future internet

MDPI

Review

Exploring In-Network Computing with Information-Centric Networking: Review and Research Opportunities

Marica Amadeo [1,2,*] and Giuseppe Ruggeri [2,3]

[1] Department of Engineering, University of Messina, C.da di Dio, Vill. S.Agata, 98166 Messina, Italy
[2] Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), Viale G.P. Usberti, 181/A, 43124 Parma, Italy; giuseppe.ruggeri@unirc.it
[3] Department of Information Engineering, Infrastructure and Sustainable Energy (DIIES), University Mediterranea of Reggio Calabria, Via Zehnder snc, 89100 Reggio Calabria, Italy
* Correspondence: marica.amadeo@unime.it

Abstract: The advent of 6G networks and beyond calls for innovative paradigms to address the stringent demands of emerging applications, such as extended reality and autonomous vehicles, as well as technological frameworks like digital twin networks. Traditional cloud computing and edge computing architectures fall short in providing their required flexibility, scalability, and ultra-low latency. Cloud computing centralizes resources in distant data centers, leading to high latency and increased network congestion, while edge computing, though closer to data sources, lacks the agility to dynamically adapt to fluctuating workloads, user mobility, and real-time requirements. In-network computing (INC) offers a transformative solution by integrating computational capabilities directly into the network fabric, enabling dynamic and distributed task execution. This paper explores INC through the lens of information-centric networking (ICN), a revolutionary communication paradigm implementing routing-by-name and in-network caching, and thus emerging as a natural enabler for INC. We review state-of-the-art advancements involving INC and ICN, addressing critical topics such as service naming, executor selection strategies, compute reuse, and security. Furthermore, we discuss key challenges and propose research directions for deploying INC via ICN, thereby outlining a cohesive roadmap for future investigation.

Keywords: in-network computing; information-centric networking; named data networking; compute reuse; edge computing

check for updates

Academic Editor: Amiya Nayak

Received: 21 December 2024
Revised: 10 January 2025
Accepted: 16 January 2025
Published: 18 January 2025

Citation: Amadeo, M.; Ruggeri, G. Exploring In-Network Computing with Information-Centric Networking: Review and Research Opportunities. *Future Internet* **2025**, *17*, 42. https://doi.org/10.3390/fi17010042

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The vision of 6G networks and beyond is driven by transformative use cases that demand unparalleled levels of connectivity, intelligence, and real-time responsiveness [1]. Applications such as eXtended Reality (XR) for immersive experiences, autonomous vehicles navigating complex environments, and smart cities powered by billions of interconnected Internet of Things (IoT) devices highlight the pressing need for advanced communication and processing capabilities. Similarly, federated learning at the edge for privacy-preserving AI, real-time disaster response systems, and Digital Twin Networks (DTNs) that synchronize physical and virtual worlds emphasize the critical importance of low-latency and scalable computing solutions [2].

Traditional centralized architectures, reliant on cloud-based processing, struggle to provide the ultra-low latency, bandwidth efficiency, and real-time decision making required by these next-generation applications. This challenge has led to the emergence of edge

computing and Multi-access Edge Computing (MEC) as intermediate solutions, which bring computational resources closer to the end-user [3]. By offloading tasks from the cloud to edge servers, these approaches reduce latency, optimize bandwidth usage, and improve responsiveness. However, these frameworks largely rely on static and predefined deployments, where computational resources are provisioned at fixed locations near the network's periphery. While effective, they often lack the flexibility and scalability needed to address the dynamic and distributed nature of many 6G applications.

In this context, *In-Network Computing* (INC) emerges as a revolutionary paradigm that goes beyond the limitations of static edge computing [4]. It enables the network itself to perform computation dynamically and adaptively, directly within its communication fabric. Unlike traditional edge architectures, INC leverages the capabilities of network nodes to process, compute, and make decisions closer to the data source, often at intermediate hops. This paradigm shift introduces unprecedented opportunities to meet the stringent demands of next-generation applications [5]; however, it also raises some distinctive challenges.

Indeed, deploying INC requires the management of distributed and heterogeneous resources across distributed network nodes, each with varying computational, storage, and energy capabilities. Dynamically coordinating these resources in real time, while ensuring efficient task allocation and execution under changing network conditions, is far from trivial [6]. The distribution of computational tasks across multiple intermediate nodes can lead to inefficiencies if routing, scheduling, or workload balancing are not optimized. The presence of client mobility adds another level of complexity, by introducing frequent topology changes and intermittent connectivity, which networks may struggle to accommodate. Another significant challenge lies in data consistency, privacy, and security. INC involves processing and storing data at multiple points within the network, increasing the risk of data breaches and inconsistencies. Balancing the need for high performance with robust security measures remains a pressing concern, especially for privacy-sensitive applications like healthcare [7].

The host-centric protocol suite of the current Internet, TCP/IP, exacerbates these challenges due to its fundamental design for end-to-end communications between fixed client and server entities. IP inherently ties communication to specific physical addresses, making it rigid and inefficient for the dynamic and distributed nature of INC. When computation tasks are allocated on demand, rather than tied to specific locations, relying on explicit host addresses for routing results in unnecessary overhead, reduced scalability, and inefficiencies in resource utilization. Additionally, TCP/IP's connection-oriented approach does not natively support the data-driven and location-independent workflows required for INC, such as dynamic task migration or localized processing within the network fabric.

This underscores the need for networking paradigms that natively integrate computation within the communication fabric [8]. In this context, Information-Centric Networking (ICN) stands out as a promising solution [9]. By shifting the focus from host addresses to data, ICN enables inherently flexible and efficient communication that aligns with the principles of INC. Its features, such as built-in caching and data-centric routing, facilitate adaptive task placement, dynamic resource allocation, and reduced data redundancy. Moreover, ICN's ability to decouple data from its physical source provides a robust foundation for addressing mobility and scalability challenges, making it a natural enabler for in-network computing.

In this paper, after introducing the INC concept, we delve into the existing literature that positions ICN as its enabling networking paradigm. We examine critical components for deploying INC via ICN, including service-naming schemes, executor selection strategies, compute reuse mechanisms, and security considerations, presenting a structured overview

of the state of the art. Moreover, we outline open challenges and potential solutions, emphasizing the need for specific advancements, including semantic routing, collaborative caching, and adaptive service placement to fully realize the potential of INC via ICN.

As shown in Table 1, several surveys have been published so far that separately consider ICN and INC approaches. However, to the best of our knowledge, this is the first work to comprehensively explore INC through the lens of ICN, with a particular emphasis on addressing network layer challenges.

Table 1. Summary of related surveys and their focus.

Reference	Year	Main Topic	Focus
[10]	2013	ICN	Comparison of distinct ICN architectures
[11]	2016	ICN	Transport-layer approaches
[12]	2016	ICN	NDN routing, caching, forwarding and mobility
[13]	2017	ICN	In-network security and privacy
[14]	2019	ICN	Forwarding strategies in wireless networks
[15]	2019	ICN	NDN in vehicular networks
[16]	2020	ICN	Naming and caching in 5G and beyond networks
[17]	2020	ICN	Caching in vehicular NDN
[18]	2021	ICN	Access control mechanisms in NDN
[19]	2022	INC	Programmable data planes and applications
[20]	2022	ICN	NDN routing
[21]	2023	INC	In-network ML
[22]	2023	ICN	In-network caching strategies
[23]	2024	INC	Smart NICs
[24]	2024	ICN	Software-defined NDN
[25]	2024	INC	Enabling technologies and architectures
[26]	2024	INC	In-network ML

The rest of this paper is organized as follows: Section 2 introduces the INC paradigm with two representative use cases. Section 3 presents the ICN paradigm and focuses on a specific implementation, namely Named Data Networking (NDN). Section 4 discusses INC via ICN, by considering several building blocks like service naming, executor selection strategies, resource discovery mechanisms, compute reuse and security. Section 5 discusses research perspectives, before concluding this paper in Section 6.

2. In-Network Computing: What and Why

INC represents a paradigm shift from the traditional host-centric models of cloud and edge computing to a distributed, network-embedded approach, where computation is integrated directly into the network fabric [19].

The original INC concept dates back to the '90s, with the theorization of active networks, a technology that allows for the injection of customized programs into network nodes, thus enabling in-network processing and storage services [27]. In active networks, routers can process user data as they flow through them, performing tasks such as caching, compression, and even data fusion or filtering. Traditional (passive) packets are replaced by active capsules, which encapsulate lightweight programs that can be executed at intermediate nodes. These capsules not only allow for dynamic computation but also enable the

propagation of code to nodes along a network path [28]. By doing so, the network itself becomes an extension of user applications, enabling faster protocol innovation. For example, active nodes can execute customized protocols to reduce retransmission latency or cache dynamic content closer to the client.

Due to the technological limitations of the 1990s, including restricted computational capacity and network resources, the active network paradigm was not adopted in practical deployments and eventually faded from focus. Today, with the advent of powerful computational capabilities and advanced networking infrastructures, the core principles of active networks have been revamped, promoting the realization of the INC paradigm.

In future 6G networks, INC can allow complex tasks to be executed on data as it traverses the network, bringing significant advantages, like reduced computing latency, lower data traffic, alleviation of network congestion, and reduced load on centralized servers. Computations can be dynamically performed closer to clients or data sources, bypassing the need to transmit data to distant cloud servers for processing. This reduction in round-trip latency is particularly critical for applications requiring real-time responses, such as autonomous vehicles, industrial automation and XR. Unlike edge computing, which relies on purpose-built servers at the network's edge, INC leverages programmable network elements such as switches and routers, thus eliminating the dependency on dedicated computing infrastructure and enabling real-time task distribution across the network, providing enhanced flexibility and scalability.

2.1. Enabling Technologies

The realization of INC is driven by advancements in networking and programmability through enabling technologies like Software-Defined Networking (SDN), Network Function Virtualization (NFV) [29] and ICN [30]. Moreover, high-performance network elements like Smart Network Interface Cards (Smart NICs), and Field-Programmable Gate Arrays (FPGAs) can contribute to make INC a reality [23].

Table 2 summarizes the main features of these technologies and their contributions to INC.

SDN is a transformative network paradigm that decouples the control plane, responsible for decision making, from the data plane, which handles packet forwarding. This separation allows centralized control and enhanced programmability, providing a comprehensive and dynamic view of the network. Centralized control is managed by an SDN controller, which collects information about the topology, resources, and traffic patterns, enabling efficient resource allocation and traffic management. Programmability is introduced through high-level APIs, enabling network operators to dynamically define and enforce policies, deploy new functionalities, and adapt to changing conditions without modifying the underlying hardware. Physical infrastructure is abstracted into a unified interface, simplifying network management and promoting rapid innovation.

In the context of INC, SDN can facilitate dynamic resource allocation by steering data flows toward nodes capable of executing computational tasks, optimizing resource usage, and minimizing delays [31]. The SDN controller can adapt to varying traffic patterns and ensure efficient priority-based task scheduling and dynamic traffic management, e.g., rerouting data in response to real-time network conditions [32].

