SoK: Distributed Computing in ICN*

Wei Geng
HKUST(GZ)†
Guangzhou
China

Yulong Zhang
HKUST(GZ)†
Guangzhou
China

Dirk Kutscher†
HKUST(GZ)†
Guangzhou
China

Abhishek Kumar
University of Oulu
Oulu, Finland

Sasu Tarkoma
University of Helsinki and University of Oulu
Helsinki, Finland

Pan Hui
HKUST(GZ)†and University of Helsinki
Guangzhou
China

ABSTRACT

Information-Centric Networking (ICN), with its data-oriented operation and generally more powerful forwarding layer, provides an attractive platform for distributed computing. This paper provides a systematic overview and categorization of different distributed computing approaches in ICN encompassing fundamental design principles, frameworks and orchestration, protocols, enablers, and applications. We discuss current pain points in legacy distributed computing, attractive ICN features, and how different systems use them. This paper also provides a discussion of potential future work for distributed computing in ICN.

CCS CONCEPTS

• **Networks** → **Network protocols**; **Network architectures**; • **Computing methodologies** → **Distributed computing methodologies**.

KEYWORDS

ICN, Distributed Computing

1 INTRODUCTION

Distributed computing – a model where distributed components for computing and storage communicate over a network to form a larger system – is the basis for all relevant applications on the Internet. Based on well-established principles [47], different mechanisms, implementations, and applications have been developed that form the foundation of the modern Web.

The Internet with its stateless forwarding service and end-to-end communication model [70] promotes certain types of communication for distributed computing. For example, IP addresses and/or DNS names provide different means for identifying computing components. Reliable transport protocols (e.g., TCP, QUIC) promote interconnecting modules. Communication patterns such as REST [19] and protocol implementations such as HTTP enable certain types of distributed computing interactions, and security frameworks such as TLS and the web PKI [55] constrain the use of public-key cryptography for different security functions.

With currently available Internet technologies, we can observe a relatively succinct layering of networking and distributed computing, i.e., distributed computing is typically implemented in overlays with Content Distribution Networks (CDNs) being prominent and ubiquitous example. Recently, there has been growing interest in revisiting this relationship, for example by the IRTF Computing in the Network Research Group (COINRG)[1] – motivated by advances in network and server platforms, e.g., through the development of programmable data plane platforms [66] and the development of different types of distributed computing frameworks, e.g., stream processing [35] and microservice frameworks. [15] This is also motivated by the recent development of new distributed computing applications such as distributed machine learning (ML) [105] and emerging new applications such as Metaverse suggest new levels of scale in terms of data volume for distributed computing and the pervasiveness of distributed computing tasks in such systems. There are two research questions that stem from these developments:

1) How can we build distributed computing systems in the network that can leverage the on-path location of compute functions, e.g., optimally aligning stream processing topologies with networked computing platform topologies?

2) How can the *network* support distributed computing in general, so that the design and operation of such systems can be simplified, but also so that different optimizations can be achieved to improve performance and robustness?

ICN (we focus mainly on CCNx/NDN-based ICN in this paper) with its data-oriented operation and generally more powerful forwarding layer provides an attractive platform for distributed computing. Several different distributed computing protocols and systems have been proposed for ICN, with different feature sets and different technical approaches, including Remote Method Invocation (RMI) as an interaction model as well as more comprehensive distributed computing platforms. RMI systems such as RICE [39] leverage the fundamental named-based forwarding service in ICN systems [108] and map requests to *Interest* messages and method names to content names (although the actual implementation is more intricate as we will explain below). Method parameters and results are also represented as content objects, which provides an elegant platform for such interactions.

This SoK paper provides a comprehensive analysis and understanding of distributed computing systems in ICN, based on a survey of more than 50 papers. Naturally, these different efforts cannot be directly compared due to their difference in nature. We categorized different ICN distributed computing systems, and individual approaches and highlighted their specific properties. The scope of this study is *technologies for **ICN-enabled distributed computing***. Specifically, we divide the different approaches into four categories, as shown in figure 1: enablers, protocols, orchestration, and applications. The contributions of this study are as follows:

*The manuscript has been accepted to appear in the proceedings of ACM ICN 2023.
†D. Kutscher is the corresponding author. dku@ust.hk
†HKUST(GZ) is The Hong Kong University of Science and Technology (Guangzhou).

ACM ICN '23, October 9–10, 2023, Reykjavik, Iceland
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1] https://irtf.org/coinrg

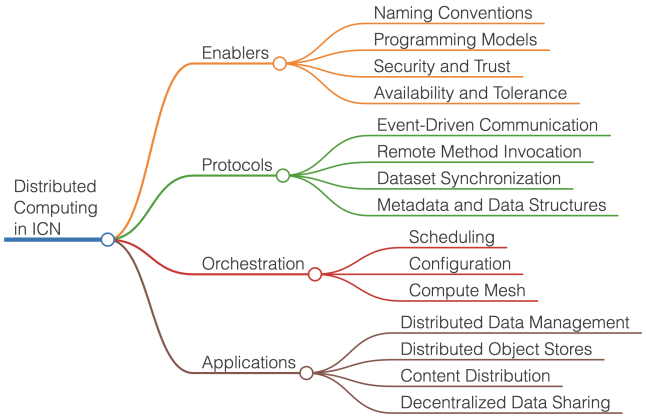

Figure 1: A category of distributed computing systems in ICN.

(1) A discussion of the benefits and challenges of distributed computing in ICN.
(2) A categorization of different proposed distributed computing systems in ICN.
(3) A discussion of lessons learned from these systems.
(4) A discussion of existing challenges and promising directions for future work.

The rest of this paper is structured as follows section 2 describes relevant general distributed computing concepts and the non-ICN state-of-the-art for reference; section 3 presents different technologies that enable distributed computing in ICN; section 4 analyzes various protocols dedicated to in-network computing; section 5 summarizes scheduling and computing approaches; section 6 presents ICN-enabled distributed computing applications; and section 7 discusses lessons learned and suggests future directions for distributed computing.

2 DISTRIBUTED COMPUTING IN ICN

Distributed computing has different facets, for example, client-server computing, web services, stream processing, distributed consensus systems, and Turing-complete distributed computing platforms. There are also different perspectives on how distributed computing should be implemented on servers and network platforms, a research area that we refer to as *Computing in the Network*. Active Networking [99], one of the earliest works on computing in the network, intended to inject programmability and customization of data packets in the network itself; however, security and complexity considerations proved to be major limiting factors, preventing its wider deployment [43]. *Dataplane programmability* [6] refers to the ability to program behavior, including application logic, on network elements and SmartNICs, thus enabling some form in-network computing. Alternatively, different types of server platforms and light-weight execution environments are enabling other forms of distributing computation in networked systems, such as architectural patterns, such as *edge computing*.

In this study, we focus on distributed computing and on how information-centricity in the network and application layer can support the development and operation of such systems. The rich set of distributed computing systems in ICN suggests that ICN provides some benefits for distributed computing that could offer advantages such as better performance, security, and productivity when building corresponding applications. We discuss issues in legacy distributed computing in section 2.1, ICN features to address some of these issues in section 2.2, and performance metrics that are frequently used by different ICN distributed computing approaches in section 2.3.

2.1 Issues in Legacy Distributed Computing

Although there are many distributed computing applications, it is also worth noting that there are many limitations and performance issues [101]. Factors such as network latency, data skew, checkpoint overhead, back pressure, garbage collection overhead, and issues related to performance, memory management, and serialization and deserialization overhead can all influence the efficiency. Various optimization techniques can be implemented to alleviate these issues, including memory adjustment, refining the checkpointing process, and adopting efficient data structures and algorithms.

Some performance problems and complexity issues stem from the overlay nature of current systems and their way of achieving the above-mentioned mechanisms with temporary solutions based on TCP/IP and associated protocols such as DNS. For example, Network Service Mesh has been characterized as architecturally complex [71] because of the so-called *sidecar* approaches and their implementation problems.

In systems that are layered on top of HTTP or TCP (or QUIC), compute nodes typically cannot assess the network performance directly – only indirectly through observed throughput and buffer under-runs. Information-centric data-flow systems, such as IceFlow [42], claim to provide better visibility and thus better joint optimization potential by more direct access to data-oriented communication resources. Then, some coordination tasks that are based on exchanging updates of shared application state can be elegantly mapped to named data publication in a hierarchical namespace, as the different dataset synchronization (Sync) protocols (discussed in section 4.3) in NDN demonstrated. [68]

2.2 Information-Centric Distributed Computing

ICN generally attempts to provide a more useful service to data-oriented applications but can also be leveraged to support distributed computing specifically.

Names: Accessing named data in the network as a native service can remove the need for mapping application logic identifiers such as function names to network and process identifiers (IP addresses, port numbers), thus simplifying implementation and run-time operation, as demonstrated by systems such as Named Function Networking (NFN) [103], RICE [39], and IceFlow [42]. It is worth noting that, although ICN does not generally require an explicit mapping of names to other domain identifiers, such networks require suitable forwarding state, e.g., obtained from configuration, dynamic learning, or routing.

Data-orientedness: ICN's notion of immutable data with strong name-content binding through cryptographic signatures and hashes seems to be conducive to many distributed computing scenarios, as both static data objects and dynamic computation results in those

systems such as input parameters and result values can be directly sent as ICN data objects. NFN has first demonstrated this.

