electronics

Article

An ICN-Based On-Path Computing Resource Scheduling Architecture with User Preference Awareness for Computing Network

Zhihui Ni [1,2], Jiali You [1,2] and Yang Li [1,2,*]

[1] National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China; nizhihui21@mails.ucas.ac.cn (Z.N.); youjl@dsp.ac.cn (J.Y.)
[2] School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: liyang@dsp.ac.cn

Abstract: The Computing Network is an emerging network paradigm that aims to realize computing resource scheduling through the intrinsic capabilities of the network. However, existing resource scheduling architectures based on conventional TCP/IP networks for the Computing Network suffer from deficiencies in routing flexibility and a lack of user preference awareness, while Information-Centric Networking (ICN) holds the potential to address these issues. ICN inherently supports dynamic routing in scenarios such as multi-homing and mobility due to its routing mechanism that is based on content names rather than host addresses, and it is further enhanced by the integration with Software-Defined Networking (SDN) technologies, which facilitate convenient network-layer route readdressing, thus offering a conducive environment for flexible routing scheduling. Furthermore, ICN introduces novel routing protocols that, compared with the more rigid protocol designs in conventional TCP/IP networks, offer greater flexibility in field usage. This flexibility allows for the incorporation of customized fields, such as "preference", enabling the perception of user preferences within the network. Therefore, this paper proposes a novel ICN-based on-path computing resource scheduling architecture named IPCRSA. Within this architecture, an original design for computing resource request packet format is developed based on the IPv6 extension header. Additionally, preference-based computing resource scheduling strategies are incorporated, which employ the technique for order preference by similarity to ideal solution (TOPSIS) combined with the entropy weight method, to comprehensively evaluate computing resource nodes and use a roulette-selection algorithm to accomplish the probability selection of destination nodes. Experimental results indicate that, in comparison to alternative scheduling schemes, IPCRSA exhibits significant advantages in enhancing scheduling flexibility, improving scheduling success rates, and catering to diverse user requirements.

Keywords: ICN; computing resource scheduling architecture; computing network

check for updates

Citation: Ni, Z.; You, J.; Li, Y. An ICN-Based On-Path Computing Resource Scheduling Architecture with User Preference Awareness for Computing Network. *Electronics* **2024**, *13*, 933. https://doi.org/10.3390/electronics13050933

Received: 18 January 2024
Revised: 21 February 2024
Accepted: 26 February 2024
Published: 29 February 2024

Copyright: © 2024 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 development of the digital economy has driven the generation of massive data, and the processing of this vast amount of data relies on powerful computing resources and widely covered network connections. The convergence of computing and network has become a crucial trend. Against this backdrop, a novel network paradigm, termed the "Computing Network", has been proposed [1–3], which integrates computing resource scheduling capabilities into the network. This paradigm aims to interconnect distributed computing resource nodes; coordinate computing resources, network resources, and user requirements; and achieve the integrated scheduling of computing and network resources [4].

Various network service providers, represented by telecommunications operators, have proposed their own Computing Network architectures. Each provider has put forth their respective ideas regarding the collaboration of computing and network resources, and they have implemented computing resource scheduling in either a centralized or distributed manner within these architectures [5–8].

It is important to note that computing resource scheduling in Computing Network is a broad concept. It can involve the scheduling of fundamental resources such as CPU, memory, and storage, as well as the scheduling of specific computing services provided on computing resource nodes. However, this paper primarily focuses on the scheduling of basic resources under service deployment requests [9]. Furthermore, computing resource scheduling, as one of the critical issues in computational systems such as Computing Network, Edge Computing, and Cloud Computing, plays a crucial role in determining the effectiveness of computing resource utilization and user service experience [10].

However, with the rapid increase in cloud-edge-end computing resource nodes, along with diverse computing demands from users, the centralized computing resource scheduling mechanism in a Computing Network faces scalability limitations. It is not suitable for large-scale network deployments [11–13]. Distributed scheduling methods, on the other hand, offer advantages in terms of scalability and reliability [14]. Therefore, this paper places emphasis on the study of distributed computing resource scheduling architectures.

Existing distributed resource scheduling architectures in a Computing Network mostly rely on a flat architecture, performing full-scale dissemination of resource status information and scheduling among all the scheduling nodes [12]. Faced with a scenario of high-frequency changes in computing resource node status, it is challenging for resource status information to synchronize rapidly [15,16]. Delayed resource status information is prone to scheduling failures. Moreover, current distributed resource scheduling architectures perform scheduling based on conventional TCP/IP networks, determining the final scheduling destination at the user entry node. This "in-out" scheduling method lacks flexibility [13] and cannot effectively address resource competition issues in distributed resource scheduling scenarios, leading to potential scheduling failures. In contrast, in Information-Centric Networking (ICN), the content-name-based routing mechanism, combined with Software-Defined Networking (SDN) technologies, allows for convenient readdressing of routing in the network layer [17–19], offering high routing flexibility. Therefore, ICN possesses advantageous conditions for implementing dynamic computing resource scheduling. Additionally, current resource scheduling architectures based on traditional TCP/IP networks often adopt single and static scheduling strategies, such as optimizing the utilization and load balancing of computing resources or minimizing resource usage costs [20]. These architectures do not specifically design scheduling schemes based on user preferences, thus failing to meet diverse user requirements. ICN, with its novel routing protocols, which allow flexible definition and use of fields, can achieve flexible routing of interest packets based on user preferences [21,22], demonstrating the potential to implement a preference-based resource scheduling mechanism.

Therefore, this paper proposes an ICN-based on-path computing resource scheduling architecture with the following main contributions:

- Designing a hierarchical scheduling architecture that supports on-path resource scheduling: by carrying resource demands and preference parameters in the computing resource request packets, the scheduling nodes can adjust their scheduling destinations promptly during the packet routing process based on demand parameters and current resource status information, thus improving scheduling success rates.
- Developing computing resource scheduling strategies tailored to different resource request preferences: scheduling nodes read the preference field in the packet header, make decisions, and forward the packet based on the corresponding scheduling strategy, meeting diverse user requirements.

The remaining sections of this paper are organized as follows: Section 2 analyzes relevant research on the Computing Network, existing resource scheduling mechanisms,

ICN, and constructing a Computing Network based on ICN. Section 3 describes the overall structure of the proposed distributed computing resource scheduling architecture IPCRSA, presenting the design of computing resource request packets and the typical process of resource scheduling within this architecture. Section 4 expounds on the new mechanisms and optimizations of IPCRSA, encompassing the preference-based computing resource scheduling strategies and the processing and forwarding mechanisms for computing resource request packets. In Section 5, a performance analysis of IPCRSA is conducted through a comparative simulation experiment. Finally, Section 6 summarizes the work presented in this paper.

2. Related Works

This section introduces the main related works pertinent to this paper. Section 2.1 begins by discussing the related works on the Computing Network from three perspectives: macro-policy, standardization, and academia; it then introduces the existing methods of computing resource scheduling within the Computing Network from two perspectives: distributed and centralized. Finally, it presents some computing resource scheduling mechanisms outside of the Computing Network that utilize multi-criteria decision-making approaches to aid in the allocation of computing resources. Section 2.2 reviews the related works on ICN, explicating the characteristics of the ICN architecture upon which this paper is based. It subsequently provides a brief introduction to the existing methods of building a Computing Network and implementing the scheduling of computing resources based on ICN.

2.1. Computing Network and Computing Resource Scheduling

In recent years, the academic and industrial sectors have conducted extensive research and exploration on the Computing Network. On a macro-policy level, Chinese telecommunications operators have issued a white paper on Computing Network research, elaborating on the demands, architecture, and key technologies of the Computing Network. Various Computing Network architectures, including Computing-aware Networking (CAN), Computing Power Network (CPN), Computing First Network (CFN), etc., have been proposed [23–25]. In terms of standardization, ITU-T has initiated and released a series of standards related to the Computing Network, such as Y.CPN-arch, Y.IMT2020-CAN-req, Q.CPN, Q.BNG-INC, etc. [26–29]. The IETF's Routing Area Working Group (RTGWG) working group has conducted research on CFN, submitting three core proposals that outline the framework, requirements, and use cases of CFN [1,7,30]. Additionally, the IRTF has established the Computing in the Network Research Group (COINRG) to conduct relevant research on computing-network integration standards. On the academic front, the authors in [2] first proposed a Computing Network solution based on deep integration of cloud, network, and edge, providing an implementation case for AI applications. The authors in [3] analyzed the research background and key technologies of the Computing Network, demonstrating their advantages in 6G business scenarios. The authors in [11] provided a detailed exposition of the overall composition and routing protocols of the Computing Network architecture: CFN.

As the core of the Computing Network, computing resource scheduling in it has also received extensive attention. In terms of distributed scheduling, the authors in [8] proposed a computing-aware bandwidth allocation method called CABA based on utility optimization. CABA selects the optimal computing resource node and routing path for resource requests in the Computing Network, enhancing the overall utilization of computing resources. Although this method considered user satisfaction as one of the decisive factors in the scheduling of computing resources, it failed to account for the users' preferences for resource requirements. The authors in [12] introduced a CAN-based Computing Network simulation prototype, validating the effectiveness of its computing-aware routing scheduling mechanism and improving the overall utilization of computing resources. The authors in [13] proposed a Computing Network routing protocol, CARP. It comprehensively consid-

ered computing and network resource information, scheduled computing resource requests to optimal service nodes, and optimized system capacity and end-to-end average latency. However, both methods above assumed that the resource status information they acquired was real-time, overlooking the resource competition issue in the scenario of distributed scheduling, which was challenging to achieve in real-world network environments. Moreover, the method of always choosing a single optimal destination resource node, in the absence of real-time information synchronization, could not achieve load balancing among nodes, potentially adversely affecting resource scheduling. Additionally, both methods above were based on the TCP/IP network for scheduling and determined the destination resource node at the ingress node, lacking the flexibility to adjust based on resource status information in a timely manner. The authors in [31] presented a Computing Network resource scheduling architecture centered around the IP network, implementing computing orchestration and scheduling at the network layer based on the Segment Routing IPv6 (SRv6). The authors in [32] built a CPN prototype experimental platform based on Kubernetes, achieving dynamic allocation of computing resources and flexible scheduling of resource requests, reducing response latency, and enhancing load balancing performance. Although these methods analyzed and optimized resource scheduling in the Computing Network from an implementation perspective, and the latter balanced the load of resource nodes, there remained an issue of lacking scheduling flexibility. In terms of centralized scheduling, the authors in [33] elaborated on the concept of Segment Identifier as a Service (SIDaaS) based on SRv6 and proposed the Programmable Service System (PSS) based on SIDaaS, providing a vital implementation method for integrated scheduling of computing and network resources. The authors in [34] proposed a lightweight, multi-cluster hierarchical edge resource scheduling solution based on a hybrid architecture of Kubernetes and K3s. This scheme achieved unified management and scheduling of a large number of edge computing resource nodes. The authors in [35] implemented coordinated scheduling of computing and network resources using SDN technologies, orchestrating computing and services on the underlay network. However, with the increase in the number of ubiquitous computing resource nodes and diversified user demands, the aforementioned centralized scheduling methods faced issues of scalability and reliability limitations, which similarly had an adverse impact on resource scheduling.

Furthermore, some computing resource scheduling mechanisms outside of Computing Network have utilized multi-criteria decision-making approaches to aid in the allocation and scheduling of computing resources, which is also of reference value to this paper. For instance, the authors in [36] employed the Analytic Hierarchy Process (AHP) within Kubernetes for resource scaling decisions, optimizing the allocation of computing resources. The authors in [37] used the TOPSIS method for placing Virtual Network Functions (VNFs) within computing resource clusters to complete resource invocation, significantly enhancing the satisfaction rate of requests. In [38], the authors introduced a two-phased Multi-Attribute Decision Making (MADM) method, utilizing the P2P formalism to discover and allocate computing resources.

Overall, due to scalability limitations associated with centralized computing resource scheduling, distributed approaches have gained more widespread application. However, the aforementioned distributed resource scheduling mechanisms still face challenges, such as a lack of flexibility and awareness of user preferences. Computing resource scheduling based on ICN has the potential to address and improve these issues. Consequently, this paper proposes a computing resource scheduling architecture based on ICN, which has advantages in terms of scheduling flexibility and meeting diverse user preferences. A detailed analysis will be further elaborated on in the following sections.

2.2. ICN and ICN-Based Computing Network

As a new network architecture, ICN aims to address scalability, flexibility, and security limitations in traditional TCP/IP network caused by IP semantic overload. Some of the

currently successful implementations include DONA [39], CCN [40], NetInf [41], NDN [42], PURSUIT [43], Mobility First [44], SEANet [45], and others.

Existing ICN architectures can be categorized into two types based on their implementation approaches: evolutionary and revolutionary. Evolutionary ICN architectures aim to incrementally improve the current network architecture, enhancing network functionalities by introducing ICN concepts and mechanisms within the existing IP network framework. This is typically achieved by adding new protocol layers, utilizing middleware, or extending existing TCP/IP protocols. As a result, they can be relatively easily deployed in current network environments, reducing the need for extensive infrastructure replacement [46,47]. On the other hand, revolutionary ICN architectures advocate for a fundamental redesign of the network, with content-centric concepts as the foundation of network construction. This approach often requires the development of entirely new network protocols and infrastructures, which may include new routing algorithms, data naming schemes, caching mechanisms, etc. Therefore, the implementation of revolutionary ICN architectures tends to be more complex, challenging, and costly. The computing resource scheduling architecture proposed in this paper is based on the evolutionary ICN architecture, which facilitates its subsequent implementation in the real world at a lower cost.