By decoupling network functions from proprietary hardware, NFV enables their virtualization and execution on general-purpose hardware, such as servers, switches, and routers. This approach not only reduces the reliance on expensive, dedicated hardware but also provides the agility and scalability needed for modern network operations [33]. NFV allows dynamic instantiation, migration, and scaling of network functions, aligning seamlessly with the distributed and adaptive nature of INC. Moreover, NFV abstracts the

underlying hardware, allowing network functions to run seamlessly on a wide variety of devices, from high-performance edge servers to resource-constrained routers. This abstraction potentially enables the network to accommodate diverse tasks and applications, and ensures that the same physical infrastructure can support multiple users or applications simultaneously, maximizing resource utilization.

Table 2. Enabling technologies for INC.

Technology	Main Features	Contributions to INC
SDN	• Decouples control/data planes and implements a centralized network controller. • Provides high-level programmability with APIs. • Abstracts physical infrastructure into a unified interface.	• Enables dynamic resource allocation and task scheduling. • Optimizes traffic management and reduces latency. • Steers data flows to computationally capable nodes. • Adapts to real-time network conditions.
NFV	• Virtualizes network functions on general-purpose hardware. • Reduces dependency on proprietary hardware. • Allows dynamic instantiation, migration, and scaling of functions. • Abstracts hardware for multi-tenancy.	• Facilitates instantiation and migration of functions on diverse devices. • Maximizes resource utilization across shared infrastructure.
ICN	• Focuses on content-based rather than host-based communication. • Supports in-network caching and computation. • Provides efficient content retrieval mechanisms.	• Reduces latency by processing and caching data in-network. • Enables dynamic task distribution closer to clients.
Programmable switches, smart NICs, FPGAs	• Offer high-speed processing within the network. • Enable customizable hardware acceleration.	• Transform network elements into active computation participants. • Facilitate real-time in-network execution of tasks.

ICN, on the other hand, shifts the focus from host-based communication to content-based communication, thus facilitating in-network caching and computation, becoming an ideal networking model for supporting INC, as it will be clarified in the next sections.

In parallel, technologies such as programmable switches, smart NICs and FPGAs enable high-speed processing capabilities within the network [23]. These components facilitate efficient handling of large volumes of data, supporting operations like packet inspection, data aggregation, and even machine learning inference directly within the data plane [34], thus transforming the network into an active participant in computation.

INC is also perfectly aligned with the convergence of communication, computing and caching (3C) resources, a foundational concept for (beyond) 6G networks [35]. The 3C paradigm envisions each network node equipped with 3C capabilities to support intelligent operations and self-evolution. Such convergence transforms the network into a dynamic and distributed computing platform that spans the entire *cloud-to-end-devices'* continuum and adapts in real-time to application demands, delivering ultra-low latency, high reliability, and context-aware services. The synergy among 3C resources underpins the transition from static infrastructure to an intelligent, self-orchestrating system capable of meeting the demands of next-generation applications.

It is worth observing that INC will not replace cloud and edge computing; rather, these paradigms will coexist and complement each other, forming a cohesive ecosystem that leverages the strengths of centralized cloud resources, edge proximity, and in-network processing to meet the diverse and stringent requirements of 6G applications. Figure 1 summarizes the overall vision, where every network entity along the cloud-to-end-devices continuum is augmented with 3C capabilities to enable INC.

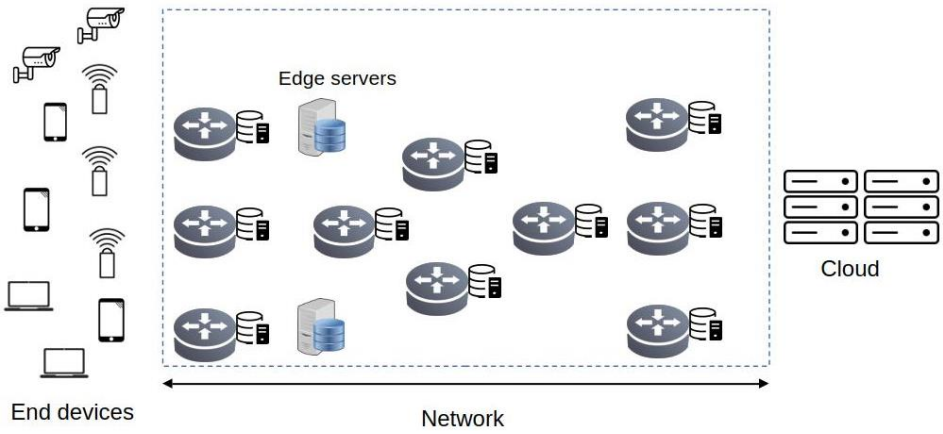

Figure 1. Reference scenario.

Numerous research initiatives and projects have emerged to promote and advance INC. Among them, the Computing in the Network Research Group (https://www.irtf.org/coinrg.html, accessed on 8 January 2024) (COINRG), established in 2019 under the Internet Research Task Force (IRTF), focuses on developing architectures, protocols, and implementations that tightly integrate computing and networking. Similarly, the IETF Routing Area Working Group (https://datatracker.ietf.org/group/rtgwg/about, accessed on 8 January 2024) (RTGWG) has also contributed to the INC research by presenting the compute first networking (CFN) framework [36], where service requests are routed to optimal edge nodes based on real-time assessments of computing resources and network conditions. In the same context, the Programming Protocol-independent Packet Processors (P4) Project (https://p4.org/, accessed on 8 January 2024) aims to standardize a high-level programming language for precise control of packet processing, enabling highly customizable in-network operations.

According to [19], INC applications can be broadly classified into five main categories:

1. In-Network Analytics: Data-intensive tasks such as data aggregation and machine learning.
2. In-Network Caching: Temporary storage of information within network elements to enable quick data retrieval.
3. In-Network Security: Implementation of security mechanisms, such as DDoS mitigation and firewalls.
4. In-Network Coordination: Offloading coordination tasks, such as consensus protocols, to network elements to enhance distributed operations.
5. Technology-Specific Applications: Offloading specific tasks, such as load balancing and resource allocation, tailored to particular technologies like NFV.

2.2. Representative Use Cases

INC can be applied to diverse 6G use cases. In the following, we consider two of the most ambitious ones, namely XR and holographic communications, as well as DTNs.

2.2.1. XR and Holographic Communications

XR and holographic communications represent the frontier of immersive technologies, offering unparalleled experiences in fields ranging from entertainment to gaming, telemedicine, and Industry 4.0 [37,38]. These applications aim to create seamless, real-time interactions that merge the physical and digital worlds, requiring ultra-low latency, high bandwidth, and precise synchronization.

Holographic communications, in particular, have been identified as a critical use case for 6G networks due to their potential to revolutionize digital interactions [39,40]. They push connectivity demands to the extreme by transmitting massive data volumes to render three-dimensional, life-like representations in real time. For instance, end-to-end latency must be less than 1 ms to ensure smooth and immersive experiences, as delays beyond this threshold disrupt the perception of real-time interactivity. Additionally, data rates of several terabits per second are required to transmit the volumetric data necessary for high-quality three-dimensional rendering [41]. Handling these requirements necessitates not only high-capacity links but also efficient data processing across distributed nodes. Additionally, the dynamic nature of holographic applications, involving multiple users and devices, introduces challenges related to maintaining seamless and synchronized delivery, especially in scenarios with rapidly changing network conditions.

Traditional cloud-based architectures struggle to meet these stringent requirements due to inherent delays in processing and data transmission over long distances. Similarly, while edge computing reduces latency by bringing computational resources closer to the user, it remains limited by its reliance on fixed infrastructure, constrained resource availability, and potential bottlenecks during peak demand or in large-scale, multi-user scenarios.

By leveraging the processing power of distributed network elements, INC reduces reliance on cloud and edge servers, minimizes end-to-end latency, and ensures efficient use of bandwidth. It can enable real-time operations such as motion tracking, rendering, and object detection to be performed closer to the user, ensuring smooth and immersive experiences. Coupled with distributed AI capabilities and communication technologies envisioned for 6G [42], including cognitive networks, sub-terahertz and Visible Light Communication (VLC), and Intelligent Reflecting Surfaces (IRSs), INC can enable the synchronized delivery of high-quality, multi-perspective holographic streams to multiple users simultaneously. Network nodes can dynamically optimize their resources, prioritize tasks, and pre-emptively manage computational workloads. For example, predictive algorithms can estimate user movements in XR or adjust holographic stream quality based on device and network conditions. Moreover, by distributing computational tasks, such as rendering and encoding, across network nodes, INC also ensures efficient and fair resource utilization.

2.2.2. DTNs

DTNs extend the traditional concept of Digital Twins (DTs) by introducing inter-twin communications, enabling a networked ecosystem of digital entities that collaboratively manage complex systems or environments [43]. Traditionally, DTs rely on intra-twin communications, focusing on interactions with the physical twin only, with the targets of synchronizing its state or analyzing its behavior. However, DTNs expand this paradigm by creating communities of DTs that interact with each other in the virtual world, thus enabling new services like cooperative decision making, distributed simulation, and collective optimization [44].

DTNs are particularly relevant in 6G for addressing real-world challenges in key domains. For example, in smart cities, DTNs can enable real-time monitoring and optimization of urban infrastructure, such as traffic flow, energy distribution, and public

safety systems, by facilitating seamless communication and coordination among the digital representations of these components [45]. In healthcare, DTNs can enable innovative solutions like remote surgery simulations or collaborative diagnostics by synchronizing DTs of medical devices, patients, and healthcare professionals [46]. Therefore, in the context of 6G, DTNs are envisioned as foundational components for achieving comprehensive network intelligence [42] and advanced user-centric services [47].

DTs can be hosted in various parts of the network, depending on their application and performance requirements [47]. For instance, they can reside in centralized cloud servers, leveraging the vast computational and storage resources of the cloud for tasks requiring significant processing power or long-term data analysis. Alternatively, DTs can be deployed closer to data sources, such as at the edge of the network, where latency-sensitive tasks and real-time interactions demand minimal delays. The choice of hosting location introduces additional challenges in managing communication and synchronization between DTs, particularly when they span multiple levels of the network infrastructure. Therefore, inter-twin communications in DTNs can vary significantly in their nature and underlying requirements. For instance, DTs hosted on the same physical device, such as an edge server, can interact with minimal latency and high reliability. Conversely, when DTs are distributed across different network devices—such as two distinct edge servers or a remote cloud server and an edge server—latency and reliability can fluctuate depending on the quality of the underlying communication networks. These variations in latency, bandwidth, and reliability, exacerbated by the additional need of maintaining consistent and synchronized interactions between DTs, make the design of robust and efficient inter-twin communication mechanisms a critical challenge for realizing DTNs. Scalability is another significant challenge, as 6G is expected to support millions of interconnected DTs.

Despite their significant potential, the practical deployment of DTNs remains largely unexplored in the literature, though emerging technologies suggest they could become a reality in the near future.

6G advancements, such as sub-terahertz communication, IRS and semantic communication, can empower DTNs by enabling ultra-reliable low-latency communications and dynamic resource allocation [42], seamlessly supporting inter-twin interactions even in highly heterogeneous and geographically dispersed environments. Additionally, pervasive AI in 6G enables DTNs to process and act on contextual information, improving their ability to dynamically adapt to network conditions and application demands.