Securing distributed computing could be supported better in so far as ICN does not require additional dependencies on public-key or pipe securing infrastructure, as keys and certificates are simply named data objects and centralized trust anchors are not necessarily needed [2, 21]. Larger data collections can be aggregated and re-purposed by manifests (FLIC, [102]), enabling "small" and "big data" computing in one single framework that is congruent to the packet-level communication in a network. IceFlow uses such an aggregation approach to share identical stream processing results objects in multiple consumer contexts.

Data-orientedness eliminates the need for connections; even reliable communication in ICN is completely data-oriented. If higher-layer (distributed computing) transactions can be mapped to the network layer data retrieval, then server complexity can be reduced (no need to maintain several connections), and consumers get direct visibility into network performance. This can enable performance optimizations, such as linking network and computing flow control loops (one realization of *joint optimization*), as showed by IceFlow.

Location independence and data sharing: Embracing the principle of accessing named and authenticated data also enables location independence, i.e., corresponding data can be obtained from any place in the network, such as replication points (*repos*) and caches. This fundamentally enables better multi-source/path capabilities as well as data sharing, i.e., multiple data retrieval operations for one named data object by different consumers can potentially be completed by a cache, repo, or peer in the network.

Stateful Forwarding: ICN provides stateful, symmetric forwarding, which enables general performance optimizations such as in-network retransmissions, more control over multipath forwarding, and load balancing. This concept could be extended to support distributed computing specifically, for example, if load balancing is performed based on RTT observations for idempotent remote-method invocations.

More Networking, less Management: The combination of data-oriented, connection-less operation, and stateful (more powerful) forwarding in ICN shifts functionality from management and orchestration layers (back) to the network layer, which can enable complexity reduction, which can be especially pronounced in distributed computing. For example, legacy stream processing and service mesh platforms typically must manage connectivity between deployment units (pods in Kubernetes [9]). In Apache Flink [8], a central orchestrator manages the connections between *task managers* (node agents). Systems such as IceFlow have demonstrated a more self-organized and decentralized stream-processing approach, and the presented principles are applicable to other forms of distributed computing.

Summary: In summary, we can observe that ICN's general approach of having the network providing a more natural (data retrieval) platform for applications benefits distributed computing in similar ways as it benefits other applications. One particularly promising approach is the elimination of layer barriers, which enables certain optimizations. In addition to NFN, there are other approaches that jointly optimize the utilization of network and computing resources to provide network service mesh-like platforms, such as edge intelligence using federated learning [85], advanced CDNs where nodes can dynamically adapt to user demands according to content popularity [25, 72], and general computing systems [40, 46, 64].

2.3 System Metrics

Table 1: Metrics Overview

Metrics	Related Works	Count
End-to-end Latency	R2 [84], NFaaS [41], [95], [37], ICedge [64], NFN [93], DICer [3]	7
Fault Tolerance	[88], ICedge [64], CFN [40]	3
Transmission Latency	C3PO [106], ICN with edge for 5G [104]	2
Computing Latency	Serving at the Edge [17], CFN [40]	2
System Overhead	Serving at the Edge [17], ICedge [64]	2
Load Balance	NFaaS [41], C3PO [106]	2
Latency Evolution	NFaaS [41]	1
Resource Utilization	DICer [3]	1
Satisfication Rate	NFaaS [41]	1
Compute Result Reuse	[95]	1
Drop Rate	C3PO [106]	1

Information-centric distributed computing systems can provide a range of optimizations, depending on the type of applications they support and the environment in which they are intended. Systems built on top of edge environments, such as ICedge [64], focus on low latency for distributed computing by leveraging knowledge about spatial proximity in edge computing scenarios, addressing applications such as IoT, mobile computing, Extended Reality (XR), and vehicular distributed computing that can benefit from offloading computation from cloud-based servers to the edge. A brief overview of these metrics is presented in Table 1. Latency is a key metric in information-centric distributed-computing systems. The end-to-end delay is mainly composed of two parts: 1) the time consumed by data packet transmission in the network, and 2) the execution time of computing tasks. In addition to studies dedicated to network transmission time [104, 106] or computation completion time [17, 40], most studies measure the total latency (end-to-end delay) [3, 37, 41, 64, 84, 93, 95] in their evaluations. It is worth noting that only a few works [32, 93, 110] have evaluated the network and computation latency respectively. To better understand systems and their optimization potential, as well as for run-time optimization, it is important to analyze network and computation latency separately. Fine-granular latency metrics can enable more accurate resource control and help us understand which resource optimization and mechanism (caching, load balancing, etc.) contributes to performance improvements. Section 3.2 discusses this from the perspective of programming models.

It is also worth noting that the distributed system should not only limit the optimization goal of the system to performance (such as latency, overhead, and balance), but also pay more attention to other metrics, especially the native security and robustness provided by ICN. Although latency accounts for a large proportion of the objective, other metrics are also being explored. For example, [17, 64] considered overhead in terms of the number of sent

packets (*Interest* and responses) and forwarder state size as important indicators for evaluating their systems. Load balancing is a key technology for improving the performance; NFaaS [41] and C3PO [106] are concerned with compute load balancing. Moreover, there are other perspectives, including fault tolerance [40, 64, 88], satisfaction rate [41], delay evolution [41], drop rate [106], times of computation result reuse [95], and resource utilization [3].

3 ENABLERS

ICN distributed computing systems are using several approaches and technologies. In this section, we discuss naming conventions, computing and performance, security and trust, and availability and tolerance.

3.1 Naming Conventions

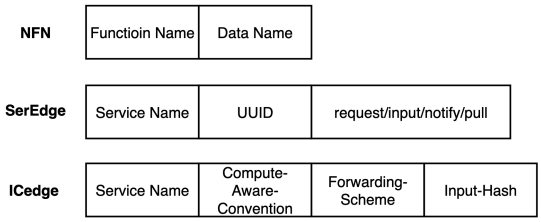

Figure 2: Three Examples of Naming Conventions, including lambda in NFN [93] hierarchical in Serving at the Edge [17], flat in ICedge [64].

Naming a computing *Interest* or process is a key concept in NFN or general distributed computing systems over ICN. As shown in figure 2, a straightforward method is to encode a lambda function into the original data name [93]. However, this simple naming convention cannot cater to complex computing process combinations and their input parameters. Many systems have proposed their own naming conventions to support complex or durable computing *Interests*. Many distributed systems name their object and computations in a hierarchical way for easier service discovery and reuse of computation results. For example, [17] proposes constructing names by concatenating service identifier, session identifier, and operating information. This convention is suitable for service *Interest* forwarding because it decouples services from server nodes. ICedge [64] uses namespaces to perform service discovery. This can facilitate service mesh implementation and a seamless user onboarding experience. More details are available in these surveys on naming conventions: [90, 98].

3.2 Programming Models

Functional programming abstracts computation as the process of mathematical function evaluation. The idea that functions are first-class citizens corresponds well with ICN's data-oriented architecture, in which data is the first-class citizen. Therefore, computation tasks are offloaded in a modular and declarative manner. Lambda is a function that can be encoded into ICN names [93, 103]. Although functional programming provides a powerful computing abstraction, other issues exist, such as simplifying concurrency management. Distributed computing systems face a similar challenge: How can multiple nodes be scheduled to work on the same task at the same time collaboratively? The actor model and message-oriented communication, such as in SmallTalk [27] and Erlang [4], try to structure systems and their communication processes to support this. In ICN, IceFlow [46] uses the actor model and dataflow concepts for parallel processing.

In the actor model, independent computing entities communicate with one another through messages. Actors own states and behaviors and can create other actors. This model enables high degrees of concurrency, scalability, and fault tolerance. Dataflow represents the task as a directed graph where nodes and edges represent computations and data flows, respectively. At the logical layer, dataflow systems typically provide multi-destination communication, as the results can be consumed by many downstream actors. This matches well with ICN's data-oriented operation and in-network replication and can be further supported by ICN caching. On the other hand, ICN's pull-based communication is not conducive to fundamentally push-based dataflow communication. IceFlow [46] addresses this by employing Sync approaches based on periodic pulling. We suggest that some push-based communication mechanisms can be adopted by dataflow-based distributed computing systems, which we discuss further in section 4.2.

Computation re-use is important for improving distributed computing systems to reduce computing redundancy. [64] defines a set of compute-aware naming conventions that cluster the computation task *Interests* at compute nodes: Subsequent identical computation requests can be forwarded to the specific compute nodes where the results already exist. [95] also aggregates identical service requests and evaluates the number of result reuse. In general, the ICN network layer can provide general caching, but cannot assist with computation re-use, which requires some computing system knowledge. Current ICN systems generally lack the ability to assess the computation re-use potential, which calls for the development of naming conventions for expressing this potential explicitly, as formulated by RICE [39].

3.3 Security and Trust

Security and trust are important enablers of distributed computing systems, especially in multiple tenant environments. Inherited from ICN, data-oriented security in distributed computing offers precise access control, confidentiality, and integrity protection at the data level by binding names and contents with digital signatures. This enables secure data sharing and resilience to system changes, thereby enhancing the overall security and reliability of distributed computing systems. Building on this, some works start from metadata. [50] proposes the use of signatures and hash codes to guarantee the security of the results from a functional chaining system. [61] adopts an access control list (ACL) structure to list all permited client identities. Through content production chains where results are produced out of results, a named object will be modified on the content load part but ACL. That means ACL keeps unchangeble to remark permited identities. Some studies have started at the protocol level. For example, RICE [39] designed a two-stage handshake protocol to provide consumer authentication and authorization.

In general, if the results of remote execution tasks are recalculated locally to verify credibility, this violates the original intention of sending the request. [62] argues that the validation of data is about verifying authenticity, while the resolution result is about correctness. This decouples authenticity from correctness, assuming that all participants are honest and that the results should be correct. This concept achieves the first verification mechanism results without recomputing the task. A decrease in the cost of verification will promote the use of distributed computing systems.