ICN adopts an information-centric communication approach, replacing the traditional host-centric model, and routes based on information or content names rather than host addresses. Different ICN architectures may employ various naming schemes for content, which are primarily categorized into hierarchical and flat naming schemes. For instance, NDN uses a human-readable hierarchical structure similar to the URL for naming content. PURSUIT employs a flat Rendezvous Identifier (RID) and a Scope identifier (SID) as a pair for content naming. Mobility First uses flat, globally unique identifiers (GUIDs) for this purpose. However, existing research demonstrates that flat naming schemes offer superior advantages in terms of security, scalability, and flexibility compared with hierarchical schemes [48]. Consequently, the ICN architecture described in this paper also utilizes flat identifiers (IDs) for content naming, akin to the aforementioned flat naming schemes, which facilitates routing within the ICN environment based on these IDs.

Since ICN routes based on content names rather than host addresses, it necessitates a name resolution service that matches or translates content names to the locators of content providers. This enables content requests to route to the content provider's location based on content names or locators [49]. The name resolution process and the content request routing can be coupled (performing name resolution during the routing process). In this scenario, content requests route directly by content names, and content providers respond along the reverse path of the content request route. This approach is known as the Name-Based Routing (NBR) method, with NDN being a typical example. Alternatively, name resolution can be an independent process, separate from message routing. Content requests first obtain locators through an independent name resolution system, then route based on those locators. This method is referred to as the Standalone Name Resolution (SNR) method, with typical examples including DONA, PURSUIT, MobilityFirst, SEANet, and others. However, NBR methods are often found in revolutionary ICN architectures, requiring substantial modifications to the existing network infrastructure and presenting high deployment challenges. On the other hand, SNR methods, based on a separate name resolution system, offer better scalability and compatibility with the existing TCP/IP network, aligning more closely with the needs of evolutionary ICN architectures. Therefore, the ICN architecture discussed in this paper employs the SNR method. Additionally, it is important to note that IP addresses are utilized as locators for content providers in the ICN environment, which is compatible with TCP/IP in this paper.

Regarding the integration of ICN and Computing Network, the authors in [4] proposed a scheme for constructing a named Computing Network based on ICN, emphasizing that the routing scheduling in both the Computing Network and ICN emphasize "content" rather than the specific location of hosts. However, this method merely presented the concept without delving into specific mechanism designs. The authors in [14] designed

a distributed resource scheduling architecture for Computing Network based on ICN, utilizing Conflict-free Replicated Data Types (CRDTs) as the underlying data structure and Remote Method Invocation for ICN (RICE) as the execution environment, demonstrating advantages in simplicity, performance, and fault recovery capability. Yet, this approach, predicated on the revolutionary ICN architecture, poses significant deployment challenges. Additionally, some novel ICNs based on the SNR method, incorporating SDN principles and technologies, have implemented a new feature of readdressing or late-binding for IP in the network layer, further enhancing the flexibility of ICN data routing [45]. Consequently, they possess advantageous conditions for realizing the dynamic scheduling of computing and network resources.

Finally, a review of the related works on computing resource scheduling in Computing Network mentioned above is conducted, with the most representative studies selected and summarized in Table 1 below for reference.

Table 1. Related works on computing resource scheduling.

Approach	Publication Year	Distributed or Centralized	Network Type	Results Achieved	Advantages	Disadvantages
CABA [8]	2021	Distributed	TCP/IP	A computing-aware bandwidth allocation method based on utility optimization	High utility in relation to bandwidth allocation	Neglect user preferences, lack flexibility
CARP [13]	2022	Distributed	TCP/IP	A computing-aware routing protocol	Optimized system capacity and end-to-end average latency	Overlook resource competition issues, lack of flexibility
PSS [33]	2022	Centralized	TCP/IP (SRv6)	A vital implementation method for integrated scheduling of computing and network resources	Increased flexibility of network programmability	Scalability and reliability limitations
K8s&K3s-Based [34]	2020	Centralized	TCP/IP	A lightweight, multi-cluster hierarchical edge resource scheduling solution	Wide applicability	Scalability and reliability limitations
SDN-Based [35]	2021	Centralized	TCP/IP	A computing power scheduling technology based on programmable network technology	Scheduling in underlay network	Scalability and reliability limitations
Named-Based [4]	2021	/	ICN	A scheme for constructing a named Computing Network based on ICN	More efficient	No specific mechanism designs
CRDTs-Based [14]	2019	Distributed	ICN	A distributed resource scheduling architecture for a Computing Network based on ICN	Advantages in simplicity, performance, and fault recovery capability	Pose significant deployment challenges

3. Architecture

This section will elaborate on the proposed on-path computing resource scheduling architecture based on ICN, which is named IPCRSA. Section 3.1 presents the overall design of the architecture, including the topology structure and relationships among various entities. Section 3.2 provides an explanation of the design of the computing resource request packet format within this architecture, while Section 3.3 introduces the typical process of computing resource scheduling in IPCRSA.

3.1. IPCRSA Overview

Figure 1 illustrates the overall design of IPCRSA. Its network topology is divided into three levels: L_3, L_2, and L_1. The main entities include the Scheduling Node (SN); Gateway Node (GN); Computing Resource Node (CRN); and User Client (Client), User, and Resolution Node (RN). There is a one-to-one connection relationship between SNs and GNs, and GNs establish either a one-to-one or one-to-many connection relationship with CRNs. The L_3, L_2, and L_1 level network topologies and various entities within their coverage areas collectively constitute the scheduling domains in IPCRSA at levels L_3, L_2, and L_1. The specific functions of each entity will be explained in the following sections.

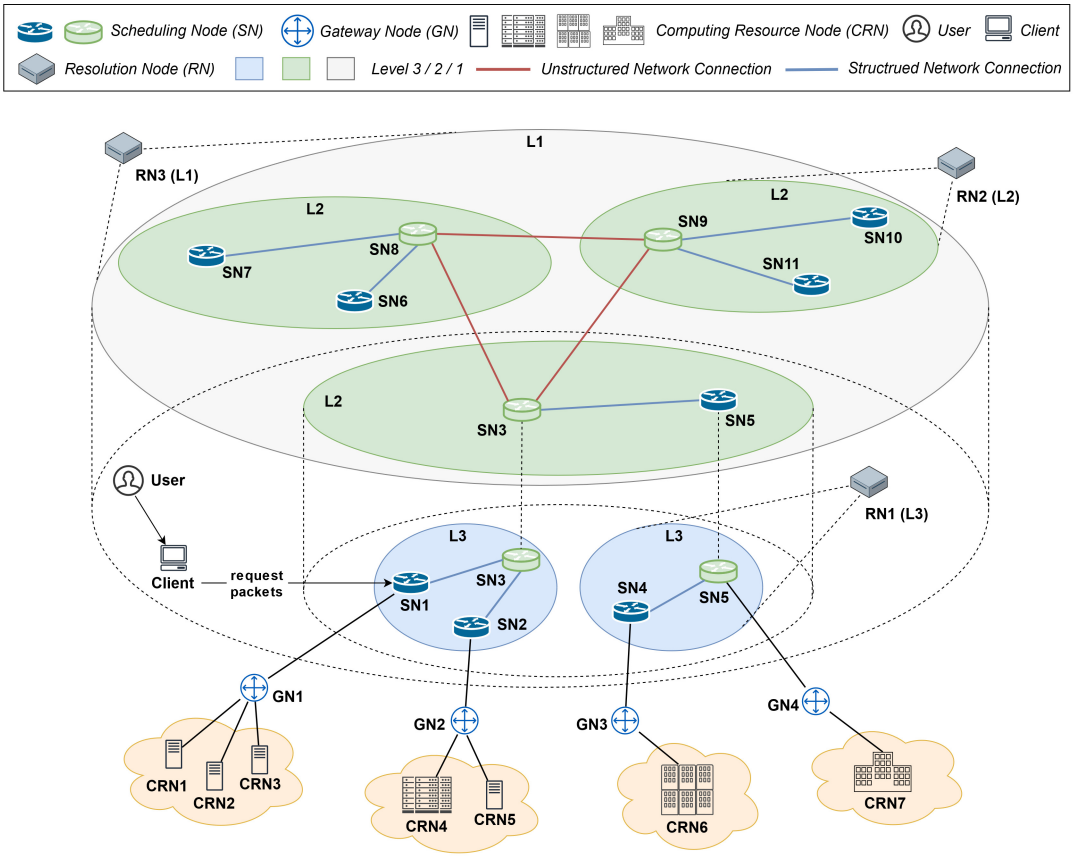


Figure 1. IPCRSA overview.

3.1.1. Scheduling Node

The SNs in IPCRSA also serve as ICN nodes, implementing computing resource scheduling based on ICN. As depicted in Figure 2, the composition structure of SNs can be divided into the control plane and the data plane. The main function modules in the control plane include the ID Management Module, Topology Construction Module, Resource Status Notification Module, and Computing Resource Scheduling Module. The data plane is responsible for receiving, processing, and forwarding computing resource request packets.

The ID Management Module oversees the management of identifiers and network addresses mapping relationships of SNs, ensuring the addressability of SNs within the ICN. Leveraging the ID Management Module, the Topology Construction Module accomplishes the construction of logical topology for resource status notification and resource scheduling, ultimately forming the hierarchical logical topology structure of IPCRSA, as illustrated in Figure 1 (The logical topology describes the way data packets flow within a network, determined by network protocols and configurations). The Resource Status Notification Module, based on the hierarchical topology structure, synchronizes resource status informa-

tion among SNs, forming a resource status information table in each SN. The Computing Resource Scheduling Module, relying on the resource status information table, executes computing resource scheduling strategies. In collaboration with the Computing Resource Scheduling Module, the data plane of each SN processes and forwards computing resource request packets, thereby conducting on-path computing resource scheduling. The specific functionalities of each module and their interrelationships will be further elucidated in the subsequent sections.

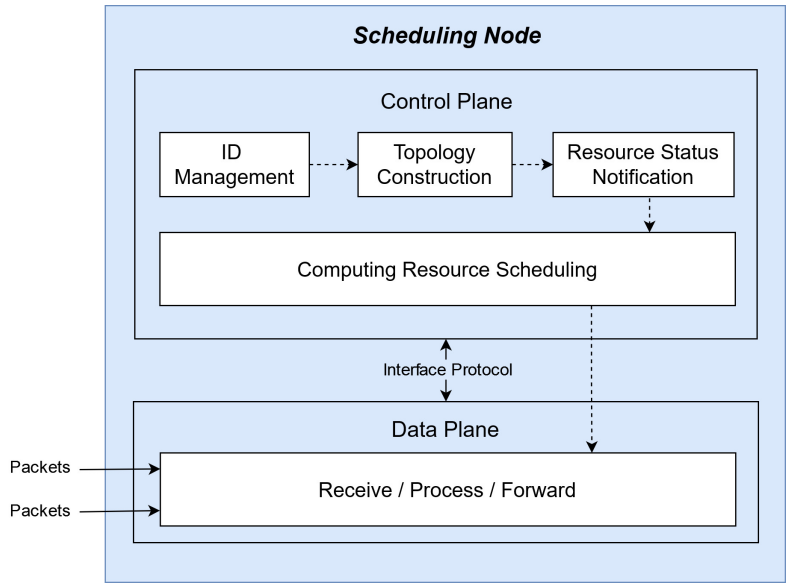


Figure 2. Architecture and function modules of SN.

- ID Management Module

As shown in Table 2, in IPCRSA, SNs within the scheduling domains at the L_x level possess identical and flat resource scheduling service identifiers, denoted as RSS_ID_L_x, indicating that they can provide resource scheduling service in the corresponding scheduling domain at level L_x. The mapping relationship between the SN's RSS_ID_L_x and IP is maintained by a hierarchical ICN name resolution system similar to the one discussed in SEANet [50].

Table 2. IDs in IPCRSA.

ID	Meaning
RSS_ID_L_1	Provide Resource Scheduling Service in L_1
RSS_ID_L_2	Provide Resource Scheduling Service in L_2
RSS_ID_L_3	Provide Resource Scheduling Service in L_3
CRS_ID_T_1	Provide Computing Resource Service with Type T_1
CRS_ID_T_2	Provide Computing Resource Service with Type T_2
CRS_ID_T_3	Provide Computing Resource Service with Type T_3
USER_ID	Represent A Computing Resource Requestor in IPCRSA

The basic structure of the hierarchical name resolution system is illustrated in Figure 3. The resolution domains (the scope within which name resolution services can be provided) are divided into three levels: L_3, L_2, and L_1, adhering to the principle of inter-layer nesting and intra-layer non-intersection. From level L_3 to L_1, the coverage scope of the resolution domains gradually expands. RNs within each level's resolution domains are the main components of this hierarchical name resolution system. They are responsible for maintaining the mapping relationships between IDs and IP addresses of all entities (such as SNs, Users, etc.) within their resolution domains in the ICN environment. The RNs themselves

follow a multi-layered tree structure, providing name resolution services within the scope of different levels' resolution domains. It is important to note that the resolution hierarchy and resolution domains of this hierarchical name resolution system correspond one-to-one with the scheduling hierarchy and scheduling domains in IPCRSA. Each scheduling domain in IPCRSA is also a resolution domain of the hierarchical name resolution system. The RN provides name resolution services within each scheduling domain, as illustrated in Figure 1.

