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

Zarin Tasnim

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science (Computer Science) at

Concordia University

Montréal, Québec, Canada

March 2025

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Zarin Tasnim**

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

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________________ Chair
Dr. Abdelhak Bentaleb

______________________________ Examiner
Dr. Sandra Cespedes

______________________________ Supervisor
Dr. Roch Glitho

Approved by ______________________________
Dr. Joey Paquet, Chair
Department of Computer Science and Software Engineering

____________ 2025

____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

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

Zarin Tasnim

In-Network Computing (INC) refers to the process that enables the distribution of computing tasks across the network instead of computing on servers outside the network. Advances in programmable hardware allow computations directly within network devices, such as switches and Smart Network Interface Cards (smart NICs), as data passes through them, and thereby reduces network congestion, minimizes reliance on distant cloud servers, and improves latency.

The growing demand for ultra-low delays, high bandwidth, and the ability to dynamically synchronize data streams in emerging applications, particularly Holographic-Type Communication application, is pushing the limits of current network infrastructures, with INC emerging as a promising solution as it aims to meet these stringent requirements. In addition, efficient provisioning of INC requires a comprehensive understanding of INC components, including their specifications, configuration information, and requirements. Nevertheless, an architecture with a description and discovery model offers a structured and standardized framework to represent INC components, defining their functionality and characteristics, and retrieving the most pertinent INC components according to the user requirements.

This thesis proposes a novel architecture for describing and discovering the most relevant INC components based on user preferences, specifically tailored for holographic-type application. The contribution of this work is threefold. First, we introduce the In-Network Computing Ontology (INCO), a domain-independent, ontology-based description model that provides a semantic representation of INC components, facilitating their discovery from a centralized repository. The description model covers both the functional and non-functional specifications of INC components and consists of two parts: a generic description for INC components and an extension detailing four specific INC components (i.e. encoder, decoder, transcoder, and renderer)- essential for our holographic application use case. Second, we present a semantic matchmaking algorithm that leverages the INCO model to automatically identify and select the most appropriate INC components based on user requests and preferences. Lastly, we validate the proposed approach through experimental simulations, demonstrating the algorithm's effectiveness in terms of response time and consistency. Response time was measured based on two criteria: query complexity and the number of retrieved instances. The simulation results indicate that response time fluctuates with increasing query complexity, while it remains comparatively stable as the number of retrieved instances grows.

Acknowledgments

First and foremost, I would like to thank Almighty for giving me the strength, wisdom, and perseverance to complete this work.

I would like to express my deepest gratitude to my supervisor, Dr. Roch Glitho, for his invaluable guidance, support, and insightful feedback throughout the development of this thesis. His advice and expertise significantly enriched my research, and I truly appreciate the time and effort he dedicated to ensuring my success. His patience and encouragement were instrumental in helping me navigate the challenges I encountered along the way.

I would also like to extend my heartfelt appreciation to Mouhamad Dieye and Felipe Estrada-Solano for their collaboration, constant encouragement, and unwavering support during this journey. I am deeply grateful for their mentorship and the time they dedicated to my progress.

To my family, words cannot express my gratitude for their unconditional love, patience, and constant belief in me. Their continuous encouragement and unwavering support has been the foundation of my strength and the greatest source of my motivation, and I am incredibly blessed to have them by my side.

Lastly, my special thanks go to my friends and labmates, whose companionship and laughter have made this journey so much more enjoyable. Their presence has been a great source of comfort and joy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 1 begins by defining the key terminologies and concepts relevant to the thesis. It then provides a description of the motivation, problem statement, and the contributions of this work. The final section offers an overview of how the remaining chapters are organized.

1.1 Definitions

In the following sections, definitions of key terms critical to this thesis are presented. These terms include the Ontology, Holographic-Type Communication and In-Network Computing. These terminologies are fundamental to the thesis.

1.1.1 Ontology

Ontology is a formal representation of knowledge that defines a set of concepts and the relationships between them within a specific domain. It allows content and services to be described in a machine-readable format, enabling the automation of tasks such as annotating, discovering, publishing, advertising, and composing services. Ontologies establish a shared understanding of a domain that can be conveyed both among people and across diverse, distributed application systems. [11]. It serves as the backbone for enabling machines to understand and process the meaning of information by providing a shared vocabulary for entities and their interactions. It facilitates knowledge sharing, interoperability, and reasoning by enabling consistent and explicit conceptualization across different systems. Furthermore, it supports the representation of conceptualizations using languages like Resource Description Framework (RDF) and Web Ontology Language (OWL), which formalize knowledge and enable logical inference [1, 11].

1.1.2 Holographic-type Communication

Holographic-type Communication (HTC) application is widely regarded as a new form of augmented reality media that provides internet users with highly immersive experiences. According to [12], HTC is an emerging form of immersive media that integrates Augmented and Virtual Reality (AR/VR) technologies to present fully 3D objects captured by RGB depth (RGB-D) sensor cameras, enabling internet users to experience a level of immersion far beyond that of traditional video-based applications. Unlike standard 3D content, HTC enables the transmission and interaction with holographic data over a network from remote locations. By incorporating parallax, holograms allow viewers to interact with the image, which dynamically adjusts based on the viewing angle. This shift transforms the viewer's role from passive, as with 2D and 3D content, to active and interactive with holograms, further increasing the demands on HTC [13]. HTC applications are considered among the most demanding content types for next-generation communication networks. In the coming years, new holographic applications are expected to provide fully immersive AR/VR experiences and enable near-real-time personal communication with holograms [2] [14].

1.1.3 In-Network Computing

INC is an emerging paradigm designed to enable computation within the network itself by exploiting programmable data plane, providing an alternative to the traditional approach of confining computation to servers outside the network [15, 16, 4]. Similarly, INC promotes the concept of leveraging network elements, such as programmable switches, Field-Programmable Gate Array (FPGAs), and smart NICs, to be programmed for the purpose of computation [17, 18]. According to the reference [15], in-network computing is defined as the process of offloading certain computational tasks from end hosts to network elements. Moreover, the authors in reference [16], describe INC as the execution of application-specific functions on programmable network hardware at line rate, achieving significantly higher throughput and lower latency than traditional servers. Additionally, INC can alleviate the computing load on data centers and offers substantial energy-saving benefits by processing tasks within the network itself [5]. At its core, INC utilizes network devices as computational resources, leveraging redundant computing capacity for data processing [5].

1.2 Motivation and Problem Statement

The rapid advancement of cutting-edge applications, such as HTC, VR gaming, and other immersive experiences [19], requires real-time processing capabilities in order to make it well-suited for immersive experiences, where time constraint is critical [20, 21]. Let's consider a distributed concert—a holographic application—where individuals worldwide can fully immerse themselves, experiencing the event as if they were physically present with the artists [22]. 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, these INC components must be carefully selected, matched, and orchestrated according to their capabilities and performance.

Furthermore, provisioning INC successfully requires detailed knowledge of its components, including their functional and non-functional properties, requirements, dependencies, and deployment specifics. Despite its potential, there is a notable gap: the absence of a comprehensive, well structured and standardized description model for INC components, making it difficult to fully understand and integrate them. The lack of a standardization and clear representation complicates the situation, as no model exists to consistently represent INC components across different environments. This gap not only limits our understanding of INC but also makes it exceedingly difficult to automate the discovery, deployment, and management of these components, resulting in inefficiencies in network operations. A standardized description model would assist network operators in configuring and provisioning INC-based services, ensuring that network resources are used optimally to meet the performance demands of emerging applications.

Likewise, 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 advanced technologies like Virtualized Network Functions (VNFs) benefit from the standardized description and orchestration provided by the ETSI NFV architectural framework [23], INC components currently lack a similarly comprehensive framework. Although VNFDs provide a framework for describing and managing virtual network functions in an NFV-based architecture, they do not adequately address the unique demands of INC-based services. The most closely related work is by Javid et al. [24], 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.

To the best of our knowledge, no existing literature addresses the issue of standardizing INC component description and discovery. This gap complicates the automation of discovery and deployment processes for INC components, highlighting the need for a description 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 essential for enabling the efficient selection and deployment of INC components across heterogeneous network environments. Without an INC-specific description model, network operators face considerable challenges in efficiently deploying and scaling INC resources. Overall, the absence of a specialized descriptor for INC hinders network operators from fully leveraging INC's potential, particularly in highly demanding, low-latency environments.

1.3 Thesis Contributions

The thesis contributions are as follows :

- An architecture for the description and discovery of In-Network Computing components tailored for holographic streaming use cases.
- A unified, ontology-based, domain-independent semantic description model for In-Network Computing components, encompassing both functional and non-functional properties.

- A reusable description model for In-Network Computing components that can be adapted or extended for other applications without needing to be rebuilt from scratch.
- A semantic matchmaking algorithm to effectively pair requested In-Network Computing resources with the most relevant ones available.
- A prototype implementation and performance evaluation.
- A simulation was conducted to demonstrate the efficiency of the semantic matchmaker in discovering In-Network Computing components.

1.4 Thesis Organization

The rest of the thesis is organized as follows :

- Chapter 2 provides an in-depth discussion and background knowledge of the key concepts relevant to this research.
- Chapter 3 reviews related literature, design consideration and challenges, introduces the motivating scenario and emphasizes the need for a comprehensive framework to address the identified challenges.
- Chapter 4 presents the proposed architectural model and provides a detailed description of each module.
- Chapter 5 describes the proof-of-concept prototype, including performance evaluation, and presents the analysis and experimental results.
- Chapter 6 concludes the thesis by summarizing the contributions and identifying potential future research directions.

Chapter 2

Background

This chapter provides the foundational concepts relevant to the research domain of this thesis. It begins with an overview of ontology and its significance in structured knowledge representation and information sharing. Subsequently, it delves into Holographic-Type Communication, discussing how the rising demand for high-performance holographic applications is shaping technological advancements. Finally, the chapter examines the emergence of In-Network Computing, emphasizing its role in meeting the demands of next-generation applications. The sections are structured to introduce and define each concept, culminating in a conclusion summarizing the chapter's key points.

2.1 Overview of Ontology

This section 2.1 provides an overview of ontology, explaining its definition and significance in knowledge-based representation.

2.1.1 Ontology

The modern concept of ontology, defined as "explicit formal specifications of the terms in the domain and relations among them" by Gruber in 1993, has become foundational for structuring and sharing knowledge across various domains, particularly in Artificial Intelligence and the World Wide Web [25]. Ontologies provide a shared vocabulary for researchers to exchange and interpret information within a specific domain. They include machine-readable definitions of core concepts

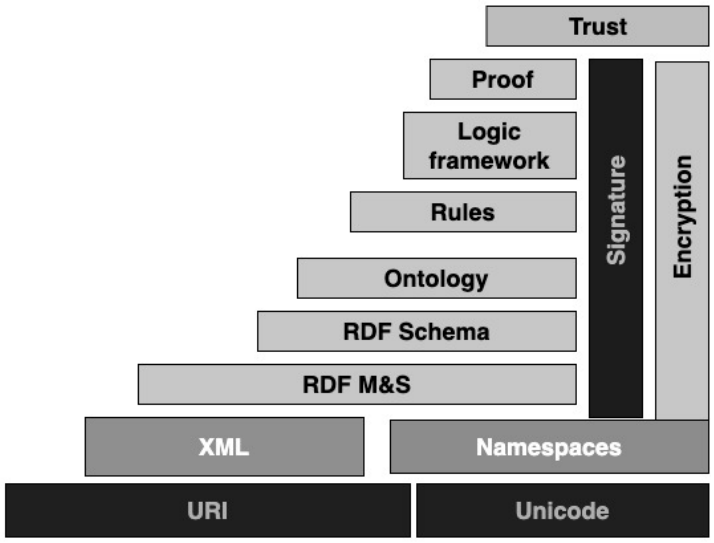


Figure 2.1: Semantic Web Layer Cake [1].

and their relationships, enabling both human and machine understanding.

Developing an ontology serves multiple purposes: fostering a common understanding of information structure among individuals or software agents, enabling the reuse of domain knowledge, clarifying domain-specific assumptions, separating domain knowledge from operational knowledge, and supporting domain knowledge analysis. The Web Ontology Language, endorsed by the W3C, is widely adopted for defining ontologies due to its XML-based syntax and correspondence with Description Logics. OWL offers significant advantages, including logical reasoning capabilities, data structuring, and seamless transmission over the web, making it superior to other ontology languages like Cycl, Loom, and KIF [26].

In the Semantic Web architecture as illustrated in Figure 2.1, semantic functionality is achieved through a layered hierarchy of evolving languages. The Resource Description Framework [27] provides a foundational graph-based reference model, while RDF Schema (RDFS) [28] introduces a basic vocabulary and axioms for object-oriented modeling. Building on this, OWL [29] extends these capabilities by offering advanced constructs and axioms specifically tailored for knowledge-based ontology development.

2.1.2 Significance of Ontology

Ontology is a structured representation of concepts within a specific domain, encompassing their attributes (or properties) and the constraints on these attributes (referred to as facets or role restrictions). When combined with instances of these concepts, an ontology constitutes a knowledge base, enabling the representation, sharing, and reuse of structured knowledge across various platforms and applications. Ontology-based semantic descriptions provide a comprehensive framework for defining entities and their interrelationships, facilitating automated and effective service discovery. Ontology bridges the gap between syntactic and semantic understanding, ensuring machine-processable data exchange between people and machines [30] [25].

Ontology enhances automation, interoperability, and service provisioning, which are critical for dynamic and personalized INC provisioning. They address interoperability challenges across diverse INC programs, network devices, and service providers, enabling automated deployment, dynamic composition, and improved reasoning. Semantic reasoning in ontology simplifies complex network management tasks, making them cost-effective for network providers. Furthermore, this approach improves resource utilization, automates decision-making, and accelerates service discovery, ensuring efficient and accurate results [31] [32] [33].

2.2 Holographic Application

This section explores high-demand holographic applications, focusing on Holographic-type Communication.

2.2.1 Holographic-type Communication

Holographic-Type Communication refers to the capability to transmit, interact with, and render holographic data in real-time over a network [13]. HTC represents a groundbreaking paradigm that goes beyond traditional multimedia and virtual/augmented reality by enabling multi-sensory holographic experiences, including 3D audio, visual parallax, and user-defined sensory inputs like touch and smell. It facilitates real-time digital presence and immersive interactions, making it essential for applications such as holographic telepresence, remote surgery, intelligent transportation,

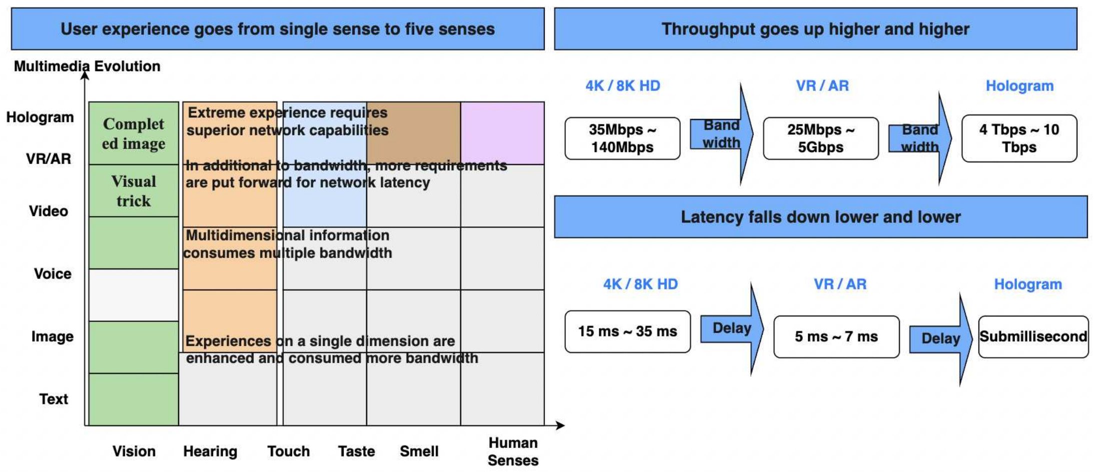

User experience goes from single sense to five senses
Multimedia Evolution
Hologram
VR/AR
Video
Voice
Image
Text
Complet ed image
Visual trick
Extreme experience requires superior network capabilities
In additional to bandwidth, more requirements are put forward for network latency
Multidimensional information consumes multiple bandwidth
Experiences on a single dimension are enhanced and consumed more bandwidth
Vision
Hearing
Touch
Taste
Smell
Human Senses
Throughput goes up higher and higher
4K / 8K HD
35Mbps ~ 140Mbps
Band width
VR / AR
25Mbps ~ 5Gbps
Band width
Hologram
4 Tbps ~ 10 Tbps
Latency falls down lower and lower
4K / 8K HD
15 ms ~ 35 ms
Delay
VR / AR
5 ms ~ 7 ms
Delay
Hologram
Submillisecond

Figure 2.2: Evolution of user experience, from text to multi-sense Holographic Type Communications [2].

and industrial systems. HTC demands ultra-high bandwidth, reaching terabits per second for full-parallax scenarios, and ultra-low latency, with delays below a millisecond, to ensure seamless user experiences [2]. There has been some research works conducted on Holographic-Type Communication. Proof-of-concept implementations by Aghaaliakbari et al. [22] and Javid et al. [24] have demonstrated holographic applications as a practical use case in their studies.

Likewise, HTC aims to integrate advanced technologies to deliver immersive, high-precision services, enabling applications like emergency response and multi-dimensional virtual interactions [2]. Figure 2.2 illustrates the progression of multimedia experiences, showing the increasing network demands for throughput (from Mbps for video to Tbps for holograms) and reduced latency (from milliseconds for video to sub-milliseconds for holograms) as applications evolve from text and video to immersive holograms. Similarly, figure 2.3 highlights the evolution from today's communication systems to future communication paradigms. Future communications envision integrating a "Holographic 5-Sense Matrix" to enable immersive applications using multi-sensory input. This shift emphasizes holographic communication and enhanced user experiences by leveraging new attributes and multi-sense integration, showcasing the transformation required for future communication, such as those anticipated in 6G.