3.4 Availability and Tolerance

A distributed computing system usually comprises heterogeneous resources, such as different architectures (ARM, X86, etc.) and functions (routers, servers, etc.). In addition, edge computing environments are usually physically fragile and unreliable, for example considering edge network mobility, battery exhaustion, and link failure. These factors make the availability of distributed computing systems, for example in edge networks, a key issue. Several studies have focused on this issue. [88] provides a straightforward solution to interact with the Process Control Block (PCB) of the request computing process to achieve fault detection, task cancelation, task stopping, and task resuming. However, this solution only tackles state detection and modification and does not provide architecture-level support.

To resolve this, CFN [40] has been designed to tolerate computing node failure and the corresponding loss of *Interest* response. The framework can either choose to recompute the request upon failure detection in a proactive manner or defer the retry until the result is eventually required. The latter strategy is similar to function resolution in functional programming languages. This approach increases the complexity and the overhead of the system. Mastorakis et al.[64] claimed that avoiding a single point of failure requires a backup mechanism, and this method increases the synchronization overhead. They moved the fault tolerance function from the overlay system to the underlay network, which can implement forwarding-based recovery to avoid node failure. Our analysis indicates that fault tolerance should consider not only the task level but also both the node and network levels. Node-level solutions, such as those in CFN [40], follow the idea of traditional distributed systems. Because CFN is built on top of ICN, a flexible forwarding plane is a reliable yet simple solution to achieve this goal.

4 PROTOCOLS

This section describes the protocols for communication between distributed processes in ICN. Generally, these protocols enable applications to leverage computing and communication resources distributed in the network via higher-level APIs. We also consider the role of metadata and data structure in ICN protocols for distributed content delivery.

4.1 Remote Method Invocation

RMI is enabling client-server and peer-to-peer distributed computing scenarios, typically with the goal to provide a transparent funcation invocation service, where the invoking application does not have to be aware of the distributed nature of the system. In this section, we introduce several ICN RMI protocols. One demo by Yamamoto et al. [110] designed a protocol capable of performing the same function on multiple IoT devices such as finding objects on multiple smart cameras. Similarly, DNMP [75] defines a Pub/Sub protocol to send measurement functions to multiple devices for execution and to retrieve execution results, based on a series of libraries facilitating distributed measurement in NDN. In Named Service Calls (NSC) [65] (figure 3-NSC), a client will publish its RMI interest as data and subscribe to the corresponding result name. An NDN Sync protocol, syncps, was adopted to update all servers with information about newly published RMI *Interests*. Subsequently, all subscribed servers execute the function and publish the result to the name to which the client has subscribed. Finally, the client retrieves the result. Unlike DNMP, RICE [39], as shown in figure 3-RICE, NSC regards RMI requests as *Interests* rather than data. A four-way handshake protocol is used to establish an interactive session-like connection, and the paired clients and servers perform subsequent RMIs. RICE inserts a second, reverse, *Interest*-data interaction (for parameter transfer) between the first interaction (for authentication and RMI confirmation). In contrast to RICE, NSC [65], as shown in figure 3-NSC, adopts NDN's native security features to authenticate the client and server to provide a drop-in framework for many scenarios. NSC and RICE have similar parameter input and result retrieval processes. They both request parameters from the server and retrieve results using a *thunk* or result location after an estimated execution time. As shown by the arrow directions in figure 3, NSC has bidirectional communication between the client and server, whereas RICE has unidirectional communication from the client to the server. This means that NSC requires the client to own a routable name, whereas RICE does not. RICE maintains the anonymity of users through pull-based operations. From the perspective of parameter input (orange color parts), RICE inserts the parameter exchange period in the first RMI call which needs to extend the PIT expiration time. The extra modifications to forwarders may cause additional system complexity. NSC does not have this problem because it uses a separate *Interest*-data interaction for parameter transmission.

RICE further highlights the inadequacy of the basic *Interest*/data exchange model of CCNx/NDN-style ICN, specifically in the context of RESTful communication that adheres to the principles of the REST architectural style. This inadequacy is particularly pronounced in scenarios involving the transmission of resource representations or request parameters from clients to servers. RESTful-ICN [44] and [29] envision an ICN-based protocol framework that can leverage key properties of the REST architectural pattern, such as security, consumer anonymity, and session continuation. We see the potential of RESTful for ICN in establishing an ICN-native infrastructure for applications, encompassing an Information-Centric web and RMI rather than merely charting existing HTTP mechanisms onto ICN.

4.2 Event-Driven Communication

Event-driven protocols facilitate the exchange of information between different endpoints based on asynchronous events (that could trigger computation), instead of continuous or periodic communication as observed in request-reply protocols. ICN has a two-fold relationship to even-driven communication: On the one hand side,

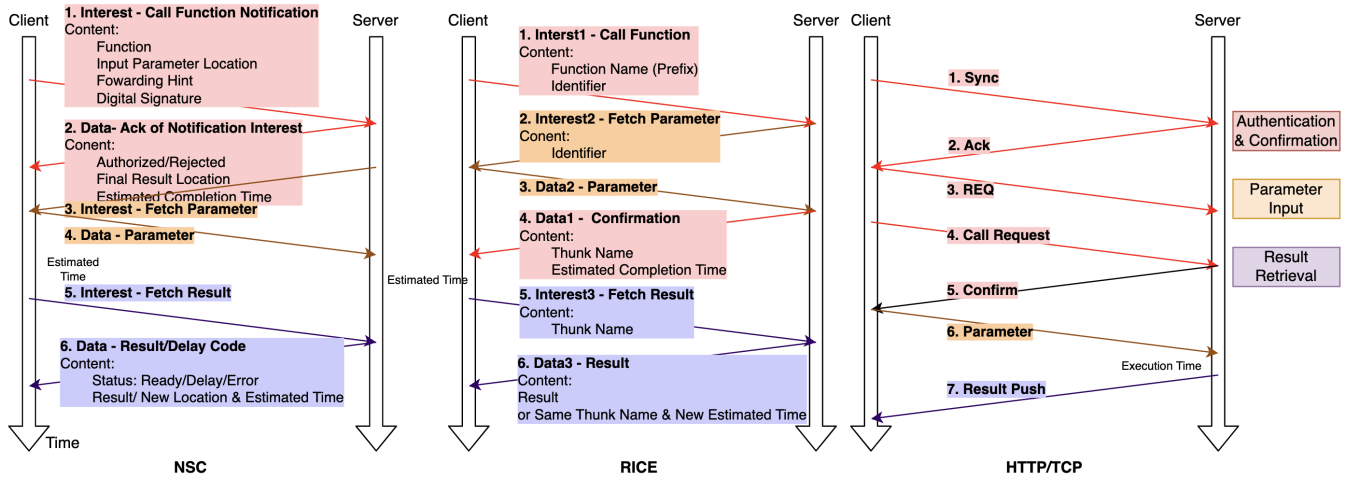

Figure 3: A Brief Comparision of NSC [65], RICE [39], and HTTP/TCP RMI

asynchronously generated data could be treated as named data as well, so the fundamental principle of accessing named data in the network seems conducive to event-driven communication, especially when considering that CCNx/NDN-based architectures feature intrinsic data replication and *Interest* aggregation (which might resemble Pub/Sub application-layer systems such as MQTT). On the other hand, communication in CCNx/NDN-based ICN is fundamentally *pull-based*, i.e., based on *Interest*/data interactions, which is the opposite of event-driven communication. We can distinguish three lines of work in ICN:

1) **Information-Centric architectures that are itself based on the Pub/Sub paradigm** which could provide more natural support for using this paradigm for distributed computing: DONA [38] and PSIRP [97] are examples of such architectures where distributed computing is leveraged on the infrastructure layer: locating objects with topology-independent identifiers across heterogeneous inter-domain settings (i.e. different administrative domains) and inter-domain rendezvous services that combine policy-based name routing between adjacent networks with hierarchical interconnection overlays for locating objects across different domains [86].

2) **Pub/Sub extensions to CCNx/NDN-based systems** to ameliorate the above-mentioned incongruence of pull-based protocols and push semantics: Content-based Publish/Subscribe Networking [10] combines pull (on-demand) with pub/sub-based communication by extending forwarders and ICN protocols, so that producers can register prefixes for which they would generate data asynchronously in the network. COPSS [12] provides a scalable pub/sub service using multicast concepts. At the ICN forwarding layer, COPSS uses a multiple-sender, multiple-receiver multicast capability, similar to PIM-Sparse-Mode, relying on Rendezvous Points (RP). Users subscribe to content based on so-called Content Descriptors (CDs). COPPS-aware routers provide additional data structures, i.e., a subscription table for maintaining CD subscriptions. Another approach that provides support for both push and pull in ICN is HoPP [28], which adopts a hybrid architecture consisting of brokers and peers. Brokers communicate in a centralized manner to manage subscriptions, and peers communicate with each other in a decentralized manner to exchange data, while gateways bridge the gap between the two types of components. This hybrid architecture balances scalability and efficiency.

3) **ICN Distributed Computing systems leveraging Pub/Sub** NFN [87] provides a mechanism to control long-lasting in-network computations in ICN by enabling debugging, timeout prevention (i.e. changing timeout on the fly), fetching intermediate results and client-side computation steering. NFN uses pub-sub to fetch intermediate computing results. IceFlow [46] leverages dataset synchronization (Sync) to inform dataflow nodes about newly available input data, which then triggers new computations.