In this context, INC can also play a transformative role. By embedding computational capabilities directly into the network, it can facilitate the dynamic instantiation and management of DTs closer to their physical counterparts or other points of interaction. Computationally intensive tasks like synchronization, state updates, and inter-twin interactions in real time can be deployed in the network, reducing reliance on centralized cloud/edge resources and alleviating latency bottlenecks.

Moreover, INC's distributed architecture allows inter-twin communications to adapt dynamically to network conditions. For example, when latency-sensitive interactions are required, DTs, or only some of their modules (if composable DT architectures are implemented), can be instantiated to nodes nearby the physical twins or the users. Additionally, INC can optimize communication paths for DTNs, ensuring reliable and efficient data exchange between DTs hosted on geographically dispersed devices. These capabilities make INC an essential enabler for realizing scalable, low-latency, and reliable DTNs.

3. Information-Centric Networking

Unlike traditional host-centric protocols, i.e., TCP/IP, ICN focuses on what data or service is being requested rather than on the node providing it [10,48]. This change reflects

the growing demand for efficient content retrieval and service provisioning in modern networks, where users are increasingly more interested in accessing information and services than communicating with specific hosts.

The initial design of ICN was presented in 2009 by Jacobson et al. in their seminal work in [49], which laid the foundation for a paradigm shift from host-based communication to content-centric communication. The objective was to enhance data retrieval processes by addressing several limitations of traditional networking, such as inefficient content distribution, scalability challenges, and security issues tied to host identities. However, ICN's flexibility and innovative design have made it an ideal candidate for more advanced applications. Today, ICN is being increasingly explored for service retrieval and in-network processing, where computation and service execution can take place within the network itself, leveraging ICN's intrinsic features like data naming, caching, and security.

In the following, we review the main ICN principles and explain how they become enabling factors for INC. Then, we focus on the most prominent ICN architecture, namely NDN.

3.1. Why ICN for INC

Named Data. ICN uses unique, persistent names to identify content. Uniqueness guarantees that a particular name corresponds to a single piece of content only. Persistency ensures that names remain valid even if the content is moved, replicated, or cached in different locations. By doing so, the data are globally addressable and retrievable without the need for dynamic updates to a location-dependent routing system. Names can be *flat* (opaque identifiers without structural semantics), or *hierarchical* (structured, human-readable identifiers) [16]. Hierarchical names are particularly common in ICN architectures, as they resemble familiar constructs like URLs. In addition, the hierarchical structure enables intuitive organization, easier routing, and discovery of related content by leveraging meaningful prefixes and namespaces.

Beyond identifying static content, ICN names can be extended to services and on-demand content. Indeed, services can also be named using a hierarchical scheme that reflects their functionality and context. Similarly, on-demand content, such as real-time sensor data or live video streams, can be named dynamically to represent the temporal or contextual state of the data (e.g., */sensor/temperature/room1/timestamp*).

As a result, named packets can be used by client applications to easily request computation services. This simplifies the invocation of in-network computations, as service requests and responses are inherently tied to named-based network primitives. Additionally, named software code can be requested by network nodes for local installation, enabling on-the-fly deployment of computation tasks or updates to existing services. This capability aligns with the principles of INC by allowing nodes to dynamically acquire and execute the required functionality without prior configuration.

For instance, a node could retrieve named software modules to perform data compression on-the-fly for downstream clients. Similarly, computational workflows can be modularized and addressed by name, enabling nodes to fetch and compose the necessary components to fulfill complex processing requests. This adaptability reduces dependency on pre-installed software and enhances the agility of the network in responding to diverse application requirements.

In-Network Caching. In ICN, data can be cached at intermediate network nodes (e.g., routers) as they traverse the network: when a user requests specific content, the network can serve it directly from a nearby cache rather than fetching it from the original producer. By providing such a core INC service, ICN not only reduces redundant data transfers but also brings content closer to the user, resulting in faster content delivery, reduced latency,

and lower network congestion. In parallel, by alleviating the load on origin servers and optimizing bandwidth utilization, in-network caching significantly enhances the scalability and efficiency of the network [22].

An important extension of this principle is the caching of service outputs [48]. In scenarios involving in-network computing, computational results (e.g., processed video/images, analytic outcomes) can also be cached at network nodes by enabling what is known as *compute reuse* [50]. This is another pillar of INC deployments, where results of a previously executed service are reused to fulfill similar requests without needing to re-execute the computation. Compute reuse is particularly advantageous in use cases such as video transcoding, machine learning inference, and IoT analytics, where identical or similar computations are frequently performed on the same data. For example, a node might cache the output of a video compression service. If another user requests the same video in the same compressed format, the network can directly serve the cached result, avoiding the need to rerun the compression process. This reduces computational overhead, conserves energy, and minimizes response times. Moreover, compute reuse enhances fault tolerance by providing a fallback mechanism. If the original service producer becomes unavailable, cached results can still fulfill requests, ensuring continuous service availability. This capability is especially valuable in dynamic environments such as edge networks and IoT systems, where network disruptions or node failures can occur frequently.

Stateful Forwarding. ICN routers maintain state information about active requests: when a request for content or service is received, the router tracks it in its memory, which allows the network to intelligently forward responses and adapt to changes [14]. In particular, stateful forwarding naturally enables multicast communication: when multiple consumers request the same content or service, the router can aggregate these requests and forwards only one copy of the message upstream. The resulting content or service output is then distributed to all consumers without requiring redundant transmissions. In the context of INC, this reduces the computational overhead at upstream nodes, by ensuring that tasks such as data processing, analytics, or transformations are performed only once for multiple consumers. Further benefits include reduced bandwidth consumption and enhanced scalability for popular or frequently accessed resources.

Stateful forwarding also allows routers to dynamically adjust packet transmissions based on real-time network conditions, thus minimizing delays and ensuring higher reliability and load balancing. For example, routers can select alternative paths to bypass congestion or recover from link failures, without requiring end-to-end reconfiguration, or they can distribute requests across multiple paths, optimizing resource utilization across the network. Tracking active requests also allows us to implement in-network congestion control mechanisms [11]. If a router detects excessive demand for certain content or congestion on specific links, it can limit the forwarding of new requests or reroute them to less congested paths. For INC, such mechanisms help manage workloads at computational nodes by preventing bottlenecks and ensuring fair distribution of processing tasks, ultimately improving the responsiveness and reliability of the overall system.

Security by Design. Unlike traditional IP-based networking, which primarily secures the communication channel, e.g., using protocols like TLS/SSL, ICN introduces a fundamentally different security paradigm by focusing on the data itself. In ICN, every piece of data is cryptographically signed at its source. Thus, any recipient is able to verify the authenticity and integrity of the content, independent of the communication path it traversed. As a result, the need for a direct, trusted connection between the sender and receiver is eliminated [13]. Data-centric security protects against content tampering and forgery, as any alteration to the data will render the signature invalid, and it mitigates

man-in-the-middle and replay attacks, as the focus shifts away from securing the channel to securing the data payload itself [51].

These features are particularly beneficial for INC, as they ensure that computations executed on-the-fly at intermediate, not a priori known, nodes are always verifiable. Integrity and authenticity of both the input data and the computational outputs are guaranteed, building trust in the distributed processing environment and ensuring secure interactions between clients and in-network services.

Last but not least, data-centric security facilitates caching at intermediate nodes, since cached data retains its security properties and can be verified by subsequent consumers without requiring a new connection to the original source.

Mobility Support. By decoupling data from their original location, ICN inherently addresses the challenges of client mobility. In traditional IP-based networks, mobility requires maintaining a stable connection to a fixed endpoint, often relying on complex mechanisms such as Mobile IP or tunneling to manage changes in user location. ICN eliminates this dependency by naming content rather than endpoints, allowing content to be retrieved from any location where it is cached or hosted [52]. This feature is particularly advantageous in dynamic and mobile environments, such as those involving smartphones, autonomous vehicles, or IoT devices. When clients move, ICN seamlessly enables requests to be satisfied in an anycast fashion, from the nearest provider. This also minimizes the need for frequent updates to routing or forwarding tables, making ICN highly scalable even as the number of mobile devices grows exponentially. For INC, the native client mobility support ensures that computational tasks and their results remain accessible regardless of client location, enabling their uninterrupted access to in-network services.

Managing mobility of content producers in ICN may present additional challenges, such as ensuring consistent routing updates. However, recent approaches, leveraging localized updates and stateful forwarding, have demonstrated that producer mobility can be effectively supported in a seamless fashion [53,54].

3.2. Named Data Networking: Basics

NDN is the most prominent architecture within the broader ICN paradigm [30]. It relies on two primary communication primitives, namely Interest packets and Data packets. Clients, e.g., data/service consumers, send an Interest packet specifying the name of the desired content or service. This packet propagates through the network until it either (i) reaches a provider node that can satisfy the request (e.g., a content source, a service executor, or a caching node) or (ii) times out or (iii) an intermediate receiver sends back a negative acknowledgment. When Interests reach the provider node, Data packets are returned along the reverse path established by the Interest. Data packets can carry static contents or service results and they are cryptographically signed by the provider to ensure authenticity and integrity. This request–response mechanism eliminates the need for end-to-end connections and allows for dynamic retrieval of content from multiple sources.

3.2.1. Node Architecture

As shown in Figure 2, each NDN node maintains three tables at the data plane:

- The Content Store (CS) is a cache that temporarily stores Data packets passing through the node. If an Interest requests a content available in the CS, the node can respond directly, without forwarding the Interest further, reducing network load.
- The Pending Interest Table (PIT) tracks pending interests that the node has forwarded but has not yet been satisfied. When a Data packet arrives, the PIT ensures that it is sent back to all downstream consumers who requested it and then removes the corresponding entry.

- The Forwarding Information Base (FIB) stores name-based forwarding rules. It is used to direct Interests to the appropriate next hops, similar to a routing table in IP networks.

In addition, the forwarding strategy module dynamically determines how and when to forward Interests, i.e., over which network or application interface, considering multiple factors such as network congestion, link quality, and application requirements [55]

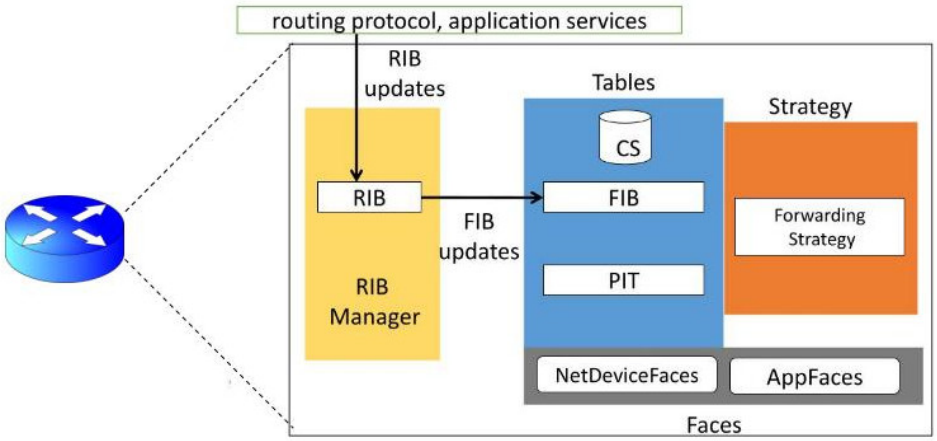

Figure 2. NDN node architecture.