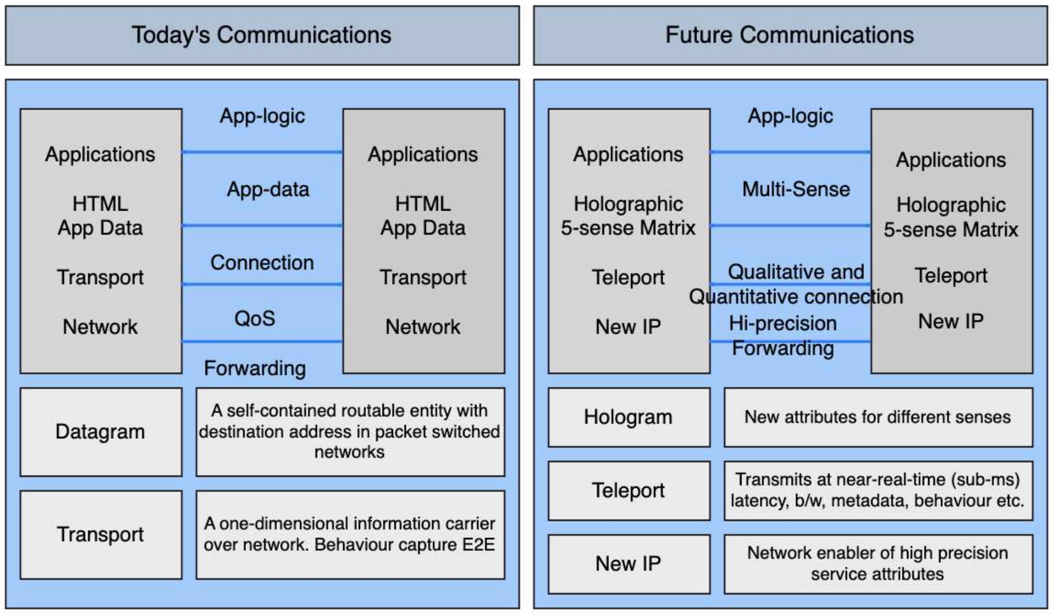

Today's Communications
Future Communications
App-logic
Applications
HTML
App Data
Transport
Network
App-data
Connection
QoS
Forwarding
Datagram
A self-contained routable entity with destination address in packet switched networks
Transport
A one-dimensional information carrier over network. Behaviour capture E2E
Holographic
5-sense Matrix
Teleport
New IP
Multi-Sense
Qualitative and Quantitative connection
Hi-precision Forwarding
Hologram
New attributes for different senses
Teleport
Transmits at near-real-time (sub-ms) latency, b/w, metadata, behaviour etc.
New IP
Network enabler of high precision service attributes

Figure 2.3: New Concepts for Future Communications [2].

2.3 The Emergence of In-Network Computing

This section provides an overview of the emergence of In-Network Computing, highlighting the pivotal roles of Software-Defined Networking and Programmable Data Planes in its development. It also explores how INC addresses and fulfill the stringent requirements of Holographic-Type Communications.

2.3.1 Software-Defined Networking

Software-Defined Networking (SDN) is a networking paradigm typically deployed in the core network that separates the control plane from the data plane, allowing network control to be programmed directly through software-based controllers [34]. This decoupling enables greater control over network management by centralizing the control plane in a software-based controller or network operating system, simplifying policy enforcement, configuration, and network updates while reducing hardware requirements [35]. As illustrated in Figure 2.4, SDN differs from traditional networks by decoupling control from individual devices and centralizing it in a network operating system.

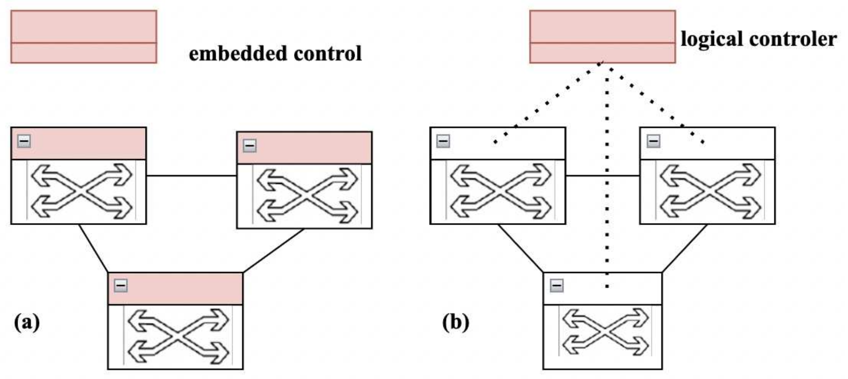
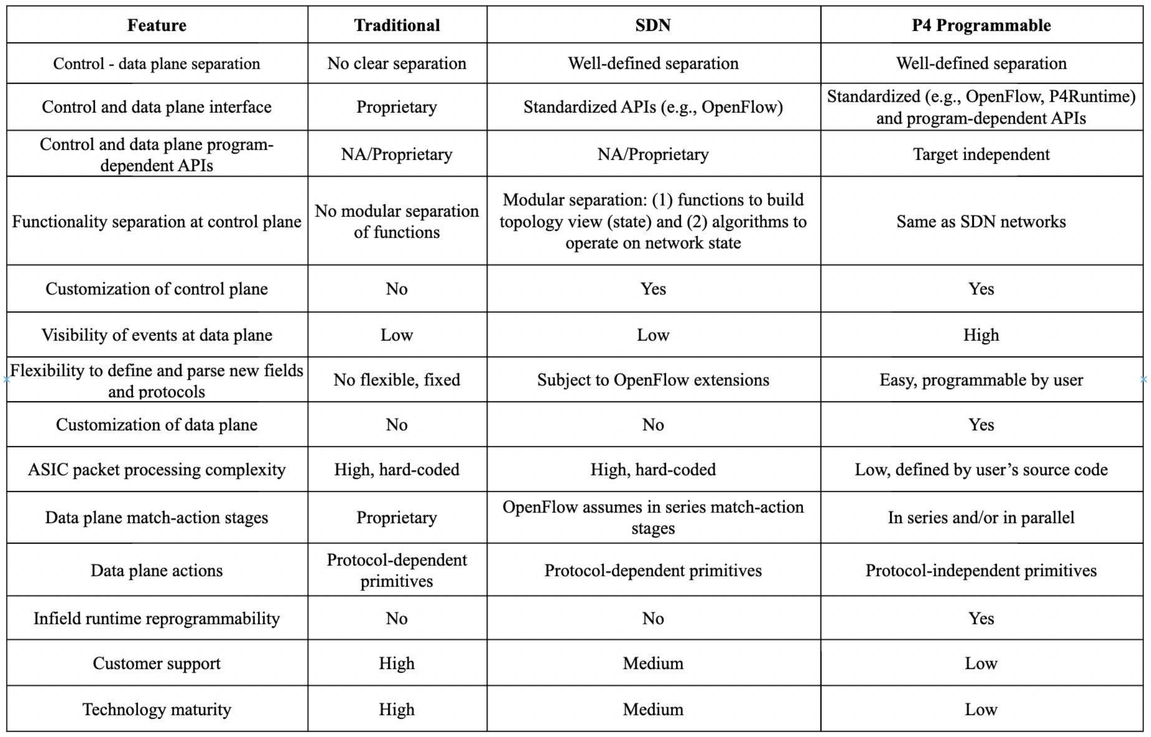


Figure 2.4: (a) Traditional network, (b) Software defined network [3].

Furthermore, SDN's separation of control and data planes also facilitates programmability in the data plane, enabling direct programmability of packet-forwarding devices through open interfaces like OpenFlow and ForCES [36], [17]. This architecture streamlines traffic engineering, supports remote management, and allows the deployment of new protocols and applications. SDN's programmability has also led to the development of programmable data planes (PDPs), which allow packet processing to be customized using high-level languages [37], [38]. For example, in the SDN-based network computing model, data processing is implemented using P4 programming and deployed alongside the flow table on network devices that handle data packets [5]. Unlike traditional fixed-function switch chips, PDPs give network operators flexibility over packet processing, enabling faster adoption of new data plane functions and facilitating prototype development [17]. However, SDN provides a clear division between control and data planes, centralizing network intelligence while maintaining a programmable, standardized data plane [39], [36].

2.3.2 Programmable Dataplane

Programmable data plane built on the SDN principle of separating control and data planes, programmable data planes utilize a standard API to manage interactions between the two planes [36]. In this model, the data plane consists of ”dumb” network elements managed by the control plane, which accesses embedded state information to control network programmability. The data plane, forming the network's core infrastructure, processes packets using a combination of hardware and specialized software distributed across elements such as ASICs, FPGAs, network processors,



Feature	Traditional	SDN	P4 Programmable
Control - data plane separation	No clear separation	Well-defined separation	Well-defined separation
Control and data plane interface	Proprietary	Standardized APIs (e.g., OpenFlow)	Standardized (e.g., OpenFlow, P4Runtime) and program-dependent APIs
Control and data plane program-dependent APIs	NA/Proprietary	NA/Proprietary	Target independent
Functionality separation at control plane	No modular separation of functions	Modular separation: (1) functions to build topology view (state) and (2) algorithms to operate on network state	Same as SDN networks
Customization of control plane	No	Yes	Yes
Visibility of events at data plane	Low	Low	High
Flexibility to define and parse new fields and protocols	No flexible, fixed	Subject to OpenFlow extensions	Easy, programmable by user
Customization of data plane	No	No	Yes
ASIC packet processing complexity	High, hard-coded	High, hard-coded	Low, defined by user's source code
Data plane match-action stages	Proprietary	OpenFlow assumes in series match-action stages	In series and/or in parallel
Data plane actions	Protocol-dependent primitives	Protocol-dependent primitives	Protocol-independent primitives
Infield runtime reprogrammability	No	No	Yes
Customer support	High	Medium	Low
Technology maturity	High	Medium	Low

Figure 2.5: Features, traditional, SDN, and P4 programmable devices [4].

and NICs, often with a packet classification engine. Technologies such as SDN, programmable data planes, edge computing, support In-Network Computing (INC) [3].

Traditionally, the data plane has been limited to fixed functions for packet forwarding with a limited set of protocols. Recently, however, data plane programmability has gained interest from researchers and industry, allowing custom packet processing functions. This evolution from SDN enables rapid design, testing, and deployment of packet processing. The P4 language (Programming Protocol-independent Packet Processors) has emerged as the standard for defining forwarding behavior [40], allowing P4-programmable switches to bring network design to a broader audience previously restricted to network vendors. The original P4 specification, known as P414, was released in 2014, and a more robust version, P416, followed in 2016, expanding P4's applicability to include ASICs, FPGAs, Network Interface Cards (NICs), and other targets [39].

As shown in Figure 2.5, P4-programmable devices offer several advantages over traditional and SDN-based solutions. Key benefits of P4-programmable devices include program-specific APIs, enabling the same P4 code to run across different targets without modification in runtime applications. These devices also use protocol-independent packet processing primitives, feature a more advanced computation model allowing match-action stages in both series and parallel, and support in-field reprogrammability at runtime [39].

2.3.3 In-Network Computing

The concept of In-Network Computing 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. [15] aptly describe INC as a 'dumb idea whose time has come.' As illustrated in Figure 2.7, INC is an emerging computing model that delegates application-layer processing functions to the network data plane, enabling data to be processed during transmission and reducing overall traffic volume. This approach alleviates computing load and energy consumption on cloud systems by offloading tasks to network nodes. INC moves data processing tasks from the host directly to network devices, which can handle traffic processing while forwarding it. By utilizing the idle resources of network devices, INC reduces data transmission demands and cloud computing pressure [5].

Based on the survey in [3], figure 2.6 illustrates the computing capabilities enabled by in-network computing. The in-network computing fabric consists of network elements in the blue cloud, positioned between end-devices and edge computing servers (e.g., MEC servers, fog nodes, cloudlets) or between end-devices and cloud data centers (and even between edge and cloud servers). The green path represents an end-to-end communication route, which can be truncated before reaching cloud servers (e.g., at point 2) by using in-network computing on intermediate network elements like programmable switches, FPGAs, or smart NICs. Similarly, traffic on the red path may be processed by network elements before reaching edge servers (e.g., at point 1), allowing results to be returned to end-devices from a closer location than the edge server.

Moreover, the authors in [5], note that recent advancements in programmability and processing power allow network devices to handle increasingly complex computing tasks, reducing the need for numerous cloud servers and decreasing overall energy consumption. This approach allows networks to dynamically adapt to varying conditions and demands. However, high-performance

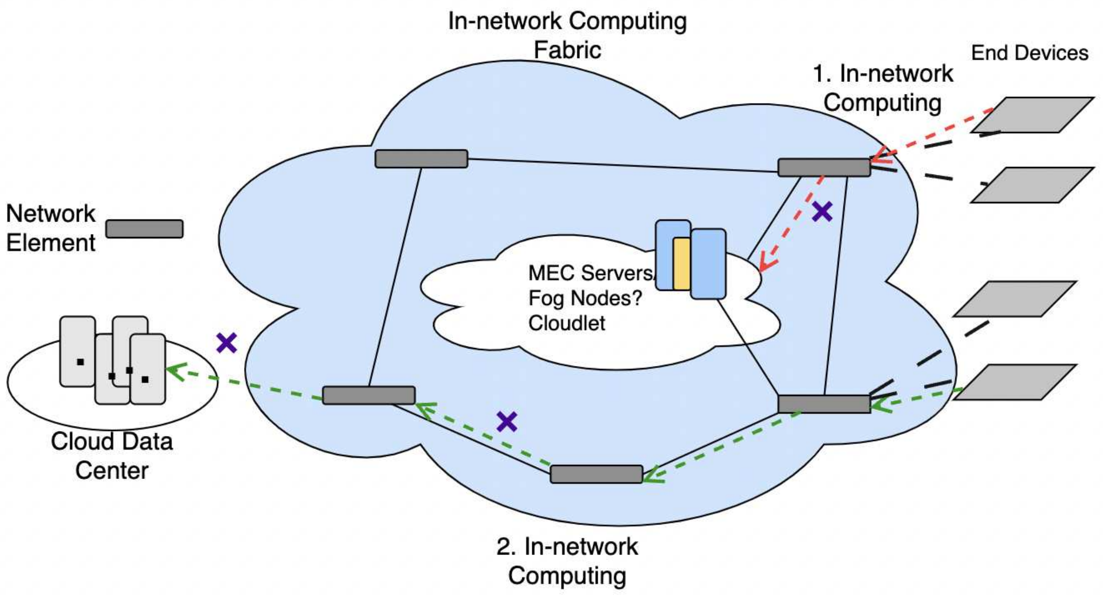


Figure 2.6: Schematic of in-network computing fabric [3].

programmable network processors, along with SDN and the development of the P4 programming language, have made active, programmable networks feasible and practical [5], [41]. Furthermore, the resurgence of INC is largely driven by 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 [39].

As a product of application-network integration, INC aims to improve processing performance and energy efficiency by merging application systems with communication infrastructure. Moreover, INC offers notable advantages: lower latency through in-transit data processing, high throughput by gradually reducing data volume during transmission, and enhanced energy efficiency by reducing data center load [5]. Few proof-of-concept works have explored INC's potential. For instance, Sapio et al. [15] developed an INC framework with the BMv2 software switch, demonstrating significant data reduction (up to 89.3%) and latency improvements.

On the other hand, applications like holographic streaming impose stringent Quality of Service (QoS) requirements that current infrastructures often struggle to meet, particularly when these demands need to be addressed concurrently [13]. In this context, In-Network Computing plays a

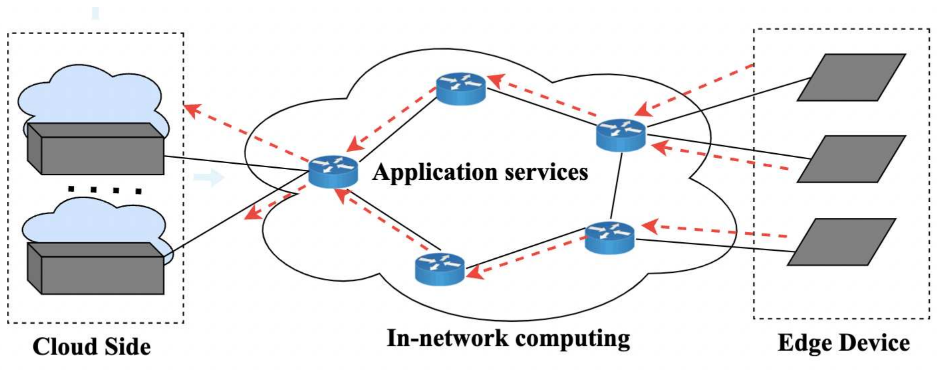


Figure 2.7: In-Network Computing model [5].

crucial role in fulfilling these requirements. For instance, among the few previous studies, Aghaali-akbari et al. [22] 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.

2.4 Conclusion

In this chapter, we provided a comprehensive overview of the foundational concepts essential to this thesis. We first explored ontology, discussing its role in structured knowledge representation and its significance in enabling automated and semantic-driven discovery mechanisms. Next, we examined holographic applications, particularly holographic-type communication, highlighting its growing demand and the stringent network requirements necessary for seamless real-time interactions. Finally, we delved into the emergence of In-Network Computing as a transformative approach to handling computational workloads directly within network infrastructure. We analyzed key networking technologies, including Software-Defined Networking and programmable data planes, which play a crucial role in enabling INC. By integrating these concepts, this chapter establishes the necessary background for understanding the motivation behind an ontology-based INC component description and discovery model, which is the focus of this thesis.

Chapter 3

Use Case and State of the Art

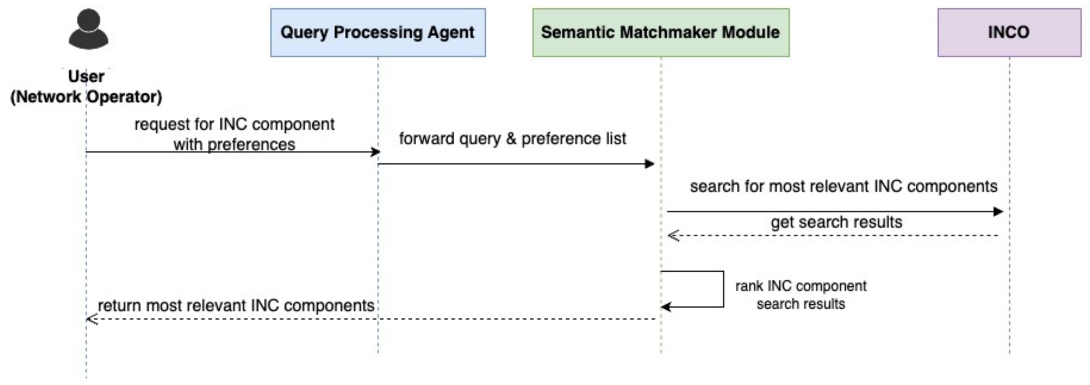
In this chapter 3, we first introduce our holographic streaming use case to outline the requirements for the INC Components description and discovery model architecture. Next, we derive the specific requirements based on this use case. Finally, we summarize the State of the Art in relation to these requirements and concluded the chapter.

3.1 Use Case

Consider a holographic streaming use case designed for immersive remote events, such as a real-time holographic concert. In this scenario, users can experience high-quality holographic content, engaging with the event in real-time as if they were physically present. To initiate a holographic streaming session, users require specific INC components tailored to their needs. For example, INC components such as a renderer, encoder, decoder or transcoder may be essential for ensuring a seamless streaming experience.

To retrieve the most relevant INC components, users (e.g., Network operator) provide their specific requirements and preferences, enabling a system to match and select the best-fit components for their request. For the holographic concert use case, the following sequence of actions is envisioned:

(1) To facilitate the retrieval of INC components, they must first be published by the INC provider in a centralized repository.



User
(Network Operator)
Query Processing Agent
Semantic Matchmaker Module
INCO
request for INC component with preferences
forward query & preference list
search for most relevant INC components
get search results
rank INC component search results
return most relevant INC components

Figure 3.1: Sequence of interaction during Holographic Streaming.