Unlike CCNx/NDN-based ICN, DONA, and PRIRP, which only enable either pull-based or push-based interactions, INetCEP [56] enables both interaction patterns in a single ICN architecture via an expressive complex event processing (CEP) query language and a CEP query engine. CEP query language can distinguish between pull and push-based traffic and supports standard operators such as windows, joins, filters and aggregators. The CEP query engine can execute these operators on CEP queries in both a centralized and distributed fashion.

4.3 Dataset Synchronization

Multiparty communication and coordination are essential for a distributed system or application, and ICN generally supports this well. For example, ICN uses data object names for de-multiplexing and enables receivers to fetch data according to their individual needs and capabilities. However, participants must know the collection of data item names in advance. Sync is a commonly used transport service for reliable name synchronization in some distributed computing approaches. Sync enables dataset synchronization through updates to a common (shared) namespace, and different variants provide different mechanisms for efficient pull-based update protocols. For example, SVS [48], as one vector-based sync protocol, encodes the dataset state in state vectors where a vector represents a participant. Participants perform synchronization by event-driven or periodic dataset state exchange. Sync introduces a trade-off between delay and overhead. More frequent synchronization operations (pull) can

reduce latency but also increase the overhead. Furthermore, current Sync protocols operate in a decentralized manner, leveraging *Interest* multicast forwarding, which can pose implementation and deployment challenges in non-local networks. A detailed discussion can be found in [68] and [91].

4.4 Metadata and Data Structures

Well-designed metadata and data structures can help improve the computational efficiency and reduce latency. For example, the R2 remote function execution mechanism [84] attempts to select the best executor in a distributed manner. Metadata is used to describe the input data which consumes network resources. R2 uses a cost estimation model for jointly optimizing network and compute resources. During the process of pulling metadata, the system can also "warm up" the execution environment. Therefore, the optimal node selection and hot runtime provisioning in R2 reduce the end-to-end computation completion time. CFN [40] adopts a compute graph to represent the computation process, which helps coordinate distributed computing nodes to work collaboratively on the same task. Similarly, Tangle-Centric Networking (TCN) [89] proposes a decentralized data structure for coordinating distributed applications. In essence, these metadata or data structures are abstractions for information sharing among ICN networks that can help to support decision making in distributed computing.

5 ORCHESTRATION

In distributed computing systems, orchestration automates and coordinates various components across nodes to achieve common objectives. Its primary tasks include resource allocation and provisioning, task scheduling and scaling, configuration management, service deployment, and state monitoring.

5.1 Scheduling

In distributed computing, resources are partitioned and allocated to different applications and services. Task scheduling is the process of allocating resources to tasks to achieve the performance objectives as described in section 2.3. In this section, we discuss provisioning and scheduling approaches in ICN-based distributed computing.

5.1.1 Resource Allocation and Provisioning. Fine-grained resource provisioning approaches are more flexible and responsive than coarse-grained ones; although they also introduce more challenges into the system, such as resource management complexity. Leveraging ICN's name-based operation, resource provisioning and allocation can be performed in a more flexible and fine-grained manner. Here, we present a brief overview of granularity systems built on ICN. From coarse to fine granular, [95] uses Raspberry Pis, i.e., physical machines, to provide a runtime environment for NDN's Forwarding Daemon (NFD). [104] uses a virtual machine (VM) as the runtime of ndnSim [63]. [17, 37] use Docker containers to provide services. [41] adopts unikernels [59], a lightweight virtualization technology compared to containers and VMs [81]. [40, 93] provide computing resources through worker processes while [40, 46] both use actors as the basic resource unit. We argue that systems that are based on ICN and thus have the ability to distribute fine-grained tasks should adopt resource allocation methods that can match their granularity. For example, creating a VM for a function request is not worth the cost, while performing tasks in VMs with a persistent context in traditional distributed computing will be more efficient.

5.1.2 Service Deployment. Using resource-provisioning technologies, schedulers (centralized or decentralized) can optimize where and when to place services or executors. For example, [104] allocates resources and prefetch caches in radio network stations according to the popularity of content and services. [32] proposes an optimal placement algorithm for Hadoop over ICN to minimize net data transfer and computation costs. CFN [40] divides resources into several pools of compute nodes. A scoped flooding resource advertisements protocol was designed for dynamic computation serving each node. [17] modeled the popularity of edge computing services and designed a service dynamic pull strategy. This mechanism instantiates services to minimize the request completion time. Similarly, NFaaS [41] uses a pull strategy to pull service unikernels to nodes where this service is popular. These methods consider the dynamicity of service popularity and adapt to changes, which leads to a responsive system.

5.2 Configuration

IP-based systems must design complicated configuration and scalability components to address and service resolution as well as scalability issues. ICN-based methods without locators can avoid some of these problems. IceFlow [46] uses Sync for managing dataflow components in a decentralized manner. Unlike conventional dataflow systems, this approach operates without complex configuration, leveraging distributed data structures and synchronization protocols, as described in section 4.4. Configuration is often considered an engineering problem, and is thus neglected in research. However, the ease of use of a system determines the utility of a technology to a large extent, and its ease of use is clearly dependent on flexible and simple configuration mechanisms.

5.3 Compute Mesh

Service mesh is a dedicated service composition approach that is based on service-to-service communications, including service discovery and dispatching. Given the similarity between service and compute functions in the context of distributed computing systems, we define *Compute Mesh* as a more general concept of *service mesh*. With respect to computing discovery, requesters in ICedge [64] send *Interests* under namespace */discovery*, and then compute nodes send a response back to the requesters with metadata describing how to call the services. We observe that most systems adopt a namespace approach to perform service discovery. This match with ICN can reduce the reliance on more complex middleware in conventional systems.

Regarding computation offloading and best executor selection as a multi-objective problem, approaches such as [3, 17, 64] all model their objective as cost and design different strategies, such as proactive and passive [64], to forward the computing request to an optimal node executor. For example, [3] designed a distributed coordination mechanism to achieve computation resolution with respect to resource utilization rate and computation completion time. Another important observation is that most approaches adopt a synchronization data structure to update information among nodes, such as SVS [48] in DICer [3], compute graphs, and PSync

[112] in [40]. However, flooding in scoped resource pools has also been adopted for resource advertisement and update [40]. We can observe that ICN can support *compute mesh* well because both naming conventions, as described in section 3.1, and name-based routing can be leveraged to elegantly support specific objectives.

6 APPLICATIONS

Various ICN distributed computing applications have been developed. We categorize them and describe their properties with respect to how they leverage information-centricity.

6.1 Distributed Microservices Communication

Microservices offer a paradigm for constructing complex software applications by dividing them into small, independent, and loosely coupled units. However, as applications scale and the number of microservices multiply, intricate inter-service communication becomes a challenge. Traditional centralized service-oriented network architectures often fail to handle the complexities of extensive distributed microservices deployments. This inadequacy manifests in multiple dimensions: performance bottlenecks, owing to increased latency when routing traffic through a central controller; scalability issues, as expanding a centralized control point becomes increasingly complex and requires substantial resources and meticulous management; and reliability constraints, where any downtime or malfunction in the central control component can disrupt the entire inter-service communication flow [49].

Some approaches have made attempts at simplifying microservice systems by leveraging ICN's name-based, locator-less operation. μNDN [60] implements a management plane to monitor and orchestrate microservices according to pre-defined rules. This system disaggregates the key functions of an NDN forwarder into specialized Virtualized Network Functions (VNFs) and utilizes the VNF manager for adaptive deployment. Although effective, this approach adds a layer of management complexity, necessitating increased orchestration and chaining tasks to maintain the network. DAMC [49] proposes leveraging name-based operation with unique service prefixes to enhance microservice communication. It introduces multiple architectural entities, such as Service Gateway (SG), Service Router (SR), Service Prefixes Authentication (SPA), and Service Mesh Communication Scheduling Center (SCSC), and coordinates communication through well-defined control signaling messages and functions, providing a powerful framework for managing the complex communication requirements of distributed microservices. However, challenges remain [33, 53, 60]. For instance, scalability issues might arise in scenarios with uneven service popularity or applications burgeoning in size.

6.2 Decentralized Blockchain Transactions

Blockchain's core data dissemination mechanism—gossip protocol over P2P—often leads to efficiency deficits [100]: 1) redundant data transmission over multiple TCP connections, leading to unnecessary network traffic; and 2) suboptimal routing due to nodes' lack of awareness of the physical network topology. While the first issue is manageable for small transactions, it becomes problematic for larger-block deliveries. Modifying the block propagation protocol can mitigate this, but at a cost: certain block deliveries may require 1.5 round-trip message exchanges. Additionally, because receiving peers are unaware of the nearest block source, they default to request the block from the first announcer, resulting in suboptimal delivery latency.

Implementing blockchain with ICN can leverage its in-network caching and implicit multi-destination delivery, leading to a more efficient blockchain system with enhanced data dissemination. Researchers have proposed solutions in various domains, including public key infrastructure [54], data security and access control [57], and vehicle networks [77]. Thai et al. [100] proposed a protocol that utilizes a P2P overlay for data announcements and then leverages NDN's pull mechanism for retrieval, based on unique naming conventions inferred from the announcement. This unique naming of data enables request aggregation, allowing the returning data packet to be cached and replicated by forwarders in the network for efficient delivery to all requesters. Feng et al. [18] addressed the scalability of blockchain storage by proposing an ICN-based approach that includes a resolution system for community division, fostering efficient blockchain node partitioning. It provides a virtual chain for faster blockchain indexing, coupled with collaborative block replica deletion, optimized for neighboring partitions. However, challenges such as specific transaction retrieval [100] persist. If a piece of data is absent, the retrieval process becomes incredibly burdensome, leading to escalated transmission costs and deteriorating user experience (refer to [5] for more details).