At the control plane, the Routing Information Base (RIB) maintains information about routing prefixes, typically received from one or more routing protocols. The entries in the RIB indicate possible paths to reach specific name prefixes and include metrics such as cost, delay, or link quality. The RIB dynamically updates the FIB with the best routes based on these metrics. NDN routing protocols operate similarly to their IP counterparts but use named prefixes instead of IP addresses.

3.2.2. Forwarding Fabric

When an Interest packet arrives at an NDN node *N*, the forwarding process begins by checking the CS. If the requested Data are found in the CS, *N* immediately responds with the cached data, without forwarding the Interest further. Otherwise, *N* accesses the PIT to determine whether the same Interest has already been forwarded. If a match exists, *N* aggregates the request by appending the incoming interface to the PIT entry and discards the Interest, thus avoiding duplicate forwarding. If the PIT check fails, *N* looks for a matching in the FIB and forwards the Interest to one or more next hops, based on the established forwarding policy.

Once the requested Data packet returns to *N*, it checks the PIT for a matching entry, forwards the Data to all interfaces listed in that entry, and then removes it.

If the Data are not returned within a predefined lifetime, the *N* can try alternative paths or send back a negative ack (NACK) to the previous node. The latter can decide to retry the request via a different interface listed in the FIB, or propagate the NACK downstream to inform the previous nodes in the Interest path about the failure. In addition to the Interest expiration, NACKs can also identify other errors [56], including (i) no data, when the requested Data are not available at the upstream node; (ii) congestion, when the upstream path is experiencing congestion or packet loss; (iii) no route, when there is no available forwarding route to reach the producer of the requested Data.

This mechanism ensures that the forwarding strategy in NDN can dynamically adapt to network conditions, such as link failures or congestion, without requiring global routing table updates [57].

4. In-Network Computing via ICN

A variety of studies have explored NDN as a promising enabler for in-network computing, primarily focusing on generic services where processing is confined to a single in-network executor.

In this section, we review the most notable approaches, structuring our analysis around the following critical deployment steps: (i) service naming schemes and communication messages, (ii) reactive and (iii) proactive strategies for selecting in-network executors, (iv) techniques for compute reuse, and (v) security considerations.

4.1. Service Naming and Messages

The first steps in enabling in-network computing via NDN is to define an effective naming scheme that can identify both the requested processing function and the data to be processed and define the communication primitives in addition to the standard NDN request and response packets, i.e., Interest and Data. Over the years, various solutions have been proposed to address these needs, as summarized in Table 3 and discussed in the following.

Table 3. Comparison of service naming and messaging approaches.

Approach	Ref.	Name Format	Key Features
NFN	[58,59]	Names represent data and functions with λ-expressions. The postfix name component */NFN* denotes the computation request.	• Interest packets can carry complex expressions combining data and functions • Commands can control long-running computations (pause, resume, stop)
NCN	[4,60]	Hierarchical names identify both data and functions. The tag */NCN* is used as a delimiter between the two name parts.	• Compatible with legacy NDN forwarding • Supports execution parameters in service names
ICedge	[48,61]	Hierarchical names with meaningful prefixes for data and services.	• Discovery Interest mechanism for finding available services • Metadata like location-aware clustering for reuse of compute results • Controls Interest lifetime to manage responses; supports dynamic service discovery
NFaas	[62]	Names include the main prefix */exec/* for execution requests. Application requirements are carried as name components.	• Functions implemented as unikernels to enable lightweight processing • Hash-based suffixes to distinguish between different requests • Optional TLV fields for task deadlines
RICE	[63]	Functions named hierarchically or via λ-expressions. *Thunk* names identify ongoing computations.	• Supports long-running computations with thunk-based targeting • Four-way handshake protocol ensures secure authentication and DoS protection

Named Function Networking (NFN) [58] was the first proposal extending NDN by integrating data processing capabilities alongside data retrieval. In NFN, names can represent data content, a function for processing the content, or even an expression that combines both. Consequently, an Interest packet can carry complex expressions that refer to named data, named functions, or a combination of the two. The network is responsible for resolving these expressions and identifying the node that will execute the required function. This process seamlessly integrates expression resolution with name-based forwarding.

To achieve this, the NDN forwarding plane is enhanced with a $\lambda-$expression resolution engine, which processes all Interests with the postfix name component/NFN, allowing the system to identify and execute functional requests dynamically. In [59], the NFN architecture is augmented with a set of *request-to-computation*, commands to allow the consumers control long-running in-network computations, including pausing, resuming, stopping, or fetching intermediate results. These commands also address challenges such as timeout management by introducing keep-alive messages and support dynamic reconfiguration of computations, making NFN more suitable for real-time and resource-constrained environments.

In the Named Computation Networking (NCN) approach [4,60], both data and functions are assigned hierarchical names, with the function name appended to the data name. The prefix NCN is used as a delimiter between the data name and the function name. If the NCN tag is absent, IoT-NCN nodes treat the Interest as a traditional NDN content request. This structure ensures compatibility with legacy NDN forwarding mechanisms. Service names can also include a limited set of input parameters for execution. For example, an Interest with the name */unirc/buildingEngineering/temp/NCN/avg/p1=10* requests the calculation of an average over the last 10 temperature values collected from building Engineering at the University of Reggio Calabria. Nodes without computing capabilities retain information only about the data name prefix (e.g., /unirc) and route the packet accordingly.

In the ICedge scheme [48], service names are structured hierarchically and carry semantic meaning. The first name component is the *service name*, which identifies the type of service being requested, e.g., */Yolo* for an image annotation service. The second name component may include *compute-aware convention* metadata, which help the network to detect if a similar task has been processed recently and enable compute reuse. For example, for location-aware services, the naming convention might include geographic coordinates to cluster tasks related to specific areas on a map. Users send "discovery" Interests under the namespace */discovery* to find out which services are available in the edge network. Edge nodes respond with details about available services, naming conventions and metadata, and wait for specific service requests from consumers. The latter can also set a lifetime for the discovery Interest to effectively control the number of responses [61].

In the Named Function as a Service (NFaaS) approach [62], the authors assume that processing functions are lightweight virtual machines implemented as unikernels. Consumers can request the function code by sending Interests including the name of the function to be processed. In parallel, they can also request the execution of a function, by including the prefix */exec/* followed by an indicator for application requirements, such as */exec/delay/* for delay-sensitive applications, or */exec/bandwidth/* for bandwidth-hungry applications. Each request name may also include a hash-based suffix to differentiate between consecutive requests with different input parameters and optional Type–Length–Value (TLV) fields to set a task deadline.

In the RICE proposal [63], functions can be named hierarchically or via *lambda*-expressions, uniquely identifying the computation and its parameters. In addition, *thunk* names identify specific service instances already being run on a specific node and they are used to retrieve long-running computations. A thunk is composed of three main components: (i) the name of the in-network executor, (ii) the name of the instantiated function, and (iii) the name of the function's internal state. A client uses the thunk name to directly target the executor, without the need of maintaining a state in intermediate nodes. In RICE, a four-way handshake protocol is introduced for authenticating clients before starting a computation, thus preventing denial-of-service attacks. The handshake begins with the client sending an initial Interest message, which includes a unique identifier and a function name to be executed. This message establishes temporary state in the PITs of intermediate

nodes to ensure correct routing of subsequent messages. Upon receiving the request, the executor replies with another Interest message that is used for authenticating the client and transmit additional instructions. The client responds with a Data message, which contains authentication credentials and input parameters needed for the computation. Once the server validates the client and prepares to allocate resources, it concludes the handshake with a confirmation Data message, which also provides a unique thunk name, allowing the computation results to be retrieved later.

4.2. Reactive Executor Selection Strategies

Network nodes, particularly those at the edge, operate with constrained resources, such as limited computational power, memory, and energy capacity [64]. As a result, selecting the optimal executor for in-network processing tasks becomes a critical decision to ensure efficient resource utilization and meet both service-level and network-level objectives. The selection process must balance multiple factors and adapt dynamically to varying conditions within the network.

From the perspective of the network administrator, a range of objectives can be prioritized to optimize the performance, including minimizing traffic congestion by strategically distributing computational workloads across the network to prevent bottlenecks, reducing overall energy consumption to ensure sustainable operations, and achieving load balancing by avoiding overburdening specific nodes while underutilizing others.

From the client's perspective, requirements are often driven by the specific nature of the service being executed. For latency-sensitive applications, such as real-time video analytical or autonomous systems, the selection process prioritizes nodes capable of providing minimal delay. For applications requiring high reliability, such as industrial automation or healthcare systems, executor selection must ensure robust fault tolerance and consistent availability of processing resources.

Consequently, research has extensively explored mechanisms to perform efficient service executor selection within the ICN forwarding fabric, as summarized in Table 4. We distinguish distributed approaches, where the decision is directly taken by network nodes, and centralized approaches, where the decision is taken by a centralized entity overseeing the network domain.

4.2.1. Distributed Approaches

In [65], the authors extend NFN with three resolution strategies for mobile and constrained scenarios, namely EdgeFox, Find-and-Execute (FaX) and Find-or-Pull-and-Execute (FoP)aX. EdgeFox is designed for stationary IoT networks, where devices are typically resource-constrained and data are generated at the network edge. Instead of transferring data to the network core, the target is to look for cached results or deploy computations on the edge node as close as possible to the IoT sources. If no edge node is available, EdgeFox asks executing the computation at the data source, if possible.

The FaX strategy is tailored for highly mobile IoT networks: it immediately starts the requested computation in the node closest to the data sources, instead of searching for cached results. If a cached result is found during execution, it is returned to the requester, and the computation is stopped. While this reduces delays, it can lead to inefficiencies, such as redundant computations. Therefore, FaX is suitable for scenarios where timeliness is more critical than computational efficiency.

Finally, the (FoP)aX strategy extends FaX by integrating the request-to-computation mechanism to fetch intermediate results from neighboring nodes. The goal is to reduce overall computation time by leveraging partial results already calculated elsewhere in the network. (FoP)aX is particularly useful for long-running computations in mobile

scenarios, where nodes may frequently disconnect or switch networks. By fetching intermediate results, it avoids redundant calculations and improves the likelihood of timely result delivery.

Table 4. Executor selection strategies.