(2) The user submits a request for the desired INC components (e.g., encoder, renderer, decoder, or transcoder), specifying their requirements and preferences to ensure seamless INC provisioning for an optimal holographic streaming experience.

(3) Upon receiving the user's request, the Query Processing Agent processes it by translating the specified requirements and preferences into a structured query and preference list, which is then forwarded to the Semantic Matchmaking Module.

(4) The Semantic Matchmaking Module receives the query and preference list from the Query Processing Agent and leverages the INC Component description model (INCO) to search for the most suitable INC components.

(5) After completing the search, the Semantic Matchmaking Module ranks the matched components based on the user's preferences and returns the results to the user.

Figure 3.1 provides a visual representation of this sequence, detailing the steps involved in INC Component discovery, ranking, and returning them to the user based on their preferences and request for successful placement of INC Components to fulfill the holographic streaming request.

3.2 Requirements

In the context of seamless INC component discovery for holographic streaming, several challenges arise. As described in the use case, INC providers design and publish components to a centralized repository, which allows to efficiently retrieve these INC components by semantic matchmaker for service orchestration.

To this end, the first requirement derived from the use case is that the proposed architecture must include a centralized repository that functions as a database to store and access INC components, allowing seamless publishing by INC providers.

Since this use case involves retrieving specific INC components for holographic streaming, a detailed description model is essential. This model should encompass a comprehensive ontology covering both functional and non-functional properties, enabling precise identification and retrieval of components like encoders, decoders, transcoders, and renderers.

Hence, the second requirement is the need for an ontology-based semantic description model, which provides a structured and comprehensive description of INC components to facilitate smooth and automated component discovery.

Nevertheless, for efficient and automatic discovery, the architecture must include a semantic matchmaker that can identify and rank the most relevant INC components based on each user's request and preferences/criteria.

Therefore, the third requirement is that the architecture should incorporate a semantic matchmaker module to retrieve and rank the most suitable INC components based on user-defined criteria.

To ensure that user requests are interpretable by the semantic matchmaker, these requests, including specific requirements and preferences, must be translated into a query format. A query processing module is needed to convert user requests into a form that the semantic matchmaker can understand and use for accurate ranking and retrieval.

Finally, the fourth requirement is to provide a query processing module that translates user requests and preferences into query and preference list, ensuring that the semantic matchmaker can effectively retrieve the required INC components.

3.3 State of the Art

In this section 3.3, we review the state of the art in the discovery and description of network components, aiming to adapt relevant insights for our work. We organize and review the state of the art in four main subsections: the first subsection discusses the role of semantics in service discovery, emphasizing its importance in enhancing discovery. Next, in sections 3.3.2 and 3.3.3 reviews prior research on VNF descriptions and discovery methods. Subsequently, we discuss the design considerations and challenges specific to this thesis. Finally, the chapter concludes with a synthesis and summary of the reviewed works.

3.3.1 The Role of Semantics in Service Discovery

As the number of web services (WS) grows, efficient service discovery and composition have become critical challenges for distributed applications. Service discovery methods are generally categorized into syntactic-based and semantic-based approaches. Syntactic matching relies on string equivalence of parameter names for a service's input and output messages, often utilizing graph theory, such as the Resource Description Framework (RDF) and DIANE Service Description [42]. In contrast, semantic matching leverages ontologies like W3C Web Ontology Language (OWL-S) [43] and Web Service Modeling Ontology (WSMO) [44] to represent a service's interface and attributes, enabling parameter type matching based on XML Schema type hierarchies [45]. For instance, OWL ontologies were utilized in the FP7 European mOSAIC project to represent heterogeneous multi-vendor cloud resources and user Service Level Agreements in cloud environments [46].

XML-based specifications provide only syntactic descriptions of Web services, requiring human involvement during discovery processes. On the other hand, semantic Web service technology aims to minimize manual intervention by enabling software agents to automatically locate, integrate, and execute services to meet user objectives. The Semantic Web vision has inspired enhancements to Web service descriptions with machine-interpretable semantics, creating ”semantic Web services” to automate core tasks like discovery, composition, selection, and invocation [10].

Moreover, to enable effective service discovery, key steps include adding interpretable metadata, documenting services consistently, storing the information in a searchable repository, and facilitating efficient searches [47]. Although syntactic-based standards like Universal Description and Discovery Interface (UDDI), Web Services Description Language (WSDL), and Simple Object Access Protocol (SOAP) provide a foundation for Web services, they rely on syntactic descriptions, leading to inefficiencies in service discovery and composition. In contrast, semantic Web technologies or services are likely to provide better qualitative and scalable solutions to these problems and facilitate more efficient and flexible service discovery and composition, emphasizing the role of semantic descriptions in overcoming the limitations of traditional, syntactic-based approaches [45] [10] [47].

Likewise, the semantic-based approach to Web service discovery also considers both functional and non-functional parameters, emphasizing interoperability. Nevertheless, numerous studies in service computing have examined service descriptions and user queries, showing that incorporating semantics improves precision and recall in service discovery. Research comparing syntactic methods, typically based on information retrieval metrics, with semantic approaches involving logical inference has shown that semantic matching outperforms syntactic matching, motivating our focus on semantic matching for our thesis work [45], [10], [9]. Table 3.1 summarizes the key comparisons discussed in this section that guided our decision to choose a semantic-based approach over syntactic methods.

Table 3.1: Semantic Matchmaking over Syntactic for Discovery [10].

Comparison Features	**Syntactic-based**	**Semantic-based**
Matches Functional Parameters	✓	✓
Matches Non-Functional Parameters	✗	✓
Automatic Discovery	✗	✓

3.3.2 Related Work on VNF Description and Discovery

In this section 3.3.2, we examine and evaluate several approaches previously used for describing Virtual Network Functions. As mentioned earlier, to the best of our knowledge, no prior work has been done on describing and discovering INC components. Therefore, we focused on VNFs to gain valuable insights that could be adapted to our proposed model. For instance, Cloud4NFV

[48], architecture is compliant with the ETSI NFV architectural guidelines, Cloud4NFV platform adheres to major NFV standard guidelines and is built on cloud infrastructure management and SDN platforms, designed for the provisioning of VNFs, delivering Network Functions as a Service to end customers. Cloud4NFV offers significant contributions to modeling and orchestration incorporating a front-end database that stores collections of VNFs with high-level descriptions (e.g., ID, name, description, location) and a back-end database that contains specific VNF details required for deployment and configuration. However, Cloud4NFV focuses solely on deployment and configuration and did not mention about the non-functional properties. Moreover, it lacks an automated discovery mechanism for VNFs.

Table 3.2: Summary of related work evaluating VNF description and discovery approaches.

Virtual Network Function / In-Network Computing Components					
Papers	**Description**		**Publication**	**Discovery**	
	Functional Properties	**Non-Functional Properties**	**Inter-operability**	**Semantic Match-making**	**User Request Builder**
Cloud4NFV [48]	VNF (✓) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)
OASIS TOSCA NFV [49]	VNF (✓) INC (✗)	VNF (✗) INC (✗)	VNF (✓) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)
T-NOVA [50]	VNF (✓) INC (✗)	VNF (Partially) INC (✗)	VNF (✓) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)

In [49],TOSCA-NFV refers to the TOSCA Simple Profile for NFV, a standard developed by OASIS. It provides a data model and guidelines using the TOSCA (Topology and Orchestration Specification for Cloud Applications) language to describe and manage the deployment, connectivity, and lifecycle of VNFs within an NFV environment. It enables interoperability between NFV components and simplifies the deployment of network services. It also enables automation of VNF deployment, scaling, and lifecycle management through orchestration tools. While TOSCA-NFV promotes interoperability and addresses functional properties of VNFs, it falls short in providing semantic discovery and does not comprehensively address non-functional properties.

The T-NOVA project [50], presents an innovative framework for delivering Network Functions as a Service through an integrated NFV marketplace. This framework enables network operators to offer VNFs to customers without requiring hardware. The architecture includes a marketplace where developers can publish VNFs, and customers can select tailored functions, supported by an advanced orchestrator for automated provision, management, monitoring and optimization of VNFs. While T-NOVA addresses interoperability, it only partially covers non-functional properties and lacks a mechanism for the semantic discovery of VNFs.

Table 3.2 summarizes the most relevant studies related to VNF description, publication, and discovery. The review in this section reveals that while some works (e.g., [48], [49], [50]) have made strides in functional properties and interoperability, none offer an automated semantic discovery process, address user requests, or comprehensively cover the non-functional properties of VNFs.

3.3.3 Related Work on Ontology-based VNF Description and Discovery

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 ontology-based VNF description and discovery methods to adapt relevant insights to our INC model.

For example, Bonfim et al. [51] 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, this approach does not address non-functional properties.

Additionally, Hoyos et al. [6] 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. In this work, two VNFD descriptors from different NFV implementations were parsed into NOn VNFD instances creating a semantic VNFD template

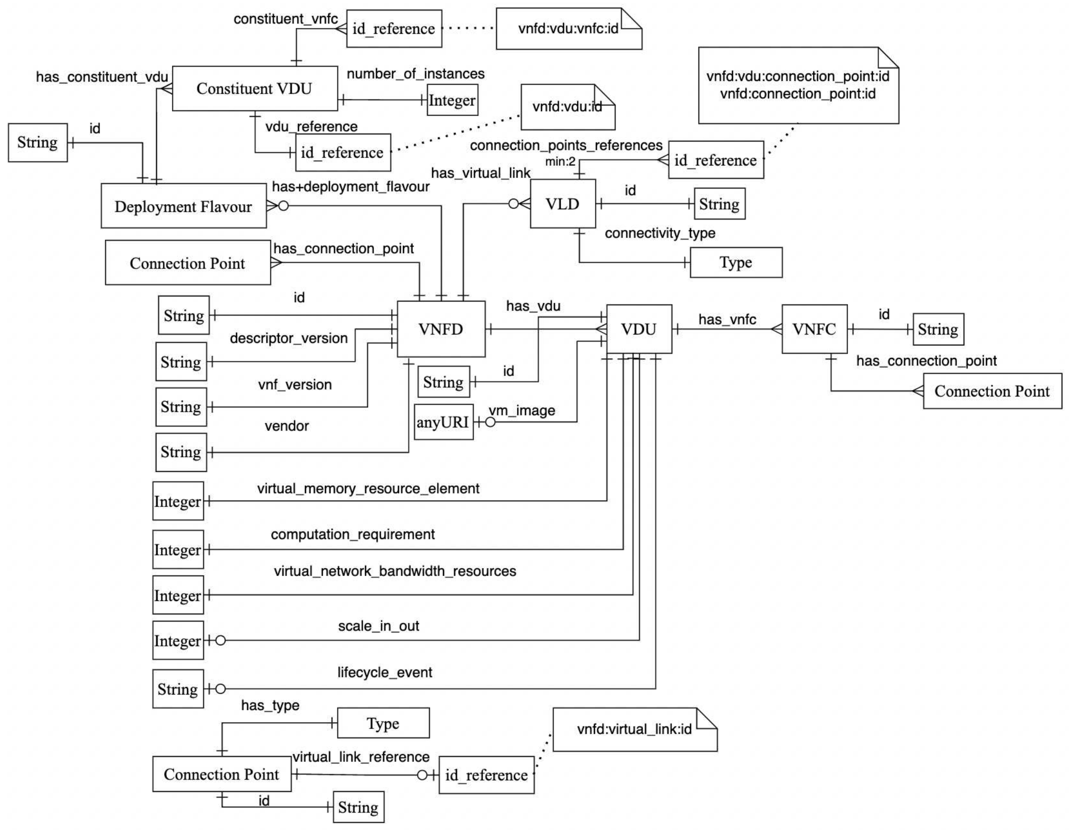


Figure 3.2: NOn Model [6].

following ETSI specification. Figure 3.2 illustrates the final design of NOn, using the entity relationship diagram annotation, containing all the abstracted elements, slots (properties) and cardinality. Moreover, user requests were created using the proposed Generic Client in their work. 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. [7] introduced TRAILS, an extension to TOSCA NFV figure 3.3 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.

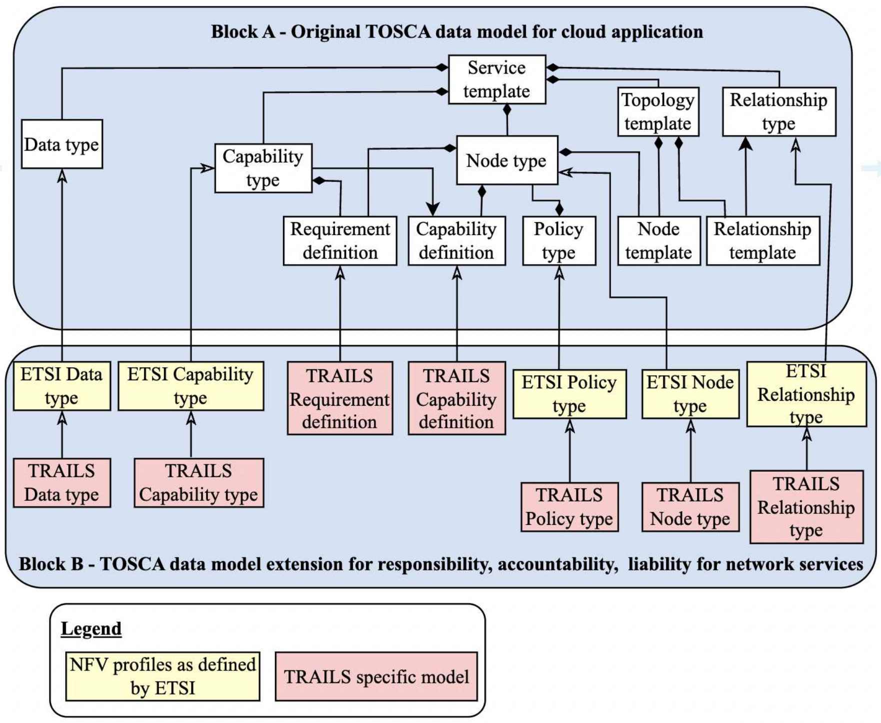


Figure 3.3: Extension of the TOSCA NFV metamodel [7].

Kim et al. [8] utilized Network Service Description (NSD) data and ontology to automate VNF management and enable the automatic generation of network services. By employing ontology reasoning and semantic annotations, they developed an ontology that captures semantic relationships between parameters. This was achieved through an annotation process that enriched NSD parameters with semantic information, allowing for the discovery of similar services and facilitating automated network service composition. Their descriptor includes functional information such as, details about VNFs and their dependencies, non-functional information such as vendor details, and optional information blocks for features like scaling and monitoring. Figure 3.4 shows the three blocks and the relationships between classes. However, their work lacks a discovery algorithm specifically for VNFs. Likewise, Oliver et al. [52], propose an ontology for NFV that formally describes the whole network resources, VNF properties, and relationships. While it covered functional aspects of VNFs, like Kim et al. [8], it lacked a discovery algorithm and non-functional property coverage.

Few studies focus on both functional and non-functional properties. Notably, Nouar et al. [9]

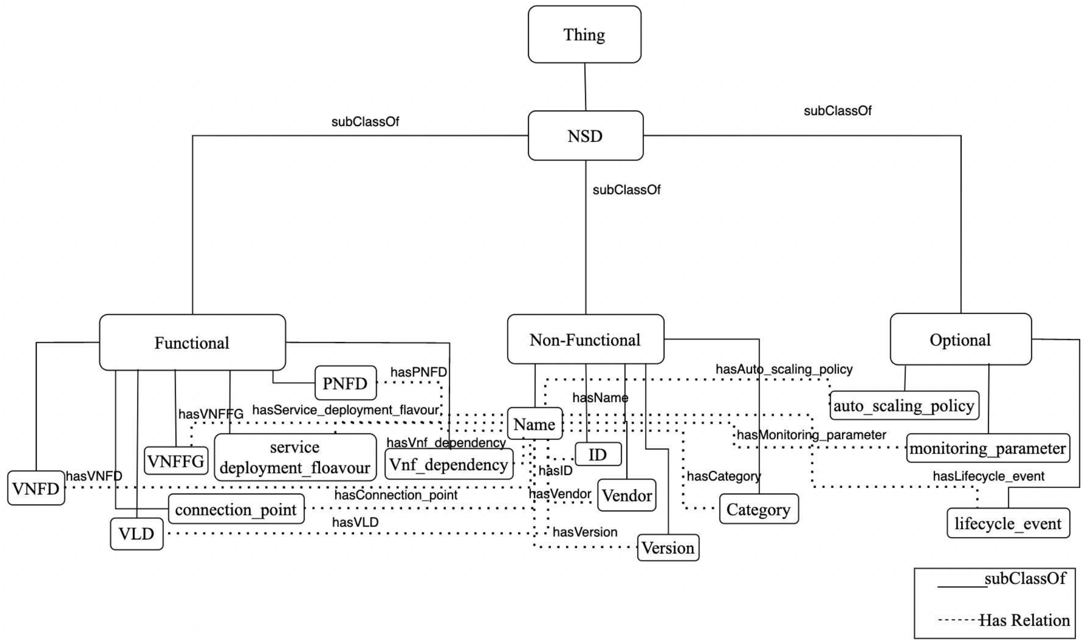


Figure 3.4: NFVO Ontology [8].

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 properties. 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. Their model supports building user requests by considering user requirements and preferences, which are clustered into three distinct categories, providing a more advanced and automated approach compared to previous methods.

Table 3.3 sums up the most relevant studied work on ontology-based VNF description, publication, and discovery. In addition, Nouar et al. [9] VIKING ontology, succeeded in covering both the functional and non-functional properties of the VNFs in their proposed description models, separately highlighted in figure 3.5 and also addressed semantic discovery.

Building on these studies and addressing identified gaps, we developed an ontology-based semantic description model that comprehensively covers both the functional and non-functional aspects of INC components. In our conference paper [53], accepted at the 20th International

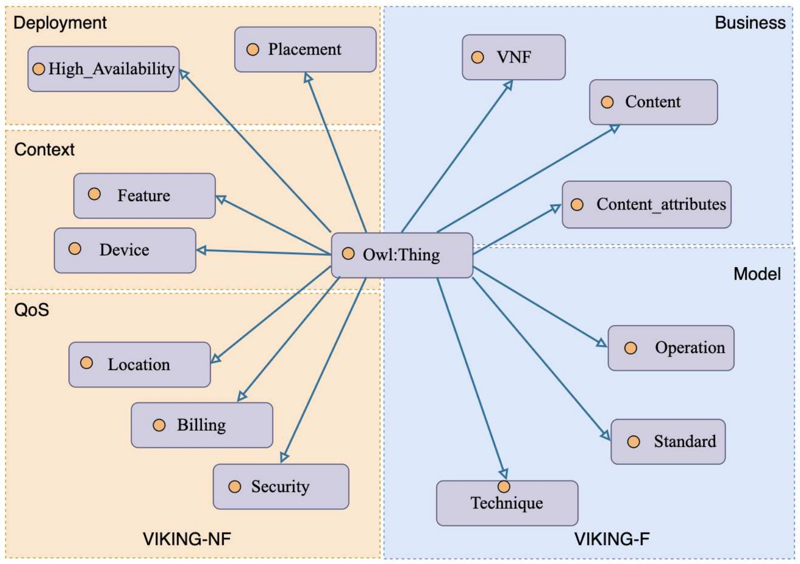


Figure 3.5: Core concepts in the VIKING framework, categorized into functional and non-functional properties [9].