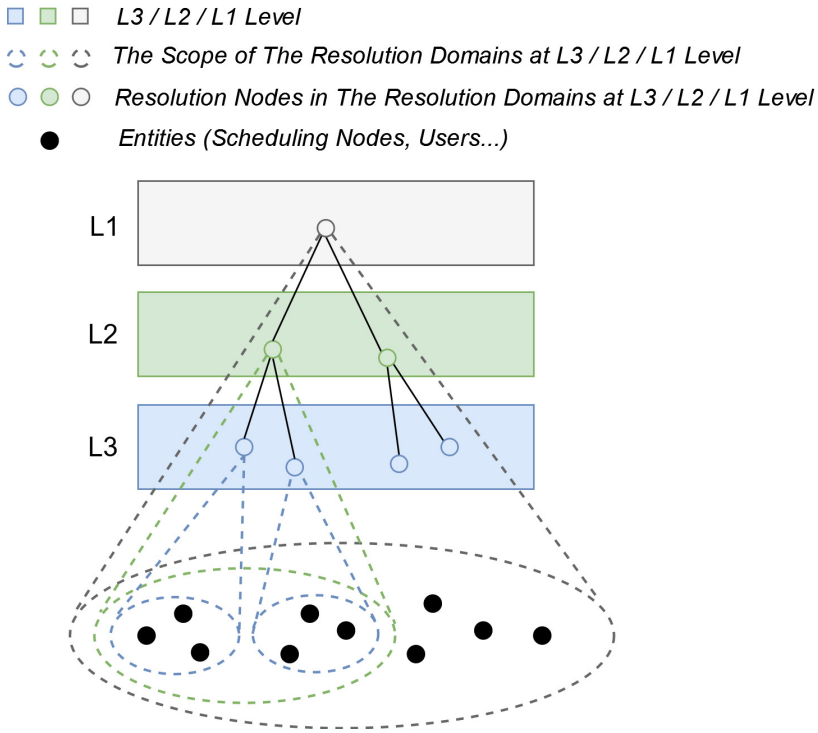


Figure 3. Hierarchical name resolution system in IPCRSA.

The ID Management Module is responsible for interacting with the RN in its scheduling domain, completing the registration, deregistration, and query of the mapping relationship between the SN's RSS_ID_L_x and IP.

- Topology Construction Module

The Topology Construction Module utilizes the ID Management Module and ICN name resolution system to build a semi-distributed structured tree-like topology and structured connections among SNs within each scheduling domain at the L_3 and L_2 levels sequentially. The SN with the minimum average communication delay to other SNs within the domain is selected as the root node, and the root node of the lower-level scheduling domain also belongs to the scheduling domain at the upper level, with the corresponding resource scheduling service identifier (L_3 is the lowest level). Subsequently, within the scheduling domain at the L_1 level, the module establishes a fully distributed unstructured topology and unstructured connections among SNs. Ultimately, the hierarchical topology structure depicted in Figure 1 is constructed.

It should be additionally noted that the Topology Construction Module, through the ID Management Module, queries the address information of other SNs within its scheduling domain from the RN and measures the communication delay with other SNs. The construction of tree-like topologies and structured connections among SNs within the L_3 and L_2 level scheduling domains primarily aims to reduce the overhead associated with the flooding of subsequent computing resource status information among SNs, akin to the method discussed in reference [51]. The establishment of unstructured topology

and connections within the L_1 level scheduling domain primarily aims at load balancing among SNs.

The construction process of the hierarchical topology is as follows:

1. When a new SN comes online, its ID Management Module registers the mapping relationship between RSS_ID_L_3 and IP with the RN within its L_3 level scheduling domain.
2. Within each scheduling domain at the L_3 level, the Topology Construction Module of each SN queries the IP address information of other SNs within its scheduling domain from the RN via the ID Management Module and measures the communication delay with other SNs.
3. The SN with the smallest average communication delay to other SNs within the domain is selected as the root SN for the corresponding L_3 level scheduling domain, with other SNs serving as leaf nodes.
4. Adjacency relationships are established between the root SN and all leaf SNs to form a tree-structured logical topology.
5. The ID Management Module of the root SN of each scheduling domain at the L_3 level registers the mapping relationship between RSS_ID_L_2 and IP with the RN within its L_2 level scheduling domain.
6. Similarly, within each scheduling domain at the L_2 level, the Topology Construction Module of each SN sequentially completes the tasks of measuring the communication delay between SNs, selecting the root SN, and constructing the tree-structured logical topology.
7. The ID Management Module of the root SN of each scheduling domain at the L_2 level registers the mapping relationship between RSS_ID_L_1 and IP with the RN within its L_1 level scheduling domain.
8. Within the scheduling domain at the L_1 level, each SN establishes adjacency relationships with each other, forming a fully distributed, unstructured topology.

Ultimately, the construction of the hierarchical logical topology is completed.

- Resource Status Notification Module

The primary responsibilities of the Resource Status Notification Module are as follows: (1) Fetching communication delay information among SNs within the domain as measured by the Topology Construction Module. (2) Receiving and processing local CRNs' status information reported by connected GNs. (3) Receiving CRNs' status information from other SNs within its scheduling domain. (4) Notifying local CRNs' status information to other SNs within the domain. (5) Aggregating CRNs' status information, performing the fetch of communication delay among SNs and resource status information notification in the corresponding scheduling domain at the upper level, in the condition that the associated SN of the module is the root node of the scheduling domain. (6) Forming a resource status information table and storing it locally.

The GNs can perceive and monitor the status information of CRNs connected to them and report it to the SNs, as elaborated in Section 3.1.2 below. CRNs can be categorized into various types based on their purposes. CRNs of the same type share the same computing resource service identifier, CRS_ID_T_x, indicating that these CRNs can provide computing resource services of the corresponding type T_x. A detailed introduction is provided in the subsequent Section 3.1.3. The notification of computing resource status information relies on the topology constructed by the Topology Construction Module and is disseminated using a flooding mechanism. The relevant fields of the resource status information table and their meanings are shown in Table 3 above. Each entry in the table records network and computing resource status information related to a CRN.

- Computing Resource Scheduling Module

The Computing Resource Scheduling Module serves as the execution unit for computing resource scheduling strategies. In IPCRSA, user resource requests are sent from the Clients to the SN's data plane in the form of computing resource request packets, and the

Clients query the IP addresses of the SNs corresponding to the RSS_ID_L$_x$ through the ICN name resolution system, then send the packets to the designated SN. This process will be elaborated in Section 3.3 of the following text. The Computing Resource Scheduling Module, relying on the status information of various CRNs in the local resource status information table, executes the corresponding computing resource scheduling strategy. It precomputes and generates a computing routing flow table, which is then sent to the data plane based on the southbound interface protocol. This flow table guides the processing and forwarding of computing resource request packets. The flow table matching process of in-transit packets is akin to the execution of scheduling strategies. In other words, by on-path matching, processing, and forwarding the request packets, the module seeks CRNs that meet the User's demands, implementing computing resource scheduling. Furthermore, in the ICN environment where routing is based on IDs, the data plane of the SN can adjust the scheduling targets of the request packets based on the current resource status information during their routing process, thereby enhancing the flexibility of resource scheduling. The specific computing resource scheduling strategies and the data plane's processing and forwarding mechanisms for computing resource request packets during the flow table matching process will be elaborated in Sections 4.1 and 4.2, respectively.

Table 3. Fields of the resource status information table and their meanings.

Field	Meaning
CRS_ID_T$_x$	Provide computing resource service with Type T$_x$
Source_IP	IP Address of the SN corresponding to the CRN
Next_Hop	Source IP of the status announcement information
Hierarchy_Information	Scheduling level of the SN corresponding to the CRN
Sequence_Number	Sequence number of the CRN
Available_CPU_Cores (Core)	Available CPU cores of the CRN
Available_Memory (GB)	Available memory of the CRN
Available_Storage (GB)	Available storage of the CRN
Available_Bandwidth (bps)	Available Bandwidth of the CRN
CPU_Usage (%)	CPU usage of the CRN
Memory_Usage (%)	Memory usage of the CRN
Storage_Usage (%)	Storage usage of the CRN
Bandwidth_Usage (%)	Bandwidth usage of the CRN
Communication_Delay (ms)	Communication Delay between SNs
PUE	Power usage effectiveness of the CRN
Price	Price for the use of this CRN

3.1.2. Gateway Node

In IPCRSA, the GNs serve the functions of User and CRN access and management. On the CRN side, the GNs perceive and monitor the real-time resource status information of each CRN. They identify the types of CRNs, generate computing resource service identifiers, and finally report the identifiers and the real-time status information of corresponding CRNs to the connected SNs. On the User side, when computing resource scheduling is successful, the GNs return a scheduling success response packet to the Clients. Based on the information in the response packet, the Clients establish a connection with the GNs. Users operate the CRNs managed by these GNs to complete relevant service deployment operations.

3.1.3. Computing Resource Node

In IPCRSA, CRNs can be classified into three types based on their purposes: Universal-CRN, Super-CRN, and Intelligent-CRN, as shown in Table 4. These nodes can take the form of physical machines, virtual machines, or containers. Specifically, Universal-CRNs are primarily used for routine computing tasks in consumer and industry internet domains. Super-CRNs are mainly employed in the scientific field to handle large-scale scientific and engineering computing tasks. Intelligent-CRNs are mainly dedicated to processing

tasks related to artificial intelligence. The same type of CRNs have the same computing resource service identifiers, CRS_ID_T_x, indicating that the node can provide corresponding computing resource services with type T_x, as shown in Table 2.

Table 4. CRN types in IPCRSA.

CRN Type	Type Name	ID
T_1	Universal-CRN	CRS_ID_T_1
T_2	Super-CRN	CRS_ID_T_2
T_3	Intelligent-CRN	CRS_ID_T_3

It is important to note that IPCRSA does not impose any restrictions on the classification methods for CRNs. The classification methods mentioned above are just examples, and IPCRSA supports other classification methods and custom computing resource service identifiers for various scenarios.

3.1.4. User and User Client

In IPCRSA, Users are the requesters of computing resources and may be individuals or organizations. Within IPCRSA, each User is assigned a unique identifier, USER_ID, which serves not only as an identity marker but also as its ID within the ICN environment, as depicted in Table 2 above. The mapping relationship between USER_ID and the User's IP address is also maintained by the aforementioned hierarchical name resolution system.

The Client is a software or hardware tool that enables Users to access IPCRSA to initiate resource requests or use resources. Its primary functions are as follows: (1) Identifying the User and generating a USER_ID. (2) Receiving input for resource requests from the User, creating corresponding computing resource request packets, and dispatching them to the SN. (3) Interacting with the hierarchical name resolution system to complete tasks such as registering the mapping relationship between USER_ID and User's IP, and querying for the IP of SN associated with RSS_ID_L_x. (4) Receiving scheduling success or failure response packets and relaying the scheduling outcome back to the User. If the scheduling is successful, the User's service deployment commands can be forwarded from the Client to GN for execution.

3.2. Packet Format Design

Within an evolutionary ICN environment compatible with TCP/IP, IPCRSA designs a computing resource request packet format based on the IPv6 extension header, as illustrated in Figure 4. Firstly, two crucial fields, Source ID and Destination ID, are added to the packet header. The Source ID is filled with USER_ID to identify the User requesting resources, while the Destination ID is filled with the computing resource service identifier CRS_ID_T_x to specify the type of computing resource service requested by the User. The inclusion of these two ID fields enables the implementation of an ID-based routing mechanism in the ICN. Combining this mechanism with SDN technologies allows the convenient implementation of late-binding for IP, enabling modification of the Destination IP field during the packet routing process based on actual requirements, which enhances routing and scheduling flexibility.

Following these two ID fields are the QoS field, Cost field, and Resource Request Quantity field. The QoS field is filled with the User's QoS constraint for resource scheduling delay: Scheduling_Delay. The Cost field is filled with the maximum cost that the User is willing to accept for CRN usage: Max_Price. The Resource Request Quantity field is filled with the specific resource request quantities in CPU, memory, storage, and bandwidth dimensions: RRQ_Core, RRQ_Memory, RRQ_Storage, and RRQ_Bandwidth, respectively. The values of these three fields, along with the Destination ID field's computing resource service identifier, collectively constitute the User's hard constraints on resources.

Next is the Preference field, which is filled with the User's specific resource request preference: RRP_x, indicating the User's inclination toward resources. This field represents a soft constraint.

Finally, there are the Scheduling State field and the Scheduling Target field. The Scheduling State field, filled with the IS_Scheduled flag, is set to 0 when the computing resource request has not been scheduled, and the corresponding Scheduling Target field is left empty. When set to 1, it signifies that the computing resource request has been scheduled, and the corresponding Scheduling Target field is filled with the sequence number of the destination CRN: Sequence_Number.

Each SN in IPCRSA can receive computing resource request packets, read the relevant fields of resource demands and demand preferences in the packet header, make intelligent decisions, and forward packets based on the corresponding computing resource scheduling strategy to accomplish resource scheduling.

