g., if it includes an attribute that changes often. Additionally, this approach can result in unnecessary network overhead if content items include attributes that are large in size and are not requested often. The existing SeEDS use cases are focused on small-size content items that are updated infrequently. For this reason SeEDS has adopted the latter approach.

C. Security

Our solution incorporates the data integrity protection mechanism proposed in [14]. At a high level, this mechanism allows a content provider to represent a JSON-LD content item as a collection of key-value pairs, referred to as *disclosures*. Each disclosure, corresponding to a distinct attribute of the content item, is individually hashed, and the resulting list of hashes is then digitally signed by the provider. This design enables selective disclosure: any intermediary can "reveal" only a chosen subset of the disclosures, while still providing strong cryptographic assurances that the content item has not been tampered with. Specifically, a data consumer can compute the hashes of the disclosed attributes, verify that these hashes are included in the original signed list, and subsequently validate the signature over the entire list of hashes to ensure content integrity.

We have integrated this selective disclosure and signing approach into the SeEDS proxies, which act as intermediaries in our architecture. The SeEDS proxies located on the provider side are responsible for computing and signing the list of disclosure hashes before distributing the content item. On the consumer side, SeEDS proxies can reveal only the attributes explicitly requested by the data consumer, while still providing verifiable integrity guarantees for the partially disclosed content, using the method described above.

Additionally, to protect RVs from malicious proxies that advertise fake content, we extend the solution proposed in [15], which leverages W3C *Decentralized Identifiers* (DIDs) [16] to enable cryptographic proofs of content ownership. Specifically, our system adopts the *did:self* method [17], which facilitates the binding of a URL-like identifier to an X.509 certificate without relying on any trusted third party. This mechanism allows a content provider to issue an X.509 certificate to a SeEDS proxy, authorizing a public key—controlled by the proxy—to sign content identifiers that are prefixed with a DID under the provider's control.

When advertising content, the SeEDS proxy uses the authorized key to sign a cryptographic token that includes a timestamp and a nonce. This signed blob is attached to INTEREST messages targeting content names of the form `<type>/<id>`, which are routed by the NDN infrastructure to the corresponding RV. Upon receiving such an INTEREST, the RV extracts the signature and the x509 certificate, it validates the signature using the public key included in the certificate, and it validates the certificate using the DID of the content provider. This process ensures that only authorized proxies can advertise content on behalf of the provider, mitigating the risk of content forgery or hijacking.

IV. IMPLEMENTATION AND EVALUATION

A. Implementation

A SeEDS prototype has been developed that interacts with the open-source NDN implementation [13]. Our SeEDS proxy implements the NGSI-LD API and intercepts the corresponding HTTP requests. Then, it uses its Pub/Sub modules to perform the appropriate operations in the NDN network. Similarly, SeEDS RVs are implemented as NDN applications. Our proof-of-concept has been validated with Mini-NDN,[3] an NDN implementation for the Mininet emulation platform, as well as with the NDN testbed, a global NDN experimentation platform. As an object storage module we currently rely on an SQL database, but ongoing efforts

[3] https://minindn.memphis.edu/

are investigating the integration of the FIWARE Orion broker to the SeEDS prototype.[4]

B. Performance evaluation

In order to evaluate our solution we emulate a topology with a single SeEDS proxy connected to an NDN network. We set the time required for the SeEDS proxy to retrieve a content item from the NDN network equal to 300 ms. A content consumer requests an object from a pool of O objects ($O = 100$ in our experiments) using a Zipf distribution for their popularity with $\alpha = 1.2$, every s seconds, where s is set randomly between 1 and 1.5. Using locust[5], we launch $0.1*O$,..., $0.4*O$ simultaneous content consumers which make requests for 2 minutes. We measure the average, min, and max response time of all requests, as well as the average response time of the requests for the 5 most popular items of the workload. Fig. 7 shows these times as a function of the number of simultaneous content consumers. It can be observed that even though these experiments are without caching, the average response time is below 300 ms (which is the configured network delay). Furthermore, as the number of simultaneous content consumers increases the average response time decreases. This happens because NDN aggregates INTEREST messages for the same content name. Therefore, supposing that a user $U1$ makes a request for content name $C1$ and another user $U2$ makes the same request within 300 ms, the INTEREST message of $U2$ will be suppressed and both users will receive a response based on a single DATA message.

