An Ontology-Based Model for In-Network Computing Components Description and Discovery

Zarin Tasnim * Mouhamad Dieye †, Felipe Estrada-Solano *‡, Roch H. Glitho * §, Halima Elbiaze †, Wessam Ajib †

*CIISE, Concordia University, Montreal, QC, Canada
†Université du Québec À Montréal, Montreal, QC, Canada
‡Fundación Universitaria de Popayán, Popayán, Colombia
§University of Western Cape, Bellville, South Africa

***Abstract*—The increasing demand for ultra-low latency and high bandwidth in emerging applications, such as virtual reality gaming and telesurgery, is challenging current network infrastructures. In-Network Computing (INC) has emerged as a promising solution to these challenges by optimizing network performance, reducing congestion, and minimizing both latency and bandwidth usage. As 6G networks strive to deliver unprecedented connectivity and ultra-low latency, INC is poised to become an integral part of future network architectures. However, a significant gap exists in the literature concerning a comprehensive model for describing INC components, which is crucial for effective INC provisioning. To address this gap, we propose the In-Network Computing Ontology (INCO), a domain-independent, ontology-based model designed to describe and discover INC components in a centralized repository. Our model covers both the functional and non-functional specifications of INC components. Furthermore, we introduce a semantic matchmaking algorithm that uses the INCO model to automatically discover and select the most relevant INC components from the repository based on user requests. Experimental simulations validate our approach, demonstrating the effectiveness of the semantic matchmaking algorithm, particularly regarding response time and consistency.**

***Index Terms*—In-Network Computing, Semantic Matchmaking, In-Network Computing Ontology**

I. Introduction

In-Network Computing (INC) is an emerging paradigm that leverages data plane programmability to enable in-network computations using devices such as switches and smart NICs, which are traditionally used for traffic forwarding [1].

INC is expected to play a critical role in 6G networks, which aim to deliver ultra-low latency, massive connectivity, and unprecedented data rates. Unlike previous generations, 6G is not merely an extension of 5G but a foundational infrastructure for emerging applications, such as holographic communication and remote healthcare [2], [3]. These applications impose stringent Quality of Service (QoS) requirements, including ultra-low latency and high bandwidth, which current network infrastructures struggle to meet—especially when these demands must be addressed simultaneously [4].

INC addresses these challenges by enabling data processing within the network itself, thereby reducing latency, minimizing reliance on distant cloud servers, and optimizing bandwidth. Furthermore, INC's ability to handle complex computational tasks within the network not only enhances the performance of 6G applications but also allows the network to dynamically adapt to varying conditions and demands. By distributing computational tasks across network elements, INC supports the 6G vision of energy-efficient networks, helping to reduce overall energy consumption [5], [6].

Traditionally, Virtual Network Functions (VNFs) have served as the foundational elements of Service Function Chains (SFCs), enabling the dynamic composition of network services by linking together multiple functions, such as firewalls and load balancers. This approach allows network operators to deploy and scale services flexibly. Similarly, INC components act as computational units within the network, executing specific tasks analogous to VNF instances. Proof-of-concept implementations by Aghaaliakbari et al. [7] and Javid et al. [8] have demonstrated the feasibility of integrating INC components within such frameworks.

Consider a distributed concert—a holographic application expected to thrive in the 6G era—where individuals worldwide can fully immerse themselves, experiencing the event as if they were physically present with the artists [7]. Such a concert relies on the seamless integration of various components, including holographic data encoding and real-time transmission, provided by different component providers across a heterogeneous network. To harness the potential of INC and achieve ultra-low latency and high bandwidth, these INC components must be carefully selected, matched, and orchestrated according to their capabilities and performance.

Therefore, network providers must implement efficient discovery and matchmaking mechanisms to identify the most relevant INC components from dedicated marketplaces or centralized repositories based on pre-established service-level agreements. While VNFs benefit from the standardized description and orchestration provided by the ETSI NFV architectural framework [9], INC components currently lack a similarly comprehensive framework. To the best of our knowledge, no existing literature addresses the issue of standardizing INC description and discovery. This gap complicates the automation of discovery and deployment processes for INC components, highlighting the need for a unified model that defines both the functional (i.e., what the component does, such as operations/actions) and non-functional (i.e., how the component performs, such as availability) capabilities of these components. Such a model is crucial for enabling the efficient selection and deployment of INC components across heterogeneous network environments.