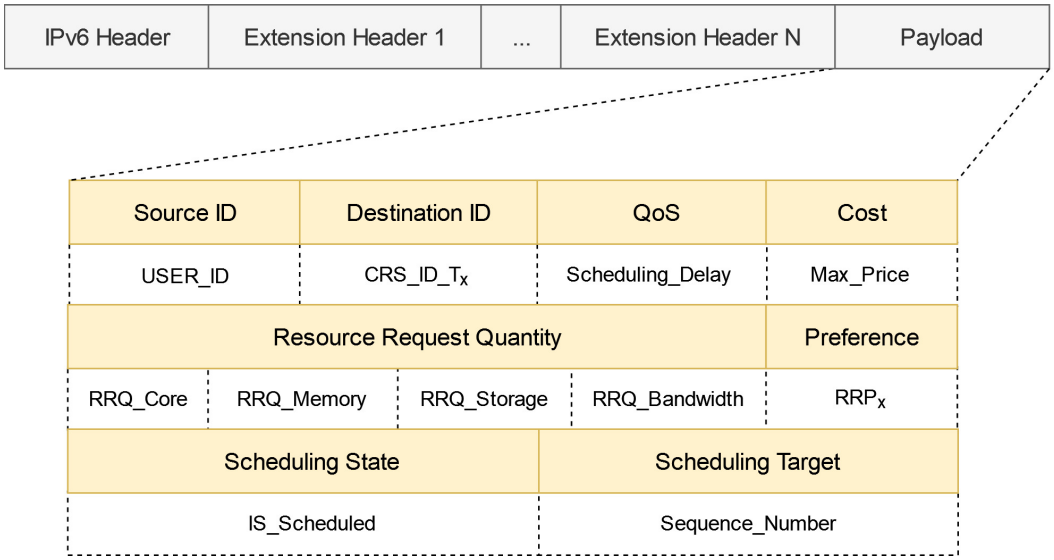


Figure 4. Format of the computing resource request packet.

3.3. Scheduling Workflow

Taking the overall structure of IPCRSA shown in Figure 1, along with its various entities, as an example, this section describes the typical process of computing resource scheduling in IPCRSA, which is illustrated in Figure 5 below.

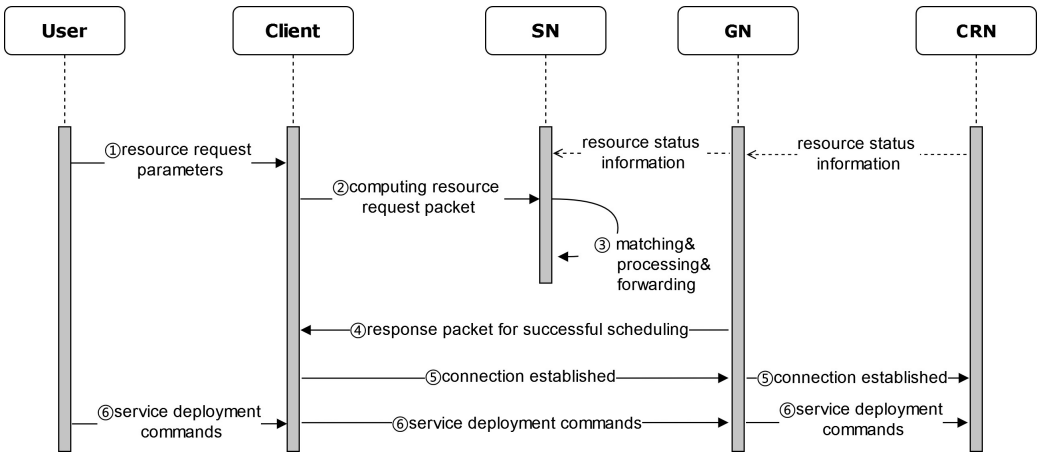


Figure 5. Workflow of computing resource scheduling in IPCRSA.

1. The Client generates a computing resource request packet based on the resource demand parameters the User provides.
2. The ICN name resolution system is used to query an SN within the L_3 level scheduling domain that has the identifier RSS_ID_L_3. The computing resource request packet is then randomly sent to any of these SNs, following the principle of proximity scheduling.
3. Through on-path packet matching, processing, and forwarding between SNs, a CRN that satisfies the User's demands and preferences can be identified.
4. The GN returns a scheduling success response packet to the Client.
5. The Client establishes a connection with the GN.
6. The User, through the GN, completes the deployment operation of the services on the target CRN.

4. Mechanisms and Optimizations

The computing resource scheduling mechanisms are the pivotal component of the computing resource scheduling architectures. It determines which destination CRN is selected and governs how CRNs are chosen. It is the synergistic interaction of these two aspects that achieves computing resource scheduling. This section will focus on introducing the new mechanisms and optimizations of IPCRSA, commencing with these two aspects. Section 4.1 delves into the computing resource scheduling strategies based on user preferences. Section 4.2 elucidates the specific processing and forwarding mechanisms of computing resource request packets in the data plane of each SN under the on-path resource scheduling mechanism.

4.1. Preference-Based Scheduling Strategies

The computing resource scheduling in IPCRSA is preference-based, with different user preferences corresponding to distinct computing resource scheduling strategies. The execution of the preference-based computing resource scheduling strategies involves two stages: The first stage evaluates various CRNs using a node-scoring algorithm corresponding to the user preferences based on the status information of each CRN in the local resource status information table. The second stage involves selecting the destination CRN based on the scoring results.

In IPCRSA, Users with varying resource request preferences are most concerned about different status parameters of computing resource nodes, which are defined in this paper as priority indicators corresponding to each preference. Currently, IPCRSA has designed specific node-scoring algorithms for various resource request preferences and preference combinations, as outlined in Table 5, with the input parameters being the priority indicators corresponding to them.

Table 5. Resource request preferences in IPCRSA.

RRP Type	Type Name	Input Parameters (Priority Indicators)
RRP_0 (Default)	Load-First	CPU_Usage/Memory_Usage/Storage_Usage/ Bandwidth_Usage
RRP_1	CPU-First	Available_CPU_Cores
RRP_2	Memory-First	Available_Memory
RRP_3	Storage-First	Available_Storage
RRP_4	Bandwidth-First	Available_Bandwidth
RRP_5	Cost-First	Price
RRP_6	Efficiency-First	PUE
RRP_7	Delay-First	Communication_Delay
...	...	...

When a User has no specific preferences, a node-scoring algorithm corresponding to the default preference, i.e., the load-first preference, is employed to score each CRN. Specifically, the SN calculates the load score for each CRN using Equations (1)–(4). Here,

u_{i_c}, u_{i_m}, u_{i_s}, and u_{i_b} represent the normalized utilization rates of CPU, memory, storage, and bandwidth, respectively, for the ith $(i \in N)$ CRN. N represents the set of CRNs' ordinal numbers. These parameters correspond to CPU_Usage, Memory_Usage, Storage_Usage, and Bandwidth_Usage in the local resource status information table. $\bar{u}_i$ is the mean of the normalized utilization rates for these four basic type resources. $MaxNodeScore$ is the predefined maximum score for CRNs, typically set to 10 or 100. $s_i(i \in N)$ represents the final score for the CRN. The higher the score, the lower the load on the CRN.

$$\bar{u}_i = \frac{(u_{i_c} + u_{i_m} + u_{i_s} + u_{i_b})}{4} \tag{1}$$

$$s_{i_1} = (1 - \sqrt{\frac{(u_{i_c} - \bar{u}_i)^2 + (u_{i_m} - \bar{u}_i)^2 + (u_{i_s} - \bar{u}_i)^2 + (u_{i_b} - \bar{u}_i)^2}{4}}) \times MaxNodeScore \tag{2}$$

$$s_{i_2} = \frac{(1 - u_{i_c}) + (1 - u_{i_m}) + (1 - u_{i_s}) + (1 - u_{i_b})}{4} \times MaxNodeScore \tag{3}$$

$$s_i = \frac{s_{i_1} + s_{i_2}}{2},\ i \in N \tag{4}$$

When Users have preferences in terms of the resource itself, cost, energy consumption, and scheduling delay, i.e., having one or more preferences from RRP_1 to RRP_7, the SN scores each CRN based on the TOPSIS comprehensive evaluation method. Firstly, an original data matrix $X_{n\times m}$ is constructed based on n CRNs and m input parameters corresponding to user preferences in the local resource status information table. The row vector represents the value vector of m input parameters for the ith $(i \in N)$ CRN in this scoring algorithm. The column vector represents the values of the n CRNs on the jth $(j \in M)$ input parameter. M represents the set of input parameters' ordinal numbers.

$$X_{n\times m} = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix} \tag{5}$$

Next, it is necessary to "Benefit-Cost" adjust the original data matrix. Among the input parameters corresponding to the preferences of RRP_1-RRP_7 mentioned above, Available_CPU_Cores, Available_Memory, Available_Storage, and Available_Bandwidth are beneficial attributes, where larger values are better. As input parameters for the scoring algorithm, their "Benefit-Cost" values are the same as their original values, i.e., $x_{ij}^+ = x_{ij}$. On the other hand, Price, PUE, and Communication_Delay are cost attributes, where smaller values are better. If they are used as input parameters, adjustment is required using Equations (6) and (7). After adjusting the original data, the "Benefit-Cost" adjusted data matrix $X_{n\times m}^+$ is obtained.

$$max\{x_j\} = max\{x_{1j}, x_{2j}, x_{3j}, \ldots x_{nj}\} \tag{6}$$

$$x_{ij}^+ = max\{x_j\} - x_{ij} \tag{7}$$

$$X_{n\times m}^+ = \begin{bmatrix} x_{11}^+ & \cdots & x_{1m}^+ \\ \vdots & \ddots & \vdots \\ x_{n1}^+ & \cdots & x_{nm}^+ \end{bmatrix} \tag{8}$$

Then, it is required to utilize Equation (9) to standardize the $X_{n\times m}^+$, eliminating the influence of different dimensions and obtaining the standardized data matrix $Z_{n\times m}$.

$$z_{ij} = \frac{x_{ij}^+}{\sqrt{\sum_{i=1}^{n} \left(x_{ij}^+\right)^2}} \tag{9}$$

$$Z_{n\times m} = \begin{bmatrix} z_{11} & \cdots & z_{1m} \\ \vdots & \ddots & \vdots \\ z_{n1} & \cdots & z_{nm} \end{bmatrix} \tag{10}$$

Finally, based on the standardized data matrix, Equations (11)–(15) are employed to calculate the comprehensive score $s_i (i \in N)$ for each CRN. A higher score indicates a better overall performance of the CRN across the dimensions of input parameters corresponding to user preferences.

$$max\{z_j\} = max\{z_{1j}, z_{2j}, z_{3j}, \ldots z_{nj}\} \tag{11}$$

$$min\{z_j\} = min\{z_{1j}, z_{2j}, z_{3j}, \ldots z_{nj}\} \tag{12}$$

$$d_i^+ = \sqrt{\Sigma_{j=1}^m \omega_j \left(max\{z_j\} - z_{ij}\right)^2},\ i \in N \tag{13}$$

$$d_i^- = \sqrt{\Sigma_{j=1}^m \omega_j \left(min\{z_j\} - z_{ij}\right)^2},\ i \in N \tag{14}$$

$$s_i = \frac{d_i^-}{d_i^- + d_i^+} \times MaxNodeScore,\ i \in N \tag{15}$$

ω_j represents the weight of each input parameter, determined through the entropy weight method. Firstly, utilize Equation (16) to standardize the original data matrix. $min\{x_j\}$ is the minimum values of $\{x_{1j}, x_{2j}, x_{3j}, \ldots x_{nj}\}$, which is opposite to $max\{x_j\}$. Subsequently, for the jth $(j \in M)$ input parameter, calculate its information entropy E_j using Equations (17) and (18). Finally, obtain the weight ω_j using (19), with $0 \leq \omega_j \leq 1$, and the sum of m weights is equal to 1.

$$x_{ij}^{'} = \frac{x_{ij} - min\{x_j\}}{\max\{x_j\} - \min\{x_j\}} \tag{16}$$

$$P_{ij} = \frac{x_{ij}^{'}}{\Sigma_{k=1}^n x_{kj}^{'}} \tag{17}$$

$$E_j = -\frac{1}{\ln n} \sum_{i=1}^{n} P_{ij} \times \ln P_{ij},\ j \in M \tag{18}$$

$$\omega_j = \frac{1 - E_j}{m - \sum_{k=1}^m E_k},\ j \in M \tag{19}$$

After obtaining scores for each CRN, it is necessary to select the target CRN based on the score results.

Initially, CRNs that meet the hard constraints indicated by the following condition (20) are added to the candidate CRNs set C. Herein, A represents the set of all CRNs.