6.3 Distributed Data Management

Distributed Data Management (DDM) in NDN applies information-centric concepts and a collection of strategies to enable the manipulation, distribution, and protection of extensive geographically dispersed data [109]. These strategies are employed in a variety of applications, including federated catalog systems [16], scalable data dissemination [113], version control [58], and federated data repositories [83], enabling the effective management of large-scale, dispersed data. For example, [16] provides a federated catalog by storing and managing NDN names to accelerate the discovery of desired data across multiple domains. DLedger [113] ensures effective data dissemination over NDN by offering content multicast and a gating function called Proof-of-Authentication (PoA), which digitally signs records, addressing security concerns and enabling participation of constrained devices. GitSync [58] enables effective distributed version control through direct peer-to-peer Git synchronization. It maintains a local repository copy and runs a sync protocol, similar to SVS, in the background. This protocol broadcasts synchronization *Interests*, carrying the local storage root hash, which allows connected peers to detect changes and synchronize as needed. Hydra [83] is an NDN-based federated file storage system that enables effective federated data repositories. In Hydra, multiple file servers process and store file segments while computing a synchronously updated 'global view' of the metadata.

However, contrary to centralized TCP/IP models, some NDN-based DDM systems might incur higher synchronization costs owing to the gap between inefficient data dissemination in heterogeneous networks [79, 113], the high data throughput necessitated by P2P networks [51], and the ICN-based pull data transmission

mode (section 4.3). Despite these challenges, NDN-based DDM systems tend to be more resilient. For instance, in Hydra [83], even if individual nodes are compromised, signed and encrypted files maintain their integrity and confidentiality.

6.4 Distributed Object Stores

In large-scale data management, Distributed Object Stores (DOS), such as Amazon S3[78], Google Cloud Storage [23], and Azure Storage[94], are used to manage and manipulate massive data chunks across distributed infrastructure. These DOSs organize data as unique objects that are identifiable by individual keys and spread across numerous storage servers or nodes. However, these DOSs have some limitations, primarily owing to the IP-centric locator-based communication model. A notable concern is the scaling challenge [80], resulting from the potential need for an excessive number of TCP connections (in the worst case, N×N connections, where N is the number of storage nodes, as in the case of *Redis Cluster* [1], a distributed implementation of Redis). In addition, there is a lack of universal IP multicast services for efficient data replication, performant transport protocols, and usable security implementations.

Chipmunk [92] and Kua [80] are examples of NDN-based DOS systems. Chipmunk is an object store service over NDN that leverages effective naming schemes and unique identifiers for object-to-metadata mapping. In **Chipmunk** [92], nodes carry a common prefix, differentiated by a node ID, eliminating the need for users to know the node prefix. Moreover, Chipmunk provides two types of storage within each node: *file storage* for data, and a *metadata store* for its metadata. Data segments are stored in the file system of the designated node, whereas metadata is stored in a node determined by the data identifier. **Kua** [80] approaches storage and location differently. Kua defines the smallest storage unit as an *Application Data Unit* that carries semantic meaning for the application. This allows applications to use semantically meaningful identifiers for object storage and retrieval in Kua clusters. For data location, Kua simply calculates the appropriate bucket for the data using a Forwarding Hint containing the bucket identifier. This potentially allows for better performance compared to Chipmunk owing to its simpler protocol and the absence of metadata fetching requirements.

6.5 Content Distribution

CDNs are overlay networks to efficiently distribute content globally by caching it near consumers for enhanced retrieval performance [24] and server offloading. They employ a relatively complex set of mechanisms, including request redirection and routing, which track network changes and content availability. ICN could simplify this by leveraging some of its inherent features (e.g., in-network caching capabilities and implicit multicast) [26, 34, 36, 73]. For example, Inaba et al. [34], employed ICN in cooperation with CDNs to enable consumers to cache contents and serve future requests for the same contents. OpenCDN [73] implements a distributed actor-model programming approach and fosters an architecture independent of namespaces. It enables any ISP or third-party entity to collaborate in content distribution, allowing complete control over content storage and routing elements. However, these solutions rely on routing protocols to disseminate content availability information across network nodes, which poses two noteworthy issues [20]. First, these protocols broadcast content availability updates to all nodes, regardless of their relevance or necessity. This can lead to potential FIB overload as the network's content volume continues to expand. Second, these protocols only calculate routes from each node to the content origins, neglecting both on-path caches, which becomes purely opportunistic, and off-path caches, whose effective utilization becomes impractical.

6.6 Decentralized Data Sharing

Decentralized Data Sharing (DDS) refers to the collection of applications and systems leveraging NDN to enable the distributed, peer-to-peer exchange of data. This can be applied across a range of contexts, including multimedia sharing [14, 22, 111], augmented reality (AR) [7, 31], and multiserver online games [11, 67, 69, 107].

For example, NDNFit [111], a distributed mobile health platform for DDS, showcases an exemplary approach to naming convention design. It utilizes consistent data naming with timestamps, facilitating easy *Interest* construction to retrieve data for specific time intervals. With additional authentication and access control measures, this diminishes the dependence on perimeter-based security and simplifies service chaining. AR Web [7] is an architecture that forwards signed data packets based on application-defined names. It employs media-specific coding such as layered coding [74] for efficient 360-degree video transmission, and "perceptual pruning" [82] with content options represented in an NDN namespace. By offering web semantics with packet granularity, the AR web can deliver low-latency, high-granularity, and context-dependent media. Matryoshka [107] employs a bespoke naming scheme to fetch information about objects near a player and uses content caching and multicast by partitioning the virtual environment into octants. G-COPSS [11] employs COPPS (section 4.2) to enable efficient decentralized information dissemination in MMORPGs.

In the context of DDS, hierarchical names offer usability but can be limiting when it comes to representing attribute-based classification schemes typically used for organizing content. Efficient synchronization techniques (as discussed in sections 4 and 5) for namespace subsets would open up significant possibilities to support continuous nearest-neighbor retrieval patterns [96] required by, for instance, AR content prefetching.

7 CONCLUSIONS

ICN provides an attractive platform for distributed computing. From the analyzed approaches, we can **identify the following factors**: 1) accessing named data can be applied to both access to static data and dynamic computation results so that general ICN features such as name-based forwarding, locator-less operation, object security, and in-network caching can be leveraged directly. 2) ICN-empowered forwarding planes can provide additional improvements, e.g., through forwarding strategies with computing-aware algorithms and by enabling concepts such as joint resource optimization. 3) Multiparty communication, e.g., for group coordination, can be achieved without centralized servers. 4) ICN, by its security model and the other features mentioned above, is more conducive to self-managing, decentralized systems, which is also beneficial concerning complexity reduction in distributed computing.

After surveying more than 50 papers on this topic, we can conclude that the above-mentioned features often play a role in motivating different approaches. We observed the following **potential for improvement** in this work: 1) Most proposed algorithms and systems are not compared with state-of-the-art systems in the non-ICN world. Admittedly, this is not always easy to do correctly (considering different software maturity levels), but we suggest more efforts be made to better understand the qualitative and quantitative potential for improvement. 2) The *Interest*-Data vehicle is often used too naively for triggering computation and transmitting results (as discussed in [39]). 3) Not enough real-world experiments are done, and technologies are not developed to a maturity level that would allow initial deployment and real-world experimentation.

Suggested Next Steps for ICN Research

We can also derive some missing features in ICN that can better support distributed computing.

Asynchronous Data: Distributed computing systems often have to deal with asynchronous events and their transmission over networks (**push**), an interaction model that ICN does not naturally provide. Different systems such as Sync, publish-subscribe, dataflow, and IoT phoning-home scenarios [76] provide different workarounds, often involving some form of active polling (unless larger changes to the forwarding plane are proposed). In our view, distributed computing and other applications could benefit from a well-defined push service that works in "point-to-point" as well as in group communication scenarios in Internet-scale scenarios.

Reflexive Forwarding: RICE [39] and RESTful ICN [45] provide an ICN-idiomatic method for client-server communication for both static data access and RMI, i.e., these systems enable longer-lasting computation and the transmission of request parameters of arbitrary size and complexity without giving up flow balance and consumer anonymity. It is achieved by a proposed ICN extension called *reflexive forwarding*, which involves installing temporary forwarding states on the reverse path from the producer to consumers so that the producer can request NDO from the consumer.

There is discussion on how such interactions should be done best in ICN and how ICN-based protocols should be extended in this direction. While it is possible to construct RMI scenarios in testbed without reflexive forwarding, ICN researchers should consider the real-world deployability concerns raised in these two papers.

Information-Centric Web: RESTful ICN [45] laid the foundation for an information-centric web by enabling client/server communication with a series of request/response interactions in a session context, leveraging reflexive forwarding for both RESTful parameter transmission and key exchange. This enables secure RESTful communication using standard ICN mechanisms such as Content Object encryption and signatures, without forcing all interactions into TLS-like tunnels; and overall the system is supposed to provide QUIC-like efficiency (without the connection overhead).

What is missing is the definition of, and further experimentation with, HTTP-like protocol features, i.e., a complete RESTful protocol that could provide a similar feature set as HTTP. Of course, an information-centric web can go beyond HTTP's limitations. For example, it should be explored how result parameters can be shared (for idempotent requests) as demonstrated by RICE [39] before.