Ontologies have been effectively used in NFV to create a structured, semantic framework for describing, discovering, and orchestrating VNFs, as demonstrated by NSChecker [10]. By standardizing both functional and non-functional properties, ontologies enhance automation, interoperability, and service provisioning. This enables automated deployment, dynamic composition, and improved reasoning, thereby reducing complexity and increasing reliability.

This paper proposes an ontology-based model for the semantic description and discovery of INC components. Leveraging ontologies facilitates dynamic and personalized INC provisioning while addressing interoperability challenges across various INC programs, network devices, and service providers. Furthermore, semantic reasoning within ontologies simplifies complex network management tasks, making this approach cost-effective for network providers. It enhances resource utilization, automates decision-making, and supports fast, accurate, and automated service discovery [11]–[13].

This paper makes two key contributions: it introduces a semantic description model for INC component capabilities from both functional and non-functional perspectives and develops a semantic matchmaker to effectively pair requested INC resources with the most relevant ones available. The approach is validated through a prototype that automates and simplifies INC component discovery and instantiation, enabling cooperation among heterogeneous providers.

The paper is structured as follows: Section II provides background information, Section III reviews related literature, Section IV introduces the architectural model, Section V describes the proof-of-concept prototype, Section VI presents the experimental results, and Section VII concludes with suggestions for future work.

II. Background and Related Work

A. In-Network Computing (INC)

The INC concept traces its origins back to the active networks of the 1990s, which first introduced customized computations on messages passing through network devices. Sapio et al. [14] aptly describe INC as a 'dumb idea whose time has come.'

The resurgence of INC is largely driven by Software-Defined Networking (SDN), which enables control plane customization but limits the data plane to protocol-dependent actions and lacks in-field runtime re-programmability. INC extends SDN by enhancing data plane programmability through domain-specific languages like P4 [15]. Although P4 does not support complex operations, it enables vendor-independent programming and protocol-neutral reconfiguration of packet processing within switches, aligning with its design goals.

INC offers substantial potential due to its numerous benefits, including high throughput, low latency, bandwidth reduction, load balancing, and energy efficiency [1], making it particularly well-suited for various applications envisioned for upcoming 6G networks.

Among the few proof-of-concept works, Sapio et al. [14] proposed an INC-based framework and implemented a prototype using the BMv2 software switch. Their results demonstrate significant data reduction (up to 89.3%) and reduced latency related to worker computation. Aghaaliakbari et al. [7] explored INC's potential for holographic applications, specifically remote holographic concerts. Their study proposed an architecture and prototype using the BMv2 switch, assuming the application operates within a cloud-edge continuum with a transcoder function. Experiments revealed that deploying the transcoder on a network device reduces network load by up to 50% compared to edge server execution. The INC scenario also showed notable latency gains over the NFV scenario, though these gains were less pronounced than the reductions in network load.

The most closely related work is by Javid et al. [8], who, to our knowledge, were the first to address the management and orchestration of INC components. They proposed extending the 5G NFV MANO architecture to manage a hybrid NFV/INC infrastructure for holographic applications. However, their work does not address the issue of INC component description and discovery.

B. Ontology-based description and discovery of network functions

Ontology-based semantic descriptions provide a structured framework that defines entities and their relationships, enabling data sharing and reuse across platforms. This facilitates automatic and effective service discovery [16].

While various ontology-based methods have been used for describing and discovering VNFs, none have specifically targeted INC components. To address this gap, we reviewed existing VNF description and discovery methods to adapt relevant insights to our INC model.

For example, Hoyos et al. [17] addressed the challenges of standardization and common understanding among stakeholders, which lead to portability and interoperability issues. They proposed an NFV Ontology (NOn) that enables Semantic NFV Services (SnS) to reduce manual intervention in integrating heterogeneous NFV domains. NOn standardizes NFV component descriptions, facilitating seamless integration across platforms. A proof of concept using a Generic Client within an OpenStack and OpenBaton-based testbed demonstrated how this approach enhances automation and reduces cross-domain integration complexity, thereby mitigating costly rework in current NFV implementations.

Similarly, Anser et al. [18] introduced TRAILS, an extension to TOSCA NFV profiles that addresses the convergence of IoT, NFV, 5G, and Fog and Edge computing. This complex Cloud-to-IoT continuum presents challenges in assigning responsibility, accountability, and liability. Traditional TOSCA NFV profiles focus on deployment but overlook these critical issues. TRAILS integrates responsibility and accountability descriptors into a unified profile, enabling consistent, liability-aware service management across IoT devices, fog, edge, and cloud nodes.