Subsequently, within the candidate CRNs set, node selection is performed based on the score results. If a greedy algorithm is employed, always choosing the highest-scored node may exacerbate resource competition issues in distributed scheduling scenarios, leading to load imbalance and scheduling failures. On the other hand, adopting a random selection algorithm fails to meet user preferences. Therefore, in IPCRSA, the SN employs a roulette-selection algorithm for CRN selection. In this algorithm, the SN randomly selects CRNs with scores from high to low in descending order of probability, simultaneously balancing the load of CRNs and meeting user preferences.

$$\begin{cases} & C = \{a \in A\} \\ a \ \textbf{s.t.} & CRS_ID_T_x \ Requested == CRS_ID_T_x \ Available \\ & Scheduling_Delay \ \geq \ Total \ Communication_Delay \\ & Max_Price \ \geq Price \\ & RRQ_Core \leq Available_CPU_Cores \\ & RRQ_Memory \leq Available_Memory \\ & RRQ_Storage \leq Available_Storage \\ & RRQ_Bandwidth \leq Available_Bandwidth \end{cases} \tag{20}$$

The roulette-selection algorithm is commonly used in genetic algorithms, where the probability of selecting a candidate node is directly proportional to its composite score. Firstly, the probability p_i of selecting each of the n^* candidate CRNs is calculated based on their scores. Subsequently, the cumulative probabilities q_i for each candidate CRN are computed. Finally, a uniformly distributed random number r is generated within $[0, 1]$, and the range in which r falls determines the ultimately selected CRN. The overall process is illustrated by Equations (21)–(23).

$$p_i = \frac{s_i}{\Sigma_{k=1}^{n^*} s_k}, \ i \in N^* \tag{21}$$

$$q_i = \Sigma_{j=1}^{i} p_j, \ i \in N^* \tag{22}$$

$$select = \begin{cases} 1, & when \quad r \leq q_1 \\ k, & when \quad q_{k-1} < r \leq q_k \ (2 \leq k \leq n^*) \end{cases} \tag{23}$$

Based on the aforementioned scoring and selection algorithms of CRNs, the preference-based computing resource scheduling algorithm in IPCRSA is formulated in Algorithm 1.

Algorithm 1 Preference-based computing resource scheduling in IPCRSA

Input: $CRS_ID_T_x$ *Requested*; *Scheduling_Delay*; *Max_Price*; *RRQ_Core*; *RRQ_Memory*; *RRQ_Storage*; *RRQ_Bandwidth*; RRP_x; *CRNs* with Status Information Table;
Output: *CRN* that meets the User's resource requirements

if $RRP_x = RRP_1$ **then**
  score each *CRN* in *CRNs* with Equations (1)-(4)
else
  score each *CRN* in *CRNs* with Equations (5)-(19)
end if
for *CRN* in *CRNs* **do**
  if *CRN* can satisfy the constraint of Condition (20) **then**
    add *CRN* to the set of *Candidate CRNs*
  end if
end for
for *CRN* in *Candidate CRNs* **do**
  calculate the probability that the *CRN* may be selected with Equation (21)
  calculate the cumulative probability of the *CRN* with Equation (22)
end for
$r \leftarrow rand()/RAND_MAX$
//*RAND_MAX* is the largest number that the *rand*() function can return
for *CRN* in *Candidate CRNs* **do**
  if *r* is in the probability interval of *CRN* **then**
    return *CRN*
  end if
end for
return *NULL*

Additionally, it is important to note that the resource scheduling in IPCRSA is not limited solely to the user preferences and corresponding scheduling strategies shown in Table 5. It has the flexibility to incorporate additional user preference options and corresponding scheduling strategies based on new user requirements, demonstrating high scalability.

Finally, the key variables involved in this section are summarized in Table 6 below for reference.

Table 6. Notation table.

Variable	Definition
u_{i_c}, u_{i_m}, u_{i_s}, u_{i_b}	The normalized utilization rates of CPU, memory, storage, and bandwidth for the ith CRN
N	The set of CRNs' ordinal numbers
$\bar{u}_i$	The mean of the normalized utilization rates for CPU, memory, storage, and bandwidth for the ith CRN
$MaxNodeScore$	The predefined maximum score for CRNs
s_i	The final score for the ith CRN
n	Total number of CRNs
m	Total number of Input Parameters
$X_{n\times m}$	The original data matrix
i	CRN's ordinal number
j	Input Parameter's ordinal number
M	The set of input parameters' ordinal numbers
$X^{+}_{n\times m}$	The "Benefit-Cost" adjusted data matrix
$Z_{n\times m}$	The standardized data matrix
ω_j	The weight for the jth input parameter
E_j	The information entropy for the jth input parameter
C	The set of the candidate CRNs
A	The set of all CRNs
n^*	Total number of candidate CRNs
p_i	The probability of selecting the ith candidate CRN
q_i	The cumulative probabilities for the ith candidate CRN

4.2. Processing and Forwarding Mechanisms for Resource Request Packets

Upon receiving a computing resource request packet in the data plane, the SN first reads the IS_Scheduled, CRS_ID_T$_x$, and RRP$_x$ parameters in the packet header to confirm the status of the request, the resource service type requested by the User, and the user preferences.

If the request has not been scheduled, the scheduling strategy corresponding to user preferences is executed, which aims to search for a CRN that satisfies the User's resource demands in all dimensions.

- If a non-local CRN is found, the Destination IP field of the packet is modified to the destination CRN's Source_IP, the IS_Scheduled is set to 1, the Scheduling Target field is modified to the destination CRN's Sequence_Number, and the packet is finally forwarded.
- If a local CRN is found, the packet's destination IP is modified to the GN's IP corresponding to the SN and the packet is forwarded. The GN then returns a scheduling success response packet to the Client.
- If no suitable CRN is found, and the SN has a superior node, the destination IP is modified to the IP of the root node in the scheduling domain, allowing exploration of potential CRNs in other scheduling domains. If there is no superior node, a scheduling failure response packet is directly returned to the Client.

When the computing resource request has been scheduled, the SN first checks whether the available resource quantity of the destination CRN can meet the User's resource demands.

- If it can, the packet is forwarded to the GN corresponding to the SN and a scheduling success response packet is returned.
- If it cannot, the SN performs a rescheduling based on the User's resource demands and preference parameters, modifying relevant fields in the packet accordingly.

It is noteworthy that, as a distributed computing resource scheduling architecture, IPCRSA also adheres to the CAP theorem for distributed systems [52]. To enhance availability, IPCRSA, like many other distributed resource scheduling systems, follows the BASE principle, which guarantees only partition tolerance while striking a balance between availability and consistency. IPCRSA ensures only the eventual consistency of resource status information across SNs within the same scheduling domain, sacrificing strong consistency to improve the availability of resource scheduling. Consequently, it must grapple with the issue of resource contention resulting from the asynchronous nature of resource status information in the soft state.

For instance, suppose a CRN has 16 CPU cores, 64 GB of memory, 1024 GB of storage, and 10 MB of available bandwidth. Three computing resource requests, A, B, and C, sequentially request resources of this node within the same resource status notification interval. The scheduling results based on the quota allocation of resources are illustrated in Figure 6. After allocating a fixed amount of resources to requests A and B, the remaining available resources are four CPU cores, 8 GB of memory, 256 GB of storage, and 1 MB of available bandwidth, which cannot satisfy the resource demand of request C.

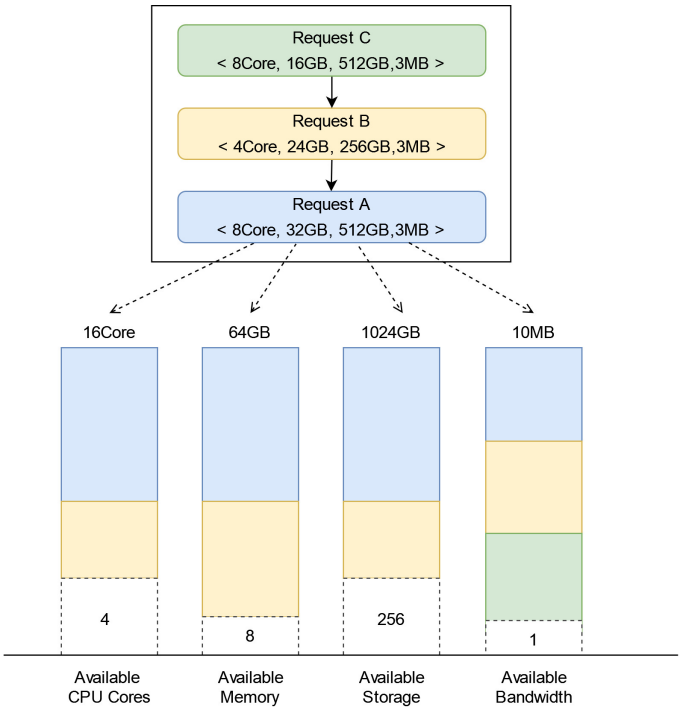


Figure 6. An example of computing resource scheduling in IPCRSA.

In traditional Computing Network resource scheduling architectures, request C would remain pending, resulting in scheduling failure. However, in IPCRSA, the hierarchical scheduling structure accelerates the synchronization of resource status information, and the roulette-selection-based CRN selection mechanism ensures load balancing among CRNs. Both mechanisms alleviate resource contention issues in the soft state to some extent, improving the scheduling success rate. Furthermore, each SN in IPCRSA maintains a computing routing flow table. When the local resource status information table is updated, the computing routing flow table is updated accordingly. Computing resource request packets in the IPCRSA route based on IDs, enabling dynamic late-binding of IP addresses. In the aforementioned scheduling scenario, the resource request packet corresponding to request C can be promptly matched

with the updated flow table based on changes in resource status information. This enables timely rescheduling, forwarding the packet to other candidate CRNs for resource invocation, and enhancing the overall scheduling success rate.

5. Simulation

This section centers on the performance analysis of IPCRSA. Section 5.1 provides a brief overview of the simulation setup and relevant assumptions. Sections 5.2 and 5.3 validate and analyze the scheduling performance of IPCRSA under the combined influence of the on-path scheduling mechanism and the preference-based scheduling strategies from the two metrics of the success rate of scheduling and the average value of priority indicators corresponding to preferences, respectively. Section 5.4 provides a comparative discussion and summary of the method described in this paper and other methods in the simulation experiment.

5.1. Simulation Setup and Assumptions

NetworkX is a Python library designed for the creation, analysis, and visualization of complex networks [53]. It offers Users a concise API that is easy to learn and use. The simulation experiment utilized NetworkX to create a hierarchical topology, as depicted in Figure 7, to simulate the topology structure of IPCRSA. In the diagram, blue and red nodes represent root SNs in the L_3 and L_2 level scheduling domains, respectively, while black nodes are leaf SNs within the L_3 level scheduling domains. Overall, the simulation topology includes nine L_3 level domains, three L_2 level domains, and one L_1 level domain. The intra-domain link delays between SNs in L_3, L_2, and L_1 level domains are set to random values within the ranges $[1\,ms, 5\,ms]$, $[5\,ms, 15\,ms]$, and $[15\,ms, 30\,ms]$, respectively. Considering the nanosecond-level high-speed matching, processing, and forwarding capabilities of the SN's data plane for packets [54], the processing delay of packets at SNs can be neglected. Additionally, in this simulation experiment, it is assumed that each SN has a stable and minimal communication delay with its corresponding GN, which is negligible compared to the link delays between SNs. Therefore, the link delays between SNs become the primary factor that affects resource scheduling delays. It is important to note that, to focus on the computing resource scheduling process itself and make simulated topology more straightforward and understandable, GNs and CRNs are not labeled in the simulated topology diagram in Figure 7. However, they still exist and maintain the connectivity relationships with the SNs, as described in Section 3.1 in this simulation experiment.

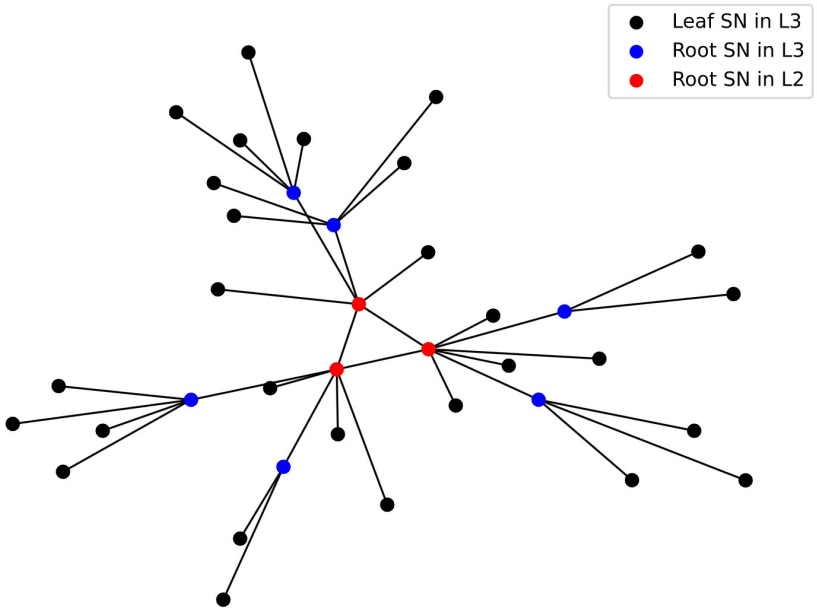


Figure 7. Topology in simulation experiment.

Furthermore, given that the simulated topology is situated within an evolutionary ICN environment that routes based on IDs, this simulation experiment introduces a hashing algorithm to generate flat IDs for Users, SNs, and CRNs within the simulation environment. When implementing the resource scheduling function on each SN, a separate subprocess is included to facilitate the late-binding for IP addresses, allowing for the late-binding of scheduling targets in subsequent experiments. Additionally, the simulation experiment also incorporates a simulated hierarchical name resolution system designed to maintain the mapping relationship between IDs and IP addresses within the simulation environment, thereby aiding in the scheduling of computing resources in this simulation experiment.