Reference	Approach	Domain	Main Features
[65]	Distributed, Reactive	IoT	Three resolution strategies (EdgeFox, FaX, (FoP)aX) for stationary and mobile networks with focus on reducing redundant computations.
[60]	Distributed, Reactive	IoT	Identifies in-network executor as the closest node to the data source; minimizes data retrieval latency but may overburden nodes with limited resources.
[4]	Distributed, Reactive	IoT	Selects best on-path executor by calculating execution cost considering data retrieval time and processing latency.
[66]	Distributed, Reactive	Personal Services	Broadcasts over the wireless channel enhanced Interest packets with task attributes; providers in the neighborhood compute weighted cost metrics; random deferral time ensures the best provider responds first.
[61]	Centralized, Reactive	IoT	Executor selection managed by a delegated node; polls edge nodes for availability, computation time, and cost; selects the node with the best offer within a decision interval.
[31]	Centralized, Reactive	5G services	SDN controller orchestrates executor selection based on global network knowledge; minimizes service provisioning time by evaluating the overall service provisioning latency.
[48]	Distributed, Proactive	IoT	Periodic resource discovery mechanism where nodes share utilization information with neighbors. Update intervals range from 15 to 60 s, with messages flooded within a hop-count scope.
[67]	Distributed, Proactive	IoT	Introduces resource breadcrumbs for proactive resource discovery, allowing edge nodes to distribute resource availability information. Updates occur only when significant resource changes happen, reducing overhead.
[62]	Distributed, Proactive	5G services	Nodes proactively advertise locally available function name prefixes via routing protocols, enabling Interests to be forwarded directly to appropriate executors. Decisions of what functions host are based on service popularity and requirements.
[68]	Distributed, Proactive	5G services	Leverages periodic and event-triggered updates on function availability and resource utilization thus nodes group into synchronization clusters. Executor selection leverages neighborhood knowledge before forwarding requests outside the group.

In [60], the authors propose a distributed strategy to identify the in-network service executor as the available node closest to the data source(s). The consumer initiates the process by transmitting an Interest packet that includes, in its name field, both the name of the data to be processed and the name of the required computing function. Additionally, a new boolean field, called *ExecAvailability*, is added to the PIT entry by intermediate nodes when forwarding the packet. This field is set to 1 if the node has the capability to execute the processing function. The Interest packet is forwarded toward the data source(s) until it reaches the *branching node*, i.e., the last NDN router before the data source(s), which is responsible for data collection. If the branching node possesses the required computing function, it executes the service locally. Otherwise, it sends a NACK back through the network. In this case, the first node with a matching PIT entry and the *ExecAvailability* field set to 1 takes over the service execution. While this approach can reduce data retrieval latency by selecting a node close to the data source(s), a significant drawback is that the

selected node may have limited computational resources, potentially leading to slower processing times.

To overcome this issue, a strategy to select the best on-path executor is presented in [4]. There, nodes along the path to the data producer(s) locally calculate the cost of executing the computation, which accounts for both the latency of retrieving the data and the local processing latency. A new header field is introduced in the Interest packet to carry the current lowest execution cost. Consequently, on-path nodes update this field only if they determine that their execution cost is lower, indicating they are the fastest option.

A cost-based strategy for the selection of the best executor is also defined in [66], when considering a distributed wireless environment where mobile nodes establish opportunistic contacts. To locate a computing node in the neighborhood, a client broadcasts an enhanced Interest packet (eInt-REQ), which includes attributes of the task to execute, such as the size of the content, task preferences, and constraints like maximum tolerable delay and energy consumption. Potential providers that receive this request first verify if they can satisfy the requirements of the service. If eligible, they compute a weighted cost metric based on the estimated task response time and energy expenditure required to process the task. Each provider assigns itself a random deferral time inversely proportional to its cost metric, ensuring the best provider with the lowest cost responds first. The first response received by the consumer is selected as the service executor, streamlining the process and minimizing decision-making latency in a distributed wireless environment.

4.2.2. Centralized Approaches

In [61], the executor selection is managed by a delegated node that oversees an edge domain. After receiving the service request from a consumer, the supervisor polls the edge nodes under its supervision to collect availability information, including the expected computation time and the cost of computation. Information is retrieved during a decision time interval and then the node with the best offer, e.g., lowest cost or shortest expected computation time, is selected for the computation.

A softwarized edge network is instead considered in [31], where an SDN controller with global knowledge of the domain orchestrates routing and in-network computing services. When a service request is received, the controller first checks if the same service has been already executed and the result can be directly reused. If this is not the case, the controller evaluates the computational and storage resources of candidate nodes, ensuring they have sufficient capacity to execute the requested service. By considering the network topology and link latencies, the overall service computing time is calculated as the sum of three time contributions: (i) the time to retrieve input data, (ii) the time to execute the service, (iii) the time to deliver the output back to the consumer. The node that minimizes the overall service provisioning time is selected as the executor. If no suitable node is available, the request may fail or be redirected to another domain.

4.3. Proactive Executor Discovery

Selection of service executors can be enhanced by proactively distributing information about the capabilities of network nodes. However, this requires additional signaling, which introduces extra overhead. Efficient dissemination strategies, such as broadcasting updates only when significant changes occur or by carefully managing the frequency and size of these updates, can minimized the overhead, while still enabling informed service executor selection. In the following, we revise some notable approaches in this context.

In [48], a periodic resource discovery mechanism is proposed, where nodes share utilization information with their neighbors. The update interval ranges from 15 to 60 s, and messages are flooded within a hop-count scope of 2 to 4. In parallel, the routers

can solicit updates by generating Interest packets. As expected, simulation results show that increasing the scope and frequency of updates improves performance but at the cost of higher signaling traffic overhead. Similarly, the work in [67] proposes a proactive resource discovery mechanism based on Resource Breadcrumbs (RBCs), which allows potential edge executor nodes to distribute information about their resource availability, such as CPU and memory status, using scoped flooding within the local network. Interest packets are then guided toward suitable nodes based on these breadcrumbs, ensuring that requests are directed to executors with sufficient capacity to handle them. Unlike [48], the resource status information is updated only when there are significant changes in the executor's resource availability (e.g., exceeding a threshold), making the process more efficient than periodic updates. Moreover, when receiving a service request, executors attach their updated resource statuses to response Data packets, reducing the need for separate signaling. Simulations demonstrate that this approach effectively reduces resource allocation failures and enhances resource utilization.

In NFaaS [62], nodes leverage a routing protocol to proactively advertise the name prefixes of locally available functions, allowing Interests to be forwarded directly to an appropriate executor based on the service name. The decision regarding which functions to store and run locally is based on a combination of service popularity, as measured by historical demand, and service-specific requirements. For instance, latency-sensitive services are predominantly deployed closer to edge nodes, while bandwidth-intensive services are allocated to core nodes to optimize resource utilization and minimize network overhead. The routing fabric is updated in real time to reflect the location of moving functions and also supports load balancing by dynamically redirecting Interests to alternate execution points when the primary node is overloaded. This feature enables the network to distribute computational tasks evenly across available resources, improving overall system performance and resilience.

The work in [68] enhances the NFN framework by introducing the State Vector Synchronization (SVS) protocol. This latter implements periodic and event-triggered notifications among NFN nodes to share updates on function availability and resource utilization. More specifically, nodes broadcast lightweight Interest packets to announce their presence. An hop limit can be set to avoid flooding the network, while loops are prevented through duplicate detection mechanisms. Receiver nodes send back Data packets containing static properties such as compute configurations (e.g., CPU, GPU, memory) and network configurations (e.g., hop distance, link status). Subsequently, nodes are grouped into clusters based on proximity (e.g., one-hop or three-hop distance) or other criteria such as compute capabilities or specialized roles. Within each synchronization group, nodes exchange updated state information. When a service needs to be executed, a node first leverages its neighborhood knowledge to determine if there is a potential executor owning the processing function. In case of failure, the request is forwarded outside the group.

4.4. Compute Reuse

Caching service outputs is a highly effective strategy that can significantly reduce the load on servers by avoiding redundant computations, decrease network traffic, and accelerate service delivery. While not all service results are suitable for caching and reuse, there are many scenarios where this approach is both feasible and advantageous. For instance, XR applications used by attendees at the same event, such as a music concert or a popular tourist attraction, can benefit from shared computation results. Similarly, in a museum setting, visitors capturing photographs of the same artwork from different angles may request similar computational tasks, such as retrieving historical information about the painting. From the client's perspective, the source of the result, i.e., whether retrieved from

the cache or dynamically generated by the server, would be entirely transparent, ensuring a seamless user experience.

Given these advantages, most existing research on INC via ICN inherently assumes the concept of compute reuse [4,58,63,68]. The core principle is that services are computed only upon the first request. The resulting output is treated as newly generated content, uniquely named based on a combination of the function name, content name, and parameters. This result is encapsulated in one or more Data packets, which are signed by the executor to ensure authentication and integrity. Similar to static content, these Data packets can be cached at the executor or at intermediate nodes and they can be retrieved using the standard Interest-Data exchange mechanism in ICN. Subsequent requests for the same service can bypass re-execution by fetching the cached results, significantly improving efficiency and reducing computational overhead.

In [50], the authors observe that reusable outputs can often be time-limited, meaning they have a finite period of validity after which they may become obsolete or inconsistent. Examples include tasks that process time-sensitive input data, such as environmental parameters or context-aware information collected by surveillance cameras. In such cases, the executor can specify a *freshness period* in the header field, which acts as a time-to-live to automatically invalidate the result once it expires.

In ICedge [48], partial and full compute reuse is considered. In the first case, the existing output is reused only partially, and additional computations are required. The authors define compute-aware naming conventions to cluster tasks and direct similar service requests to the same computing nodes to maximize the chances of reusability. Examples include clustering tasks by location (/<building_X/floor_y/Room_z>), geographic coordinates (/<X_coordinate/Y_coordinate>), or configuration parameters (/<config_X/Param_Y>). In parallel, naming conventions are associated with a compute-aware forwarding schemes. For example, a zone-based policy would cluster tasks by regions, such as ZIP codes, forwarding requests based on location metadata.

4.5. Security

By shifting computation from centralized, trusted environments to distributed nodes within the network, INC introduces critical security and privacy challenges. Without adequate mechanisms to ensure trust in these entities, malicious nodes could tamper with results, disrupt services, or misuse sensitive data. Ensuring the integrity of computation results is essential for maintaining trust, particularly in scenarios where errors or tampering could lead to severe consequences, such as in healthcare or autonomous systems. Moreover, INC systems must resist malicious attacks, including replay and DoS, while holding nodes accountable for their actions through mechanisms like provenance tracking and tamper-proof auditing. Privacy concerns are also significant, as INC often processes personal or sensitive information at distributed nodes, necessitating encryption and access control to prevent unauthorized access or data leakage.

Security concerns of INC via ICN, with a special focus on the NFN architecture, are discussed in [69]. The authors propose the use of provenance records, which document the computation process and allow consumers to verify the correctness and authenticity of computation results without needing to re-execute the function themselves. Provenance metadata includes the computing entity's identity, input data and function signatures, the produced result, and a statement covering all elements. This creates transparency in NFN, mitigating the risks of relying on potentially untrustworthy nodes. In terms of verification, the provenance system enables the identification of malicious nodes by allowing users to recompute and validate intermediate steps. Furthermore, provenance records

establish accountability by acting as indisputable proofs of a node's behavior. This not only aids in detecting malicious actors but also protects honest nodes from false accusations.