Additionally, Bonfim et al. [10] developed NSChecker, a semantic verification system designed to detect and diagnose policy conflicts in NFV environments. Conflicting policies often arise when different restrictions are applied to shared resources. NSChecker uses the Onto-NFV ontology to describe NFV infrastructure, network services, and associated policies, employing description logic (DL) for comprehensive conflict detection. A Java-based prototype demonstrated NSChecker's effectiveness in identifying conflicts related to network function precedence, resource usage, and location in scenarios involving up to 50,000 nodes.

However, these works do not address non-functional specifications.

Few studies focus on both functional and non-functional specifications. Notably, Nouar et al. [19] observed that existing discovery approaches are provider-specific, using unique methods for parsing VNF descriptors and relying on manual selection. Current VNF description models often lack comprehensive coverage of both functional and non-functional specifications. To address these gaps, Nouar et al. introduced the VIKING ontology, which provides detailed VNF descriptions and incorporates a semantic matchmaking algorithm for more efficient discovery and selection, representing a more advanced approach than previous methods.

Building on these studies and addressing the identified gaps, we have developed an ontology-based semantic description model that comprehensively covers both the functional and non-functional aspects of INC components.

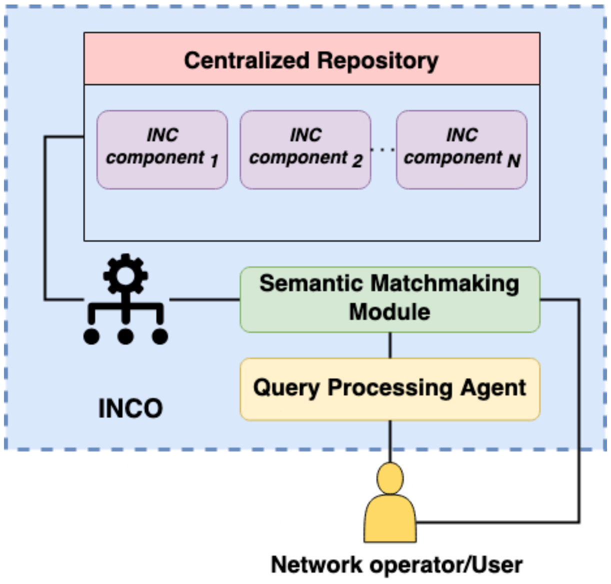

Fig. 1. Architecture overview.

III. The Proposed Architecture

This section outlines our proposed architecture, as depicted in Fig. 1, which enables the ontology-based description of INC components and their discovery using a semantic matchmaking algorithm.

The following subsections provide a detailed explanation of each module in the proposed system.

A. Module description

*1) **Centralized repository**:* The centralized repository functions as a database for storing and accessing INC components, serving as a marketplace where providers publish components and operators retrieve them seamlessly. Each INC component includes a standardized INC Descriptor, outlining specifications and operational details such as the operation name, ID, and deployment information.

*2) **INC Ontology (INCO) Module**:* The INCO module includes our ontology-based description model, detailed below, which semantically represents INC components using INC descriptors. The model defines the domain scope, manages queries, and supports the description, publication, and discovery of INC components.

*3) **Semantic Matchmaking Module**:* The Semantic Matchmaking Module identifies the best-matched INC components using the algorithm outlined in Algorithm 1, returning a list of relevant URIs.

This algorithm leverages the INCO model to align with user preferences by discarding irrelevant components, retrieving relevant ones, and ranking them. Algorithm 1 includes two matching types: `matchAll`, which verifies if an INC component meets all mandatory preferences, and `MatchSome_putPriorityValue`, which ranks the retrieved components based on the user's high and optional preferences.