Conference on Network and Service Management, we proposed a model that extends semantic matchmaking-based discovery for INC components—a core idea of this thesis. This thesis further expands on the work presented in our conference paper, as discussed in the following chapter.

Table 3.3: Summary of related work evaluating on ontology-based VNF description and discovery approaches.

Virtual Network Function / In-Network Computing Components					
Papers	**Description**		**Publication**	**Discovery**	
	Functional Properties	**Non-Functional Properties**	**Inter-operability**	**Semantic Match-making**	**User Request Builder**
Bonfim et al. [51]	VNF (✓) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)
Hoyos et al. [6]	VNF (✓) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)	VNF (✓) INC (✗)
Anser et al. [7]	VNF (✓) INC (✗)	VNF (✓) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)
Kim et al. [8]	VNF (✓) INC (✗)	VNF (✓) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)
Oliver et al. [52]	VNF (✓) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)	VNF (✗) INC (✗)
Nouar et al. [9]	VNF (✓) INC (✗)	VNF (✓) INC (✗)	VNF (✓) INC (✗)	VNF (✓) INC (✗)	VNF (✓) INC (✗)

3.4 Design Considerations

In VNF architecture, virtualization techniques are used to manage network functions, with the VNF Descriptor (VNFD) playing a critical role in defining and specifying the characteristics, configuration details, and performance requirements of VNFs [54]. The VNFD serves as a standardized template that includes essential properties such as version, name, and vendor, providing an overview of each function and its role in the network. In contrast, there is no standardized descriptor for INC components to define their specific characteristics and properties.

Constructing an optimal INC ontology to support an efficient discovery mechanism was particularly challenging. An ideal ontology should be well-structured, clearly defining concepts, relationships, and rules governing interactions. To build a coherent and comprehensive INC ontology, INC providers must supply a descriptor that contains essential information related to INC operations and deployment. In some studies (e.g., [9], [6]), the VNFD has been used as a blueprint to build an ontology defining the functional and non-functional properties of VNFs. However, INC components lack a similar descriptor, making the task of description and discovery significantly challenging in this thesis.

Deployment related information of INC components including bandwidth, latency, and supported compilers, is critical and must be given by the INC provider. Without such details in the INC descriptor, developing a comprehensive ontology would be challenging. For example, specifying the supported compiler, such as the requirement for P4-enabled switches, is essential for the correct deployment of INC components. The VNFD specifies how VNFs should be deployed and managed; for instance, it incorporates Virtual Deployment Units (VDUs) that support specific deployment resources essential for hosting VNF components. Likewise, a similar descriptor for INC components should exits, incorporating essential deployment-related information. Moreover, to build a comprehensive ontology, INC descriptors should also include details about INC operations. These parameters are essential for constructing an optimal ontology and enabling efficient INC component discovery.

Considering these factors, we designed an ontology model that encapsulates all necessary information to describe each INC component, including functional and non-functional properties. In this thesis, we assume that INC providers will provide an INC descriptor for each component, containing essential information to construct our ontology model and facilitate an effective discovery mechanism.

3.5 Conclusion

In this chapter, we explored the foundation and context for our thesis by presenting a detailed use case for holographic streaming, which highlights the challenges and requirements for the description and discovery of In-Network Computing components. Through this use case, we derived key requirements, including the necessity for a centralized repository, an ontology-based semantic description model, a semantic matchmaking module, and a query processing module, to enable efficient and accurate INC component discovery. We then reviewed the state of the art in service discovery and network function virtualization description and discovery, focusing on the role of semantics and ontology-based approaches. The review underscored the gaps in INC component descriptions, further emphasizing the need for a tailored ontology-based solution. Finally, we discussed the design considerations and challenges associated with developing a comprehensive INC ontology and efficient discovery mechanism. The groundwork laid in this chapter sets the stage for the subsequent chapters.

Chapter 4

Proposed Architecture

In the previous chapter, we identified the requirements for an In-Network Computing Component Description and Discovery Model tailored to the holographic streaming use case. This chapter introduces a proposed architecture designed to address these requirements. It begins with an overview of the architecture and its main functional entities. Subsequently, detailed descriptions of the key modules are provided, including the Centralized Repository, INC Ontology Module, Semantic Matchmaking Module, and Query Processing Agent. The chapter further elaborates on the design and development process of the INC component description model. Additionally, it highlights the INC discovery process, detailing the steps involved in building user requests, query and explaining how the architecture fulfills the predefined requirements for efficient INC component discovery. Finally, the flow of INC component discovery and the interactions between the various modules are outlined.

4.1 General Overview of the Architecture

This section 4.1 describes our proposed architecture, as depicted in Figure 4.1. The architecture enables the ontology-based description of INC Components and their discovery through a semantic matchmaking algorithm. It is designed to support advanced applications like holographic-type communication by leveraging a structured and modular approach.

As shown in Figure 4.1, the architecture comprises several key modules, each playing a critical

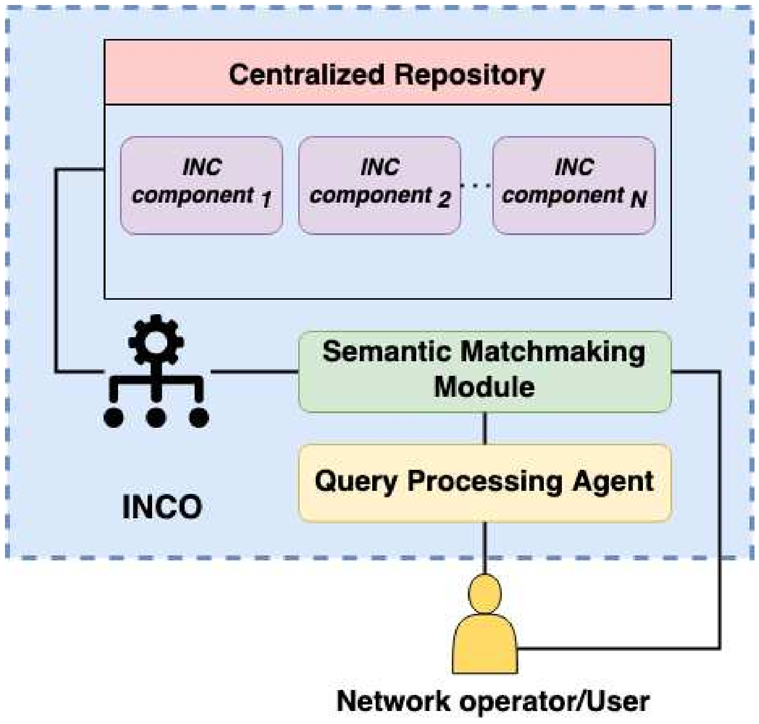


Figure 4.1: Proposed Architecture for INC Components Description and Discovery.

role in ensuring efficient INC discovery. These modules include the following entities: `Centralized Repository`, `Semantic Matchmaking Module`, `Query Processing Agent`, the ontology-based semantic description model `INCO`, and the `Network Operator/User`.

At the core of the architecture lies the `Centralized Repository`, which serves as a database for storing INC components. Each component is described using an ontology-based approach called the `In-Network Computing Ontology (INCO) model`, which provides detailed and structured description of both functional and non-functional properties ensuring interoperability and reusability. The `Semantic Matchmaking Module` plays a pivotal role in enabling efficient component discovery by leveraging the structured descriptions in the ontology. Acting as a bridge between users (e.g., Network operator) and the `Semantic Matchmaking Module`, the `Query Processing Agent` processes user-submitted requirements and preferences, translating them into actionable inputs for the matchmaking process. `User/Network Operator` interacts with the `Query Processing Agent` to specify their requirements and preferences. The architecture is designed to meet the stringent demands of next-generation applications, such as holographic-type communication, by providing a semantic solution for INC component description and discovery. In the following subsections, we provide a detailed explanation of each module within the proposed system.

4.2 Module Description

This section discusses the modules within the proposed architecture. As mentioned earlier, the architecture comprises the following key entities: `Centralized Repository`, `INCO`, `Semantic Matchmaking Module`, `Query Processing Agent`, and `User/Network Operator`. The subsequent sections provide a detailed description of each module.

4.2.1 Centralized repository

The Centralized Repository serves as a database for storing and accessing INC components, functioning as a marketplace. It enables INC providers to publish INC components seamlessly and ensures efficient access to the retrieval and discovery of these components. Each INC Component (e.g, $INC\_Component_1....INC\_Component_N$) includes a INC Descriptor—a structured template defining the component's functional and non-functional properties, computational requirements, and deployment parameters. The INC Descriptor outlines essential details, such as the operation name, ID, and deployment information.

4.2.2 INC Ontology / INCO Module

The INC Ontology Module (INCO) serves as the semantic description model for INC components. It employs an ontology-based approach to provide comprehensive semantic descriptions for both generic and specific INC components tailored to holographic streaming. INCO semantically represents INC components using INC descriptors, defining the domain scope, managing queries, and facilitating the description, publication, and discovery of INC components.

4.2.3 Semantic Matchmaking Module

The Semantic Matchmaking Module (SMA) serves as a semantic matchmaker for retrieving the best-matched INC components. Utilizing the algorithm outlined in Algorithm 1, this module identifies and returns a ranked list of relevant INC URIs. By leveraging the INCO model, the algorithm aligns with user preferences and requirements, discarding irrelevant components and ranking the relevant ones. The module relies on the ontology-based description model to ensure efficient and accurate component discovery.

4.2.4 Query Processing Agent

The Query Processing Agent (QPA) acts as a bridge between users and the SMA, translating user requests into a system-processable format. It converts user requests into SQWRL (Semantic Query-Enhanced Web Rule Language) queries and generates a preference list categorized into mandatory, high, and optional preferences (to be discussed in the following section). The QPA is also responsible for forwarding the query and preference list to the Semantic Matchmaking Module for processing.

4.3 Ontology-based approach for INC Component Description

This section details the INCO model. It explains how the INCO model is designed and constructed, covering aspects such as class definitions, class hierarchies, relationships, property definitions (slots) of classes, facets of slots, instance creation, and determining the domain and scope of INCO. Then it discusses the functional and non-functional properties of INC components, along with their key concepts. Additionally, we further elaborates INCO for holographic-type communication specific INC components.

4.3.1 INCO Model Design and Description

INCO is an OWL-based ontology designed to describe INC Components. This model was developed following the principles outlined in Ontology Development 101: A Guide to Creating

Your First Ontology [25]. The development adhered to a structured approach, ensuring a comprehensive design that captures both functional and non-functional properties of INC components. Additionally, to enhance the model's relevance, insights from existing references were incorporated. Technologies related to encoding, decoding, and transcoding, such as H.265/HEVC and H.264/AVC [55, 56, 57], along with rendering techniques like OpenGL [58, 59, 60, 61, 62, 63, 64], were studied and adapted for our work. Additional references consulted include [6, 9].

The following steps summarize the key actions undertaken during the INCO design process:

(1) **Determine the domain and scope of the ontology:** To design INCO, we first defined the specific domain and scope of the ontology. The domain and scope of INCO were defined to focus on In-Network Computing.

(2) **Define the intended purposes of INCO:** The goals of the ontology were defined to encompass the description and discovery of INC components specifically designed for Holographic-Type Communication.

(3) **Define the classes and the class hierarchy:** Core concepts were organized into classes, and a hierarchy was established to represent their relationships. A top-down strategy was adopted, starting with general concepts and refining them into specialized subclasses.

(4) **Define the properties of classes (slots):** Properties (slots) were defined to align with the characteristics of each class, such as `bandwidth, latency`.

(5) **Define the cardinality of the slots:** Cardinality restriction and type values were defined for each property to ensure that concept instances are correctly related to the appropriate instance(s) and belong to the right concepts. For example, the property `descriptor_version` has a cardinality of "1" and a value of type `String`.

(6) **Specify the types of queries INCO should address:** The ontology was designed to handle queries focused on similarity, ensuring efficient discovery processes.

(7) **Create instances:** Specific examples (instances) were created to validate and demonstrate the ontology's functionality.

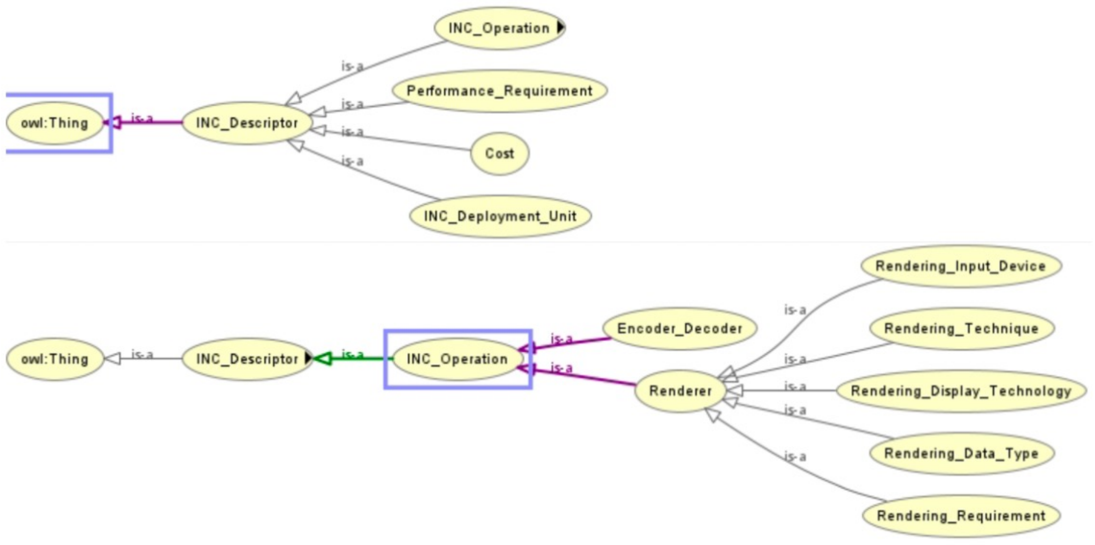


Figure 4.2: INCO's core concepts.

Concepts in INCO are organized hierarchically, where subclasses represent more specific concepts while inheriting properties from their superclasses. Properties connect these concepts through semantic relationships. INCO leverages OWL's reasoning capabilities to enhance discovery and classification.

Figure 4.2 illustrates INCO's core concepts. In OWL, the `owl:Thing` class serves as the universal superclass, encompassing all individuals by default. Key classes, such as `INC_Descriptor`, `Cost`, `Performance_Requirement`, `INC_Deployment_Unit`, and `INC_Operation` are distinguished by their functional or non-functional properties, highlighted in blue or purple. The class hierarchy represents an "`is-a`" relationship, indicating that subclasses inherit the properties (slots) of their parent class. For example, `Encoder_Decoder` and `Renderer` are specific subclasses of the `INC_Operation` class. Although these subclasses share a common superclass, they each have unique attributes and functionalities. Overall, Figure 4.2 provides a hierarchical overview of the INCO model.

The following Tables 4.1, 4.2, 4.3, 4.4, and 4.5 summarize the INCO relationships, detailing the concepts, properties (slots), cardinality, and slot values. Each column is described as follows:

- **Concepts:** Represented in the first column, these are the core elements of the ontology.
- **Properties/Slots:** Listed in the second column, they describe the attributes or relationships

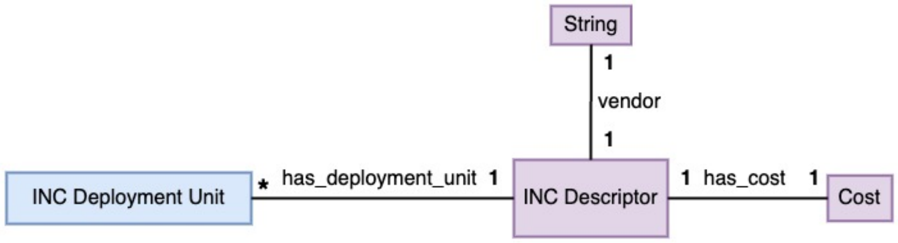


Figure 4.3: Modeling INCO.

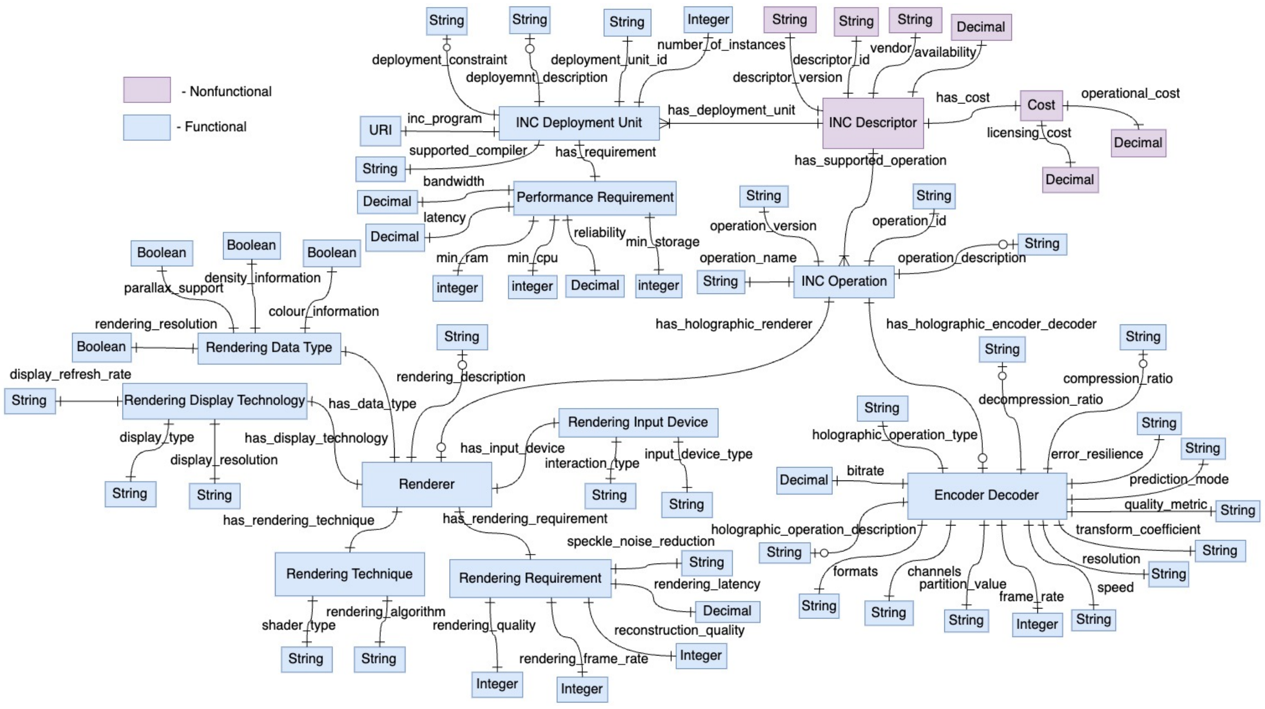


Figure 4.4: INCO Model Representation.

associated with each concept.