The ShieldDINC framework in [7] focuses on data confidentiality and integrates Homomorphic Encryption (HE) in ICN to enable secure computations on encrypted data. ShieldDINC performs computations directly on encrypted data using the Brakerski-Fan-Vercauteren HE scheme, ensuring that sensitive information remains confidential. This approach not only prevents data leakage but also enables secure result sharing by caching computed results at intermediate routers. To ensure data integrity and authenticity, computed results are signed by the computing nodes and can be verified by the requesting entities.

5. Open Challenges and Research Perspectives

In this section, we summarize the main open challenges and research perspectives involved in the realization of INC via ICN.

5.1. Line-Speed Executors

Deploying line-speed executors, i.e., systems capable of performing real-time analysis and decision making at the speed of network traffic, is still a significant technical challenge and only a few works so far have practically evaluated their performance [70].

A preliminary architectural description of an NDN-enabled software router integrating in-network computing capabilities is presented in [71]. Services are abstracted as distinct virtual machines (VMs) and managed by a local monitor that oversees resource utilization and service deployment. The proposed architecture includes mechanisms to dynamically optimize VM placement based on service demands and resource availability. However, the paper provides only simulation results as a proof-of-concept, using a simple line topology, with evaluations focused on traffic processing efficiency and service invocation delay. Practical implementation is left for future work.

Currently, programmable switches and smart NICs are designed primarily for high-speed packet forwarding, not for complex computations: they have limited memory, computational power, and processing stages. For example, a high-performance switch like Intel Tofino offers 12–20 stages in its pipeline [72], and these must be shared between essential network functions (e.g., routing, ACLs) and in-network computations. Also, depending on their position, network devices must handle from hundreds to millions of packets per second, making problematic the parallel execution of different services. Although technologies like Intel Tofino or Smart NICs are making line-speed executors feasible for specific tasks, scaling these solutions to support diverse, complex workloads remains a challenge requiring careful optimization and continued innovation.

The integration of ICN service names in the forwarding fabric introduces additional considerations for line-speed execution, due to the semantic information about the service or data being requested. This content-awareness can facilitate policy-driven and context-aware service execution, enabling SmartNICs to act as efficient intermediaries that prioritize traffic based on service-level requirements. However, the processing of ICN service names, which often requires name matching, hierarchical resolution, or context-specific decisions, can strain the limited processing resources of SmartNICs and switches. Advancements in both hardware and software technologies are required to fully exploit the potential of content-aware and policy-driven service execution.

5.2. Optimal Service Placement

Optimal placement of services that maximize performance, while minimizing resource utilization and latency, is a key factor in INC scenarios. Placement decisions must account for a complex interplay of factors, including application requirements, network condi-

tions, and resource availability, while ensuring scalability and adaptability to dynamic traffic patterns [73].

Centralized placement decision systems offer the advantage of holistic network visibility, enabling comprehensive optimization of in-network task placement. However, they require the real-time collection of extensive data from across the network, which can introduce communication overhead and potential bottlenecks. On the other hand, distributed placement decisions leverage localized information to reduce overhead and improve responsiveness. While these approaches are more scalable, they can lead to suboptimal placements if the global network state is not adequately considered.

In both centralized and distributed approaches, the selection of input parameters is pivotal to the success of placement decisions. Input parameters may include application-specific requirements such as latency tolerance, bandwidth needs, and computational load, as well as network-specific factors like link capacity, node resource availability, and energy consumption constraints. Incorporating dynamic traffic recognition methods adds a critical layer of adaptability to the decision-making process. By classifying data flows into categories such as latency-sensitive or bandwidth-intensive, traffic recognition can guide service placement to dynamically adjust to current network conditions [74]. In parallel, ICN can support traffic-aware placement strategies, e.g., where popular or latency-sensitive services are proactively placed near end-users, minimizing delays and enhancing overall performance.

Cross-layer technologies like Cognitive Radio (CR) could assist service placement by identifying nodes with optimal connectivity and channel availability, ensuring seamless communication between them [75]. Integrated with ICN, CR can dynamically manage spectrum access in highly variable 6G environments, enabling ICN nodes to operate across frequency bands that best meet the QoS requirements of applications. For instance, CR can complement ICN's content-based forwarding and caching by adapting to fluctuating network conditions, ensuring timely and reliable content delivery and computation, even in spectrum-constrained scenarios [76].

Emerging 6G technologies such as Reconfigurable Intelligent Surfaces (RIS) can further enhance ICN's service placement strategies. For example, RIS can dynamically reconfigure the wireless environment by altering signal propagation characteristics, enabling ICN nodes to maintain optimized communication paths and reduce service provisioning latency. AI techniques can also complement placement strategies by dynamically identifying content and service popularity trends and predicting access patterns [77].

5.3. Service Function Chaining

Service Function Chaining (SFC) is essential for organizing and managing service delivery at the network edge. By enabling the dynamic sequencing of virtualized service functions in the network, this approach can optimize resource utilization and distribute the load, which is crucial in environments where computational resources are limited [78]. From a network performance optimization point of view, deploying service chains poses several challenges, including (i) strategically placing virtualized functions while designing efficient resource allocation mechanisms, and (ii) formulating routing policies to manage traffic flows across service nodes, particularly when multiple instances of the same network function exist.

The integration of INC and ICN in the context of SFC presents numerous research opportunities [79]. Unlike traditional approaches, which direct traffic towards specific nodes, ICN can directly address service functions at the network layer. At each step of the service chain, a request can be fulfilled by any available node hosting the requested function, rather than a single pre-defined endpoint. This reduces reliance on static and predefined routes and inherently improves the flexibility and resilience of service delivery. However, intelligent algo-

rithms are required to dynamically place and migrate named service functions across nodes to optimize latency, resource utilization, and service availability. In addition, new forwarding mechanisms that consider function availability, network congestion, and computational load are needed to allow efficient and robust traffic forwarding across service chains.

5.4. Collaborative Caching

With computational capabilities integrated into network nodes, caching extends beyond traditional content delivery to end-users, enabling cached data to also serve as input for in-network processing. This shift requires caching strategies that consider both storage and computation needs, ensuring that data are placed not only near users but also close to execution spots to minimize processing delays and maximize efficiency.

As cached data may be transformed or modified during in-network computations, maintaining cache consistency becomes a critical challenge [80]. Inconsistent or outdated cache entries can lead to incorrect processing results or unnecessary re-computation, undermining the benefits of INC. Effective cache management requires mechanisms to propagate updates across the network efficiently, particularly in dynamic environments with frequent data changes. Techniques such as versioning, metadata tagging, and lightweight synchronization protocols can help ensure that all nodes access consistent and up-to-date data while minimizing overhead.

In parallel, coordinating caching and computation across a large-scale network with diverse applications is non-trivial. Decentralized caching decisions may lead to suboptimal global performance, while centralized approaches may not scale efficiently. Collaborative protocols and distributed algorithms are needed to enable nodes to share information and make coordinated caching and computation decisions.

5.5. Semantic Routing

Semantic routing, which involves routing decisions based on the meaning and context of data and services rather than purely syntactic identifiers, offers a promising direction for advancing network efficiency and adaptability.

The possibility of deploying semantic routing in ICN has been preliminary discussed in [81]. There, a fuzzy Interest forwarding approach that leverages semantic similarity is proposed to enable approximate name matching for Interest packets. Fuzzy matching is incorporared in both the CS and FIB, supported by a semantic similarity function which dynamically balances precision and uncertainty in data retrieval. As a result, data retrieval is performed even when consumers and producers do not share exact naming conventions.

The integration of INC and ICN may introduce transformative opportunities in this domain, fostering the development of advanced service discovery mechanisms. A key research perspective lies in designing semantic-aware forwarding mechanisms that leverage ICN's data-centric nature. This involves embedding semantic information into ICN's hierarchical naming schemes to enable efficient routing decisions based on content type, application requirements, or context-specific metadata. Combining this with INC capabilities allows intermediate nodes to process and aggregate data directly within the network, reducing latency and minimizing unnecessary data transfers. Another critical direction is the exploration of machine learning techniques to enhance semantic routing. INC-enabled nodes can employ federated learning or distributed inference models to dynamically predict optimal paths and service placements based on the semantic characteristics of traffic flows. These models, integrated in the ICN forwarding fabric, can adapt to real-time network conditions, such as traffic congestion or node availability, enhancing both routing accuracy and overall network performance.

From a resource optimization perspective, there is an opportunity to investigate mechanisms that allocate computational and networking resources for semantic routing in ICN-based environments. This includes developing algorithms for joint optimization of storage, processing, and bandwidth at INC-enabled nodes, ensuring that semantic processing does not overwhelm the network's resource constraints.

5.6. Security and Privacy

Coupling INC with ICN introduces intricate security and privacy challenges that demand innovative solutions. Research should focus on developing end-to-end trust frameworks that enable the verification of computation correctness and data authenticity. As discussed in [69], provenance tracking systems can play a pivotal role by documenting the lifecycle of computational tasks, allowing consumers to validate results without requiring task re-execution. Preserving privacy during computation is equally critical [82], particularly for sensitive data in applications like healthcare and manufacturing. Approaches such as homomorphic encryption allow computations on encrypted data, safeguarding privacy without compromising functionality. Complementary to this, trusted execution environments provide a secure enclave for isolated computations, ensuring that even resource-constrained nodes can execute sensitive tasks safely.

In addition to trust and privacy, resilience to adversarial environments is paramount. Distributed networks face threats such as tampered computations and DoS attacks, which can undermine system integrity and availability. Effective solutions should include mechanisms for real-time anomaly detection and fault tolerance, supported by redundancy and request authentication to maintain reliability. Furthermore, ensuring the integrity of cached computation results is vital. Cryptographic signatures can authenticate outputs, while strategies for maintaining cache freshness and consistency will prevent the reuse of outdated or corrupted results.

Lastly, as INC deployments grow in scale, efficient key management will become critical to ensure seamless and secure operation. Scalable methods for key distribution and rotation must be devised to support large-scale networks with minimal overhead. Application-specific security protocols, tailored to the unique demands of fields like healthcare and autonomous systems, will also be necessary to mitigate risks where errors or breaches could have significant consequences.

6. Conclusions

In this paper, we explored the synergy among INC and ICN paradigms. By embedding computational capabilities directly into the communication fabric, INC can enable ultra-low latency, efficient resource utilization, and scalable task distribution, making it a promising approach for 6G and beyond environments. Moreover, ICN's distinguished features, such as in-network caching, stateful forwarding, and flexible naming, complement INC by providing a robust framework for content and service provisioning.

We reviewed several methodologies, including service naming schemes and executor selection strategies, to enable INC via ICN. Additionally, we discussed several research perspectives in different contexts, ranging from the deployment of line-speed executors to SFC and semantic routing.

In conclusion, the synergy between INC and ICN represents a significant step toward realizing a seamless convergence of 3C resources, paving the way for intelligent and adaptive network systems.

Author Contributions: All authors have contributed equally to the work. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by the national research project PRIN 2022, entitled "TOGETHER: models and algoriThms for 6G Era digital Twin orcHEstratoR", CUP: C53D23000450006, and partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001—program "RESTART").