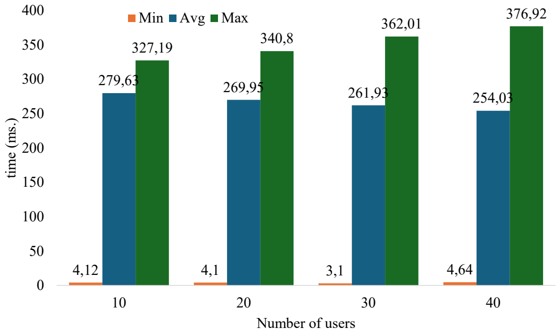

Fig. 7. Min, Average, and Max response time as a function of the number of simultaneous content consumers.

Fig. 8 shows the average response time for the top 5 ranked content items in our dataset. It can be observed that the response time for the top ranked item is ≈ 200 ms when there are $0.4*O$ simultaneous content consumers. This is a 20% improvement compared to the global average and 33% compared to the fixed network delay, again without adding support for caching.

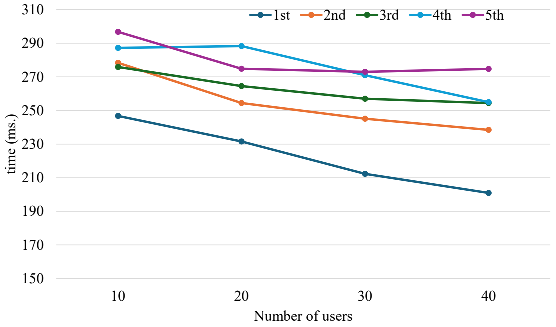

Fig. 8. Average response time for the top 5 ranked content items as a function of the number of simultaneous content consumers.

We then repeat the same experiments using a cache of size $0.05*O$ that implements the *Least Frequently Used* (LFU) cache eviction policy. Figures 9 and 10 illustrate the corresponding results. As it can be observed, not only the total average response time has been reduced, but when it comes to the top 4 ranked content items, when the number of users is $\geq 0.2*O$, almost no request is sent to the NDN network.

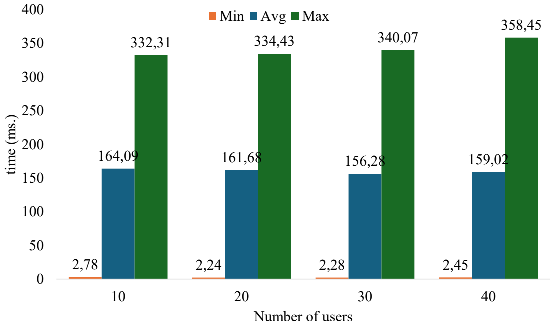

Fig. 9. Min, Average, and Max response time as a function of the number of simultaneous content consumers with 5% caching

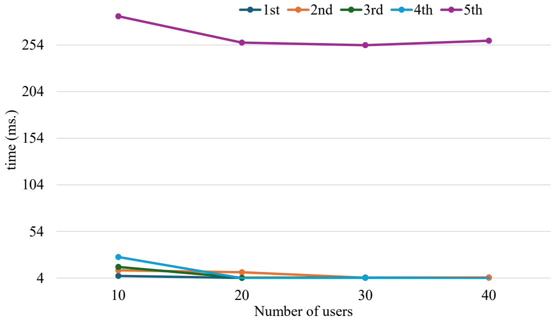

Fig. 10. Average response time with 5% caching for the top 5 ranked content items as a function of the number of simultaneous content consumers.

[4] https://fiware-orion.readthedocs.io/en/master/

[5] https://locust.io

C. *Security properties*

While the core security properties of the data integrity and the DID-based security mechanisms that we employ are thoroughly discussed in the original publications, our solution introduces additional security benefits stemming from its decentralized design. Traditional data space implementations often rely on a centralized and trusted brokering service, which introduces a single point of failure and a concentration of trust. In contrast, our architecture enables each data provider and consumer to operate through a SeEDS proxy that they explicitly trust—either because they host the proxy themselves or because they have established trust relationships or contractual agreements with a trusted proxy provider. This decentralized trust model significantly enhances the sovereignty of data providers, granting them full control over their data while reducing their dependency on external, centralized intermediaries. Moreover, this approach inherently improves system resilience, as the failure or compromise of a single proxy does not disrupt the operation of the entire data space.

In addition, we must point out that our RV approach is also decentralized, as each content type is associated with a distinct RV. This reduces bottlenecks and further mitigates the risks associated with centralization. Ongoing research aims to advance this design by introducing multiple RVs per content type, which will provide even greater fault tolerance and scalability.