- **Cardinality:** Indicated in the third column, it specifies slot facets such as "one to many (1...*)", "exactly one (1)", or "optional (0...1)". For example, the property `has_deployment_unit` in `INC_Descriptor` has a cardinality of "1...*", meaning one to many `INC_Deployment_Unit` instances can be associated (see Table Table 4.1).

- **Values:** Shown in the fourth column, these represent the slot values, which can be either a data type (e.g., String, Integer) or an object type (e.g., another concept). For example, the `vendor` field in `INC_Descriptor` is a data property with a "String" value and a cardinality of "1" (see Table 4.1). Similarly, the property `has_cost` links an `INC_Descriptor` instance to a `Cost` instance with a cardinality of "1".

Table 4.1 details the relationships of the `INC_Descriptor`, `Cost`, and `INC_Operation` concepts. Figure 4.3 illustrates the modeling of the concepts and their associated properties, establishing the relationships detailed in Table 4.1. Table 4.2 describes the relationships of the `INC_Deployment_Unit` and `Performance_Requirement` concepts. For instance, `INC_Deployment_Unit` and `Performance_Requirement` are semantically connected through the relation `has_requirement` (see Table 4.2). Table 4.3 focuses on the relationships of the `Encoder_Decoder` concept, while Table 4.4 addresses the `Renderer` concept. Lastly, Table 4.5 captures the relationships of all subclasses of the `Renderer` concept, including `Rendering_Input_Device`, `Rendering_Requirement`, `Rendering_Technique`, `Rendering_Display_Technology`, and `Rendering_Data_Type`. These tables collectively describe the semantic relationships within the INCO ontology-based model, highlighting its structure and connections.

INCO, as illustrated in Figure 4.4, serves as our ontology-based model for the semantic representation of INC components. It is structured into two primary segments: functional properties (blue boxes) and non-functional properties (purple boxes). This figure provides a comprehensive overview of how the INCO model is designed, highlighting the concepts, their associated properties, and the semantic relationships that connect them.

The organization of these concepts facilitates detailed descriptions, effective query construction, and efficient retrieval of INC components. Figure 4.4 employs an entity-relationship diagram

Table 4.1: Summary of Relationships for the `Descriptor`, `Cost`, and `Operation` Classes in INCO.

Concepts	**Properties/Slots**	**Cardinality**	**Values**
INC_Descriptor	descriptor_version	1	String
	descriptor_id	1	String
	vendor	1	String
	availability	1	Decimal
	has_cost	1	Cost
	has_deployment_unit	1...*	INC_Deployment_Unit
	has_supported_operation	1...*	INC_Operation
Cost	operational_cost	1	Decimal
	licensing_cost	1	Decimal
INC_Operation	operation_name	1	String
	operation_version	1	String
	operation_description	0...1	String
	operation_id	1	String
	has_holographic_renderer	0...1	Renderer
	has_holographic_encoder_decoder	0...1	Encoder_Decoder

annotation, showcasing all concepts, properties (slots), slot facets, and their relationships. The final design of INCO is based on this conceptual framework, which forms the foundation for its implementation. These elements and their relationships are discussed in detail in the following subsections, emphasizing their roles in achieving precise semantic representation and supporting query-building mechanisms.

4.3.2 Functional Properties of INCO

As mentioned earlier, functional properties define the capabilities and semantic relationships of an INC component. They refer to the formal attributes specifying what an INC component can perform. These functional properties include ten key concepts, three of which are discussed in this section. Detailed explanations of all the properties associated with these concepts are presented in

Table 4.2: Summary of Relationships for the `Deployment Unit` and `Performance Requirement` Classes in INCO.

Concepts	Properties/Slots	Cardinality	Values
INC_Deployment_Unit	number_of_instances	1	Integer
	deployment_unit_id	1	String
	deployment_description	0..1	String
	deployment_constraint	0..1	String
	inc_program	1	URI
	supported_compiler	1	String
	has_requirement	1	Performance_Requirement
Performance_Requirement	bandwidth	1	Decimal
	latency	1	Decimal
	min_ram	1	Integer
	min_cpu	1	Integer
	reliability	1	Decimal
	min_storage	1	Integer

Tables 4.6, 4.7, and 4.8. The following section elaborates on this three selected concepts in detail.

- **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.

In Tables 4.6, 4.7 and 4.8 the Properties/Slots column lists the data type properties associated with the concepts `INC_Operation`, `INC_Deployment_Unit`, `Performance_Requirement`

Table 4.3: Summary of Relationships for the `Encoder_Decoder` Class in INCO.

Concepts	**Properties/Slots**	**Cardinality**	**Values**
Encoder_Decoder	holographic_operation_type	1	String
	bitrate	1	Decimal
	holographic_operation_description	0..1	String
	formats	1	String
	channels	1	String
	partion_value	1	String
	frame_rate	1	Integer
	speed	1	String
	resolution	1	String
	transform_coeffient	1	String
	quality_metric	1	String
	prediction_mode	1	String
	error_resilience	1	String
	compression_ratio	0...1	String
	decompression_ratio	0...1	String

Table 4.4: Summary of Relationships for the `Renderer` Class in INCO.

Concepts	**Properties/Slots**	**Cardinality**	**Values**
Renderer	rendering_description	0...1	String
	has_input_device	1	Rendering_Input_Device
	has_rendering_requirement	1	Rendering_Requirement
	has_rendering_technique	1	Rendering_Technique
	has_display_technology	1	Rendering_Display_Technology
	has_data_type	1	Rendering_Data_Type

in the ontology. The Description column provides definitions for each of these data type properties.

Table 4.5: Summary of Relationships for all the Subclasses of `Renderer` Class in INCO.

Concepts	**Properties/Slots**	**Cardinality**	**Values**
Rendering_Input_Device	input_device_type	1	String
	interaction_type	1	String
Rendering_Requirement	speckle_noise_reduction	1	String
	rendering_latency	1	Decimal
	reconstruction_quality	1	Integer
	rendering_frame_rate	1	Integer
	rendering_quality	1	Integer
Rendering_Technique	shader_type	1	String
	rendering_algorithm	1	String
Rendering_Display_Technology	diplay_resolution	1	String
	display_type	1	String
	display_refresh_rate	1	String
Rendering_Data_Type	colour_information	1	Boolean
	rendering_resolution	1	Boolean
	parallax_support	1	Boolean
	density_information	1	Boolean

Table 4.6: Properties of `INC_Operation.`

Properties/Slots	**Description**
operation_name	Name of the INC operation, e.g., ”Compression”, ”Decompression”.
operation_version	Version of the INC operation.
operation_description	Any textual description of the INC operation.
operation_id	Unique identifier for the INC operation.

4.3.3 Non-Functional Properties of INCO

Non-functional properties focus on the requirements essential for the proper functioning of INC components. These properties define what an INC component needs or requires to function effectively. They encompass two primary concepts, which are explained in detail along with their

Table 4.7: Properties of `INC_Deployment_Unit`.

Properties/Slots	Description
number_of_instances	Number of INC Component instances to be deployed, e.g., 5 instances.
deployment_unit_id	Unique identifier for the deployment unit, e.g., ”DU-101”.
deployment_description	Any textual description of the deployment unit’s purpose.
deployment_constraint	Any textual descriptions of restrictions or limits on deploying a component.
inc_program	URI of the INC component in the centralized repository, e.g., ”http://inc-programs/holographic-streaming”.
supported_compiler	Supported compiler for deploying the INC component, e.g., ”P4 compiler”.

Table 4.8: Properties of `Performance_Requirement`.

Properties/Slots	Description
bandwidth	Required network bandwidth for processing and streaming holographic data, e.g., 500 Mbps.
latency	Maximum acceptable delay for processing and streaming holographic data, e.g., 150 ms.
min_ram	Minimum RAM required for holographic data storage and processing, e.g., 8 GB.
min_cpu	Minimum CPU cores needed for task execution, e.g., 4 cores.
reliability	Consistency of operation over time, e.g., 90% reliable for 100 continuous hours.
min_storage	Minimum storage capacity required, e.g., 200 GB.

associated properties in Tables 4.9 and 4.10. These two concepts are defined and discussed in detail as follows:

- **INC_Descriptor**: Defines the general characteristics of INCO components and serves as the central element of the INCO model. The INC_Descriptor is categorized into four specialized subclasses: INC_Operation, INC_Deployment_Unit, Performance_Requirement, Cost. 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.

Table 4.9: Properties of `INC_Descriptor`.

Properties/Slots	**Description**
descriptor_version	Specifies the version of the INC Descriptor, e.g., ”Descriptor 1.0”.
descriptor_id	A unique identifier of INC Descriptor, e.g., ”Descriptor-001”.
vendor	Provides information about the INC vendor.
availability	Percentage of time the INC component instance is operational and accessible, e.g., 99.9%.

Table 4.10: Properties of `Cost`.

Properties/Slots	**Description**
operational_cost	The ongoing cost to run or maintain an INC Component, calculated based on resource usage, e.g., $0.05 per GB processed.
licensing_cost	The licensing fee for using an INC component instance over a specific period, e.g., $500 per month.

4.3.4 INCO-Holographic-Type Communication

As discussed in Section 4.3.3, both the Functional and Non-functional properties of INCO encompass core concepts essential for INC components, irrespective of the application domain. For this work, we focus on Holographic-Type Communication as the illustrative application domain. The previously discussed `INC_Operation` of INCO has been extended to include Holographic-Type Communication-specific classes and properties, such as `Encoder_Decoder` and `Renderer`. All related properties and concepts are treated as functional in this context. For the HTP use case, we have identified four critical INC components: Encoder, Decoder, Transcoder, and Renderer.

- **Encoder_Decoder**: This class encompasses all aspects of encoding and decoding holographic data. It is associated with the concept that includes details related to compression and decompression for efficient holographic data transmission and processing, such as supported `resolution`, `compression_ratio`, and `decompression_ratio`, among others.

- **Renderer**: This class encompasses all aspects of rendering holographic data into visual output. It includes one optional property that provides a textual description of the holographic data rendering process. To ensure a detailed and precise representation, this class is further divided into five specialized subclasses: `Rendering_Input_Device`, `Rendering_Requirement`, `Rendering_Technique`, `Rendering_Display_Technology` and `Rendering_Data_Type`.
- **Rendering_Input_Device**: Represents information related to rendering input devices, including device type and interaction methods.
- **Rendering_Requirement**: Defines essential requirements for rendering, such as `rendering quality`, `reconstruction quality`, `speckle noise reduction` etc.
- **Rendering_Technique**: Specifies the rendering methods used for holographic content, such as the `shader type` and `rendering algorithm` used for rendering.
- **Rendering_Display_Technology**: Refers to the display technologies used to project holographic content, including properties like `display type`, `display resolution` etc.
- **Rendering_Data_Type**: Indicates whether specific attributes are supported for rendering holographic data, such as `colour information`, `density information` etc.

4.4 INC Component Discovery

This section discusses the discovery process of INC components. A novel discovery process based on INCO is proposed in this thesis. The INC component discovery process consists of three main steps: user request formulation, query building, and semantic matchmaker. The process begins with the user (e.g., a network operator) formulating their INC component requests in terms of

Table 4.11: Properties of `Encoder_Decoder` for Holographic Operations.

Properties/Slots	**Description**
holographic_operation_type	Type of operation, can be any encoding or decoding standard such as ”Encoder” or ”Decoder”.
holographic_operation_description	Refers to any textual description of a holographic operation such as ”HEVC” or ”AV1”.
formats	Data format used, e.g., ”PLY”.
bitrate	Encoding/decoding rate, e.g., ”10 Mbps”.
channels	Number of input and output holographic data streams, e.g., Input: 3, Output: 2.
partition_value	Holographic data partitioning method, e.g., ”16x16 block”.
frame_rate	Frames processed per second, e.g., ”60 FPS”.
speed	Speed of the operation, e.g., ”Real-time encoding”.
resolution	Supported resolution, e.g., ”1920x1080 pixels”.
transform_coefficient	Contribution of transform coefficients to frequency components, e.g., ”0.85”.
quality_metric	Measure of output quality, often compared to the original.
prediction_mode	Technique for predicting holographic data to reduce redundancy, e.g., inter-frame prediction.
error_resilience	Level of error resilience supported during encoding/decoding, e.g., ”Error Concealment”.
compression_ratio	Holographic data compression ratio, e.g., ”4:1”.
decompression_ratio	Holographic data decompression ratio, e.g., ”4:1”.

Table 4.12: Properties of `Rendering_Input_Device.`

Properties/Slots	**Description**
input_device_type	Type of input device used for rendering. Specifies the type of input device for rendering.
interaction_type	Refers to the method of interaction.

requirements and preferences. Next, the query building process and the semantic matchmaking algorithm are discussed. Lastly, we talked about the discovery flow process of the most relevant INC components based on the user preferences. Each step in this process is detailed in the following

Table 4.13: Properties of `Rendering_Technique`.

Properties/Slots	**Description**
shader_type	Type of shader used in rendering, example: ”Vertex Shader”.
rendering_algorithm	Algorithm used to generate the rendered hologram, example: ”Angular Spectrum Method”.

Table 4.14: Properties of `Rendering_Requirement`.

Properties/Slots	**Description**
speckle_noise_reduction	Refers to the technique used for speckle noise reduction during rendering.
rendering_latency	Refers to the maximum allowable delay during the rendering process.
reconstruction_quality	Refers to the quality of the reconstructed hologram.
rendering_frame_rate	Refers to the frames per second needed for rendering.
rendering_quality	Refers to the quality of the rendering process.

Table 4.15: Properties of `Rendering_Display_Technology`.

Properties/Slots	**Description**
display_resolution	Refers to the resolution of the display.
display_type	Refers to the type of display used.
display_refresh_rate	Refers to the display refresh rate.

Table 4.16: Properties of `Rendering_Data_Type`.

Properties/Slots	**Description**
colour_information	Indicates if the data includes color details for accurate object rendering.
rendering_resolution	Specifies the resolution details included in the holographic data.
parallax_support	Indicates if the holographic data supports parallax effects.
density_information	Specifies if the data includes density details for realistic volume-based rendering.

sections.

4.4.1 User Request

The user's request for the required INC component is built by the Query Processing Agent. Users can define their preferences among functional and/or non-functional requirements to identify the most appropriate INC components. These preferences are classified into three levels: mandatory, high, and optional. Users specify requirement fields and values based on their preferences, which are then used to rank and filter INC components.

- **Mandatory preferences (non-negotiable)** : All mandatory preferences must be fulfilled. INC components that do not satisfy these preferences are discarded.
- **High preference (important but flexible)** : These preferences are optional but are given a higher priority when ranking components. They represent highly preferred specifications.
- **Optional preferences (desirable but not essential)** : These preferences are considered less important and have lower priority compared to high preferences.

Equation 1 formally represents a user request, which includes all relevant functional and non-functional properties 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).

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). Table 4.17 provides a summary of the symbols used in the user request equation and their corresponding descriptions.

An example of an user request, is as follows :

UR_i = {(M, INC_Operation, operation_name, exactly, Firewall),(H, Performance_Requirement, latency, max, 110),(O, Cost, licensing_cost, max, 500)};

Table 4.17: Symbols and Their Descriptions for User Request Representation.

Symbol	**Description**
UR_i	User i
$Pr_{i,j}$	User i's Mandatory (M), High (H), or Optional (O) Preferences
$Req\_f_{i,j}$	Requirement fields representing all classes of INCO
$Req_{i,j}$	Requirements corresponding to all data type properties of INCO
$Condition_{i,j}$	Conditions such as max, min, or exactly associated with $Val_{i,j}$
$Val_{i,j}$	Value specified by User i for their requirement

4.4.2 Building Query

The Query Processing Agent is responsible for converting the user request into a query and a preference list. The user submits their request based on preferences, which the QPA processes to generate the corresponding SQWRL query and preference list. This process can be expressed as:

User Request $\rightarrow$ SQWRL Query + Preference List.

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 SQWRL query (QR_i) / Q_i and a preference list ($pref\_list_i$) are formally represented in

Equation 2 and 4, respectively.

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

It is important to note that the Requirement Field (e.g., Req_f) may appear multiple times if the user selects more than one INC operation (see Equation 2). Mandatory preferences are used to keep relevant INC components, discarding those that do not meet the specified mandatory conditions. Retrieved INC components are ranked based on high and optional preferences. Here, in Equation 2 Rel defines the structure of the ontology, helping build the query for each user request such that:

$$Rel = INC\_Descriptor(?d) \wedge has\_cost(?d, ?c) \wedge Cost(?c) \wedge supported\_operation(?d, ?o) \wedge INC\_Operation(?o) \wedge has\_deployment\_unit(?d, ?du) \wedge INC\_Deployment\_Unit(?du) \wedge has\_requirement(?du, ?p) \wedge Performance\_Requirement(?p) \wedge has\_holographic\_encoder\_decoder(?o, ?ed) \wedge Encoder\_Decoder(?ed) \wedge has\_holographic\_renderer(?o, ?r) \wedge Renderer(?r) \wedge has\_input\_device(?r, ?id) \wedge Rendering\_Input\_Device(?id) \wedge has\_rendering\_requirement(?r, ?rr) \wedge Rendering\_Requirement(?rr) \wedge has\_rendering\_technique(?r, ?rt) \wedge rendering\_technique(?rt) \wedge has\_display\_technology(?r, ?rdt) \wedge Rendering\_Display\_Technology(?rdt) \wedge has\_data\_type(?r, ?dt) \wedge Rendering\_Data\_Type(?dt) \wedge inc\_program(?du, ?inc) \tag{3}$$

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

The reader should note that only high and optional preferences are considered for the preference list (see Equation 4).

An example of a query and pref_list are as follows :

QR_i = INC_Descriptor(?d)∧has_cost(?d,?c)∧Cost(c)∧supported_operation(?d,?o)∧INC_Operation(?o)∧has_deployment_unit(?d,?du)∧INC_Deployment_Unit(?du)∧has_requirement(?du,?p)∧Perform ance_Requirement(?p)$\wedge$inc_program(?du,?inc)$\wedge$operation_name(?o,'Firewall"^^rdf:PlainLiteral)$\wedge$late ncy(?p,?l)$\wedge$licensing_cost(?c,?lc) $\rightarrow$ sqwrl:select(?inc,?l,?lc);

$pref\_list_i$ = (H, latency, max, 110), (O, licensing_cost, max, 500).

4.4.3 INC Component Matchmaking

Algorithm Algorithm 1 outlines the semantic matchmaking process used to retrieve a relevant set of INC components, `Retrieved_inc_list`. It utilizes INCO to match INC components in the repository `Repo` with user requests and their preferences. To ensure proper matching and ranking of relevant INC components, both QR_i and $pref\_list_i$ must not be null (Line 2).