Scalable Dataset Synchronization: Sync in NDN [68] is an attractive "transport protocol service" that can enable new ways to build distributed computing, consensus protocols, etc. in a decentralized manner. Current Sync systems, however, rely on *Interest* multicast which has scalability and deployability issues in the Internet. In addition, the efficiency can be low because of the update overhead, especially in larger groups with frequent namespace updates. This could be an area for future research, potentially on hybrid decentralized systems that employ relays at strategic points.

Distributed Machine Learning as a New Application

Distributed ML systems could be a potential research direction owing to the data-oriented security and communication efficiency features offered by ICN. For example, ICN supports caching and replication of data within the network in a decentralized manner. This reduces not only the reliance on centralized repositories that are vulnerable to attacks but also the centralized provider's dominant control over the data. Furthermore, as pre-trained models become basic building blocks for training new models or various applications, integrity threats [30] can be alleviated to an extent by name-data signature binding security mechanisms. Regarding network transmission efficiency, ICN can improve the efficiency of ML systems that rely heavily on data by caching frequently accessed datasets throughout the network. This becomes particularly beneficial for training models that utilize common datasets, as they can be readily available in the network once they are cached.

Moreover, the future generation of communication system is expected to move from Shannon's information theory-based communication paradigm to a semantic communication paradigm that can maximize effective information transmission across wireless networks [52]. One major proposal towards this objective is the transmission of a parameterized data model that describes the data instead of the transmission of raw data [13]. Thus, revisiting how ML is distributed and managed is crucial for the next generation of networks and wireless systems. We envision that ICN can play a crucial role in the distribution of ML models for a given application in communication networks.

ACKNOWLEDGMENTS

This work was supported in part by the Guangdong Provincial Key Laboratory of Integrated Communications, Sensing and Computation for Ubiquitous Internet of Things and by the *Guangzhou Municipal Science and Technology Project 2023A03J0011*, *Research Council of Finland, 6G Flagship program under Grant 346208*, and *Business Finland Project diary number 8754/31/2022*

REFERENCES

[1] 2022. Scaling with Redis Cluster. https://redis.io/docs/manual/scaling/.
[2] Eslam G. AbdAllah, Hossam S. Hassanein, and Mohammad Zulkernine. 2015. A Survey of Security Attacks in Information-Centric Networking. *IEEE Communications Surveys & Tutorials* 17 (2015), 1441–1454. https://api.semanticscholar.org/CorpusID:10311979
[3] Uthra Ambalavanan, Dennis Grewe, Naresh Ganesh Nayak, Liming Liu, Nitinder Mohan, and Jörg Ott. 2022. DICer: distributed coordination for in-network computations. *Proceedings of the 9th ACM Conference on Information-Centric Networking* (2022), 45–55.
[4] Joe Armstrong. 2013. Programming Erlang: software for a concurrent world. *Programming Erlang* (2013), 1–548.

[5] Khizra Asaf, Rana Asif Rehman, and Byung-Seo Kim. 2020. Blockchain technology in named data networks: A detailed survey. *Journal of Network and Computer Applications* 171 (2020), 102840.

[6] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-Independent Packet Processors. *SIGCOMM Comput. Commun. Rev.* 44, 3 (jul 2014), 87–95. https://doi.org/10.1145/2656877.2656890

[7] Jeff Burke. 2017. Browsing an augmented reality with named data networking. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 1–9.

[8] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. 2015. Apache flink: Stream and batch processing in a single engine. *The Bulletin of the Technical Committee on Data Engineering* 38, 4 (2015).

[9] Carmen Carrión. 2022. Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges. *ACM Comput. Surv.* 55, 7, Article 138 (dec 2022), 37 pages. https://doi.org/10.1145/3539606

[10] Antonio Carzaniga, Michele Papalini, and Alexander L Wolf. 2011. Content-based publish/subscribe networking and information-centric networking. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*. 56–61.

[11] Jiachen Chen, Mayutan Arumaithurai, Xiaoming Fu, and K. K. Ramakrishnan. 2011. G-COPSS: A Content Centric Communication Infrastructure for Gaming Applications. *2012 IEEE 32nd International Conference on Distributed Computing Systems* (2011), 355–365.

[12] Jiachen Chen, Mayutan Arumaithurai, Lei Jiao, Xiaoming Fu, and K.K. Ramakrishnan. 2011. COPSS: An Efficient Content Oriented Publish/Subscribe System. In *2011 ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems*. 99–110. https://doi.org/10.1109/ANCS.2011.27

[13] Mingzhe Chen, Deniz Gündüz, Kaibin Huang, Walid Saad, Mehdi Bennis, Aneta Vulgarakis Feljan, and H Vincent Poor. 2021. Distributed learning in wireless networks: Recent progress and future challenges. *IEEE Journal on Selected Areas in Communications* 39, 12 (2021), 3579–3605.

[14] Damian Coomes, Ashlesh Gawande, Nicholas Gordon, and Lan Wang. 2018. Android multimedia sharing application over NDN. *Proceedings of the 5th ACM Conference on Information-Centric Networking* (2018).

[15] Hai Dinh-Tuan, Maria Mora-Martinez, Felix Beierle, and Sandro Rodriguez Garzon. 2020. Development Frameworks for Microservice-Based Applications: Evaluation and Comparison. In *Proceedings of the 2020 European Symposium on Software Engineering* (Rome, Italy) *(ESSE '20)*. Association for Computing Machinery, New York, NY, USA, 12–20. https://doi.org/10.1145/3393822.3432339

[16] Chengyu Fan, Susmit Shannigrahi, Steve DiBenedetto, Catherine Olschanowsky, Christos Papadopoulos, and Harvey Newman. 2015. Managing scientific data with named data networking. In *Proceedings of the Fifth International Workshop on Network-Aware Data Management*. 1–7.

[17] Zhenyu Fan, Wang Yang, Fan Wu, Jing Cao, and Weisong Shi. 2021. Serving at the Edge: An Edge Computing Service Architecture Based on ICN. *ACM Transactions on Internet Technology (TOIT)* 22 (2021), 1 – 27.

[18] Hangwei Feng, Jinlin Wang, and Yang Li. 2022. A Blockchain Storage Architecture Based on Information-Centric Networking. *Electronics* 11, 17 (2022), 2661.

[19] Roy Thomas Fielding. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. Ph. D. Dissertation. University of California, Irvine. http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm.

[20] Ashley Flavel, Pradeepkumar Mani, David Maltz, Nick Holt, Jie Liu, Yingying Chen, and Oleg Surmachev. 2015. FastRoute: A scalable load-aware anycast routing architecture for modern CDNs. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*. 381–394.

[21] Xiaoming Fu, Dirk Kutscher, Satyajayant Misra, and Ruidong Li. 2018. Information-Centric Networking Security. *IEEE Commun. Mag.* 56 (2018), 60–61. https://api.semanticscholar.org/CorpusID:53926462

[22] Ashlesh Gawande, J. Matt Clark, Damian Coomes, and Lan Wang. 2019. Decentralized and Secure Multimedia Sharing Application over Named Data Networking. *Proceedings of the 6th ACM Conference on Information-Centric Networking* (2019).

[23] John J JJ Geewax. 2018. *Google Cloud platform in action*. Simon and Schuster.

[24] Chavoosh Ghasemi, Hamed Yousefi, and Beichuan Zhang. 2020. Far cry: Will cdns hear ndn's call?. In *Proceedings of the 7th ACM Conference on Information-Centric Networking*. 89–98.

[25] Chavoosh Ghasemi, Hamed Yousefi, and Beichuan Zhang. 2020. iCDN: An NDN-based CDN. *Proceedings of the 7th ACM Conference on Information-Centric Networking* (2020).

[26] Chavoosh Ghasemi, Hamed Yousefi, and Beichuan Zhang. 2020. icdn: An ndn-based cdn. In *Proceedings of the 7th ACM Conference on Information-Centric Networking*. 99–105.

[27] Adele Goldberg and David J. Robson. 1983. Smalltalk-80: The Language and Its Implementation.

[28] Cenk Gündoğan, Peter Kietzmann, Thomas C Schmidt, and Matthias Wählisch. 2018. HoPP: Robust and resilient publish-subscribe for an information-centric Internet of Things. In *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. IEEE, 331–334.

[29] Cenk Gündoğan, Christian Amsüss, Thomas C. Schmidt, and Matthias Wählisch. 2020. Toward a RESTful Information-Centric Web of Things: A Deeper Look at Data Orientation in CoAP. *Proceedings of the 7th ACM Conference on Information-Centric Networking* (2020).

[30] Shangwei Guo, Chunlong Xie, Jiwei Li, L. Lyu, and Tianwei Zhang. 2022. Threats to Pre-trained Language Models: Survey and Taxonomy. *ArXiv* abs/2202.06862 (2022). https://api.semanticscholar.org/CorpusID:246822818

[31] Peter Gusev, Jeff Thompson, and Jeff Burke. 2019. Data-centric video for mixed reality. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 1–11.

[32] Hatem Ibn-Khedher, Hossam Afifi, and Hassine Moungla. 2017. Optimal Hadoop over ICN Placement Algorithm for Networking and Distributed Computing. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 1–6.

[33] Muhammad Imran, Muhammad Salah Ud Din, Muhammad Atif Ur Rehman, and Byung-Seo Kim. 2023. MIA-NDN: Microservice-Centric Interest Aggregation in Named Data Networking. *Sensors* 23, 3 (2023), 1411.

[34] Yutaro Inaba, Yosuke Tanigawa, and Hideki Tode. 2015. Content retrieval method in cooperation with CDN and ICN-based in-network guidance over IP network. In *2015 IEEE 40th Conference on Local Computer Networks (LCN)*. IEEE, 454–457.