Algorithm 1 Semantic Matchmaking

```
1: procedure INC_MATCHMAKER($QR_i$, pref_list)
2:     $Retrieved\_inc \leftarrow []$
3:     if MATCHALL($QR_i, INC_i$) then
4:         APPEND($Retrieved\_inc, INC_i$)
5:     end if
6:     for each $INC_i$ in $Retrieved\_inc$ do
7:         MATCHSOME_PUT_PRIORITY_VALUE($INC_i$, pref_list)
8:     end for
9:     RANKBYPRIORITYVALUE($Retrieved\_inc$)
10:    return $Retrieved\_inc$
11: end procedure
```

*4) **Query Processing Agent (QPA)**:* The QPA acts as an intermediary, translating user requests into a system-processable format by converting them into SQWRL (Semantic Query-Enhanced Web Rule Language) queries, generating a preference list (categorizing preferences as mandatory, high, and optional), and forwarding this list to the semantic matchmaking module. A user request includes all relevant functional and

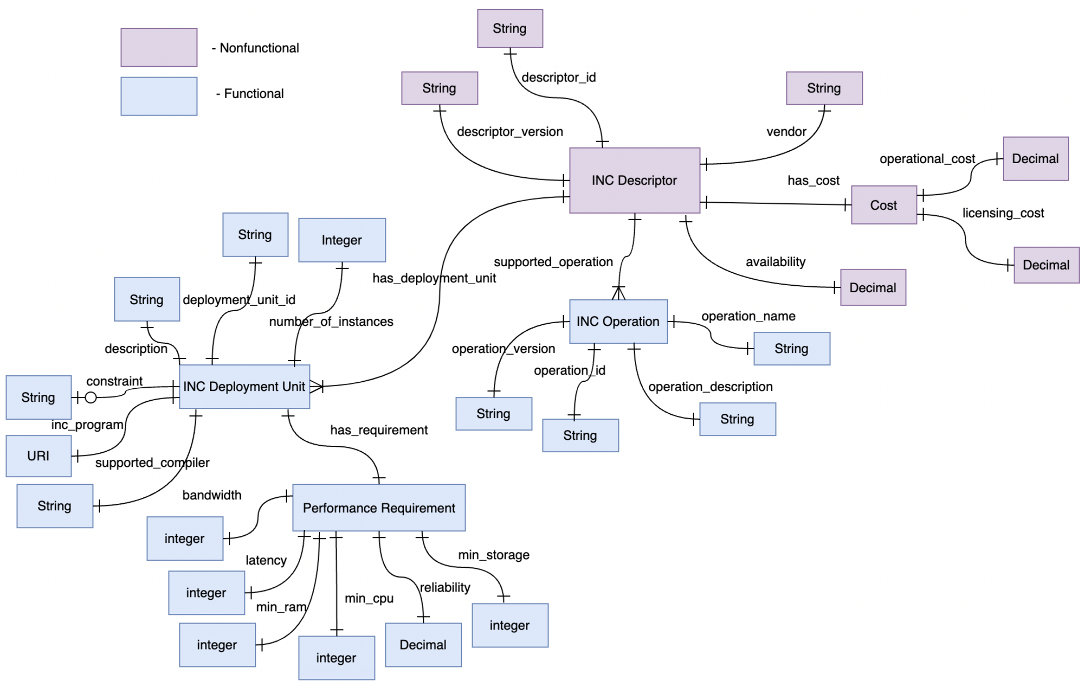

Fig. 2. Overview of the INCO model.

non-functional specifications for the desired INC component, clearly distinguishing between the levels of user preferences: mandatory (non-negotiable), high (important but flexible), and optional (desirable but not essential).

For clarity in the SQWRL formalism, the key symbols are as follows: $\wedge$ (logical AND) connects multiple true conditions, $\rightarrow$ denotes implication, mapping conditions to selected variables, $\vee$ (logical OR) requires at least one condition to be true, and $?$ serves as a placeholder for variables (e.g., $?p, ?f, ?v$) within the query.

The request from user i received by the QPA is formally defined as UR_i such that:

$$UR_i = \{(Pr_{i,j}, Req\_f_{i,j}, Req_{i,j}, Condition_{i,j}, Val_{i,j})\}_{j=1..n} \tag{1}$$

UR_i is a set of tuples, each representing a user specification for either functional or non-functional properties. The variable j indexes each user specification, and n is the total number of specifications in the request. $Pr_{i,j}$ indicates the user preference level (mandatory, high, optional), $Req\_f_{i,j}$ corresponds to an INCO concept (e.g., INC_Operation, Cost), $Req_{i,j}$ maps to an INCO data property (e.g., bandwidth, latency), $Condition$ specifies the operator (e.g., maximum, minimum, exactly), and $Val_{i,j}$ represents the value of the data property (e.g., 'Encoder', 110).