The algorithm first applies the `MatchAll` function to each INC component in `Repo` (Line 4). This function validates all mandatory preferences. Mandatory preferences are strictly enforced, ensuring that only components that meet all such preferences are included in `Retrieved_inc_list`. The `MatchAll` function checks for exact matches to all mandatory preferences (Lines 5 and 6). If no matches are found, `Retrieved_inc_list` remains null.

Once the mandatory preferences are enforced, the algorithm ranks the retrieved components based on high and optional preferences using the `MatchSome_Put_Priority_Value` function (Lines 10-12). This function assigns priority values to the retrieved INC components, allowing them to be ranked accordingly. Finally, the `RankByPriorityValue` function orders the `Retrieved_inc_list` based on the assigned priority values (Line 13). This ensures that the most relevant INC components appear first considering user requested preferences.

4.4.4 Functional Entities Interactions

This section outlines the flow of the INC component discovery process for holographic streaming requests within the proposed architecture, as illustrated in Figure 4.5.

This section illustrates the interaction among the functional entities in INC component discovery, using the holographic streaming use case. The proposed system operates in an environment where INC providers design and develop INC components, which are subsequently published in a centralized repository. This repository enables network operator to leverage these components in

Algorithm 1 Semantic Matchmaking

```
procedure INC_MATCHMAKER($QR_i$, $pref\_list_i$, $Repo$)
    if $QR_i \neq null$ and $pref\_list_i \neq null$ then
        $Retrieved\_inc\_list \leftarrow []$
        for each $INC$ in $Repo$ do
            if MATCHALL($INC, QR_i$) then
                APPEND($INC, Retrieved\_inc\_list$)
            end if
        end for
        if $Retrieved\_inc\_list \neq null$ then
            for each $INC$ in $Retrieved\_inc\_list$ do
                MATCHSOME_PUT_PRIORITY_VALUE($INC, pref\_list_i$)
            end for
            RANKBYPRIORITYVALUE($Retrieved\_inc\_list$)
        end if
        return $Retrieved\_inc\_list$
    end if
end procedure
```

INC-based Service Function Chains (SFCs). Upon publication, the INC descriptor is submitted to the INCO module for parsing and semantic enrichment. When a user, such as a network operator, initiates a holographic streaming request and seeks to compose an INC-based SFC for seamless hologram streaming, the system attempts to match the user's requirements with suitable INC components from the centralized repository, based on their specified preferences.

Relevant INC Components discovery for holographic streaming request, we envision the following sequence of actions:

The process begins when INC components are published in a centralized repository and submitted to the INCO module for parsing and semantic enrichment (Step 1). The user initiates a holographic streaming session request. Once the session is started, it submits a request to the orchestrator (Step 2). Acting as a centralized manager, the orchestrator receives the request and ensures the session runs smoothly. It sends a request (for INC components, e.g, encoder, decoder, transcoder and rendere) to the Query Processing Agent including all user preferences and requirements (Step 3). The Query Processing Agent then converts the request into a SQWRL query, and a preference list, and forwards it to the Semantic Matchmaking Module (Step 4). The Semantic Matchmaking Module acknowledges receipt of the query (Step 5). Using the ontology model in the INCO Module, the Semantic Matchmaking Module searches for and ranks the INC components

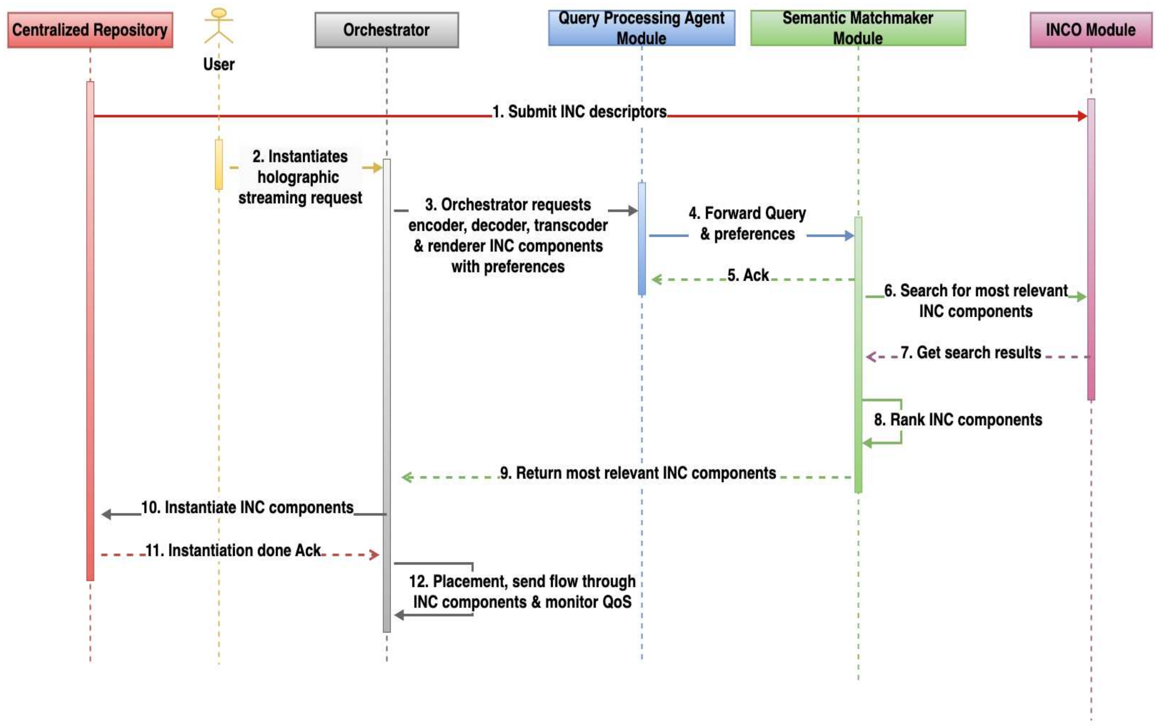

Centralized Repository
User
Orchestrator
Query Processing Agent Module
Semantic Matchmaker Module
INCO Module
1. Submit INC descriptors
2. Instantiates holographic streaming request
3. Orchestrator requests encoder, decoder, transcoder & renderer INC components with preferences
4. Forward Query & preferences
5. Ack
6. Search for most relevant INC components
7. Get search results
8. Rank INC components
9. Return most relevant INC components
10. Instantiate INC components
11. Instantiation done Ack
12. Placement, send flow through INC components & monitor QoS

Figure 4.5: Sequence of Interactions among the Functional Entities for INC Components discovery.

in the centralized repository based on the user's requirements and preferences (Steps 6, 7, and 8). Next, the Semantic Matchmaking Module returns the Uniform Resource Identifiers (URIs) of the most relevant INC component(s) to the orchestrator (Step 9). The orchestrator proceeds to instantiate the selected INC components (Step 10). Upon successful instantiation, an acknowledgment is sent back to the orchestrator (Step 11). Finally, the orchestrator manages the placement of these components, initiates the flow through them, and monitors Quality of Service (QoS) to ensure an optimal holographic streaming experience for the user (Step 12). Figure 4.5 provides a visual representation of this sequence, detailing the steps involved in INC component discovery, ranking, and orchestration to fulfill the holographic streaming request.

4.5 Conclusion

In this chapter, we presented the proposed architecture for an ontology-based INC Component Description and Discovery Model, focusing on the holographic streaming use case. A detailed overview of the architecture's design was provided, emphasizing its modular structure. We also described the ontology-based approach for INC component description, highlighting the definition of classes, properties, and relationships to semantically represent both functional and non-functional requirements. Additionally, the INC discovery process was detailed, encompassing user requests, query generation, and the semantic matchmaking algorithm to retrieve and rank INC components effectively. Finally, we introduced the flow of interactions among functional entities, demonstrating how the architecture seamlessly supports the discovery of INC components for holographic streaming. This architecture was evaluated against the predefined requirements derived from the use case, proving its ability to fulfill all those requirements. In the next chapter, we will discuss the implementation of the proposed architecture, including a proof-of-concept validation. Furthermore, we will analyze the experimental results and the insights gained from the evaluation.

Chapter 5

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 implementation details, experiments conducted to evaluate our approach, providing details on the experimental setup, metrics used for comparative analysis, and the performance measurements obtained.

5.1 System Implementation

In this section we will talk about the implementation details of the INC component description and discovery model. Then we talked detailed about how INCO's classes, subclasses, data and object properties of the ontology were implemented. Then,we describe the experiment setup and the performance metrics considered to evaluate it. Finally, we discuss the results and analysis of the experiments. A concluding section summarizes this chapter at the end.

5.1.1 Implementation

INCO was implemented using Protégé 5.6.3 modeling tool, 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

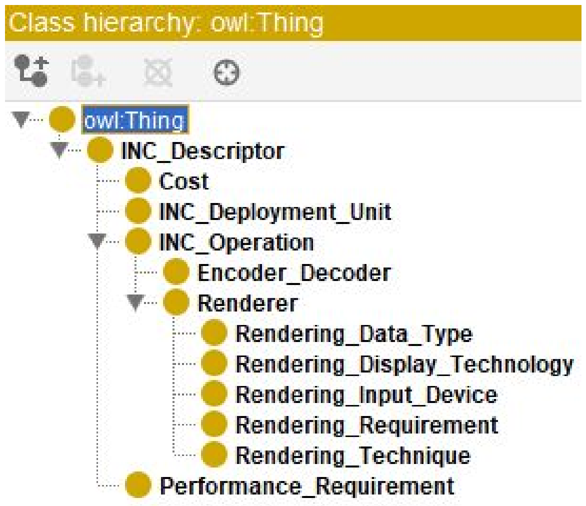
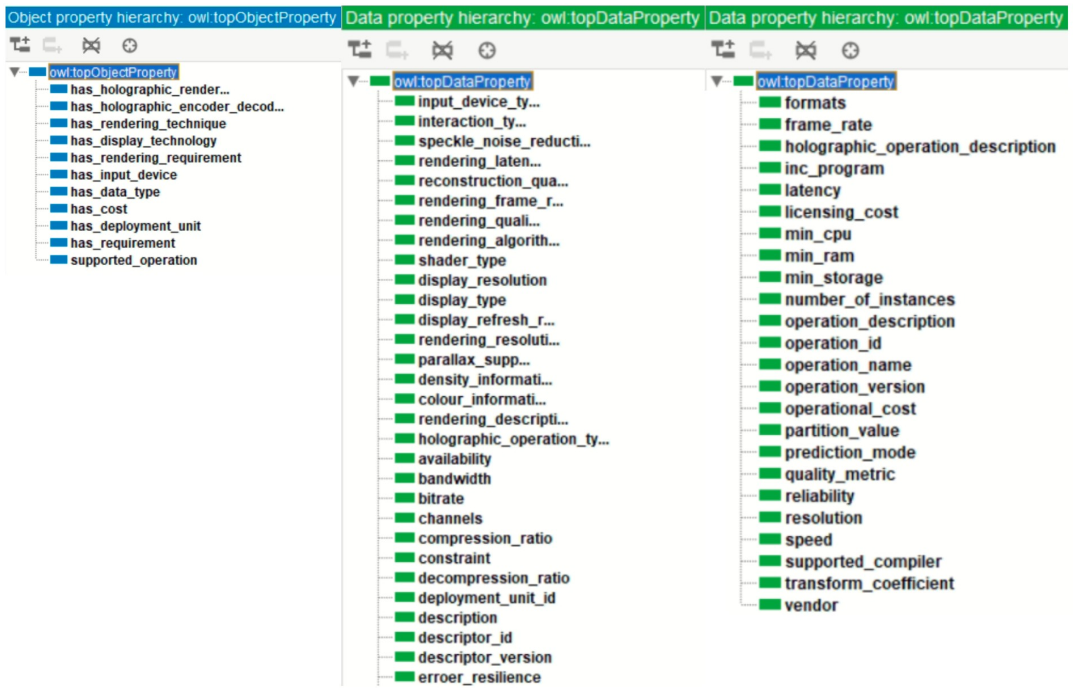


Figure 5.1: INCO Classes and Subclasses - Protégé.

relevant INC components based on user requirements and preferences.

5.1.2 INCO Classes and Sub-classes

The implementation process of INCO began with the creation of its classes and subclasses. Figure 5.1 illustrates the resulting hierarchical tree implemented using Protégé. A top-down development approach was adopted, starting with the definition of the most general concepts in the domain and progressively specializing them. For instance, the implementation began with creating broad classes, such as INC_Descriptor etc. Then we further specialized with the INC_Operation class branching into specific subclasses, such as Encoder_Decoder and Renderer. We can further created the rest of the classes and subclasses. Additional classes and subclasses were developed following a similar methodology (see Figure 5.1). Once the class hierarchy was completed in Protégé, the focus shifted to defining the data and object properties of INCO. Each class and subclass in the ontology uses capitalized class names for clarity and consistency. For concept names comprising multiple words (e.g., INC Deployment Unit), underscores were used to delimit the words, resulting in naming conventions such as, `INC_Deployment_Unit`.


Object property hierarchy: owl:topObjectProperty

- owl:topObjectProperty
  - has_holographic_render...
  - has_holographic_encoder_decod...
  - has_rendering_technique
  - has_display_technology
  - has_rendering_requirement
  - has_input_device
  - has_data_type
  - has_cost
  - has_deployment_unit
  - has_requirement
  - supported_operation

Data property hierarchy: owl:topDataProperty

- owl:topDataProperty
  - input_device_ty...
  - interaction_ty...
  - speckle_noise_reducti...
  - rendering_laten...
  - reconstruction_qua...
  - rendering_frame_r...
  - rendering_quali...
  - rendering_algorith...
  - shader_type
  - display_resolution
  - display_type
  - display_refresh_r...
  - rendering_resoluti...
  - parallax_supp...
  - density_informati...
  - colour_informati...
  - rendering_descripti...
  - holographic_operation_ty...
  - availability
  - bandwidth
  - bitrate
  - channels
  - compression_ratio
  - constraint
  - decompression_ratio
  - deployment_unit_id
  - description
  - descriptor_id
  - descriptor_version
  - erroer_resilience

Data property hierarchy: owl:topDataProperty

- owl:topDataProperty
  - formats
  - frame_rate
  - holographic_operation_description
  - inc_program
  - latency
  - licensing_cost
  - min_cpu
  - min_ram
  - min_storage
  - number_of_instances
  - operation_description
  - operation_id
  - operation_name
  - operation_version
  - operational_cost
  - partition_value
  - prediction_mode
  - quality_metric
  - reliability
  - resolution
  - speed
  - supported_compiler
  - transform_coefficient
  - vendor

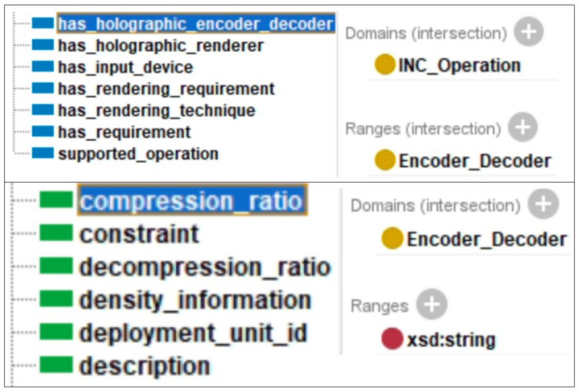
Figure 5.2: INCO Data and Object Properties - Protégé.

5.1.3 INCO Data and Object Properties

After implementing the ontology classes and subclasses, the next step was defining the data and object properties of INCO. Each class and subclass was enriched with its corresponding properties. Figure 5.2 illustrates the implemented data and object properties. For consistency, lower case was used for property names. Object properties were prefixed with `has_`, while data properties were named directly without any prefixes. The left side of the figure represents the object properties, and the right side displays the data properties.

In addition to the property definitions, relationships between ontology concepts were established by specifying domain and range for each property. The domain specifies the classes that can use the property, while the range defines the data type or class associated with the property. Figure 5.3 shows the process for defining domains and ranges. For example, the data property `compression_ratio` is shown with a range of `String` (slot type) and a domain of `Encoder_Decoder` (see Figure 5.3).

Once all properties and relationships were defined, cardinality facets were added to the slots. Cardinality defines constraints on the number of values a property can have. Figure 5.4 illustrates


has_holographic_encoder_decoder
has_holographic_renderer
has_input_device
has_rendering_requirement
has_rendering_technique
has_requirement
supported_operation

Domains (intersection)
INC_Operation

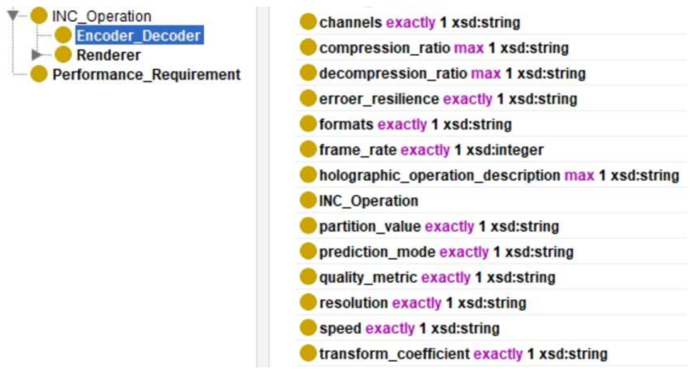
Ranges (intersection)
Encoder_Decoder

compression_ratio
constraint
decompression_ratio
density_information
deployment_unit_id
description

Domains (intersection)
Encoder_Decoder

Ranges
xsd:string

Figure 5.3: INCO Properties Range and Domain - Protégé.


INC_Operation
Encoder_Decoder
Renderer
Performance_Requirement

channels exactly 1 xsd:string
compression_ratio max 1 xsd:string
decompression_ratio max 1 xsd:string
erroer_resilience exactly 1 xsd:string
formats exactly 1 xsd:string
frame_rate exactly 1 xsd:integer
holographic_operation_description max 1 xsd:string
INC_Operation
partition_value exactly 1 xsd:string
prediction_mode exactly 1 xsd:string
quality_metric exactly 1 xsd:string
resolution exactly 1 xsd:string
speed exactly 1 xsd:string
transform_coefficient exactly 1 xsd:string

Figure 5.4: INCO Property Cardinality - Protégé.

the cardinality settings applied to the elements (objects and slots) of the relevant class. Cardinality was implemented using minimum (>=), maximum (<=), and exactly (=) restrictions as needed. With all properties, relationships, and cardinality facets finalized, the INCO ontology was completed. This resulted in a semantic, ontology-based description model ready for instance creation and experimental simulations.

5.2 Performance Measurement

In this section, we begin with a description of the experiment setup. Then, we present the considered performance metrics. Afterwards, the obtained results are presented and analysed. Finally, we conclude the section with a discussion and summary of the gained analysis from the obtained results.

5.2.1 Experiment Setup

In the experiment setup, we utilized sample queries to evaluate the performance of our Semantic Matchmaker in retrieving relevant INC components. These queries were categorized based on two parameters: query complexity (determined by the number of properties involved) and number of retrieved instances (the total count of instances returned by a query). The primary performance metric was the response time, measured in milliseconds.

