applied sciences

MDPI

Article

A Distributed Multicast QoS Routing Construction Approach in Information-Centric Networking

Jianping Song [1,2], Hong Ni [1,2] and Xiaoyong Zhu [1,2,*]

[1] National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, No. 21, North Fourth Ring Road, Haidian District, Beijing 100190, China; songjp@dsp.ac.cn (J.S.); nih@dsp.ac.cn (H.N.)
[2] School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, No. 19(A), Yuquan Road, Shijingshan District, Beijing 100049, China
* Correspondence: zhuxy@dsp.ac.cn

Abstract: Many applications suitable for multicast transmission, such as video conferencing and live e-commerce, demand high Quality of Service (QoS) and require data delivery to be completed within specified delay constraints. Some methods have been proposed for constructing delay-constrained multicast routing based on network state. However, obtaining precise network latency can be challenging, resulting in inaccuracies in delay-constrained routing calculations and, ultimately, the inability to meet application requirements. Additionally, many methods engage in an indiscriminate exploration of potential paths in the network, causing significant message processing overhead. This paper proposes an Information-Centric Networking (ICN)-based approach for delay-constrained multicast routing. Our method dynamically constructs multicast paths from tree nodes to receivers based on real-time path status detection during the join message propagation phase. Additionally, we present a method for acquiring neighborhood state information to facilitate real-time routing decisions. To curtail indiscriminate path exploration, our approach uses the ICN Name Resolution System (NRS) to obtain and select potential optimal tree nodes. For this purpose, we design a multicast service registration and resolution mechanism using the ICN Name Resolution System (NRS). Simulation results indicate that our approach exhibits a higher success ratio and concurrently incurs lower message processing overhead than some other methods, particularly in situations with stringent delay constraints.

Keywords: multicast; QoS routing; ICN

Citation: Song, J.; Ni, H.; Zhu, X. A Distributed Multicast QoS Routing Construction Approach in Information-Centric Networking. *Appl. Sci.* **2023**, *13*, 13349. https://doi.org/10.3390/app132413349

Academic Editors: Yutaka Ishibashi and Amalia Miliou

Received: 24 October 2023
Revised: 14 December 2023
Accepted: 15 December 2023
Published: 18 December 2023

1. Introduction

With the rapid advancement of terminal and multimedia technologies, real-time streaming applications, such as video conferencing, live e-commerce, online gaming, and distance education, have gained popularity among a growing audience [1–3]. These applications represent a contemporary trend in streaming media. Data dissemination to multiple destinations is a common requirement across various applications [4]. However, unicast transmission often results in redundant data packet transmissions, leading to a wastage of network resources. To address the issue of efficient group communication, multicast [5,6] has emerged as a viable solution, which builds a multicast tree between the source and the receivers to replicate and forward data. This approach allows for simultaneous data delivery to multiple destinations, reducing redundant bandwidth utilization and alleviating network device load. Furthermore, numerous high-value applications impose strict Quality of Service (QoS) [7] requirements, expecting audio and video data to be delivered within specific time constraints.

Therefore, latency constraints for the transmission of data from the source to group members must be considered. Timely delivery of data packets within specified time constraints is crucial [8,9]. QoS-driven multicast routing involves determining the optimal

distribution tree under certain performance-related constraints. This process aims to optimize network resources and fulfill the QoS requirements of applications. Therefore, the design of QoS-driven (or constraint-based) multicast routing methods holds paramount significance. To ensure QoS support for packet transmission, several fundamental techniques are required: the design of QoS-driven multicast routing protocols, packet classification techniques, packet scheduling algorithms, and resource reservation techniques. The specific operations of resource reservation or priority configuration are typically integrated into the process of configuring multicast routing information in routers. For instance, the traditional RSVP [10] protocol comprises two primary components: firstly, the establishment of a path, and secondly, the reservation of resources along the path during the path setup process. Concerns, such as resource reservation in routers, have been extensively studied in the unicast domain. Through straightforward adjustments, these techniques can be easily integrated with routing algorithms. Therefore, our research is dedicated to addressing multicast routing issues with latency constraints, with a focus on the construction of multicast trees.

To enable QoS routing, network nodes require accurate information about network resource states, such as latency and bandwidth. Owing to the dynamic fluctuations in network status, obtaining precise real-time network status information poses a challenge. In a large, distributed network, the network state information available for routing decisions at each node is often imprecise [11,12]. This can significantly deteriorate the efficiency and performance of routing algorithms due to outdated information, further worsening network conditions. Consequently, the inaccuracy of path state information should be considered when designing multicast routing algorithms.

Additionally, this issue can be formulated as a delay-constrained minimum cost tree problem, also known as a constrained Steiner tree (CST) problem [13,14], which is an NP-Complete problem [15]. Persistent issues remain despite introducing various heuristic algorithms [16] to find approximate solutions. Due to the challenges associated with acquiring real-time global network status and the inherent complexities in solving the CST problem, our approach is focused on seeking approximate optimal solutions rather than pursuing precise resolutions.