The SQWRL query (QR_i) and a preference list ($pref\_list_i$) are formally represented in Eq. (2) and Eq. (4), respectively.

$$\begin{aligned} QR_i = \text{Rel} \wedge & \\ & \Big[\text{Req\_f}_{i,\{j,\text{Req\_f} = \text{INC\_Operation} \wedge \text{num of INC\_Operation} > 1\}}(?f) \\ & \wedge \text{Req}_{i,j}(?f, ?v) \wedge (\text{Condition, Val})_{i,\{j,\ \text{Pr} = \text{mandatory}\}} \Big] \\ & \rightarrow \text{sqwrl:select}(?inc, ?v_{i,\{j,\ \text{Pr} = \text{high} \vee \text{Pr} = \text{optional}\}}) \end{aligned} \tag{2}$$

Here, Rel defines the structure of the ontology, helping build the query for each user request such that:

$$\begin{aligned} Rel = \ & INC\_Descriptor(?d) \wedge has\_cost(?d, ?c) \wedge Cost(?c) \\ & \wedge supported\_operation(?d, ?o) \wedge INC\_Operation(?o) \\ & \wedge has\_deployment\_unit(?d, ?du) \\ & \wedge INC\_Deployment\_Unit(?du) \wedge has\_requirement(?du, ?p) \\ & \wedge Performance\_Requirement(?p) \wedge inc\_program(?du, ?inc) \end{aligned} \tag{3}$$

$$\begin{aligned} \text{Pref\_list}_i = \{ & (Pr_{i,j}, Req_{i,j}, Condition_{i,j}, \\ & Val_{i,j})\}_{\{j,\ \text{Pr} = \text{high} \vee \text{Pr} = \text{optional}\}} \end{aligned} \tag{4}$$

B. INC Ontology (INCO) model description

INCO, as shown in Fig. 2, is our ontology-based description model for the semantic representation of INC components. It is organized into two segments: functional and non-functional specifications. These characteristics are semantically linked to their corresponding attributes, enabling detailed descriptions, effective query construction, and efficient component retrieval.

Functional specifications define the capabilities and semantic relationships of an INC component and include three key concepts:

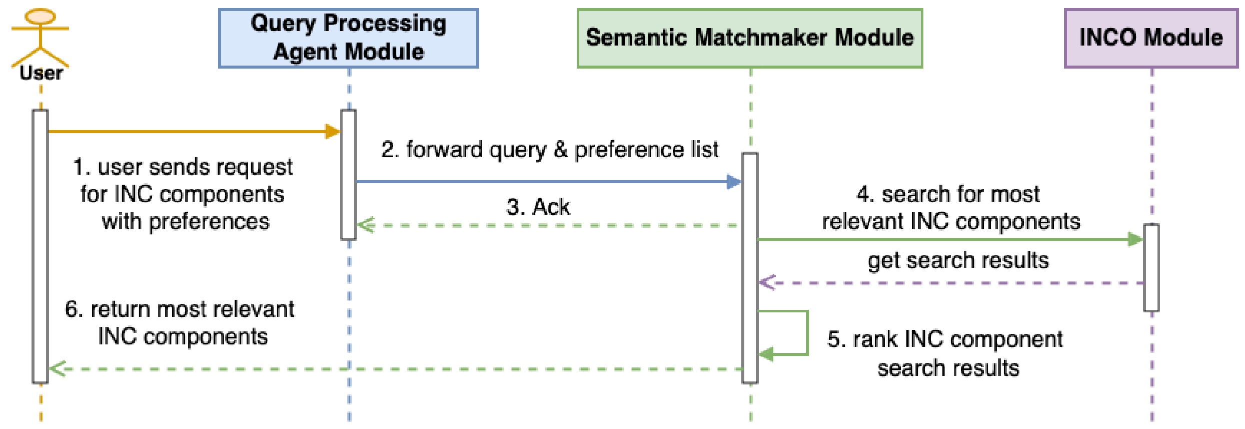
User
Query Processing Agent Module
Semantic Matchmaker Module
INCO Module
1. user sends request for INC components with preferences
2. forward query & preference list
3. Ack
4. search for most relevant INC components
get search results
5. rank INC component search results
6. return most relevant INC components

Fig. 3. Sequence diagram of INC components discovery.