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

3C	Communication, caching and computing
ACL	Access control list
AI	Artificial intelligence
API	Application programming interface
CFN	Compute first networking
CR	Cognitive radio
CS	Content store
CPU	Central processing unit
DDoS	Distributed denial of service
DT	Digital twin
DTN	Digital twin network
FIB	Forwarding information base
FPGA	Field-programmable gate array
GPU	Graphics processing unit
ICN	Information-centric networking
ICedge	Information-centric edge
IETF	Internet engineering task force
INC	In-network computing
IRS	Intelligent reflecting surfaces
IRTF	Internet research task force
IoT	Internet of Things
IP	Internet protocol
MEC	Multi-access edge computing
NACK	Negative acknowledgment
NCN	Named computation networking
NDN	Named data networking
NFaaS	Named function as a service
NFN	Named function networking
NFV	Network function virtualization
NIC	Network interface card
PIT	Pending interest table
QoS	Quality of service
RIB	Routing information base
RICE	Remote invocation in named function networking
SDN	Software-defined networking
SFC	Service Function Chaining
SSL	Secure socket layer
SVS	State vector synchronization
TCP	Transmission control protocol
TLV	Type–length–value
TLS	Transport layer security

URL	Uniform resource locator
VLC	Visible light communications
VM	Virtual machine
XR	Extended reality

References

1. Wang, C.X.; You, X.; Gao, X.; Zhu, X.; Li, Z.; Zhang, C.; Wang, H.; Huang, Y.; Chen, Y.; Haas, H.; et al. On the road to 6G: Visions, requirements, key technologies, and testbeds. *IEEE Commun. Surv. Tutor.* **2023**, *25*, 905–974. [CrossRef]
2. Giordani, M.; Polese, M.; Mezzavilla, M.; Rangan, S.; Zorzi, M. Toward 6G networks: Use cases and technologies. *IEEE Commun. Mag.* **2020**, *58*, 55–61. [CrossRef]
3. Zhang, X.; Debroy, S. Resource management in mobile edge computing: A comprehensive survey. *ACM Comput. Surv.* **2023**, *55*, 1–37. [CrossRef]
4. Amadeo, M.; Ruggeri, G.; Campolo, C.; Molinaro, A. IoT services allocation at the edge via named data networking: From optimal bounds to practical design. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 661–674. [CrossRef]
5. Zheng, C.; Tang, H.; Zang, M.; Hong, X.; Feng, A.; Tassiulas, L.; Zilberman, N. DINC: Toward distributed in-network computing. *Proc. ACM Netw.* **2023**, *1*, 1–25. [CrossRef]
6. Silva, R.; Corujo, D.; Quevedo, J.; Aguiar, R. In-network computing—challenges and opportunities. *Internet Technol. Lett.* **2024**, *7*, e487. [CrossRef]
7. Hlaing, H.H.; Asaeda, H. ShieldDINC: Privacy-Preserving Distributed In-Network Computations with Efficient Homomorphic Encryption. In Proceedings of the IEEE 49th Conference on Local Computer Networks (LCN), Caen, France, 8–10 October 2024; pp. 1–6.
8. Nour, B.; Mastorakis, S.; Mtibaa, A. Compute-less networking: Perspectives, challenges, and opportunities. *IEEE Netw.* **2020**, *34*, 259–265. [CrossRef] [PubMed]
9. Hurali, L.C.M.; Patil, A.P. Application areas of information-centric networking: State-of-the-art and challenges. *IEEE Access* **2022**, *10*, 122431–122446. [CrossRef]
10. Xylomenos, G.; Ververidis, C.N.; Siris, V.A.; Fotiou, N.; Tsilopoulos, C.; Vasilakos, X.; Katsaros, K.V.; Polyzos, G.C. A survey of information-centric networking research. *IEEE Commun. Surv. Tutor.* **2013**, *16*, 1024–1049. [CrossRef]
11. Chen, Q.; Xie, R.; Yu, F.R.; Liu, J.; Huang, T.; Liu, Y. Transport control strategies in named data networking: A survey. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2052–2083. [CrossRef]
12. Saxena, D.; Raychoudhury, V.; Suri, N.; Becker, C.; Cao, J. Named data networking: A survey. *Comput. Sci. Rev.* **2016**, *19*, 15–55. [CrossRef]
13. Tourani, R.; Misra, S.; Mick, T.; Panwar, G. Security, privacy, and access control in information-centric networking: A survey. *IEEE Commun. Surv. Tutor.* **2017**, *20*, 566–600. [CrossRef]
14. Tariq, A.; Rehman, R.A.; Kim, B.S. Forwarding strategies in NDN-based wireless networks: A survey. *IEEE Commun. Surv. Tutor.* **2019**, *22*, 68–95. [CrossRef]
15. Khelifi, H.; Luo, S.; Nour, B.; Moungla, H.; Faheem, Y.; Hussain, R.; Ksentini, A. Named data networking in vehicular ad hoc networks: State-of-the-art and challenges. *IEEE Commun. Surv. Tutor.* **2019**, *22*, 320–351. [CrossRef]
16. Serhane, O.; Yahyaoui, K.; Nour, B.; Moungla, H. A survey of ICN content naming and in-network caching in 5G and beyond networks. *IEEE Internet Things J.* **2020**, *8*, 4081–4104. [CrossRef]
17. Chen, C.; Wang, C.; Qiu, T.; Atiquzzaman, M.; Wu, D.O. Caching in vehicular named data networking: Architecture, schemes and future directions. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2378–2407. [CrossRef]
18. Nour, B.; Khelifi, H.; Hussain, R.; Mastorakis, S.; Moungla, H. Access control mechanisms in named data networks: A comprehensive survey. *Acm Comput. Surv. (cSuR)* **2021**, *54*, 1–35. [CrossRef]
19. Kianpisheh, S.; Taleb, T. A survey on in-network computing: Programmable data plane and technology specific applications. *IEEE Commun. Surv. Tutor.* **2022**, *25*, 701–761. [CrossRef]
20. Karim, F.A.; Aman, A.H.M.; Hassan, R.; Nisar, K.; Uddin, M. Named data networking: A survey on routing strategies. *IEEE Access* **2022**, *10*, 90254–90270. [CrossRef]
21. Zheng, C.; Hong, X.; Ding, D.; Vargaftik, S.; Ben-Itzhak, Y.; Zilberman, N. In-network machine learning using programmable network devices: A survey. *IEEE Commun. Surv. Tutor.* **2023**, *26*, 1171–1200. [CrossRef]
22. Pruthvi, C.; Vimala, H.; Shreyas, J. A systematic survey on content caching in ICN and ICN-IoT: Challenges, approaches and strategies. *Comput. Netw.* **2023**, *233*, 109896.
23. Kfoury, E.F.; Choueiri, S.; Mazloum, A.; AlSabeh, A.; Gomez, J.; Crichigno, J. A comprehensive survey on smartnics: Architectures, development models, applications, and research directions. *IEEE Access* **2024**, *12*, 107297–107336. [CrossRef]
24. Alhawas, A.; Belghith, A. Software-Defined Named Data Networking in Literature: A Review. *Future Internet* **2024**, *16*, 258. [CrossRef]

25. Nickel, M.; Göhringer, D. A Survey on Architectures, Hardware Acceleration and Challenges for In-Network Computing. *ACM Trans. Reconfig. Technol. Syst.* **2024**, *18*, 1–34. [CrossRef]
26. Zhu, H.; Jiang, W.; Hong, Q.; Guo, Z. When In-Network Computing Meets Distributed Machine Learning. *IEEE Netw.* **2024**, *38*, 238–246. [CrossRef]
27. Tennenhouse, D.L.; Wetherall, D.J. Towards an active network architecture. *ACM SIGCOMM Comput. Commun. Rev.* **1996**, *26*, 5–17. [CrossRef]
28. Legedza, U.; Wetherall, D.; Guttag, J. Improving the performance of distributed applications using active networks. In Proceedings of the IEEE INFOCOM'98, San Francisco, CA, USA, 29 March–2 April 1998; Volume 2, pp. 590–599.
29. Bouras, C.; Kollia, A.; Papazois, A. SDN & NFV in 5G: Advancements and challenges. In Proceedings of the 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris, France, 7–9 March 2017; pp. 107–111.
30. Zhang, L.; Afanasyev, A.; Burke, J.; Jacobson, V.; Claffy, K.; Crowley, P.; Papadopoulos, C.; Wang, L.; Zhang, B. Named data networking. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 66–73. [CrossRef]
31. Amadeo, M.; Campolo, C.; Ruggeri, G.; Molinaro, A.; Iera, A. SDN-managed provisioning of named computing services in edge infrastructures. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 1464–1478. [CrossRef]
32. Al-Hammadi, I.; Li, M.; Islam, S.M.; Al-Mosharea, E. Collaborative computation offloading for scheduling emergency tasks in SDN-based mobile edge computing networks. *Comput. Netw.* **2024**, *238*, 110101. [CrossRef]
33. Shih, Y.Y.; Lin, H.P.; Pang, A.C.; Chuang, C.C.; Chou, C.T. An NFV-based service framework for IoT applications in edge computing environments. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 1419–1434. [CrossRef]
34. Karrakchou, O.; Samaan, N.; Karmouch, A. Endn: An enhanced ndn architecture with a p4-programmabie data plane. In Proceedings of the 7th ACM Conference on Information-Centric Networking, Virtual, 29 September–1 October 2020; pp. 1–11.
35. Zhou, Y.; Liu, L.; Wang, L.; Hui, N.; Cui, X.; Wu, J.; Peng, Y.; Qi, Y.; Xing, C. Service-aware 6G: An intelligent and open network based on the convergence of communication, computing and caching. *Digit. Commun. Netw.* **2020**, *6*, 253–260. [CrossRef]
36. Li, Y.; HE, J.; Geng, L.; Liu, P.; Cui, Y. Framework of Compute First Networking (CFN). Internet-Draft Draft-li-Rtgwg-cfn-Framework-00, Internet Engineering Task Force. 2019. Work in Progress. Available online: https://datatracker.ietf.org/doc/draft-li-rtgwg-cfn-framework/ (accessed on 8 January 2024).
37. Clemm, A.; Vega, M.T.; Ravuri, H.K.; Wauters, T.; De Turck, F. Toward truly immersive holographic-type communication: Challenges and solutions. *IEEE Commun. Mag.* **2020**, *58*, 93–99. [CrossRef]
38. Cardenas-Robledo, L.A.; Hernández-Uribe, Ó.; Reta, C.; Cantoral-Ceballos, J.A. Extended reality applications in industry 4.0.–A systematic literature review. *Telemat. Inform.* **2022**, *73*, 101863. [CrossRef]
39. Strinati, E.C.; Barbarossa, S.; Gonzalez-Jimenez, J.L.; Ktenas, D.; Cassiau, N.; Maret, L.; Dehos, C. 6G: The next frontier: From holographic messaging to artificial intelligence using subterahertz and visible light communication. *IEEE Veh. Technol. Mag.* **2019**, *14*, 42–50. [CrossRef]
40. Akyildiz, I.F.; Guo, H. Holographic-type communication: A new challenge for the next decade. *ITU J. Future Evol. Technol.* **2022**, *3*, 421–442. [CrossRef]
41. Petkova, R.; Bozhilov, I.; Manolova, A.; Tonchev, K.; Poulkov, V. On the Way to Holographic-Type Communications: Perspectives and Enabling Technologies. *IEEE Access* **2024**, *12*, 59236–59259. [CrossRef]
42. Hou, W.; Bai, B.; Lyu, L. Evaluation Model of Holographic Communication Experience in the 6G Era Based on Light Field Display. *Appl. Sci.* **2023**, *13*, 12381. [CrossRef]
43. Guo, Q.; Tang, F.; Rodrigues, T.K.; Kato, N. Five disruptive technologies in 6G to support digital twin networks. *IEEE Wirel. Commun.* **2023**, *31*, 149–155. [CrossRef]
44. Wu, Y.; Zhang, K.; Zhang, Y. Digital twin networks: A survey. *IEEE Internet Things J.* **2021**, *8*, 13789–13804. [CrossRef]
45. Yigit, Y.; Ahmadi, H.; Yurdakul, G.; Canberk, B.; Hoang, T.; Duong, T.Q. Digi-infrastructure: Digital twin-enabled traffic shaping with low-latency for 6G smart cities. *IEEE Commun. Stand. Mag.* **2024**, *8*, 28–34. [CrossRef]
46. Okegbile, S.D.; Cai, J.; Niyato, D.; Yi, C. Human digital twin for personalized healthcare: Vision, architecture and future directions. *IEEE Netw.* **2022**, *37*, 262–269. [CrossRef]
47. Amadeo, M.; Marche, C.; Ruggeri, G.; Ranjbaran, S.; Nitti, M. Composing Digital Twins for Internet of Everything Applications: A User-Centric Perspective. In Proceedings of the IEEE International Mediterranean Conference on Communications and Networking (MeditCom), Madrid, Spain, 8–11 July 2024; pp. 73–78.
48. Mastorakis, S.; Mtibaa, A.; Lee, J.; Misra, S. Icedge: When edge computing meets information-centric networking. *IEEE Internet Things J.* **2020**, *7*, 4203–4217. [CrossRef]
49. Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.F.; Briggs, N.H.; Braynard, R.L. Networking named content. In Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, Rome, Italy, 1–4 December 2009; pp. 1–12.
50. Amadeo, M.; Campolo, C.; Lia, G.; Molinaro, A.; Ruggeri, G. In-network placement of reusable computing tasks in an SDN-based network edge. *IEEE Trans. Mob. Comput.* **2023**, *23*, 1456–1471. [CrossRef]