Moreover, multicast is a point-to-multipoint distribution pattern, whereas IP was initially designed for point-to-point communication networks. Addressing distribution issues through point-to-point communication protocols is complex and error-prone. This complexity arises from the dual nature of current IP addresses, serving as host and location identifiers. Moreover, in addition to distribution challenges, scalability, mobility, security, and other issues have hindered the rapid progress of the Internet [17].

The Information-Centric Network (ICN) [18,19] is an innovative network architecture following the 'Name and Address-Separation' paradigm. It is proposed to overcome the limitations of traditional IP networks and make content distribution within the network more efficient and reasonable. ICN has attracted extensive research attention for its potential to promote revolutionary network advances.

It achieves identity and location separation by naming content and using these names instead of IP addresses as identifiers in network transmissions [20]. Within ICN, two content forward mechanisms exist, name-based routing and name-based lookup [21].

In name-based routing, requests are forwarded in the network based on information names, and content routers deliver the data from the content holder to requesting users by following the reverse path of the request [17,22]. However, this approach necessitates content routers to cache substantial routing states [23]. In the name-based lookup approach [24,25], a Name Resolution System (NRS) [26] maps names to Network Addresses (NAs) of the underlying network, such as IP addresses, and data are routed based on these NAs.

1.1. Background Work and Limitations

Several studies propose to distribute state information by utilizing extended versions of existing routing protocols, such as Open Shortest Path First (OSPF) [27,28]. However,

these studies overlook the challenges and costs of acquiring accurate state information [29]. The precision of state information depends on the frequency of updates, and frequent updates by numerous nodes as the network scales can result in substantial overhead [12].

The distributed multicast routing algorithm can be categorized into two distinct types: source-driven and receiver-driven path discovery. Numerous algorithms conduct indiscriminate exploration operations when selecting joining nodes for multicast services. Source-driven approaches may explore all nodes along the multicast tree to calculate appropriate joining nodes. Similarly, receiver-driven methodologies may explore a plethora of paths from the receivers to multicast tree nodes. Such practices result in intricate message processing overhead, consequently elevating the complexity of multicast tree construction.

Many algorithms assume that the set of group members is deterministic at the beginning of the algorithm. However, this aligns differently from many real-world scenarios, where group members often generate and join multicast groups randomly, making it impossible to pre-determine the set of group member nodes. In addition, some algorithms may not handle dynamic changes in group membership effectively. Although some algorithms support dynamic member joining and leaving, they may incur significant message processing overhead when handling requests.

1.2. Contributions

In this paper, to address the challenges mentioned above in multicast QoS routing, we propose a distributed delay-constrained multicast routing construction approach based on ICN. Our routing scheme is applicable to networks with a static topology. The primary contributions of this paper are as follows:

To construct delay-constrained multicast trees, we propose a QoS multicast routing architecture based on ICN, consisting of three essential components: delay-constrained tree construction algorithm, neighborhood state awareness mechanism, and tree node selection mechanism. Leveraging the ICN system based on name-based lookup, our research employs globally unique names to identify multicast services, and the NRS maintains the mapping between multicast service names and Network Addresses (NA) of multicast tree nodes. The receiving node interacts with NRS to obtain tree nodes and select the optimal node to send joining requests. Join messages collect path state information during propagation to the selected tree node. Routers construct paths from the multicast tree to receivers hop-by-hop based on routing tables, neighborhood state information, and path state information. To circumvent acquiring accurate information regarding the status of the complete network, we design a path construction algorithm that makes decisions based on precise path state information and imprecise routing information. We have designed a tree node registration and resolution mechanism based on NRS. This allows receivers to access a viable set of nodes within the multicast tree through the NRS to select the optimal tree node for joining requests. Additionally, we have introduced a cost-effective neighborhood state awareness mechanism for network nodes to acquire relatively precise neighborhood state information. We have developed and implemented the proposed multicast routing method in NS-3 [30] and compared it with existing methods. Experimental results show that our approach has a significantly higher success ratio than other algorithms and has stable and low message-processing overhead.

The remaining structure of this research is as follows. Section 2 provides a summary of related work. We present the details of the proposed approach in Section 3. In Section 4, we conduct comparative experiments with some other methods to assess the effectiveness of the proposed method. Section 5 concludes and discusses future directions for this work.

2. Related Work

The IP Multicast technology was initially introduced by Deering in 1988 [31] and has since garnered significant recognition in both academic and industrial domains [32]. Many research institutes and organizations worldwide have conducted extensive studies on multicast routing, considering it a crucial and indispensable technology.