V. Related Work

Several studies have investigated the applicability of NDN to content distribution. In [8], the authors discuss how NDN be used to implement a *Content Distribution Network* (CDN) by reducing the server workload and enhancing failure resiliency through ubiquitous in-network caching and adaptive forwarding. They also claim that the distributed in-network nature of NDN reduces operating costs in terms of hardware, maintenance, and administration, and they underline the efficiency of NDN's native multicast, multipath and multisource support. The same authors sketch an NDN-based CDN design in [9]: they focus on routing table size and propose a forwarding strategy utilizing content availability information, to enhance resilience and performance. In [10], an NDN-based CDN over IP is explored. This hybrid design delivers native multicast and multisource, also facilitating routing to the "best" content copies and real-time network monitoring, in order to promptly address congestion and link or node failures. The performance of CDNs and NDN is compared in [11], revealing that NDN can offer significant gains in terms of error resilience, by quickly recovering from lost or corrupted packets via in-network packet-level caching. The loose coupling of end-points allows connectionless transfers, thus accelerating the delivery of small content items and supporting seamless handoffs. Finally, the "best" content copy can be located by the intelligent forwarding strategy of NDN, without a centralized monitoring authority.

Although these efforts recognize and demonstrate the efficiency of the NDN architecture for content distribution and caching, they typically provide only rudimentary data access APIs that cater to basic content retrieval. While this level of functionality aligns with the core requirements of traditional CDNs, it falls short in supporting more sophisticated data interaction models. In contrast, our approach extends beyond simple content dissemination by enabling advanced operations on stored content items. Specifically, we integrate support for the ETSI NGSI-LD API, which offers a rich interface for both coarse-grained access to collections of content entities and fine-grained access to individual attributes within those entities. This API allows clients to perform context-aware queries, subscribe to updates, and interact with content in a semantically meaningful manner—essential capabilities for many emerging applications.

Furthermore, a growing body of research explores the use of NDN for network-based storage solutions. PythonRepo [18] implements in-network persistent storage for content items and provides a basic data access API, enabling the retrieval of stored objects directly from the network. Hydra [19] develops a distributed data storage platform tailored to scientific research, leveraging NDN's naming and caching mechanisms to manage and distribute experimental data efficiently. Similarly, mGuard [20] utilizes NDN to securely store and access m-health data in a decentralized manner. These efforts are more aligned with our objectives, as they explore the use of NDN beyond mere content distribution.

However, our approach differs in several key aspects. First, it abstracts the underlying NDN APIs from end users, providing a higher-level and more developer-friendly interface. Second, it supports the storage of content items outside the NDN network, which enhances interoperability with existing data space technologies and protocols. This decoupling of content storage from the NDN infrastructure allows seamless integration with legacy systems and broader adoption across heterogeneous environments. At the same time, our architecture remains compatible with future extensions that could incorporate in-network NDN-based storage. Such an integration would allow the deployment of an alternative, native "data storage component". We consider this a promising direction for future work.

Since the introduction of *Next Generation Service Interfaces* (NGSI) [6] and its recent evolution into *Next Generation Service Interfaces-Linked Data* (NGSI-LD) [21], the information model for publishing, discovering, monitoring, and maintaining context data for smart applica-

[6] http://www.openmobilealliance.org/release/NGSI/

tions has drawn the attention of the research community, especially in the field of IoT. In [22], the authors rely on NGSI-LD to enrich the data models offered to IoT applications for transportation systems. In [23], the authors integrate NGSI-LD with IoT-based unmanned vehicular networks, so as to accommodate context data transfer between data producers and consumers; the query, update, subscribe and notify context operations are exploited to transfer context values. In [24], NGSI is the cornerstone for designing fog-based IoT services for smart cities, leading to a definition of service programming models, data models and interfaces. In [25], the authors introduce an architecture for IoT-enhanced communities where inter-community communication is built over NGSI-LD. All these efforts are complementary to our approach. We envision that our solution can be used to enhance the security and performance of these efforts, with minimal integration effort.

Our solution is designed to operate entirely within the boundaries of the existing NDN architecture, without requiring modifications to its core protocols. This architectural alignment ensures compatibility with standard NDN deployments while maintaining flexibility and modularity. Nevertheless, recent advances in NDN extensions introduce new capabilities that could be leveraged to further enhance the scalability, robustness, and feature set of our system. As our architecture evolves toward a decentralized Rendezvous (RV) model and incorporates support for versioning of content items, the need for distributed consistency becomes increasingly important. In this context, emerging state synchronization frameworks (e.g., [18], [26]) provide mechanisms for efficiently propagating and reconciling state across distributed nodes. By integrating such synchronization protocols, our system can maintain a coherent and up-to-date view of published content items across multiple RV instances, thereby improving content discovery latency, fault tolerance, and overall system resilience in geographically distributed deployments.