51. Shah, M.S.M.; Leau, Y.B.; Anbar, M.; Bin-Salem, A.A. Security and integrity attacks in named data networking: A survey. *IEEE Access* **2023**, *11*, 7984–8004. [CrossRef]
52. Jiang, X.; Bi, J.; Wang, Y. What benefits does NDN have in supporting mobility. In Proceedings of the IEEE Symposium on Computers and Communications (ISCC), Madeira, Portugal, 23–26 June 2014; pp. 1–6.
53. Augé, J.; Carofiglio, G.; Grassi, G.; Muscariello, L.; Pau, G.; Zeng, X. Map-me: Managing anchor-less producer mobility in content-centric networks. *IEEE Trans. Netw. Serv. Manag.* **2018**, *15*, 596–610. [CrossRef]
54. Wang, X.; Cai, S. Edge-assisted NDN-based IoT framework with provider and consumer mobility support. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 1713–1725. [CrossRef]
55. Wang, L.; Lane, A.; Serban, C.; Elwell, J.; Afanasyev, A.; Zhang, L. Investigating the Synergy between Routing and Forwarding Strategy in NDN Networks. In Proceedings of the 10th ACM Conference on Information-Centric Networking, Reykjavik, Iceland, 8–10 October 2023; pp. 67–77.
56. Compagno, A.; Conti, M.; Ghali, C.; Tsudik, G. To NACK or not to NACK? negative acknowledgments in information-centric networking. In Proceedings of the IEEE 24th International Conference on Computer Communication and Networks (ICCCN), Las Vegas, NV, USA, 3–6 August 2015; pp. 1–10.
57. Yi, C.; Abraham, J.; Afanasyev, A.; Wang, L.; Zhang, B.; Zhang, L. On the role of routing in named data networking. In Proceedings of the 1st ACM Conference on Information-Centric Networking, Paris, France, 24–26 September 2014; pp. 27–36.
58. Tschudin, C.; Sifalakis, M. Named functions and cached computations. In Proceedings of the IEEE CCNC, Las Vegas, NV, USA, 10–13 January 2014; pp. 851–857.
59. Scherb, C.; Faludi, B.; Tschudin, C. Execution state management in named function networking. In Proceedings of the IFIP Networking Conference (IFIP Networking) and Workshops, Stockholm, Sweden, 12–16 June 2017; pp. 1–6.
60. Amadeo, M.; Campolo, C.; Molinaro, A.; Ruggeri, G. IoT data processing at the edge with named data networking. In Proceedings of the European Wireless, Catania, Italy, 2–4 May 2018; VDE: Berlin, Germany, 2018; pp. 1–6.
61. Pirmagomedov, R.; Srikanteswara, S.; Moltchanov, D.; Arrobo, G.; Zhang, Y.; Himayat, N.; Koucheryavy, Y. Augmented computing at the edge using named data networking. In Proceedings of the IEEE Globecom Workshops (GC Wkshps), Virtual, 7–11 December 2020; pp. 1–6.
62. Król, M.; Psaras, I. NFaaS: Named function as a service. In Proceedings of the 4th ACM Conference on Information-Centric Networking, Berlin, Germany, 26–28 September 2017; pp. 134–144.
63. Król, M.; Habak, K.; Oran, D.; Kutscher, D.; Psaras, I. Rice: Remote method invocation in icn. In Proceedings of the 5th ACM Conference on Information-Centric Networking, Boston, MA, USA, 21–23 September 2018; pp. 1–11.
64. Ye, Y.; Qiao, Y.; Lee, B.; Murray, N. PIoT: Programmable IoT using information centric networking. In Proceedings of the IEEE/IFIP Network Operations and Management Symposium, Istanbul, Turkey, 25–29 April 2016; pp. 825–829.
65. Scherb, C.; Grewe, D.; Wagner, M.; Tschudin, C. Resolution strategies for networking the IoT at the edge via named functions. In Proceedings of the IEEE CCNC, Las Vegas, NV, USA, 12–15 January 2018; pp. 1–6.
66. Amadeo, M.; Campolo, C.; Molinaro, A. NDNe: Enhancing named data networking to support cloudification at the edge. *IEEE Commun. Lett.* **2016**, *20*, 2264–2267. [CrossRef]
67. Kondo, D.; Ansquer, T.; Tanigawa, Y.; Tode, H. Resource Breadcrumbs: Discovering Edge Computing Resources Over Named Data Networking. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 3305–3316. [CrossRef]
68. Ambalavanan, U.; Grewe, D.; Nayak, N.; Liu, L.; Mohan, N.; Ott, J. DICer: Distributed coordination for in-network computations. In Proceedings of the 9th ACM Conference on Information-Centric Networking, Osaka, Japan, 19–21 September 2022; pp. 45–55.
69. Marxer, C.; Tschudin, C. Result Provenance in Named Function Networking. In Proceedings of the 7th ACM Conference on Information-Centric Networking, Virtual, 29 September–1 October 2020; pp. 24–29.
70. Xing, J.; Qiu, Y.; Hsu, K.F.; Sui, S.; Manaa, K.; Shabtai, O.; Piasetzky, Y.; Kadosh, M.; Krishnamurthy, A.; Ng, T.E.; et al. Unleashing SmartNIC packet processing performance in P4. In Proceedings of the ACM SIGCOMM 2023 Conference, New York, NY, USA, 10–14 September 2023; pp. 1028–1042.
71. Liang, B.; Tian, J.; Zhu, Y. A Named In-Network Computing Service Deployment Scheme for NDN-Enabled Software Router. In Proceedings of the 5th International Conference on Hot Information-Centric Networking (HotICN), Guangzhou, China, 24–26 November 2022; pp. 25–29.
72. Karlos, G.; Bal, H.; Wang, L. NetCL: A Unified Programming Framework for In-Network Computing. In Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, Atlanta, GA, USA, 17–22 November 2024; pp. 1–20.
73. Malazi, H.T.; Chaudhry, S.R.; Kazmi, A.; Palade, A.; Cabrera, C.; White, G.; Clarke, S. Dynamic service placement in multi-access edge computing: A systematic literature review. *IEEE Access* **2022**, *10*, 32639–32688. [CrossRef]
74. Serrano, S.; Sahbudin, M.A.B.; Chaouch, C.; Scarpa, M. A new fingerprint definition for effective song recognition. *Pattern Recognit. Lett.* **2022**, *160*, 135–141. [CrossRef]

75. Barrak, S.E.; Lyhyaoui, A.; Gonnouni, A.E.; Puliafito, A.; Serrano, S. Application of MVDR and MUSIC spectrum sensing techniques with implementation of node's prototype for cognitive radio ad hoc networks. In Proceedings of the 2017 International Conference on Smart Digital Environment, Rabat, Morocco, 21–23 July 2017; pp. 101–106.
76. Boukerche, A.; Coutinho, R.W.; Loureiro, A.A. Information-centric cognitive radio networks for content distribution in smart cities. *IEEE Netw.* **2019**, *33*, 146–151. [CrossRef]
77. Krishnendu, S.; Bharath, B.; Bhatia, V.; Nebhen, J.; Dobrovolny, M.; Ratnarajah, T. Wireless edge caching and content popularity prediction using machine learning. *IEEE Consum. Electron. Mag.* **2022**, *13*, 32–41.
78. Hantouti, H.; Benamar, N.; Taleb, T. Service function chaining in 5G & beyond networks: Challenges and open research issues. *IEEE Netw.* **2020**, *34*, 320–327.
79. Liu, L.; Peng, Y.; Bahrami, M.; Xie, L.; Ito, A.; Mnatsakanyan, S.; Qu, G.; Ye, Z.; Guo, H. ICN-FC: An information-centric networking based framework for efficient functional chaining. In Proceedings of the IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–7.
80. Feng, B.; Tian, A.; Yu, S.; Li, J.; Zhou, H.; Zhang, H. Efficient cache consistency management for transient IoT data in content-centric networking. *IEEE Internet Things J.* **2022**, *9*, 12931–12944. [CrossRef]
81. Chan, K.; Ko, B.; Mastorakis, S.; Afanasyev, A.; Zhang, L. Fuzzy Interest forwarding. In Proceedings of the 13th Asian Internet Engineering Conference, Bangkok, Thailand, 20–22 November 2017; pp. 31–37.
82. Sundarakantham, K.; Sivasankar, E. A hybrid deep learning framework for privacy preservation in edge computing. *Comput. Secur.* **2023**, *129*, 103209.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.