- **INC_Operation**: Describes the supported operations, detailing their purpose and specific functionalities. This includes operation names, descriptions, versions, and associated identifiers.
- **INC_Deployment_Unit**: Provides essential deployment information for INCO components, such as the required compiler (e.g., P4) for executing an INCO program, the program's URI, and other relevant details.
- **Performance_Requirement**: Specifies the conditions necessary for an INCO component to operate efficiently, including computational and storage needs, such as minimum RAM and CPU requirements.

Non-functional specifications focus on the requirements for the proper functioning of INC components and include two main concepts:

- **INC_Descriptor**: Defines the general characteristics of INCO components and serves as the central element of the INCO model. It connects directly or indirectly to every other concept, encompassing details such as component availability, descriptor versions, and vendor information.
- **Cost**: Addresses the financial aspects of an INCO component, including operational and licensing expenses.

C. *INC component discovery flow description*

This section presents the flow of the INC component discovery process within our proposed architecture, as illustrated in Fig. 3.

The proposed system operates in an environment where INC providers design and develop INC components, which are then published to a centralized repository. This repository enables network operators to utilize these components in INC-based SFCs. Upon publication, the INC descriptor—a structured template defining the component's functional and non-functional properties, computational requirements, and deployment parameters—is submitted to the INCO for parsing and semantic enrichment.

In this setup, when a user seeks to compose an INC-based SFC, the system attempts to match their requirements with an INC component from the centralized repository based on specific preferences. The process begins when the user submits a request to the Query Processing Agent (Step 1), which converts the request into a SQWRL query, extracts the preference list, and forwards it to the Semantic Matchmaking Module (Step 2). The Semantic Matchmaking Module then acknowledges receipt of the query (Step 3). Using the ontology model in the INCO Module, the Semantic Matchmaking Module searches for and ranks the INC components in the centralized repository according to the user's request and preferences (Steps 4 and 5). Finally, in Step 6, the Semantic Matchmaking Module returns the Uniform Resource Identifiers (URIs) of the most relevant INC component(s) to the user.

IV. Proof-of-Concept Validation

This section presents the proof-of-concept developed to validate the effectiveness of the proposed semantic description model and matchmaking algorithm. We outline the experiments conducted to evaluate our approach, providing details on the experimental setup, metrics used for comparative analysis, and the performance measurements obtained.

A. *Implementation:*

INCO was implemented using Protégé 5.6.3, with the semantic matchmaking algorithm developed in Python. The implementation was executed on a system running Windows 10 Enterprise, equipped with an Intel Xeon E5645 processor (2.40 GHz, 6 cores), 16 GB RAM, and a 64-bit architecture. The Pellet reasoner ensured logical consistency, validated the ontology, and handled complex SQWRL queries, which were essential for the matchmaking algorithm to retrieve relevant INCO components based on user requirements and preferences.

B. *Performance Metrics*

The experiments utilized sample queries categorized by their complexity and the number of retrieved instances, with response time (in milliseconds) as the primary performance metric. Two parameters were considered: query complexity (determined by the number of properties involved) and the number of retrieved instances (the total count of instances returned by a query).

Response time was measured using two approaches. The first approach varied query complexity while keeping the number of retrieved instances constant; six queries ($Q1$ to $Q6$) were executed, with complexity increasing by adding three properties per query. The second approach maintained constant query complexity while varying the number of retrieved instances by incrementally adding instances to the ontology. These approaches assessed how response time scales with both query complexity and the number of retrieved instances.

For example, $Q1$ represents the least complex query, retrieving all INC components that support encoder operations. This query involves only one property and serves as the baseline for our complexity analysis.

$$\begin{aligned} Q1 = {} & \text{INC\_Descriptor}(?d) \land \text{has\_cost}(?d, ?c) \land \text{Cost}(?c) \\ & \land \text{supported\_operation}(?d, ?o) \land \text{INC\_Operation}(?o) \\ & \land \text{has\_deployment\_unit}(?d, ?du) \land \text{INC\_Deployment\_Unit}(?du) \\ & \land \text{has\_requirement}(?du, ?p) \land \text{Performance\_Requirement}(?p) \\ & \land \text{inc\_program}(?du, ?inc) \\ & \land \text{operation\_name}(?o, \texttt{"Encoder"\^{}rdf:PlainLiteral}) \\ & \rightarrow \text{sqwrl:select}(?inc) \end{aligned}$$

In $Q2$, three additional properties—availability, licensing cost, and operational cost—were appended to $Q1$, with complexity increasing incrementally up to $Q6$.