The sample queries were designed to challenge the Semantic Matchmaker, as described in Section 4.4.3. For this purpose, SQWRL queries (Q) were created to evaluate the matchmaker's capability to handle multiple requirement effectively. These queries were run through the Semantic Matchmaking module to retrieve the relevant INC components, and the average response time was calculated for evaluation purposes.

As shown in Table 5.1, the two types of queries were designed to assess distinct aspects of the matchmaker's performance. The first type evaluated the matchmaker's ability to handle increasing query complexity, while the second type tested its ability to manage varying numbers of retrieved instances. By combining these two dimensions, we aimed to comprehensively evaluate the efficiency of the Semantic Matchmaker.

Table 5.1: Query Category.

Query (Q)	**Query Type**
$Q_1 - Q_6$	Different Query Complexity.
Q_7	Different Number of Retrieved Instances.

5.2.2 Performance Metrics

As previously mentioned, response time (measured in milliseconds) served as the primary performance metric in this evaluation. The measurement was conducted using two distinct approaches to assess the efficiency of the Semantic Matchmaker.

The first approach involved varying the query complexity while keeping the number of retrieved instances constant. Six queries ($Q1$ to $Q6$) were executed, with complexity increasing incrementally by adding more properties to each subsequent query (see Figures 5.5, 5.6, 5.7). 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.

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

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

In contrast, the second approach maintained constant query complexity while varying the number of retrieved instances by incrementally adding instances to the ontology. For this scenario, the query structure remained unchanged (see $Q7$ in Figure 5.7). $Q7$ exemplifies the second approach, where the query structure remains unchanged, but the number of retrieved instances varies as new instances are added.

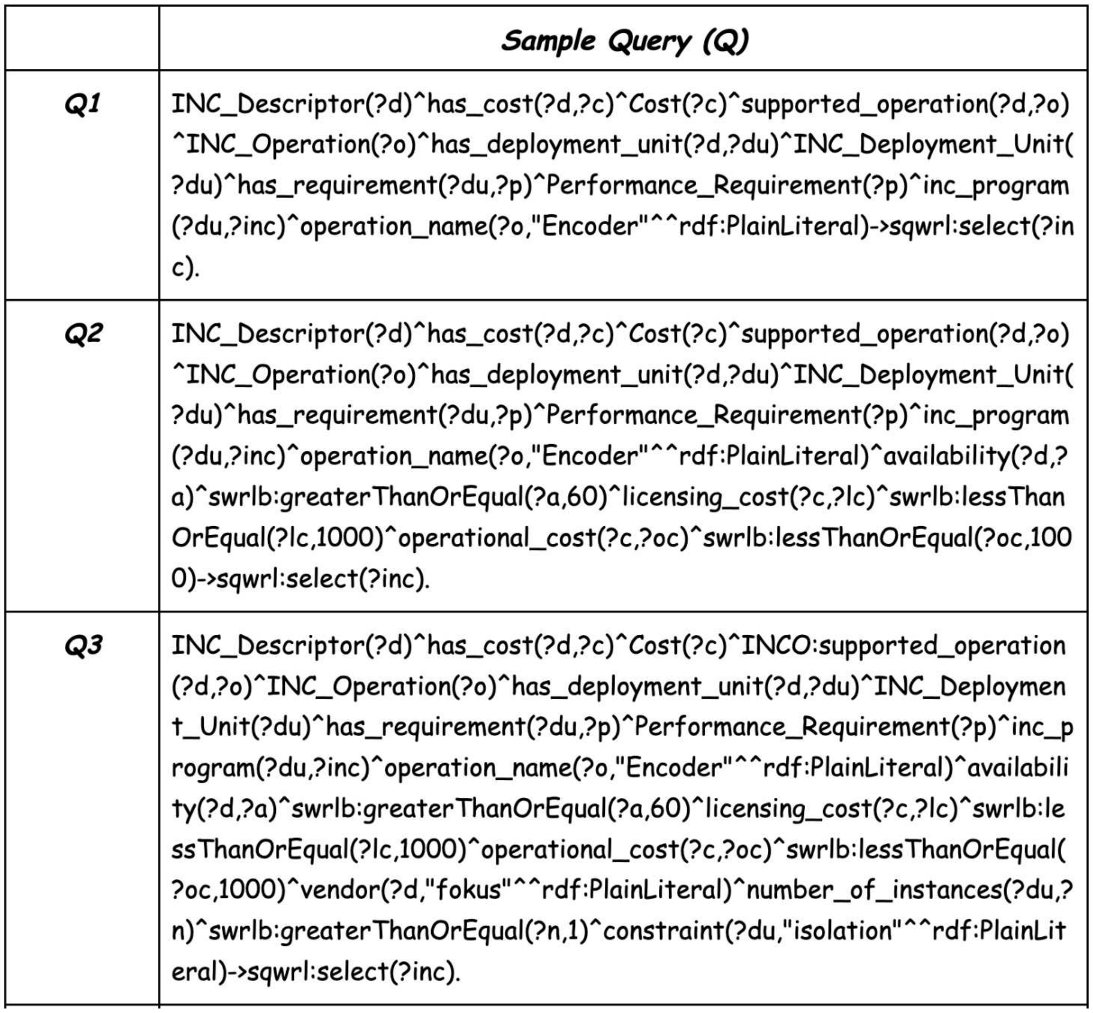
$$
\begin{aligned}
Q7 = {} & \text{INC\_Descriptor}(?d) \wedge \text{has\_cost}(?d, ?c) \wedge \text{Cost}(?c) \\
& \wedge \text{supported\_operation}(?d, ?o) \wedge \text{INC\_Operation}(?o) \\
& \wedge \text{has\_deployment\_unit}(?d, ?du) \\
& \wedge \text{INC\_Deployment\_Unit}(?du) \\
& \wedge \text{has\_requirement}(?du,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

Together, these two approaches enabled a comprehensive evaluation of the system's response time under varying conditions, offering valuable insights into its performance when handling increased query complexity and larger instance retrievals. This dual assessment effectively demonstrated how response time scales with both query complexity and the number of retrieved instances.

5.2.3 Results and Analysis

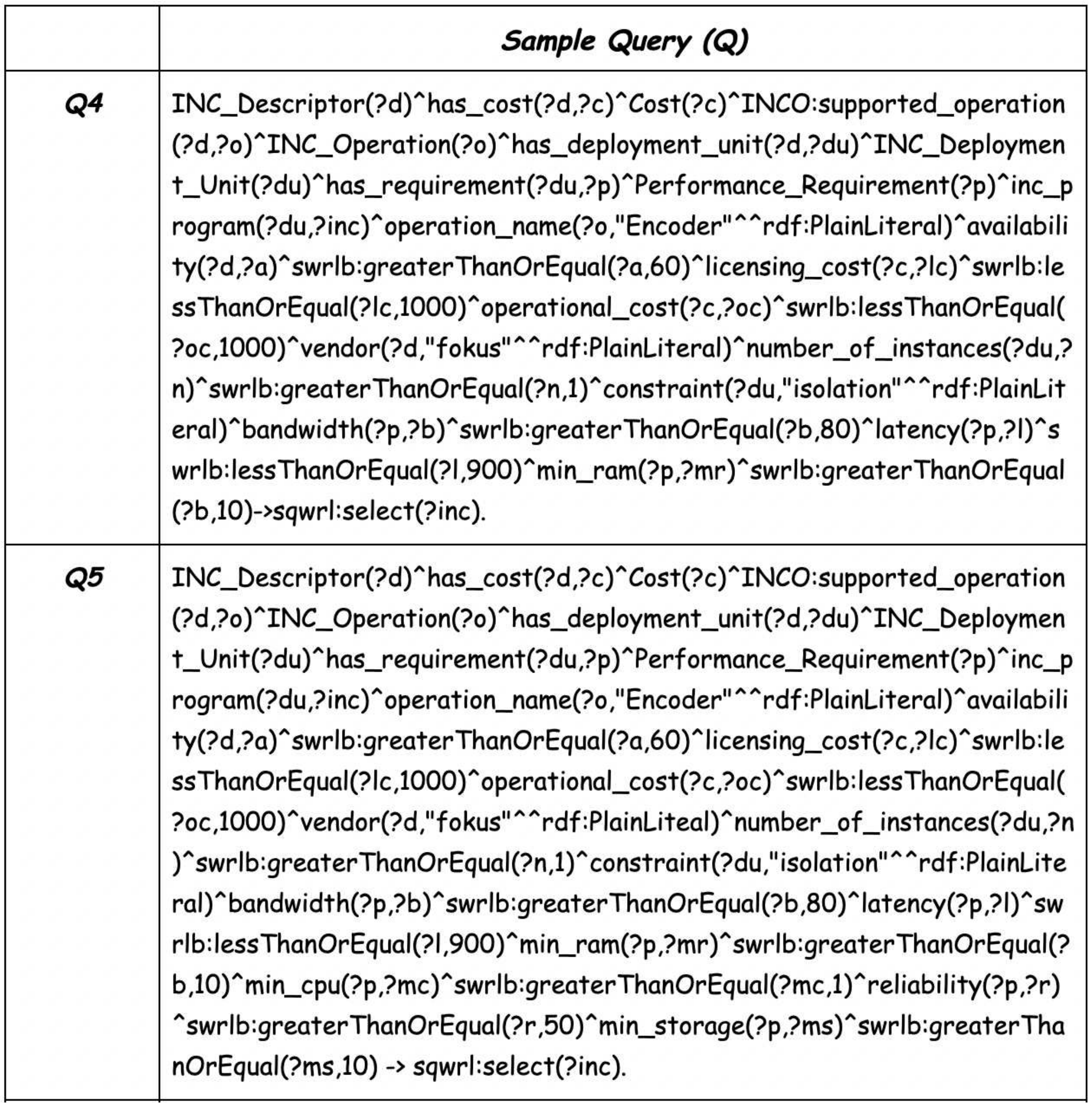
The experimental results in Fig. 5.8, show a clear increase in response time as query complexity escalates from $Q1$ to $Q6$, with the number of properties rising from 1 to 15. This trend reflects the additional computational overhead needed to evaluate more conditions and properties as the query complexity 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.

Conversely, Fig. 5.9 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



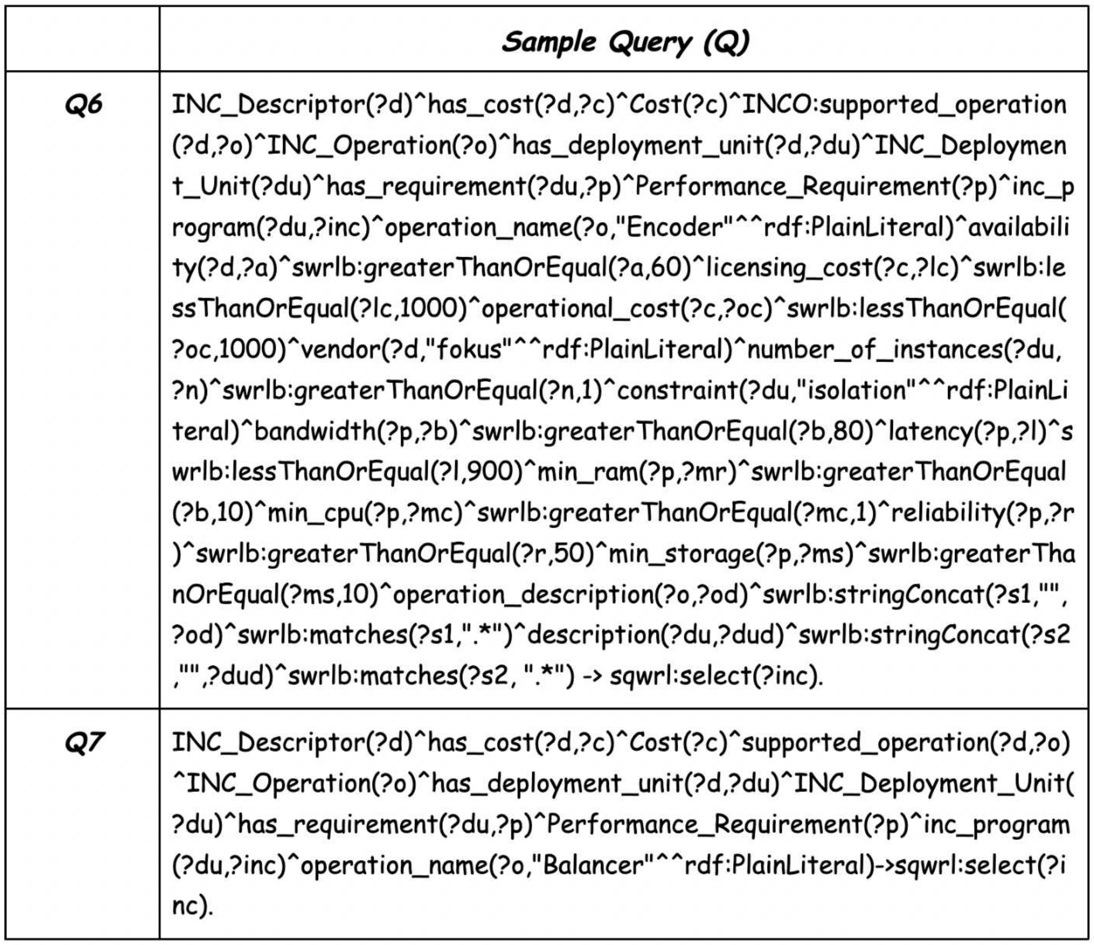
	Sample Query (Q)
Q1	INC_Descriptor(?d)^has_cost(?d,?c)^Cost(?c)^supported_operation(?d,?o)^INC_Operation(?o)^has_deployment_unit(?d,?du)^INC_Deployment_Unit(?du)^has_requirement(?du,?p)^Performance_Requirement(?p)^inc_program(?du,?inc)^operation_name(?o,"Encoder"^^rdf:PlainLiteral)->sqwrl:select(?inc).
Q2	INC_Descriptor(?d)^has_cost(?d,?c)^Cost(?c)^supported_operation(?d,?o)^INC_Operation(?o)^has_deployment_unit(?d,?du)^INC_Deployment_Unit(?du)^has_requirement(?du,?p)^Performance_Requirement(?p)^inc_program(?du,?inc)^operation_name(?o,"Encoder"^^rdf:PlainLiteral)^availability(?d,?a)^swrlb:greaterThanOrEqual(?a,60)^licensing_cost(?c,?lc)^swrlb:lessThanOrEqual(?lc,1000)^operational_cost(?c,?oc)^swrlb:lessThanOrEqual(?oc,1000)->sqwrl:select(?inc).
Q3	INC_Descriptor(?d)^has_cost(?d,?c)^Cost(?c)^INCO:supported_operation(?d,?o)^INC_Operation(?o)^has_deployment_unit(?d,?du)^INC_Deployment_Unit(?du)^has_requirement(?du,?p)^Performance_Requirement(?p)^inc_program(?du,?inc)^operation_name(?o,"Encoder"^^rdf:PlainLiteral)^availability(?d,?a)^swrlb:greaterThanOrEqual(?a,60)^licensing_cost(?c,?lc)^swrlb:lessThanOrEqual(?lc,1000)^operational_cost(?c,?oc)^swrlb:lessThanOrEqual(?oc,1000)^vendor(?d,"fokus"^^rdf:PlainLiteral)^number_of_instances(?du,?n)^swrlb:greaterThanOrEqual(?n,1)^constraint(?du,"isolation"^^rdf:PlainLiteral)->sqwrl:select(?inc).

Figure 5.5: Sample Query ($Q1 - Q3$)



	Sample Query (Q)
Q4	INC_Descriptor(?d)^has_cost(?d,?c)^Cost(?c)^INCO:supported_operation(?d,?o)^INC_Operation(?o)^has_deployment_unit(?d,?du)^INC_Deployment_Unit(?du)^has_requirement(?du,?p)^Performance_Requirement(?p)^inc_program(?du,?inc)^operation_name(?o,"Encoder"^^rdf:PlainLiteral)^availability(?d,?a)^swrlb:greaterThanOrEqual(?a,60)^licensing_cost(?c,?lc)^swrlb:lessThanOrEqual(?lc,1000)^operational_cost(?c,?oc)^swrlb:lessThanOrEqual(?oc,1000)^vendor(?d,"fokus"^^rdf:PlainLiteral)^number_of_instances(?du,?n)^swrlb:greaterThanOrEqual(?n,1)^constraint(?du,"isolation"^^rdf:PlainLiteral)^bandwidth(?p,?b)^swrlb:greaterThanOrEqual(?b,80)^latency(?p,?l)^swrlb:lessThanOrEqual(?l,900)^min_ram(?p,?mr)^swrlb:greaterThanOrEqual(?b,10)->sqwrl:select(?inc).
Q5	INC_Descriptor(?d)^has_cost(?d,?c)^Cost(?c)^INCO:supported_operation(?d,?o)^INC_Operation(?o)^has_deployment_unit(?d,?du)^INC_Deployment_Unit(?du)^has_requirement(?du,?p)^Performance_Requirement(?p)^inc_program(?du,?inc)^operation_name(?o,"Encoder"^^rdf:PlainLiteral)^availability(?d,?a)^swrlb:greaterThanOrEqual(?a,60)^licensing_cost(?c,?lc)^swrlb:lessThanOrEqual(?lc,1000)^operational_cost(?c,?oc)^swrlb:lessThanOrEqual(?oc,1000)^vendor(?d,"fokus"^^rdf:PlainLiteal)^number_of_instances(?du,?n)^swrlb:greaterThanOrEqual(?n,1)^constraint(?du,"isolation"^^rdf:PlainLiteral)^bandwidth(?p,?b)^swrlb:greaterThanOrEqual(?b,80)^latency(?p,?l)^swrlb:lessThanOrEqual(?l,900)^min_ram(?p,?mr)^swrlb:greaterThanOrEqual(?b,10)^min_cpu(?p,?mc)^swrlb:greaterThanOrEqual(?mc,1)^reliability(?p,?r)^swrlb:greaterThanOrEqual(?r,50)^min_storage(?p,?ms)^swrlb:greaterThanOrEqual(?ms,10) -> sqwrl:select(?inc).