In this simulation experiment, computing resource requests and the status information of CRNs are sourced from an open dataset provided by Alibaba Cloud [55]. In terms of requests, the dataset only includes the quantity of resources requested by Users in dimensions of CPU, memory, and storage. Therefore, it can be assumed that Users have no strict constraints on bandwidth, price, scheduling delay, or computing resource service types in this simulation experiment. Regarding the status information of CRNs, the dataset offers the total amount of resources and corresponding utilization rates in terms of CPU, memory, and storage for each CRN. For subsequent performance validation, the PUE values and usage prices for each CRN are set to random values within the ranges $(1, 4]$ and $(0, 20]$, respectively. It is also assumed that all CRNs can provide the same computing resource service types and share identical computing resource service identifiers.

Under the aforementioned assumptions, based on the request information provided by the dataset, $5\,w/10\,w/15\,w/20\,w$ computing resource request packets are formed. These resource request packets are randomly dispatched to various SNs within the topology illustrated in Figure 7. The intervals between the arrival of each request at the SN's data plane follow an exponential distribution with a scale parameter set to $40\,w$. The dataset's CRNs are randomly linked to the GNs corresponding to the SNs in the simulated topology, and they report resource status information to the SNs through the GNs. The SNs then exchange resource status information to form a resource status information table. For cross-level aggregation, the TOPSIS and entropy weight method are employed to select CRNs with relatively higher values in all resource dimensions, representing lower-level scheduling domains. Each SN, based on the demand information from the computing resource request packets and the state information from the resource status information table, executes the corresponding computing resource scheduling strategy to accomplish the scheduling. Detailed experimental parameter configurations are provided in Table 7 for reference.

Table 7. Simulation configuration.

Parameter	Description
Topology	NetworkX
Number of SNs	37
Number of L_3 scheduling domains	9
Number of L_2 scheduling domains	3
Number of L_1 scheduling domains	1
Communications delay between SNs in L_3	$[1\,ms, 5\,ms]$
Communications delay between SNs in L_2	$[5\,ms, 15\,ms]$
Communications delay between SNs in L_1	$[15\,ms, 30\,ms]$
Data Set	https://github.com/alibaba/clusterdata (accessed on 1 January 2018)
Number of resource request packets	$5\,w/10\,w/15\,w/20\,w$
The scale of the exponential distribution	$40\,w$
PUE of CRNs	$(1, 4]$
Price for the use of CRNs	$(0, 20]$

The main metrics emphasized in this simulation experiment include the success rate of scheduling and the average value of priority indicators corresponding to user

preferences. The former refers to the proportion of successfully scheduled request packets in all request packets. In the context of distributed resource scheduling, a higher success rate of scheduling indicates a stronger capability of the scheduling architecture to address resource contention issues when the total quantity of request packets is held constant. The latter denotes the average value of the preference-associated priority indicators on the CRNs occupied by successfully scheduled request packets. When the number of successfully scheduled request packets is the same, the magnitude of the average value of priority indicators reflects the capability of the computing resource scheduling strategies to meet diverse user demands and preferences.

5.2. Success Rate of Scheduling

In the context of Computing Network, reference [56] compares three scheduling schemes: CFN-dyncast, Native DNS scheduling, and Compute-Aware DNS scheduling. CFN-dyncast directs requests to a single "optimal" CRN based on the status information of both computing and network resources. The Native DNS scheduling scheme guides requests to various CRNs according to statically configured weights. Meanwhile, the Compute-Aware DNS scheduling scheme dynamically adjusts weights based on the status information of resources, facilitating dynamic load balancing. It is noteworthy that these schemes operate within a flat architecture and rely on the traditional TCP/IP network for scheduling.

This simulation experiment compares the three aforementioned schemes with the IPCRSA. To ensure comparability, the same simulation topology, as shown in Figure 7, is employed for all three schemes, treating the hierarchical topology as a flat topology for scheduling. In terms of specific scheduling strategies, IPCRSA utilizes the computing resource scheduling strategy corresponding to the default preference. This involves scoring each CRN based on its load status and subsequently selecting the CRN using the roulette-selection algorithm. Under the same scoring method, the CFN-dyncast scheme directly chooses the CRN with the highest score. The Compute-Aware DNS scheduling scheme adopts the same scheduling strategy as IPCRSA, achieving dynamic weight adjustments. The Native DNS scheduling scheme defaults to equal static weights, i.e., round-robin scheduling. However, in contrast to the scheduling of service requests in [56], the candidate set of CRNs for basic resource requests in service deployment scenarios undergoes high dynamic changes, making it challenging to perform effective scheduling based on static weights. Additionally, considering the similarity in the performance of random scheduling and round-robin scheduling when the number of requests is large, this experiment replaces the static-weight setting method with a random selection method for scheduling. Finally, the performance of the four schemes is compared under the Scheduling_Delay of $10\,ms/50\,ms/100\,ms/150\,ms$.

Figure 8 present the scheduling performance comparison of the four schemes under different scheduling delay constraints. It can be observed from the figures that, regardless of the scheduling delay constraint, IPCRSA consistently achieves an overall higher scheduling success rate compared with the other three schemes. The advantage of IPCRSA becomes more pronounced when the number of computing resource requests is relatively small. Particularly, when the total number of requests is $5\,w$, IPCRSA demonstrates a minimum improvement of 24% and a maximum improvement of 63% in scheduling success rate compared with those of the other three schemes. As the number of requests increases, resource competition intensifies, and constrained by the total available resources of CRNs, the scheduling success rates of all four schemes decrease. However, IPCRSA maintains its advantageous position. These conclusions thoroughly demonstrate the superiority of IPCRSA in improving scheduling success rates, primarily attributed to its on-path computing resource scheduling mechanism. This mechanism enables computing resource request packets to dynamically adjust their scheduling destinations during the routing process based on changes in resource status information, enhancing the flexibility of routing and scheduling.

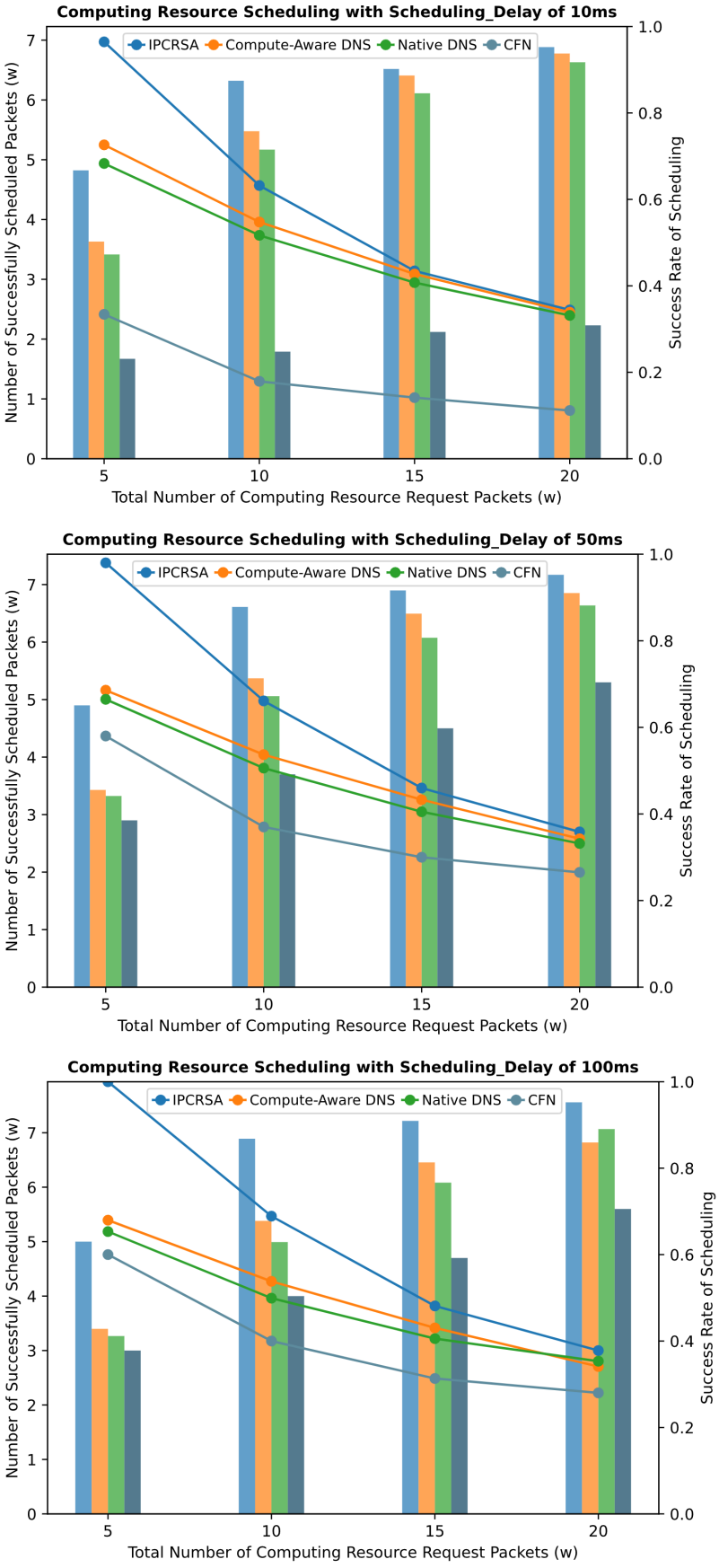


Figure 8. *Cont.*

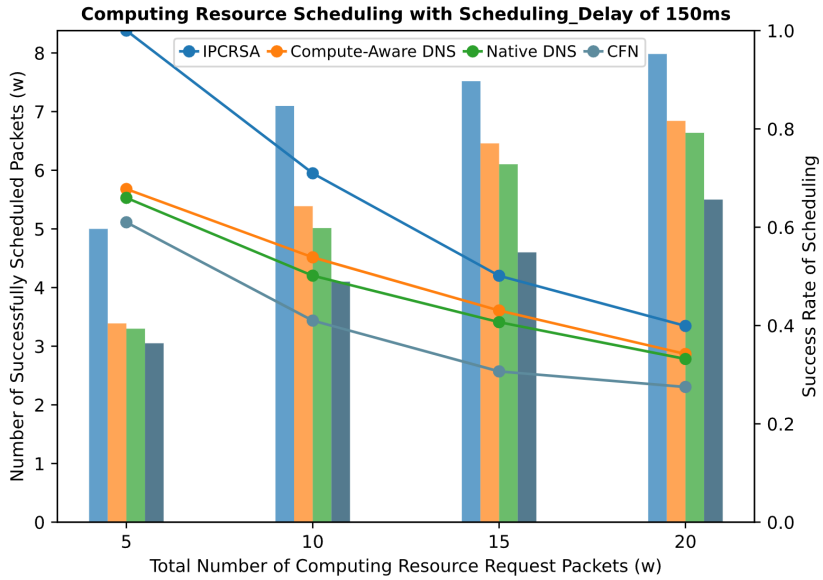


Figure 8. Success rate of scheduling with different Scheduling_Delay.

Additionally, this experiment analyzes the impact of different scheduling delay constraints on the scheduling success rate of IPCRSA. The results from Figure 9 indicate that as the constraints on scheduling delay gradually loosen, the scheduling success rate of IPCRSA increases accordingly. The reason behind this observation is that relaxed scheduling delay constraints provide computing resource request packets with more opportunities for matching, processing, forwarding, and rescheduling, thereby enhancing the scheduling success rate.

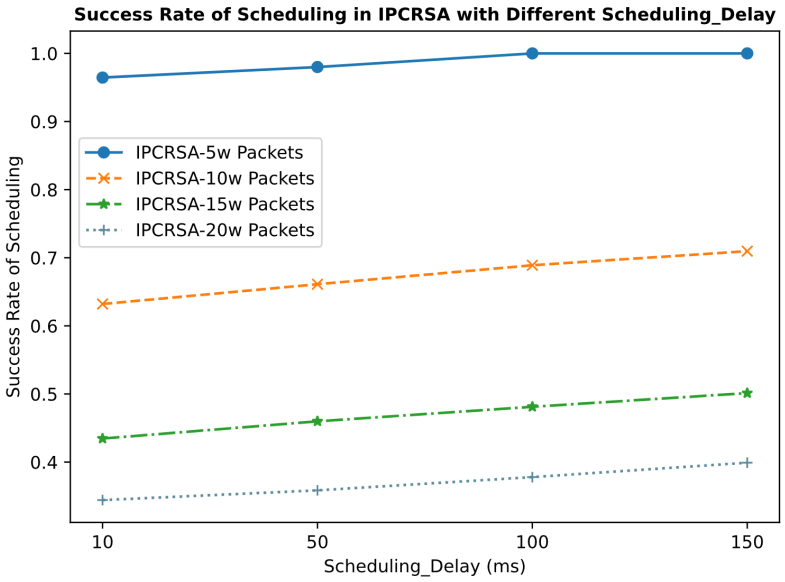


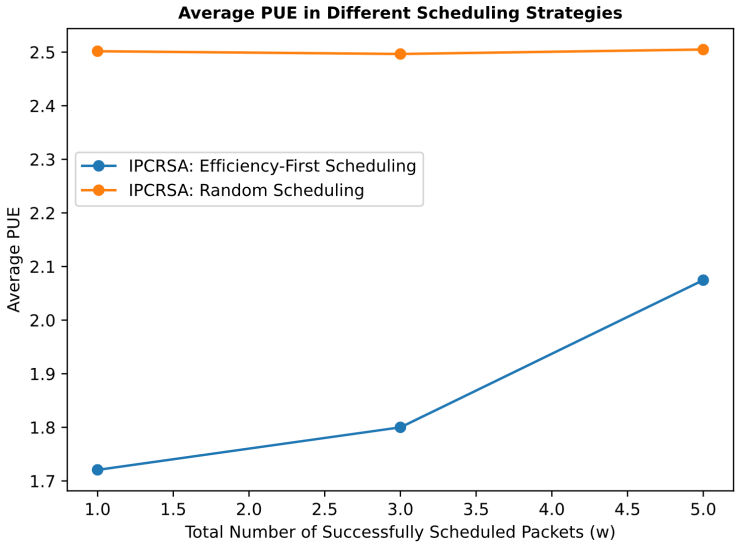
Figure 9. Success rate of scheduling in IPCRSA with different Scheduling_Delay.