Another critical dimension of our architecture is security. Our system adopts Decentralized Identifiers (DIDs) as a foundational element for establishing trust without relying on centralized trusted entities such as Certificate Authorities. This approach enhances autonomy and resilience but introduces certain trade-offs—most notably, the use of non-human-readable identifiers, since DIDs in our framework are effectively cryptographic digests of public keys. While this design decision aligns with decentralized security principles, it may pose usability challenges in scenarios requiring human-friendly naming. Alternative approaches such as the system proposed in [27], which implements a decentralized trust management framework based on verifiable certificates, offer a compelling trade-off. These approaches retain a degree of decentralization while enabling more readable and structured trust relationships. Such mechanisms could serve as complementary or alternative components in future iterations of our system, depending on deployment requirements and usability considerations.

VI. Conclusions and Future Work

In this paper, we presented a secure and efficient data space solution built upon the Named Data Networking (NDN) architecture. Our system implements the ETSI NGSI-LD API, ensuring full compatibility with existing end-user applications while providing the benefits of an information-centric networking model. By utilizing NDN's decentralized content storage capabilities, our solution eliminates the need for centralized brokers, empowering data providers with greater sovereignty and enhancing the resilience of the system. Additionally, by taking advantage of NDN's native support for caching and request aggregation, our approach delivers notable performance improvements, reducing both network load and response times.

Looking ahead, future work will focus on extending our solution to fully support the subscription mechanisms defined by the NGSI-LD API. This feature allows data consumers to subscribe to specific events and receive real-time notifications when relevant data is updated. By leveraging NDN's inherent multicast capabilities, we expect to implement this functionality with minimal overhead, providing a scalable and efficient notification system across distributed environments. Additionally, we are exploring enhancements to our RV approach through the use of multiple RVs per data type, which could further optimize the dissemination and discovery of context information. Ongoing research also involves adapting our solution to a variety of application domains and the assessment of the performance of different caching strategies, as well as strategies for implementing attribute-based queries

Acknowledgement

The work reported in this paper has been partly funded by the EU's Horizon 2020 Programme through the subgrant Secure and Efficient Data Spaces (SeEDS) (NGISARGASSO-2024-CALL4-2-SeEDS) of project NGI SARGASSO (grant agreement No 101092887).

References

[1] A. Ionescu, K. Patroumpas, K. Psarakis, G. Chatzigeorgakidis, D. Collarana, K. Barenscher, D. Skoutas, A. Katsifodimos, and S. Athanasiou, "Topio: An open-source web platform for trading geospatial data," in *Proceedings of the International Conference on Web Engineering*. Springer, 2023, pp. 336–351.

[2] I. Koren, S. Braun, M. Van Dyck, and M. Jarke, "Dynamic strategic modeling for alliance-driven data platforms: The case of smart farming," in *Intelligent Information Systems*, S. Nurcan and A. Korthaus, Eds. Springer International Publishing, 2021, pp. 92–99.

[3] S. Scheider, F. Lauf, F. Möller, and B. Otto, "A reference system architecture with data sovereignty for human-centric data ecosystems," *Business & Information Systems Engineering*, pp. 1–19, 2023.

[4] P. Pinto, C. Sousa, and C. Cardeiro, "Data spaces based approach for B2B data exchange: A footwear industry case," *Procedia Computer Science*, vol. 219, pp. 933–940, 2023.

[5] B. Farahani and A. K. Monsefi, "Smart and collaborative industrial IoT: A federated learning and data space approach," *Digital Communications and Networks*, vol. 9, no. 2, pp. 436–447, 2023.

[6] A. Seidel, K. Wenzel, A. Hänel, U. Teicher, A. Weiß, U. Schäfer, S. Ihlenfeldt, H. Eisenmann, and H. Ernst, "Towards a seamless data cycle for space components: considerations from the growing European future digital ecosystem Gaia-X," *CEAS Space Journal*, pp. 1–15, 2023.

[7] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the ACM CoNEXT*, 2009. [Online]. Available: https://named-data.net/publications/networkingnamedcontent/

[8] C. Ghasemi, H. Yousefi, and B. Zhang, "Far cry: Will CDNs hear NDN's call?" in *Proceedings of the 7th ACM Conference on Information-Centric Networking*, 2020, pp. 89–98.