[35] Haruna Isah, Tariq Abughofa, Sazia Mahfuz, Dharmitha Ajerla, Farhana Zulkernine, and Shahzad Khan. 2019. A Survey of Distributed Data Stream Processing Frameworks. *IEEE Access* 7 (2019), 154300–154316. https://doi.org/10.1109/ACCESS.2019.2946884

[36] Xiaoke Jiang and Jun Bi. 2014. ncdn: Cdn enhanced with ndn. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 440–445.

[37] Kenji Kanai, Toshitaka Tsuda, Hidenori Nakazato, and J. Katto. 2022. Information-centric service mesh for autonomous in-network computing. *Proceedings of the 9th ACM Conference on Information-Centric Networking* (2022).

[38] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. 2007. A Data-Oriented (and beyond) Network Architecture. *SIGCOMM Comput. Commun. Rev.* 37, 4 (aug 2007), 181–192.

[39] Michal Król, Karim Habak, Dave Oran, Dirk Kutscher, and Ioannis Psaras. 2018. RICE: remote method invocation in ICN. *Proceedings of the 5th ACM Conference on Information-Centric Networking* (2018).

[40] Michal Król, Spyridon Mastorakis, Dave Oran, and Dirk Kutscher. 2019. Compute First Networking: Distributed Computing meets ICN. *Proceedings of the 6th ACM Conference on Information-Centric Networking* (2019).

[41] Michal Król and Ioannis Psaras. 2017. NFaaS: named function as a service. *Proceedings of the 4th ACM Conference on Information-Centric Networking* (2017).

[42] Dirk Kutscher, Laura Al Wardani, and T M Rayhan Gias. 2021. Vision: Information-Centric Dataflow: Re-Imagining Reactive Distributed Computing. In *Proceedings of the 8th ACM Conference on Information-Centric Networking* (Paris, France) *(ICN '21)*. Association for Computing Machinery, New York, NY, USA, 52–58. https://doi.org/10.1145/3460417.3482975

[43] Dirk Kutscher, Teemu Karkkainen, and Joerg Ott. 2023. *Directions for Computing in the Network*. Internet-Draft draft-irtf-coinrg-dir-00. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-irtf-coinrg-dir/00/ Work in Progress.

[44] Dirk Kutscher and Dave Oran. 2022. RESTful information-centric networking: statement. *Proceedings of the 9th ACM Conference on Information-Centric Networking* (2022).

[45] Dirk Kutscher and Dave Oran. 2022. RESTful information-centric networking: statement. *Proceedings of the 9th ACM Conference on Information-Centric Networking* (2022). https://api.semanticscholar.org/CorpusID:252091211

[46] Dirk Kutscher, Laura Al Wardani, and T. M. Rayhan Gias. 2021. Vision: information-centric dataflow: re-imagining reactive distributed computing. *Proceedings of the 8th ACM Conference on Information-Centric Networking* (2021).

[47] Leslie Lamport and Nancy Lynch. 1990. Distributed computing: Models and methods. In *Formal models and semantics*. Elsevier, 1157–1199.

[48] Tianxiang Li, Wentao Shang, Alexander Afanasyev, Lan Wang, and Lixia Zhang. 2018. A Brief Introduction to NDN Dataset Synchronization (NDN Sync). *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)* (2018), 612–618.

[49] Xueting Li, Aijun Wang, Wei Wang, Dirk Kutscher, and Yue Wang. 2023. *Distributed architecture for microservices communication based on Information-Centric Networking (ICN)*. Internet-Draft draft-li-icnrg-damc-01. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-li-icnrg-damc/01/ Work in Progress.

[50] Lei Liu, Liguang Xie, Mehdi Bahrami, Yang Peng, Akira Ito, Sevak Mnatsakanyan, Gang Qu, Zilong Ye, and Huiping Guo. 2016. Demonstration of a Functional Chaining System Enabled by Named-Data Networking. *Proceedings of the 3rd*