Figure 5.6: Sample Query ($Q4 - Q5$)



	Sample Query (Q)
Q6	INC_Descriptor(?d)^has_cost(?d,?c)^Cost(?c)^INCO:supported_operation(?d,?o)^INC_Operation(?o)^has_deployment_unit(?d,?du)^INC_Deployment_Unit(?du)^has_requirement(?du,?p)^Performance_Requirement(?p)^inc_program(?du,?inc)^operation_name(?o,"Encoder"^^rdf:PlainLiteral)^availability(?d,?a)^swrlb:greaterThanOrEqual(?a,60)^licensing_cost(?c,?lc)^swrlb:lessThanOrEqual(?lc,1000)^operational_cost(?c,?oc)^swrlb:lessThanOrEqual(?oc,1000)^vendor(?d,"fokus"^^rdf:PlainLiteral)^number_of_instances(?du,?n)^swrlb:greaterThanOrEqual(?n,1)^constraint(?du,"isolation"^^rdf:PlainLiteral)^bandwidth(?p,?b)^swrlb:greaterThanOrEqual(?b,80)^latency(?p,?l)^swrlb:lessThanOrEqual(?l,900)^min_ram(?p,?mr)^swrlb:greaterThanOrEqual(?b,10)^min_cpu(?p,?mc)^swrlb:greaterThanOrEqual(?mc,1)^reliability(?p,?r)^swrlb:greaterThanOrEqual(?r,50)^min_storage(?p,?ms)^swrlb:greaterThanOrEqual(?ms,10)^operation_description(?o,?od)^swrlb:stringConcat(?s1,"",?od)^swrlb:matches(?s1,".*")^description(?du,?dud)^swrlb:stringConcat(?s2,"",?dud)^swrlb:matches(?s2, ".*") -> sqwrl:select(?inc).
Q7	INC_Descriptor(?d)^has_cost(?d,?c)^Cost(?c)^supported_operation(?d,?o)^INC_Operation(?o)^has_deployment_unit(?d,?du)^INC_Deployment_Unit(?du)^has_requirement(?du,?p)^Performance_Requirement(?p)^inc_program(?du,?inc)^operation_name(?o,"Balancer"^^rdf:PlainLiteral)->sqwrl:select(?inc).

Figure 5.7: Sample Query ($Q6 - Q7$)

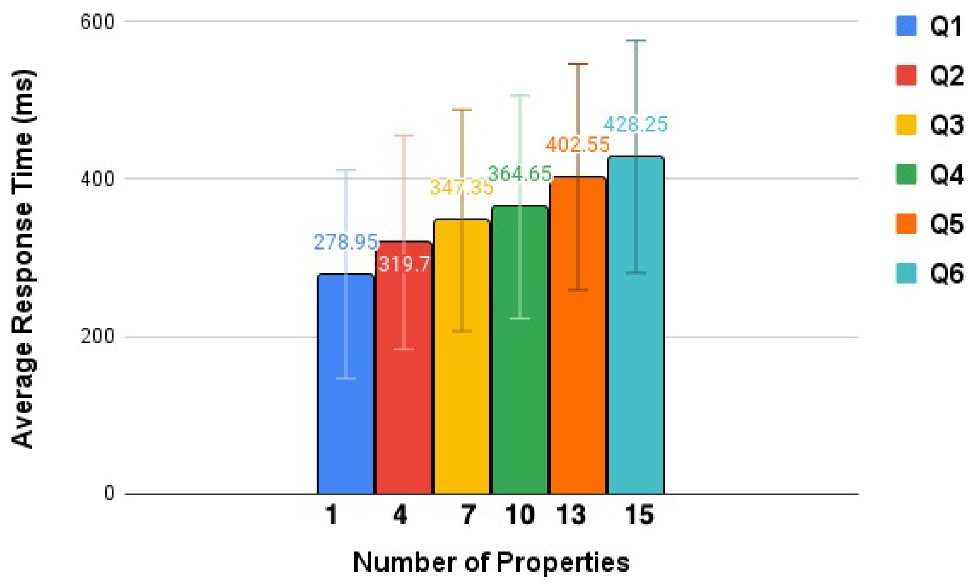


Figure 5.8: Average response time by different query comeplexity.

query complexity results, indicates that the system is well-optimized for scalable data retrieval.

5.3 Conclusion

In this chapter, we presented the proof-of-concept and the technologies used for its implementation. The process included developing the INCO ontology using Protégé, defining classes, subclasses, data and object properties, and incorporating cardinality facets to establish a comprehensive semantic framework. The semantic matchmaking algorithm was implemented in Python, utilizing SQWRL queries for the effective retrieval of INC components based on user requirements and preferences. We detailed the experimental setup and described the performance metrics used to evaluate the model. The performance was assessed using two categories of sample queries, focusing on varying query complexities and the number of retrieved instances. Finally, the obtained results were presented, discussed, and analyzed to demonstrate the model's effectiveness. In the next chapter, the thesis concludes with a summary of the work and an outline of potential directions for future research.

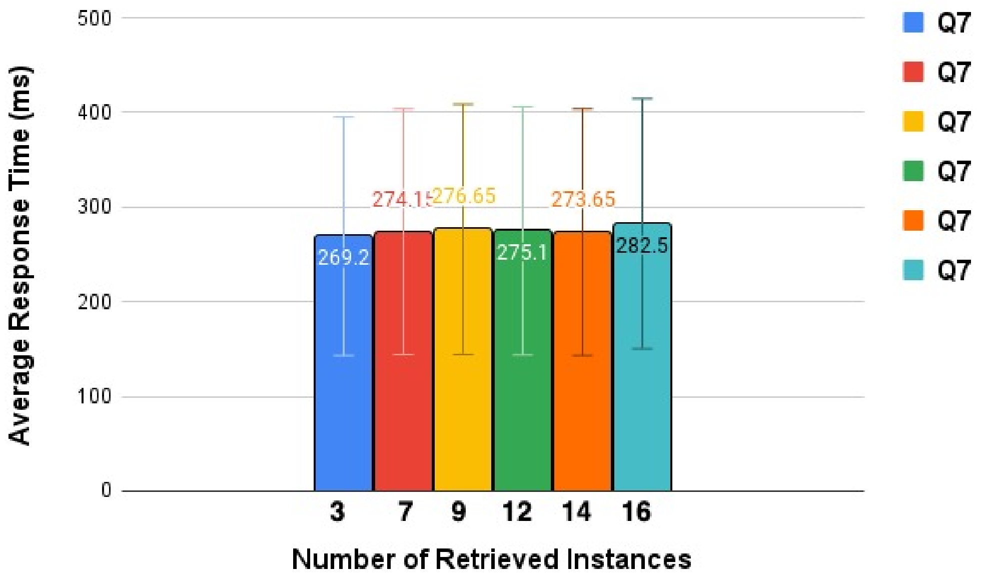


Figure 5.9: Average response time by number of retrieved instances.

Chapter 6

Conclusion

In this chapter, we provide a comprehensive overview of the key contributions of this thesis. Following that, we explore potential directions for future research.

6.1 Contributions Summary

Over the last few decades, the rise of advanced applications has significantly increased the demand for high-performance networking solutions. To meet these growing demands, technologies like In-Network Computing have emerged as vital enablers. However, enabling truly immersive holographic streaming applications such as, Holographic-Type Communication presents significant challenges to current networking infrastructures, including the need for innovative data compression techniques, optimized streaming methods, and better synchronization of concurrent data streams. Seamless holographic streaming sessions require network operators to obtain specific INC components, such as encoders, decoders, transcoders and renderers, tailored to their unique requirements. This necessity underscores the critical need for a comprehensive INC component description and discovery model. Despite its importance, no prior work has addressed the description and discovery of INC components, and the challenge of identifying relevant components based on user preferences remains largely unresolved.

To address these challenges, this thesis began by comprehensively analyzing the holographic streaming use case. This analysis informed the development of four key requirements for publishing, describing, building user requests and discovering INC components. After identifying these requirements, the thesis reviewed the state of the art, including semantic approaches to web service discovery and analised the significance of semantic based approach for automated and efficient service discovery. Then we studies existing methodologies for Virtual Network Function description and discovery. It was determined that none of the reviewed solutions fully satisfied the requirements derived from the holographic streaming use case.

Subsequently, the proposed architecture for an ontology-based semantic description and discovery model for INC components was presented. The architecture was evaluated to determine whether it fulfilled the predetermined requirements. The architectural modules were introduced, followed by an in-depth discussion of the INCO semantic description model and its role in facilitating accurate discovery.

In terms of meeting the requirements, the proposed architecture achieved all four objectives. To address the first requirement, a centralized repository was developed for publishing INC components by providers. For the second requirement, INCO, a comprehensive semantic description model, was designed to describe both functional and non-functional properties of INC components. The design process was detailed, including the principles and considerations that guided its development. The final two requirements—building user requests and discovering relevant INC components—were addressed through the Query Processing Agent and the Semantic Matchmaking Module. The Query Processing Agent translates user requests into SQWRL queries and preference lists, enabling precise communication with the Semantic Matchmaking Module. Users can define their requirements and preferences, such as mandatory, high, or optional criteria, to tailor the discovery and ranking process. The Semantic Matchmaking Module relies on the INCO model to retrieve and rank the most relevant INC components based on these preferences, ensuring accurate and efficient discovery.

To validate the proposed architecture, an exhaustive implementation process was carried out. This included designing and implementing the INCO ontology using Protégé, creating classes, subclasses, object and data properties, and establishing cardinality facets. The implementation of the semantic matchmaking algorithm was done in Python, with SQWRL queries facilitating effective retrieval of components. The performance evaluation utilized two categories of sample queries to analyze the system's response time and consistency. The results demonstrated that response time increased with query complexity but remained stable as the number of retrieved instances grew. This consistency underscores the scalability of the proposed framework for real-world applications.

This thesis presents a novel ontology-based model for the semantic description and discovery of INC components, addressing a critical gap in the literature. By leveraging ontologies, the proposed framework facilitates automated discovery of INC components, enhancing interoperability across diverse networks operator and INC providers. The incorporation of functional and non-functional properties into the semantic descriptions ensures a comprehensive and effective discovery process tailored to network operators' specific needs. The proof-of-concept implementation validates the efficacy of this approach, demonstrating its potential to streamline the discovery and deployment of INC components for advanced applications like holographic streaming.

6.2 Future Research Direction

While promising, future work should focus on scaling the ontology-based framework to accommodate the increasing complexity of INC components in large-scale network environments. Additionally, integrating advanced machine learning techniques could enhance the semantic matchmaking process, leading to more precise and context-aware recommendations. Additionally, real-world deployments must handle significantly higher query volumes and dynamic changes in the provisioning of INC components. A critical challenge in practical scenarios is that INC descriptors provided by INC providers may not be comprehensive enough or often be incomplete, making it difficult to construct a comprehensive ontology. To address this issue, further exploration of relevant literature and methodologies will be necessary. Furthermore, extending the ontology model to support real-time updates and dynamic adaptations will ensure responsiveness to evolving demands. Finally, collaborating with industry stakeholders to develop a standardized INC description framework could facilitate broader adoption and interoperability across various platforms and networks.

References

[1] L. Ding, P. Kolari, Z. Ding, and S. Avancha, “Using ontologies in the semantic web: A survey,” *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*, pp. 79–113, 2007.

[2] R. Li *et al.*, “Towards a new internet for the year 2030 and beyond,” in *Proc. 3rd Annu. ITU IMT-2020/5G Workshop Demo Day*, 2018, pp. 1–21.

[3] 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.

[4] 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.

[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] 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.

[7] 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.

[8] S. I. Kim and H. S. Kim, “Semantic ontology-based nfv service modeling,” in *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 2018, pp. 674–678.

[9] 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.

[10] A. Adala, N. Tabbane, and S. Tabbane, “A framework for automatic web service discovery based on semantics and nlp techniques,” *Advances in Multimedia*, vol. 2011, no. 1, p. 238683, 2011.

[11] M. M. Taye, “Understanding semantic web and ontologies: Theory and applications,” *arXiv preprint arXiv:1006.4567*, 2010.

[12] R. Li and Y. Miyake, “New services and capabilities for network 2030: Description, technical gap and performance target analysis,” *Doc. NET2030-O-027 in FOCUS GROUP ON TECHNOLOGIES FOR NETWORK*, vol. 2030, 2019.

[13] 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.

[14] D. Lopez-Perez, A. Garcia-Rodriguez, L. Galati-Giordano, M. Kasslin, and K. Doppler, “Ieee 802.11 be extremely high throughput: The next generation of wi-fi technology beyond 802.11 ax,” *IEEE Communications Magazine*, vol. 57, no. 9, pp. 113–119, 2019.

[15] A. Sapio, I. Abdelaziz, A. Aldilaijan, M. Canini, and P. Kalnis, “In-network computation is a dumb idea whose time has come,” in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, 2017, pp. 150–156.

[16] D. R. Ports and J. Nelson, “When should the network be the computer?” in *Proceedings of the Workshop on Hot Topics in Operating Systems*, 2019, pp. 209–215.

[17] 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, 2022.

[18] Y. Tokusashi, H. Matsutani, and N. Zilberman, "Lake: The power of in-network computing," in *2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. IEEE, 2018, pp. 1–8.

[19] I. Kunze, K. Wehrle, D. Trossen, M.-J. Montpetit, X. de Foy, D. Griffin, and M. Rio, "Use cases for in-network computing," *Internet Engineering Task Force, Internet-Draft draft-kunze-coin-industrial-use-cases-04, Nov. 2020, work in Progress*, 2022.

[20] 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.

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

[22] 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.

[23] 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

[24] 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.

[25] N. F. Noy, D. L. McGuinness *et al.*, "Ontology development 101: A guide to creating your first ontology," 2001.

[26] Z. Ma, G. Wang, B. Ren, and J. Wang, "Research on owl-based process model trasformation for interoperability," in *2006 International Technology and Innovation Conference (ITIC 2006).* IET, 2006, pp. 1546–1552.

[27] G. Klyne, "Resource description framework (rdf): Concepts and abstract syntax," *http://www. w3. org/TR/rdf-concepts/*, 2004.

[28] B. Dan, "Rdf vocabulary description language 1.0: Rdf schema," *http://www. w3. org/TR/rdf-schema/*, 2004.

[29] M. Dean, A. Schreiber, S. Bechofer, F. van Harmelen, J. Hendler, I. Horrocks, D. MacGuinness, P. Patel-Schneider, and L. A. Stein, "Owl web ontology language reference," 2004.

[30] 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).* IEEE, 2016, pp. 117–124.

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

[32] 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.

[33] 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, 2011.

[34] O. T. Eluwole, N. Udoh, M. Ojo, C. Okoro, and A. J. Akinyoade, "From 1g to 5g, what next?" *IAENG International Journal of Computer Science*, vol. 45, no. 3, 2018.

[35] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[36] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” *IEEE Communications surveys & tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[37] R. Bifulco and G. Rétvári, “A survey on the programmable data plane: Abstractions, architectures, and open problems,” in *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 2018, pp. 1–7.

[38] S. Han, S. Jang, H. Choi, H. Lee, and S. Pack, “Virtualization in programmable data plane: A survey and open challenges,” *IEEE Open Journal of the Communications Society*, vol. 1, pp. 527–534, 2020.

[39] 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.

[40] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, “P4: Programming protocol-independent packet processors,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[41] 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.

[42] U. Küster, B. König-Ries, M. Stern, and M. Klein, “Diane: an integrated approach to automated service discovery, matchmaking and composition,” in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 1033–1042.

[43] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne *et al.*, “Owl-s: Semantic markup for web services,” *W3C member submission*, vol. 22, no. 4, 2004.

[44] J. Domingue, D. Roman, and M. Stollberg, “Web service modeling ontology (wsmo)-an ontology for semantic web services,” 2005.

[45] S.-C. Oh, H. Kil, D. Lee, and S. R. Kumara, “Algorithms for web services discovery and composition based on syntactic and semantic service descriptions,” in *The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE’06).* IEEE, 2006, pp. 66–66.

[46] D. Petcu, B. D. Martino, S. Venticinque, M. Rak, T. Máhr, G. E. Lopez, F. Brito, R. Cossu, M. Stopar, S. Šperka *et al.*, “Experiences in building a mosaic of clouds,” *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 2, pp. 1–22, 2013.

[47] M. Malaimalavathani and R. Gowri, “A survey on semantic web service discovery,” in *2013 International Conference on Information Communication and Embedded Systems (ICICES).* IEEE, 2013, pp. 222–225.

[48] J. Soares, M. Dias, J. Carapinha, B. Parreira, and S. Sargento, “Cloud4nfv: A platform for virtual network functions,” in *2014 IEEE 3Rd international conference on cloud networking (cloudnet).* IEEE, 2014, pp. 288–293.

[49] O. TOSCA, “Tosca simple profile for network functions virtualization (nfv) version 1.0,” 2015.

[50] G. Xilouris, E. Trouva, F. Lobillo, J. M. Soares, J. Carapinha, M. J. McGrath, G. Gardikis, P. Paglierani, E. Pallis, L. Zuccaro *et al.*, “T-nova: A marketplace for virtualized network functions,” in *2014 European Conference on Networks and Communications (EuCNC).* IEEE, 2014, pp. 1–5.

[51] 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.

[52] I. Oliver, S. Panda, K. Wang, and A. Kalliola, "Modelling nfv concepts with ontologies," in *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 2018, pp. 1–7.

[53] Z. Tasnim, M. Dieye, F. Estrada-Solano, R. H. Glitho, H. Elbiaze, and W. Ajib, "An ontology-based model for in-network computing components description and discovery."

[54] G. M. Yilma, Z. F. Yousaf, V. Sciancalepore, and X. Costa-Perez, "Benchmarking open source nfv mano systems: Osm and onap," *Computer communications*, vol. 161, pp. 86–98, 2020.

[55] H. Koumaras, M.-A. Kourtis, and D. Martakos, "Benchmarking the encoding efficiency of h. 265/hevc and h. 264/avc," in *2012 Future Network & Mobile Summit (FutureNetw)*. IEEE, 2012, pp. 1–7.

[56] P. Seeling and M. Reisslein, "Video traffic characteristics of modern encoding standards: H. 264/avc with svc and mvc extensions and h. 265/hevc," *The Scientific World Journal*, vol. 2014, no. 1, p. 189481, 2014.

[57] V. Sze and M. Budagavi, "Design and implementation of next generation video coding systems (h. 265/hevc tutorial)," in *IEEE international symposium on circuits and systems (ISCAS), Melbourne, Australia*, 2014.

[58] A. Jones, I. McDowall, H. Yamada, M. Bolas, and P. Debevec, "Rendering for an interactive 360 light field display," in *ACM SIGGRAPH 2007 papers*, 2007, pp. 40–es.

[59] F. Yaraş, H. Kang, and L. Onural, "State of the art in holographic displays: a survey," *Journal of display technology*, vol. 6, no. 10, pp. 443–454, 2010.

[60] J. Kessenich, G. Sellers, and D. Shreiner, *OpenGL Programming Guide: The official guide to learning OpenGL, version 4.5 with SPIR-V*. Addison-Wesley Professional, 2016.

[61] T. Akenine-Moller, E. Haines, and N. Hoffman, *Real-time rendering*. AK Peters/crc Press, 2019.

[62] R. K. d. Anjos, J. M. Pereira, and J. A. Gaspar, "Video-based rendering techniques: A survey," *arXiv preprint arXiv:2312.05179*, 2023.

[63] X. Wang, Z. He, and L. Cao, “Analysis of reconstruction quality for computer-generated holograms using a model free of circular-convolution error,” *Optics Express*, vol. 31, no. 12, pp. 19 021–19 035, 2023.

[64] R. Corda, D. Giusto, A. Liotta, W. Song, and C. Perra, “Recent advances in the processing and rendering algorithms for computer-generated holography,” *Electronics*, vol. 8, no. 5, p. 556, 2019.