[9] ——, "ICDN: An NDN-based CDN," in *Proceedings of the 7th ACM Conference on Information-Centric Networking*, 2020, pp. 99–105.

[10] X. Jiang and J. Bi, "NCDN: CDN enhanced with NDN," in *Proceedings of the IEEE INFOCOM Workshops*, 2014, pp. 440–445.

[11] R. K. Thelagathoti, S. Mastorakis, A. Shah, H. Bedi, and S. Shannigrahi, "Named data networking for content delivery network workflows," in *Proceedings of the 9th IEEE International Conference on Cloud Networking (CloudNet)*, 2020, pp. 1–7.

[12] "Context information management (CIM); NGSI-LD API," ETSI Draft., 2023.

[13] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moiseenko, Y. Yu, W. Shang, Y. Huang, J. P. Abraham, S. DiBenedetto *et al.*, "NFD developer's guide," *Dept. Comput. Sci., Univ. California, Los Angeles, Los Angeles, CA, USA, Tech. Rep. NDN-0021*, vol. 29, p. 31, 2014.

[14] N. Fotiou, G. Xylomenos, and Y. Thomas, "Data integrity protection for data spaces," in *Proceedings of the 17th European Workshop on Systems Security*, ser. EuroSec '24, 2024, p. 44–50.

[15] N. Fotiou, Y. Thomas, V. A. Siris, G. Xylomenos, and G. C. Polyzos, "Securing named data networking routing using decentralized identifiers," in *Proceedings of the IEEE International Conference on High Performance Switching and Routing (HPSR)*, 2021, pp. 1–6.

[16] W3C Credentials Community Group. (2020) Decentralized identifiers (dids) v0.13. Https://w3c-ccg.github.io/did-primer/.

[17] N. Fotiou, V. A. Siris, and G. C. Polyzos, "did:self – A registryless DID method," in *Proceedings of the International Workshop on Trends in Digital Identity (TDI)*, 2025.

[18] T. Yu, X. Ma, V. Patil, Y. Kocaogullar, and L. Zhang, "Exploring the design of collaborative applications via the lens of NDN workspace," in *Proceedings of the IEEE International Conference on Metaverse Computing, Networking, and Applications (MetaCom)*. IEEE, 2024, pp. 89–96.

[19] J. Presley, X. Wang, T. Brandel, X. Ai, P. Podder, T. Yu, V. Patil, L. Zhang, A. Afanasyev, F. A. Feltus, and S. Shannigrahi, "Hydra – a federated data repository over ndn," 2022. [Online]. Available: https://arxiv.org/abs/2211.00919

[20] S. Dulal, N. Ali, A. R. Thieme, T. Yu, S. Liu, S. Regmi, L. Zhang, and L. Wang, "Building a secure mhealth data sharing infrastructure over ndn," in *Proceedings of the 9th ACM Conference on Information-Centric Networking*, ser. ICN '22, 2022, p. 114–124. [Online]. Available: https://doi.org/10.1145/3517212.3558091

[21] G. Privat and A. Medvedev, "Guidelines for modelling with NGSI-LD," ETSI White Paper. https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp_42_NGSI_LD.pdf, 2021.

[22] G. Bouloukakis, C. Zeginis, N. Papadakis, P. Zervakis, D. Plexousakis, and K. Magoutis, "Enabling IoT-enhanced transportation systems using the NGSI protocol," in *Proceedings of the 12th International Conference on the Internet of Things*, 2022, pp. 33–40.

[23] O. Vermesan, R. Bahr, M. Falcitelli, D. Brevi, I. Bosi, A. Dekusar, A. Velizhev, M. B. Alaya, C. Firmani, J.-F. Simeon *et al.*, "IoT technologies for connected and automated driving applications," in *Internet of Things–The Call of the Edge*. River Publishers, 2022, pp. 255–306.

[24] 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, 2017.

[25] 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 (DEBS)*, 2023.

[26] V. Patil, S. Song, G. Xiao, and L. Zhang, "Scaling state vector sync," in *Proceedings of the 9th ACM Conference on Information-Centric Networking*, ser. ICN '22, 2022, p. 168–170. [Online]. Available: https://doi.org/10.1145/3517212.3559485

[27] T. Yu, X. Ma, H. Xie, Y. Kocaoğullar, and L. Zhang, "A new API in support of NDN trust schema," in *Proceedings of the 10th ACM Conference on Information-Centric Networking*, 2023, pp. 46–54.