5.3. Average Value of Priority Indicators Corresponding to Preferences

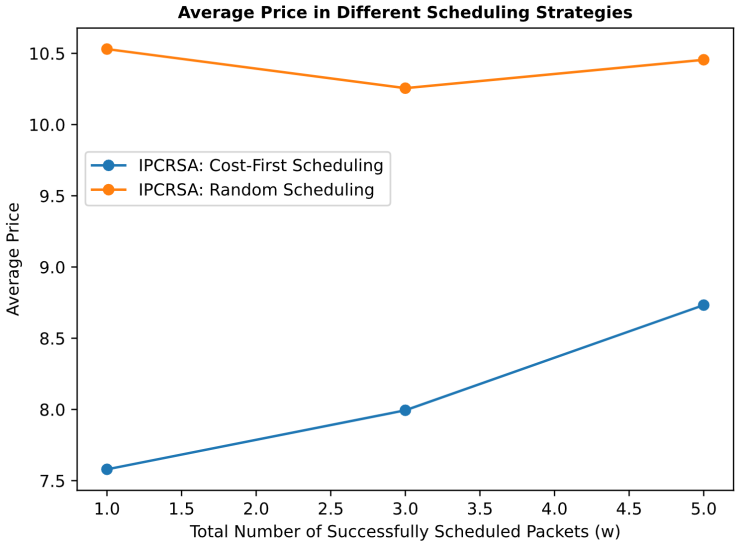
To validate the scheduling performance of the preference-based computing resource scheduling strategies in IPCRSA, this simulation experiment takes the examples of two strategies corresponding to two user preferences: Efficiency-First and Cost-First (referred to as the Efficiency-First strategy and Cost-First strategy, respectively). These strategies are implemented in the simulated topology shown in Figure 7 and compared against the random-scheduling strategy. The total number of computing resource request packets is set to 10 w, with a Scheduling_Delay of 100 ms. The vital comparative metrics are the average

values of the priority indicators corresponding to the above two user preferences, i.e., the average PUE and the average usage price.

Figure 10a presents the average PUE of CRNs occupied by requests under the Efficiency-First strategy and random-scheduling strategy, with successfully scheduled packet quantities of $1\,w/3\,w/5\,w$. The figure shows that the Efficiency-First strategy achieves an overall lower average PUE compared with the random-scheduling strategy. Particularly, when the number of successfully scheduled requests reaches $1\,w$, the average PUE of occupied CRNs is reduced by up to 31%. Similarly, Figure 10b provides a comparison of the Cost-First strategy and random-scheduling strategy. It shows that the Cost-First strategy obtains a lower overall CRN usage price compared with the random-scheduling strategy. Especially with $1\,w$ successfully scheduled packets, the average usage price is reduced by approximately 28%. Regarding scheduling delay, Figure 11 reveals that although the average scheduling delay for both Efficiency-First and Cost-First strategies is slightly higher than that of the random-scheduling strategy, the difference does not exceed $1\,ms$. This difference is less than 1% of the scheduling delay constraint, which is significantly higher than the average scheduling delay. Given the obtained benefits, such overhead is evidently acceptable.



(**a**) Average PUE in Different Strategies



(**b**) Average Price in Different Strategies

Figure 10. Average value of priority indicators in IPCRSA with different scheduling strategies.

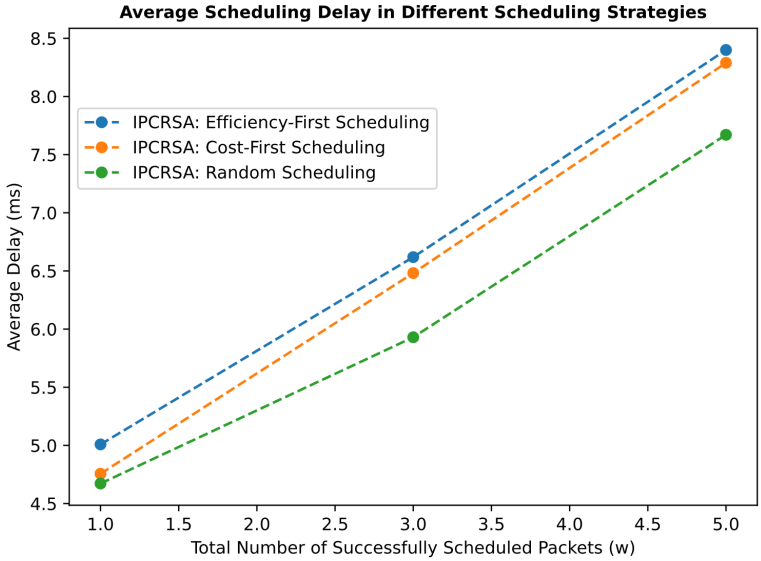


Figure 11. Average scheduling delay in IPCRSA with different scheduling strategies.

The conclusions above strongly demonstrate the advantages of the preference-based computing resource scheduling strategies in IPCRSA. It simultaneously takes into account Users' preferences for resource demands and the load balancing of resource scheduling, meeting diverse user requirements with low scheduling delay overhead.

5.4. Discussion

Regarding the success rate of scheduling, the summary of the comparative schemes in the aforementioned simulation experiment is presented in the following Table 8.

Table 8. Comparative schemes in the simulation experiment.

Approach	Distributed or Centralized	Network Type	Scheduling Strategy	Discussions
CFN-dyncast	Distributed	TCP/IP	Select optimal CRN	Unable to perform load balancing and exacerbating resource competition
Native DNS	Distributed	TCP/IP	Select Randomly	Load balancing was performed, but it was unable to further alleviate resource contention
Compute-Aware DNS	Distributed	TCP/IP	Probabilistic selection	Load balancing was performed, but it was unable to further alleviate resource contention
IPCRSA	Distributed	ICN	Probabilistic selection with rescheduling (late-binding)	Load balancing was performed, and it can further alleviate resource contention

The scheduling scheme based on CFN-dyncast always selects the optimal CRN, which does not allow for load balancing among CRNs and may even exacerbate the problem of resource competition in distributed resource scheduling scenarios. The scheme based on Native DNS randomly selects the CRN, which, to some extent, balances the load and alleviates the resource competition issue. However, due to the limited flexibility of resource scheduling in the TCP/IP network, the resource competition problem cannot be further improved. The scheduling scheme based on Compute-Aware DNS makes a probabilistic selection of the CRN using the roulette-selection algorithm, which also allows for load balancing. However, it also faces the limitation of resource scheduling flexibility in the TCP/IP environment. The IPCRSA scheme proposed in this paper also uses the roulette-selection algorithm for the probabilistic selection of resource nodes, but its on-path computing resource scheduling mechanism in the ICN environment allows the computing resource request packets to adjust the scheduling targets in a timely manner based on the

current resource status information during the routing process, thereby greatly enhancing the flexibility of routing and scheduling. Therefore, it can further alleviate the resource competition problem while achieving load balancing, thus having an advantage in the success rate of scheduling compared with that of the previous three schemes.

In terms of the average value of priority indicators, while the scheme of randomly selecting CRNs can effectively balance the load, it fails to accommodate user preferences. On the other hand, the IPCRSA's computing resource scheduling mechanism, based on user preferences, can balance the load among CRNs with minimal and negligible scheduling latency overhead while also taking into account user preferences, thus meeting diverse user requirements.

Finally, a retrospective analysis of the simulation experiment is conducted. Although it successfully validated and analyzed the scheduling performance of the architecture discussed in this paper, demonstrating its superiority, there are still areas for improvement. Firstly, due to the scarcity of accessible open-source datasets of real-world computing resource clusters, the simulation experiment was based on a single dataset, which may lack generality due to its limited information. In future work, more comprehensive datasets could be employed to further validate the performance of the proposed architecture. Secondly, some modules within the current simulation environment, such as the hierarchical ICN name resolution system, were developed in previous works. While these modules fulfill basic name resolution functions, they may not match the performance of the latest standalone name resolution systems. Future efforts could focus on upgrading and improving these modules to align with the latest research findings. Lastly, the computing resource scheduling architecture discussed in this paper is based on an evolutionary ICN architecture, with the core aim of deploying the proposed scheduling methods in the real world at a lower cost. Future work could involve validating these methods in the real world to achieve a more "realistic" "simulation environment".

6. Conclusions

This paper proposes a novel ICN-based on-path computing resource scheduling architecture, IPCRSA, in the context of the Computing Network, primarily addressing the fundamental resource scheduling issues under service deployment requests. IPCRSA adopts a hierarchical structure. By embedding resource requirements and preference parameters in the computing resource request packets, the SNs can dynamically adjust their scheduling destinations based on demand parameters and current resource status information during the packet routing process. This effectively alleviates resource contention issues in distributed scheduling scenarios, enhancing the overall scheduling success rate. Additionally, IPCRSA introduces preference-based computing resource scheduling strategies, where SNs execute corresponding resource scheduling strategies based on the Preference field in the packet header, catering to diverse user requirements. Finally, this paper presents a simulation implementation and performance analysis of IPCRSA. Experimental results demonstrate that IPCRSA has advantages in improving scheduling success rates. Moreover, as the scheduling delay constraints become loose, it exhibits even better scheduling performance. The preference-based computing resource scheduling strategies in IPCRSA achieve a balance between user preferences and resource scheduling load balancing, meeting diverse user requirements with minor overhead.

Although IPCRSA shows significant advantages in scheduling performance, there are still some challenges that need to be addressed in future research. For instance, as a hierarchical scheduling architecture, if the root SN within an L_3 or L_2 level scheduling domain experiences faults or becomes disconnected, rapid fault detection and handling are critical issues. Typically, a viable solution involves monitoring faults and implementing corrective measures by periodically sending heartbeat signals between root and leaf SNs. Additionally, efficient aggregation of resource status information at the root SNs within L_3 and L_2 level scheduling domains is another pressing concern that requires attention. Furthermore, IPCRSA is based on an evolutionary ICN architecture that employs the SNR

method. Given the significant differences between the SNR method and the NBR method, IPCRSA cannot be directly applied to ICN architectures that utilize the NBR method. However, the concept of integrating computing resource scheduling capabilities into the network itself, as well as the idea of conducting computing resource scheduling based on ICN, are equally applicable to ICN architectures based on the NBR method. Therefore, how to modify the IPCRSA under an NBR-based ICN architecture to enhance its broad applicability is also one of the issues worth exploring in the future.

Author Contributions: Conceptualization, Z.N., J.Y. and Y.L.; methodology, Z.N., J.Y. and Y.L.; software, Z.N.; validation, Z.N.; formal analysis, Z.N.; investigation, Z.N.; writing—original draft preparation, Z.N.; writing—review and editing, J.Y. and Y.L.; visualization, Z.N.; supervision, J.Y. and Y.L.; project administration, J.Y. and Y.L.; funding acquisition, J.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China: Application Demonstration of Polymorphic Network Environment for computing from the eastern areas to the western. (Project No. 2023YFB2906404).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. These data can be found here: https://github.com/alibaba/clusterdata (accessed on 1 January 2018).