In contrast, $Q7$ exemplifies the second approach, where the query structure remains unchanged, but the number of retrieved instances varies as new instances are added to the ontology.

$$\begin{aligned} Q7 = {} & \text{INC\_Descriptor}(?d) \land \text{has\_cost}(?d, ?c) \land \text{Cost}(?c) \\ & \land \text{supported\_operation}(?d, ?o) \land \text{INC\_Operation}(?o) \\ & \land \text{has\_deployment\_unit}(?d, ?du) \\ & \land \text{INC\_Deployment\_Unit}(?du) \\ & \land \text{has\_requirement}(?du, ?p) \\ & \land \text{Performance\_Requirement}(?p) \land \\ & \text{inc\_program}(?du, ?inc) \\ & \land \text{operation\_name}(?o, \texttt{"Bal."\^{}rdf:PlainLiteral}) \\ & \rightarrow \text{sqwrl:select}(?inc) \end{aligned}$$

C. Experimental Results and Analysis:

The experimental results in Fig. 4, show a clear increase in response time as query complexity escalates from $Q1$ to $Q6$, with the number of properties rising from 1 to 15. This trend reflects the additional computational overhead needed to evaluate more conditions and properties as the query complexity increases. The broader error bars at higher complexity levels indicate variability in response times, likely due to the increased computational burden of processing more complex logic or larger data sets, which causes certain property combinations to impose varying demands on the system.

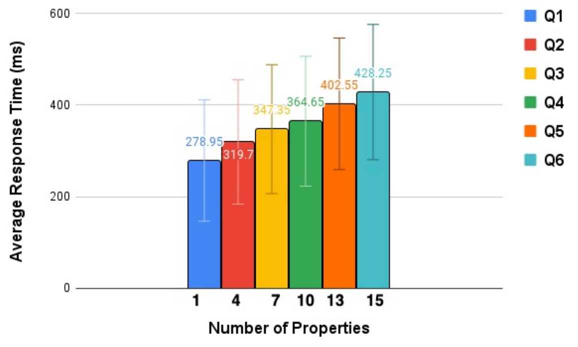

Fig. 4. Average response time by different query comeplexity.

Conversely, Fig. 5 demonstrates the efficiency of our approach in handling varying numbers of retrieved instances. Even as the number of instances increases from 3 to 16, the response time remains stable. This consistency, along with smaller and more uniform error bars compared to the query complexity results, indicates that the system is well-optimized for scalable data retrieval.

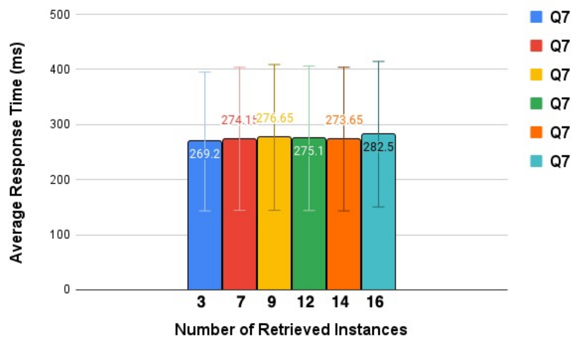

Fig. 5. Average response time by number of retrieved instances.

V. Conclusion and Future Work

This paper presents a novel ontology-based model for the semantic description and discovery of INC components, addressing a critical gap in the current literature. By leveraging ontologies, the proposed model enables automated provisioning of INC components, thereby enhancing interoperability across diverse networks and component providers. Incorporating both functional and non-functional aspects into the semantic descriptions ensures a comprehensive and efficient discovery process, tailored to the specific requirements of network operators. The proof-of-concept implementation validates the effectiveness of this approach, demonstrating its capability to streamline the discovery and deployment of INC components.

While promising, future work should not only focus on scaling the ontology-based framework for the increasing complexity of INC components in large-scale 6G environments but also on exploring specific use cases, such as holographic streaming applications. Additionally, integrating advanced machine learning techniques could enhance the semantic matchmaking process, leading to more precise and context-aware recommendations. Extending the ontology model to support real-time updates and dynamic adaptations will ensure responsiveness to evolving 6G demands. Finally, collaborating with industry stakeholders to develop a standardized INC description framework could facilitate broader adoption and interoperability across various platforms and networks.