ACM Conference on Information-Centric Networking (2016).
[51] Siqi Liu, Varun Patil, Tianyuan Yu, Alexander Afanasyev, Frank Alex Feltus, Susmit Shannigrahi, and Lixia Zhang. 2021. Designing hydra with centralized versus decentralized control: A comparative study. In *Proceedings of the Interdisciplinary Workshop on (de) Centralization in the Internet*. 4–10.
[52] Yating Liu, Xiaojie Wang, Zhaolong Ning, MengChu Zhou, Lei Guo, and Behrouz Jedari. 2023. A survey on semantic communications: technologies, solutions, applications and challenges. *Digital Communications and Networks* (2023).
[53] Kim Bao Long, HyunSik Yang, and YoungHan Kim. 2017. Icn-based service discovery mechanism for microservice architecture. In *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 773–775.
[54] Junjun Lou, Qichao Zhang, Zhuyun Qi, and Kai Lei. 2018. A blockchain-based key management scheme for named data networking. In *2018 1st IEEE international conference on hot information-centric networking (HotICN)*. IEEE, 141–146.
[55] Meng Luo, Bo Feng, Long Lu, Engin Kirda, and Kui Ren. 2023. On the Complexity of the Web's PKI: Evaluating Certificate Validation of Mobile Browsers. *IEEE Transactions on Dependable and Secure Computing* (2023). https://api.semanticscholar.org/CorpusID:257527288
[56] Manisha Luthra, Boris Koldehofe, Jonas Höchst, Patrick Lampe, Ali Haider Rizvi, Ralf Kundel, and Bernd Freisleben. 2019. INetCEP: In-Network Complex Event Processing for Information-Centric Networking. In *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS 2019, Cambridge, United Kingdom, September 24-25, 2019*. IEEE, 1–13.
[57] Qiuyun Lyu, Yizhen Qi, Xiaochen Zhang, Huaping Liu, Qiuhua Wang, and Ning Zheng. 2020. SBAC: A secure blockchain-based access control framework for information-centric networking. *Journal of Network and Computer Applications* 149 (2020), 102444.
[58] Xinyu Ma and Lixia Zhang. 2021. GitSync: Distributed Version Control System over NDN. (2021).
[59] Anil Madhavapeddy and Dave Scott. 2013. Unikernels: Rise of the Virtual Library Operating System. *Queue* 11 (2013), 30 – 44.
[60] Xavier Marchal, Thibault Cholez, and Olivier Festor. 2018. μ NDN: an orchestrated microservice architecture for named data networking. In *Proceedings of the 5th ACM Conference on Information-Centric Networking*. 12–23.
[61] Claudio Marxer, Christopher Scherb, and Christian F. Tschudin. 2016. Access-Controlled In-Network Processing of Named Data. *Proceedings of the 3rd ACM Conference on Information-Centric Networking* (2016).
[62] Claudio Marxer and Christian F. Tschudin. 2020. Result Provenance in Named Function Networking. *Proceedings of the 7th ACM Conference on Information-Centric Networking* (2020).
[63] Spyridon Mastorakis, Alexander Afanasyev, and Lixia Zhang. 2017. On the Evolution of ndnSIM. *ACM SIGCOMM Computer Communication Review* 47 (2017), 19 – 33.
[64] Spyridon Mastorakis, Abderrahmen Mtibaa, Jonathan Lee, and Satyajayant Misra. 2020. ICedge: When Edge Computing Meets Information-Centric Networking. *IEEE Internet of Things Journal* 7 (2020), 4203–4217.
[65] Daniel Meirovitch and Lixia Zhang. 2021. *NSC–Named Service Calls, or a Remote Procedure Call for NDN*. Technical Report. Technical Report NDN-0074, Revision 1. NDN.
[66] Oliver Michel, Roberto Bifulco, Gábor Rétvári, and Stefan Schmid. 2021. The Programmable Data Plane: Abstractions, Architectures, Algorithms, and Applications. *ACM Comput. Surv.* 54, 4, Article 82 (may 2021), 36 pages. https://doi.org/10.1145/3447868
[67] Philipp Moll, Selina Isak, Hermann Hellwagner, and Jeff Burke. 2021. A Quadtree-based synchronization protocol for inter-server game state synchronization. *Computer Networks* 185 (2021), 107723.
[68] Philipp Moll, Varun Patil, Lan Wang, and Lixia Zhang. 2022. SoK: The evolution of distributed dataset synchronization solutions in NDN. *Proceedings of the 9th ACM Conference on Information-Centric Networking* (2022).
[69] Philipp Moll, Sebastian Theuermann, Natascha Rauscher, Hermann Hellwagner, and Jeff Burke. 2019. Inter-Server game state synchronization using named data networking. In *Proceedings of the 6th ACM Conference on Information-Centric Networking*. 12–18.
[70] Micah Beck Terry Moore. 2023. How We Ruined The Internet. *CoRR* abs/2306.01101 (2023). https://doi.org/10.48550/arXiv.2306.01101 arXiv:2306.01101
[71] Bill Mulligan. 2023. The Future of Service Mesh is Networking — infoq.com. https://www.infoq.com/articles/service-mesh-networking/. [Accessed 19-Jun-2023].
[72] Arvind Narayanan, Eman Ramadan, and Zhi-Li Zhang. 2018. OpenCDN: An ICN-based open content distribution system using distributed actor model. *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2018), 268–273.
[73] Arvind Narayanan, Eman Ramadan, and Zhi-Li Zhang. 2018. OpenCDN: An ICN-based open content distribution system using distributed actor model. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 268–273.
[74] Afshin Taghavi Nasrabadi, Anahita Mahzari, Joseph D Beshay, and Ravi Prakash. 2017. Adaptive 360-degree video streaming using layered video coding. In *2017 IEEE Virtual Reality (VR)*. IEEE, 347–348.
[75] Kathleen M. Nichols. 2019. Lessons Learned Building a Secure Network Measurement Framework using Basic NDN. *Proceedings of the 6th ACM Conference on Information-Centric Networking* (2019).
[76] David R. Oran and Dirk Kutscher. 2023. *Reflexive Forwarding for CCNx and NDN Protocols*. Internet-Draft draft-oran-icnrg-reflexive-forwarding-05. IETF Secretariat.
[77] Victor Ortega, Faiza Bouchmal, and Jose F Monserrat. 2018. Trusted 5G vehicular networks: Blockchains and content-centric networking. *IEEE Vehicular Technology Magazine* 13, 2 (2018), 121–127.
[78] Mayur R Palankar, Adriana Iamnitchi, Matei Ripeanu, and Simson Garfinkel. 2008. Amazon S3 for science grids: a viable solution?. In *Proceedings of the 2008 international workshop on Data-aware distributed computing*. 55–64.
[79] Yan Pan, Shining Li, Bingqi Li, Yu Zhang, Zhe Yang, Bin Guo, and Ting Zhu. 2020. CDD: coordinating data dissemination in heterogeneous IoT networks. *IEEE Communications Magazine* 58, 6 (2020), 84–89.
[80] Varun Patil, Hemil Desai, and Lixia Zhang. 2022. Kua: a distributed object store over named data networking. *Proceedings of the 9th ACM Conference on Information-Centric Networking* (2022).
[81] Max Plauth, Lena Feinbube, and Andreas Polze. 2017. A Performance Evaluation of Lightweight Approaches to Virtualization.
[82] Pedram Pourashraf and Farzad Safaei. 2017. Perceptual pruning: A context-aware transcoder for immersive video conferencing systems. *IEEE Transactions on Multimedia* 19, 6 (2017), 1327–1338.
[83] Justin Presley, Xi Wang, Tym Brandel, Xusheng Ai, Proyash Podder, Tianyuan Yu, Varun Patil, Lixia Zhang, Alex Afanasyev, F Alex Feltus, et al. 2022. Hydra–A Federated Data Repository over NDN. *arXiv preprint arXiv:2211.00919* (2022).
[84] Jianpeng Qi and Rui Wang. 2021. R2: A Distributed Remote Function Execution Mechanism With Built-In Metadata. *IEEE/ACM Transactions on Networking* 31 (2021), 710–723.
[85] Anichur Rahman, K. M. Azharul Hasan, Dipanjali Kundu, Md. Jahidul Islam, Tanoy Debnath, Shahab S. Band, and Neeraj Kumar. 2022. On the ICN-IoT with federated learning integration of communication: Concepts, security-privacy issues, applications, and future perspectives. *Future Gener. Comput. Syst.* 138 (2022), 61–88.
[86] Jarno Rajahalme, Mikko Särelä, Kari Visala, and Janne Riihijärvi. 2011. On name-based inter-domain routing. *Computer Networks* 55, 4 (2011), 975–986.
[87] Christopher Scherb, Balázs Faludi, and Christian Tschudin. 2017. Execution state management in named function networking. In *2017 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, 1–6.
[88] Christopher Scherb, Balázs Faludi, and Christian F. Tschudin. 2017. Execution state management in named function networking. *2017 IFIP Networking Conference (IFIP Networking) and Workshops* (2017), 1–6.
[89] Christopher Scherb, Dennis Grewe, and Christian F. Tschudin. 2021. Tangle Centric Networking (TCN). *ArXiv* abs/2108.06710 (2021).
[90] Oussama Serhane, Khadidja Yahyaoui, Boubakr Nour, and Hassine Moungla. 2021. A Survey of ICN Content Naming and In-Network Caching in 5G and Beyond Networks. *IEEE Internet of Things Journal* 8 (2021), 4081–4104.
[91] Wentao Shang, Yingdi Yu, Lijing Wang, Alexander Afanasyev, and Lixia Zhang. 2017. A Survey of Distributed Dataset Synchronization in Named Data Networking.
[92] Yong Yoon Shin, Sae Hyong Park, Namseok Ko, and Arm Jeong. 2020. Chipmunk: Distributed Object Storage for NDN. In *Proceedings of the 7th ACM Conference on Information-Centric Networking*. 161–162.
[93] Manolis Sifalakis, B Kohler, Christopher Scherb, and Christian F. Tschudin. 2014. An information centric network for computing the distribution of computations. In *Information-Centric Networking*.
[94] Julian Soh, Marshall Copeland, Anthony Puca, Micheleen Harris, Julian Soh, Marshall Copeland, Anthony Puca, and Micheleen Harris. 2020. Azure Storage. *Microsoft Azure: Planning, Deploying, and Managing the Cloud* (2020), 271–291.
[95] Xiaobin Tan, Yang Jin, Weiwei Feng, Shunyi Wang, and Yubin Yang. 2019. Scheduling of Distributed Collaborative Tasks on NDN based MANET. *Proceedings of the ACM SIGCOMM 2019 Workshop on Mobile AirGround Edge Computing, Systems, Networks, and Applications* (2019).
[96] Yufei Tao, Dimitris Papadias, and Qiongmao Shen. 2002. Continuous nearest neighbor search. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier, 287–298.
[97] Sasu Tarkoma, Mark Ain, and Kari Visala. 2009. The Publish/Subscribe Internet Routing Paradigm (PSIRP): Designing the Future Internet Architecture. In *Towards the Future Internet - A European Research Perspective*, Georgios Tselentis, John Domingue, Alex Galis, Anastasius Gavras, David Hausheer, Srdjan Krco, Volkmar Lotz, and Theodore B. Zahariadis (Eds.). IOS Press, 102–111.
[98] Pouyan Fotouhi Tehrani, Eric Osterweil, Thomas C. Schmidt, and Matthias Wählisch. 2022. SoK: Public key and namespace management in NDN. *Proceedings of the 9th ACM Conference on Information-Centric Networking* (2022).

[99] David L Tennenhouse and David J Wetherall. 1996. Towards an active network architecture. *ACM SIGCOMM Computer Communication Review* 26, 2 (1996), 5–17.

[100] Quang Tung Thai, Namseok Ko, Sung Hyuk Byun, and Sun-Me Kim. 2022. Design and implementation of NDN-based Ethereum blockchain. *Journal of Network and Computer Applications* 200 (2022), 103329.

[101] Tri Minh Truong, Aaron Harwood, Richard O Sinnott, and Shiping Chen. 2018. Performance analysis of large-scale distributed stream processing systems on the cloud. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 754–761.

[102] Christian Tschudin, Christopher A. Wood, Marc Mosko, and David R. Oran. 2022. *File-Like ICN Collections (FLIC)*. Internet-Draft draft-irtf-icnrg-flic-04. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-irtf-icnrg-flic/04/ Work in Progress.

[103] Christian F. Tschudin and Manolis Sifalakis. 2014. Named functions and cached computations. *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)* (2014), 851–857.

[104] Rehmat Ullah, Muhammad Atif Ur Rehman, Muhammad Ali Naeem, Byung-Seo Kim, and Spyridon Mastorakis. 2020. ICN with edge for 5G: Exploiting in-network caching in ICN-based edge computing for 5G networks. *Future Gener. Comput. Syst.* 111 (2020), 159–174.

[105] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. 2020. A Survey on Distributed Machine Learning. *ACM Comput. Surv.* 53, 2, Article 30 (mar 2020), 33 pages. https://doi.org/10.1145/3377454

[106] Liang Wang, Mário Almeida, Jeremy Blackburn, and Jon A. Crowcroft. 2016. C3PO: Computation Congestion Control (PrOactive). *Proceedings of the 3rd ACM Conference on Information-Centric Networking* (2016).

[107] Zhehao Wang, Zening Qu, and Jeff Burke. 2014. Demo overview-Matryoshka: design of NDN multiplayer online game. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*. 209–210.

[108] Bastiaan Wissingh, Alexander Afanasyev, Lixia Zhang, Christopher A. Wood, Dave Oran, and Christian F. Tschudin. 2020. Information-Centric Networking (ICN): CCNx and NDN Terminology.

[109] Yuanhao Wu, Faruk Volkan Mutlu, Yuezhou Liu, Edmund Yeh, Ran Liu, Catalin Iordache, Justas Balcas, Harvey Newman, Raimondas Sirvinskas, Michael Lo, et al. 2022. N-DISE: NDN-based data distribution for large-scale data-intensive science. In *Proceedings of the 9th ACM Conference on Information-Centric Networking*. 103–113.

[110] Yoji Yamamoto, Yuki Koizumi, Toru Hasegawa, Giulio Rossi, Andrea Detti, Onur Ascigil, and Ioannis Psaras. 2019. Multiple Network Function Execution in ICN-based Crowdsensing Service: Demo. *Proceedings of the 6th ACM Conference on Information-Centric Networking* (2019).

[111] Haitao Zhang, Zhehao Wang, Christopher Scherb, Claudio Marxer, Jeff Burke, Lixia Zhang, and Christian Tschudin. 2016. Sharing mhealth data via named data networking. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. 142–147.

[112] Lixia Zhang, Deborah Estrin, Jeff Burke, Van Jacobson, James D. Thornton, Diana K. Smetters, Beichuan Zhang, and Gene Tsudik. 2010. Named Data Networking (NDN) Project.

[113] Zhiyi Zhang, Vishrant Vasavada, Xinyu Ma, and Lixia Zhang. 2019. Dledger: An iot-friendly private distributed ledger system based on dag. *arXiv preprint arXiv:1902.09031* (2019).