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

Abbreviations

The following abbreviations are used in this manuscript:

ICN	Information-Centric Networking
SDN	Software-Defined Networking
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
CAN	Computing-aware Networking
CPN	Computing Power Network
CFN	Computing First Network
ITU-T	International Telecommunication Union's Telecommunication Standardization Sector
IETF	Internet Engineering Task Force
RTGWG	IETF's Routing Area Working Group
IRTF	Internet Research Task Force
COINRG	IRTF's Computing in the Network Research Group
SRv6	Segment Routing IPv6
SIDaaS	Segment Identifier as a Service
RID	Rendezvous Identifier
SID	Scope Identifier
GUID	Globally Unique Identifier
ID	Identifier
PSS	Programmable Service System
AHP	Analytic Hierarchy Process
VNFs	Virtual Network Functions
MADM	Multi-Attribute Decision-Making
NBR	Name-Based Routing
SNR	Standalone Name Resolution
NA	Network Address
CRDTs	Conflict-free Replicated Data Types
RICE	Remote Method Invocation for ICN
SN	Scheduling Node
GN	Gateway Node
CRN	Computing Resource Node

Client	User Client
RN	Resolution Node
RSS_ID_L_x	Resource Scheduling Service Identifier in L_x level scheduling domains
PUE	Power Usage Effectiveness
CRS_ID_T_x	Computing Resource Service Identifier with type T_x
RRQ_Core	Resource Request Quantities in CPU Cores
RRQ_Memory	Resource Request Quantities in Memory
RRQ_Storage	Resource Request Quantities in Storage
RRQ_Bandwidth	Resource Request Quantities in Bandwidth
RRP	Resource Request Preference

References

1. Geng, L.; Willis, P. *Compute First Networking (CFN) Scenarios and Requirements*; Internet-Draft draft-geng-rtgwg-cfn-req-00; Internet Engineering Task Force: Fremont, CA, USA, 2019.
2. Lei, B.; Liu, Z.; Wang, X.; Yang, M.; Chen, Y. Computing network: A new multi-access edge computing. *Telecommun. Sci.* **2019**, *35*, 44–51.
3. Tang, X.; Cao, C.; Wang, Y.; Zhang, S.; Liu, Y.; Li, M.; He, T. Computing power network: The architecture of convergence of computing and networking towards 6G requirement. *China Commun.* **2021**, *18*, 175–185. [CrossRef]
4. Jia, Q.; Ding, R.; Liu, H.; Zhang, C.; Xie, R. Survey on research progress for compute first networking. *Chin. J. Netw. Inf. Secur.* **2021**, *7*, 1–12.
5. Lei, B.; Zhou, G. Exploration and practice of Computing Power Network (CPN) to realize convergence of computing and network. In Proceedings of the Optical Fiber Communication Conference, San Diego, CA, USA, 6–10 March 2022; Optica Publishing Group: Washington, DC, USA, 2022; p. M4A-2.
6. Lei, B.; Zhao, Q.; Mei, J. Computing power network: An interworking architecture of computing and network based on IP extension. In Proceedings of the 2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR), Paris, France, 7–10 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.
7. 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: Fremont, CA, USA, 2019.
8. Han, X.; Zhao, Y.; Yu, K.; Huang, X.; Xie, K.; Wei, H. Utility-optimized resource allocation in computing-aware networks. In Proceedings of the 2021 13th International Conference on Communication Software and Networks (ICCSN), Chongqing, China, 4–7 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 199–205.
9. Farhadi, V.; Mehmeti, F.; He, T.; La Porta, T.F.; Khamfroush, H.; Wang, S.; Chan, K.S.; Poularakis, K. Service placement and request scheduling for data-intensive applications in edge clouds. *IEEE/ACM Trans. Netw.* **2021**, *29*, 779–792. [CrossRef]
10. Hussain, H.; Malik, S.U.R.; Hameed, A.; Khan, S.U.; Bickler, G.; Min-Allah, N.; Qureshi, M.B.; Zhang, L.; Yongji, W.; Ghani, N.; et al. A survey on resource allocation in high performance distributed computing systems. *Parallel Comput.* **2013**, *39*, 709–736. [CrossRef]
11. Tian, L.; Yang, M.; Wang, S. An overview of compute first networking. *Int. J. Web Grid Serv.* **2021**, *17*, 81–97. [CrossRef]
12. Zhao, Y.; Chong, Z.; Han, X.; Du, Z.; Yu, K.; Huang, X. Simulation Study of Routing Mechanism in the Computing-aware Network. In Proceedings of the 2021 10th International Conference on Networks, Communication and Computing, Beijing, China, 10–12 December 2021; pp. 126–134.
13. Yao, H.; Duan, X.; Fu, Y. A computing-aware routing protocol for Computing Force Network. In Proceedings of the 2022 International Conference on Service Science (ICSS), Zhuhai, China, 13–15 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 137–141.
14. Król, M.; Mastorakis, S.; Oran, D.; Kutscher, D. Compute first networking: Distributed computing meets ICN. In Proceedings of the 6th ACM Conference on Information-Centric Networking, Macao, China, 24–26 September 2019; pp. 67–77.
15. Tao, H.E.; Yang, Z.; Cao, C.; Zhang, Y.; Tang, X. Analysis of some key technical problems in the development of computing power network. *Telecommun. Sci.* **2022**, *38*, 62–70.
16. Du, Z.; Li, Z.; Duan, X.; Wang, J. Service Information Informing in Computing Aware Networking. In Proceedings of the 2022 International Conference on Service Science (ICSS), Zhuhai, China, 13–15 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 125–130.
17. Vahlenkamp, M.; Schneider, F.; Kutscher, D.; Seedorf, J. Enabling information centric networking in IP networks using SDN. In Proceedings of the 2013 IEEE SDN for Future Networks and Services (SDN4FNS), Trento, Italy, 11–13 November 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 1–6.
18. Ravindran, R.; Liu, X.; Chakraborti, A.; Zhang, X.; Wang, G. Towards software defined icn based edge-cloud services. In Proceedings of the 2013 IEEE 2nd International Conference on Cloud Networking (CloudNet), San Francisco, CA, USA, 11–13 November 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 227–235.
19. Melazzi, N.B.; Detti, A.; Mazza, G.; Morabito, G.; Salsano, S.; Veltri, L. An openflow-based testbed for information centric networking. In Proceedings of the 2012 Future Network & Mobile Summit (FutureNetw), Estrel, Berlin, Germany, 4–6 July 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 1–9.
20. Wang, S.; Sun, J.; Wang, P.; Yang, A. Resource scheduling optimization in cloud-edge collaboration. *Telecommun. Sci.* **2023**, *39*, 163–170.

21. Zhang, Y.; Wang, Y. SDN based ICN architecture for the future integration network. In Proceedings of the 2016 16th International Symposium on Communications and Information Technologies (ISCIT), Qingdao, China, 26–28 September 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 474–478.
22. Wang, M.; Xu, C.; Jia, S.; Guan, J.; Grieco, L.A. Preference-aware fast interest forwarding for video streaming in information-centric VANETs. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–7.
23. China Mobile. *White Paper of Computing-Aware Networking*; Technical Report; China Mobile: Beijing, China, 2019.
24. China Unicom Research Institute. *White Book of Computing Power Network Architecture and Technology System*; Technical Report; China Unicom Research Institute: Beijing, China, 2020.
25. China Mobile Research Institute. *White Book of Computing-Aware Network Technology*, 2021 ed.; Technical Report; China Mobile Research Institute: Beijing, China, 2021.
26. SG13, International Telecommunication Union. *Y.CPN-Arch: "Framework and Architecture of Computing Power Network"*; Technical Report; International Telecommunication Union: Geneva, Switzerland, 2020.
27. Geng, L.; Lei, B.; Fu, Y.; Chen, L.; You, J. *Recommendation ITU-T Y.IMT2020-CAN-req: "Use Cases and Requirements of Computing-Aware Networking for Future Networks Including IMT-2020"*; Technical Report; International Telecommunication Union: Geneva, Switzerland, 2020.
28. SG11, International Telecommunication Union. *Q.CPN: "Signalling Requirements for Computing Power Network"*; Technical Report; International Telecommunication Union: Geneva, Switzerland, 2020.
29. SG11, International Telecommunication Union. *Q.BNC-INC: "Requirements and Signalling of Intelligence Control for the Border Network Gateway in Computing Power Network"*; Technical Report; International Telecommunication Union: Geneva, Switzerland, 2020.
30. Gu, S.; Zhuang, G.; Yao, H.; Li, X. *A Report on Compute First Networking (CFN) Field Trial*; Internet-Draft draft-gu-rtgwg-cfn-field-trial-01; Internet Engineering Task Force: Fremont, CA, USA, 2019.
31. Huang, G.; Shi, W.; Tan, B. Computing power network resources based on SRv6 and its service arrangement and scheduling. *ZTE Technol. J.* **2021**, *27*, 23–28.
32. Liu, J.; Sun, Y.; Su, J.; Li, Z.; Zhang, X.; Lei, B.; Wang, W. Computing power network: A testbed and applications with edge intelligence. In Proceedings of the IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), New York, NY, USA, 2–5 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–2.
33. Zhang, Y.; Cao, C.; Tang, X.; Pang, R.; Wang, S.; Wen, X. Programmable Service System Based on SIDaaS in Computing Power Network. In Proceedings of the 2022 5th International Conference on Hot Information-Centric Networking (HotICN), Guangzhou, China, 24–26 November 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 67–71.
34. Li, M.; Cao, C.; Tang, X.; He, T.; Li, J.; Liu, Q. Research on edge resource scheduling solutions for computing power network. *Front. Data Domputing* **2020**, *2*, 80–91.
35. Li, M.; Cao, C.; Yang, J. Computing Power Scheduling Mechanism Based on Programmable Network. *ZTE Technol. J.* **2021**, *27*, 18–22.
36. Dimolitsas, I.; Spatharakis, D.; Dechouniotis, D.; Papavassiliou, S. AHP4HPA: An AHP-based Autoscaling Framework for Kubernetes Clusters at the Network Edge. In Proceedings of the GLOBECOM 2022—2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 4–8 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 2566–2571.
37. Zeydan, E.; Mangues-Bafalluy, J.; Baranda, J.; Martínez, R.; Vettori, L. A multi-criteria decision making approach for scaling and placement of virtual network functions. *J. Netw. Syst. Manag.* **2022**, *30*, 32. [CrossRef]
38. Kaur, M.; Kadam, S.S. Discovery of resources using MADM approaches for parallel and distributed computing. *Eng. Sci. Technol. Int. J.* **2017**, *20*, 1013–1024. [CrossRef]
39. Koponen, T.; Chawla, M.; Chun, B.G.; Ermolinskiy, A.; Kim, K.H.; Shenker, S.; Stoica, I. A data-oriented (and beyond) network architecture. In Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan, 27–31 August 2007; pp. 181–192.
40. 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.
41. Dannewitz, C.; Kutscher, D.; Ohlman, B.; Farrell, S.; Ahlgren, B.; Karl, H. Network of information (netinf)—An information-centric networking architecture. *Comput. Commun.* **2013**, *36*, 721–735. [CrossRef]
42. 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]
43. Fotiou, N.; Nikander, P.; Trossen, D.; Polyzos, G.C. Developing information networking further: From PSIRP to PURSUIT. In Proceedings of the Broadband Communications, Networks, and Systems: 7th International ICST Conference, BROADNETS 2010, Athens, Greece, 25–27 October 2010; Revised Selected Papers 7; Springer: Berlin/Heidelberg, Germany, 2012; pp. 1–13.
44. Raychaudhuri, D.; Nagaraja, K.; Venkataramani, A. Mobilityfirst: A robust and trustworthy mobility-centric architecture for the future internet. *ACM SIGCOMM Mob. Comput. Commun. Rev.* **2012**, *16*, 2–13. [CrossRef]
45. Wang, J.; Chen, G.; You, J.; Sun, P. Seanet: Architecture and technologies of an on-site, elastic, autonomous network. *J. Netw. New Media* **2020**, *6*, 1–8.

46. Fayazbakhsh, S.K.; Lin, Y.; Tootoonchian, A.; Ghodsi, A.; Koponen, T.; Maggs, B.; Ng, K.; Sekar, V.; Shenker, S. Less pain, most of the gain: Incrementally deployable icn. *ACM SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 147–158. [CrossRef]
47. Lee, H.; Nakao, A. Efficient user-assisted content distribution over information-centric network. In Proceedings of the NETWORKING 2012: 11th International IFIP TC 6 Networking Conference, Prague, Czech Republic, 21–25 May 2012; Proceedings, Part I 11; Springer: Berlin/Heidelberg, Germany, 2012; pp. 1–12.
48. Ghodsi, A.; Koponen, T.; Rajahalme, J.; Sarolahti, P.; Shenker, S. Naming in content-oriented architectures. In Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking, Toronto, ON, Canada, 19 August 2011; pp. 1–6.
49. Dong, L.; Wang, G. A hybrid approach for name resolution and producer selection in information centric network. In Proceedings of the 2018 International Conference on Computing, Networking and Communications (ICNC), Maui, HI, USA, 5–8 March 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 574–580.
50. Xie, W.; You, J.; Wang, J. A Tree Structure Protocol for Hierarchical Deterministic Latency Name Resolution System. *Electronics* **2022**, *11*, 2425. [CrossRef]
51. Kaneko, T.; Shudo, K. Broadcast with Tree Selection from Multiple Spanning Trees on an Overlay Network. *IEICE Trans. Commun.* **2023**, *106*, 145–155. [CrossRef]
52. Gilbert, S.; Lynch, N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM Sigact News* **2002**, *33*, 51–59. [CrossRef]
53. Hagberg, A.; Swart, P.; S Chult, D. *Exploring Network Structure, Dynamics, and Function Using NetworkX*; Technical Report; Los Alamos National Lab (LANL): Los Alamos, NM, USA, 2008.
54. Wang, J.; Jing, L.; Chen, X.; You, J. Programmable data processing method and system design for polymorphic network. *J. Commun.* **2022**, *43*, 14.
55. Alibaba. Alibaba Open Trace. Available online: https://github.com/alibaba/clusterdata (accessed on 1 January 2018).
56. Liu, B.; Mao, J.; Xu, L.; Hu, R.; Chen, X. CFN-dyncast: Load Balancing the Edges via the Network. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Nanjing, China, 29 March 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.

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.