Acknowledgements

This work is partly funded NSERC through a discovery grant and by Québec FQRNT in the context of SCORING (an EU CHIST ERA project). Many thanks to all the SCORING project members for the very fruitful discussions on in-network computing.

References

[1] S. Kianpisheh and T. Taleb, "A survey on in-network computing: Programmable data plane and technology specific applications," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 701–761, 2023.

[2] W. Jiang, B. Han, M. A. Habibi, and H. D. Schotten, "The road towards 6g: A comprehensive survey," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 334–366, 2021.

[3] S. Dang, O. Amin, B. Shihada, and M.-S. Alouini, "What should 6g be?" *Nature Electronics*, vol. 3, no. 1, pp. 20–29, 2020.

[4] A. Clemm, M. T. Vega, H. K. Ravuri, T. Wauters, and F. De Turck, "Toward truly immersive holographic-type communication: Challenges and solutions," *IEEE Communications Magazine*, vol. 58, no. 1, pp. 93–99, 2020.

[5] N. Hu, Z. Tian, X. Du, and M. Guizani, "An energy-efficient in-network computing paradigm for 6g," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 4, pp. 1722–1733, 2021.

[6] S. Schwarzmann, R. Trivisonno, S. Lange, T. E. Civelek, D. Corujo, R. Guerzoni, T. Zinner, and T. Mahmoodi, "An intelligent user plane to support in-network computing in 6g networks," in *ICC 2023-IEEE International Conference on Communications*. IEEE, 2023, pp. 1100–1105.

[7] F. Aghaaliakbari, Z. A. Hmitti, M. Rayani, M. Gherari, R. H. Glitho, H. Elbiaze, and W. Ajib, "An architecture for provisioning in-network computing-enabled slices for holographic applications in next-generation networks," *IEEE Communications Magazine*, vol. 61, no. 3, pp. 52–58, 2023.

[8] F. G. Javid, M. Dieye, F. Estrada-Solano, R. H. Glitho, H. Elbiaze, and W. Ajib, "A hybrid nfv/in-network computing mano architecture for provisioning holographic applications in the metaverse," in *2024 IEEE 25th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2024, pp. 99–104.

[9] European Telecommunications Standards Institute (ETSI), "Network functions virtualisation (nfv); architectural framework," European Telecommunications Standards Institute (ETSI), Tech. Rep. ETSI GS NFV 002 V1.2.1, December 2014, accessed: 2024-08-20. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf

[10] M. Bonfim, F. Freitas, and S. Fernandes, "A semantic-based policy analysis solution for the deployment of nfv services," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 1005–1018, 2019.

[11] G. Lu, T. Wang, G. Zhang, and S. Li, "Semantic web services discovery based on domain ontology," in *World Automation Congress 2012*, 2012, pp. 1–4.

[12] M. Uschold, M. Healy, K. Williamson, P. Clark, and S. Woods, "Ontology reuse and application," in *Formal ontology in information systems*, vol. 179. Citeseer, 1998, p. 192.

[13] A. V. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics-based automated service discovery," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 260–275, 2012.

[14] A. Sapio, I. Abdelaziz, A. Aldilaijan, M. Canini, and P. Kalnis, "In-network computing is a dumb idea who's time has come," *Proceedings of Hot-Nets*, 2017.

[15] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends," *IEEE Access*, vol. 9, pp. 87 094–87 155, 2021.

[16] K. Rabahallah, F. Azouaou, and M. T. Laskri, "Ontology-based approach for semantic description and the discovery of e-learning web services," in *2016 International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, 2016, pp. 117–124.

[17] L. C. Hoyos and C. E. Rothenberg, "Non: Network function virtualization ontology towards semantic service implementation," in *2016 8th IEEE Latin-American Conference on Communications (LATINCOM)*. IEEE, 2016, pp. 1–6.

[18] Y. Anser, C. Gaber, J.-P. Wary, S. N. M. García, and S. Bouzefrane, "Trails: Extending tosca nfv profiles for liability management in the cloud-to-iot continuum," in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*. IEEE, 2022, pp. 321–329.

[19] N. el houda Nouar, S. Yangui, N. Faci, K. Drira, and S. Tazi, "A semantic virtualized network functions description and discovery model," *Computer Networks*, vol. 195, p. 108152, 2021.