(1) Multicast QoS routing

Multicast QoS routing algorithms can be categorized into centralized and distributed approaches. In centralized schemes, network nodes have comprehensive information of the entire network topology and state [9], and a central control node computes the full routing tree. Unfortunately, there are several limitations to this strategy, such as scalability concerns and potential single points of failure.

Protocol Independent Multicast-Sparse Mode (PIM-SM) [33] is a multicast routing architecture capable of establishing distribution trees efficiently across the wide-area Internet. It constructs multicast trees based on unicast routing from multicast members to the root of the multicast tree. The Kou, Markowsky, and Berman's algorithm (KMB) [34] uses the Prim algorithm [35] to generate a minimal spanning tree in a centralized manner. This tree is then pruned to produce a multicast tree by removing unnecessary branches. The Bounded Shortest Multicast Algorithm (BSMA) [36] starts by determining the least delay tree using the well-known Dijkstra method. The overall cost of the tree is then decreased by replacing expensive edges with cheaper pathways, subject to satisfying latency constraints. Nevertheless, the computation process necessitates many iterations, each involving calculating k shortest paths. Due to the high time complexity, this method is unsuitable for resolving large-scale networks.

Ref. [37] is a source-driven tree search algorithm proposed by Jia (Jia's Algo). The source node keeps track of routing data from tree nodes to receivers in a state information table. Whenever a member sends a join request, information about the multicast status is transferred from the source to leaf nodes and updated along the way. The source node collects responses from leaf nodes and selects the best leaf node to construct a path to the receiver. Nevertheless, this method relies on the unrealistic assumption that the shortest delay path between any two nodes is invariably the most cost-efficient, which is inconsistent with actual circumstances. Furthermore, it has been demonstrated that Jia's Algo can result in loops under some conditions [38].

Kompella et al. proposed a distributed algorithm [39] rooted in Minimum Spanning Tree (MST) with a similar concept to Jia's Algo. Initially, FIND messages are propagated downwards from the root throughout the tree. Each tree node receiving a FIND message computes the optimal path independently and responds to the source with a message. Ultimately, the source selects an optimal path from all received responses to construct branches of the multicast tree. However, Kompella et al.'s algorithm shares similar issues with Jia's Algo, exhibiting significant time and message processing complexity.

In Spanning-Joins [40], the receiver broadcasts a join request message within the neighborhood to locate tree nodes. Tree nodes receiving the request send response messages to the receiver, collecting QoS information along the path. Receivers select a path based on the information in the response messages. When a satisfactory path is not found, the search scope is expanded, significantly increasing the search cost.

In [41], the authors propose the QoS-sensitive Multicast Internet Protocol (QoSMIC). Initially, receivers conduct a local search; in the event of failure, they dispatch messages to designated management nodes to initiate a tree search. During the tree search phase, request messages are multicast throughout the tree, and a path selection mechanism similar to Spanning-Joins is executed.

In the QoS-aware Multicast Routing Protocol algorithm (QMRP) [42], receivers deliver requests based on the underlay route to the source node, searching for a path that meets the QoS requirement hop by hop. If the current search path is deemed insufficient to meet the QoS requirement, the algorithm returns to the predecessor node and initiates a search along several alternative paths. This results in significant message processing overhead and ignores the problem of link cost.

Table 1 briefly summarizes the methods in related work.

(2) Multicast in Information-Centric Networking (ICN)

Many ICN architectures offer their multicast solutions. Some ICN solutions, such as Content-Centric Networking/Named Data Networking (CCN/NDN) [17,22] and Data-Oriented Network Architecture (DONA) [43], inherently support multicast without explicit computation or constructing multicast trees. They achieve this by consolidating requests for the same content into an interest table and forwarding data based on this interest table. The router aggregates the same interests originating from different downstream routers or receivers and stores them in the Pending Interest Table (PIT). Subsequently, once the pertinent content is acquired, it is transmitted to the designated machine recorded in PIT.

Publish-Subscribe Internet Technologies (PURSUIT) [44] achieves multicast by computing the entire multicast tree and encoding it as a single Bloom filter. In the MobilityFirst [24] architecture, every multicast group is assigned a globally unique identifier (GUID), and the Global Name Resolution Service (GNRS) maintains the mapping between multicast GUIDs (MID) and the membership sets of multicast groups. Two multicast protocols, Named-Object Multicast (NOMA) [45] and Multicast Design for the MobilityFirst (MMF) [46], have been designed for use within MobilityFirst. NOMA sustains multicast trees through the recursive mapping of multicast group GUIDs to branch node GUIDs within the multicast tree. However, NOMA uses a centralized approach to compute multicast trees, which can be inefficient and require updates every time group membership changes. In MMF, multicast sources manage join and leave requests from each receiver. This design can lead to scalability challenges in large networks.

Table 1. Comparison of related work.

Algorithm	Type	Initiator	Comments
KMB	Steiner tree	Source	Centralized algorithm. No delay constraints are provided.
Jia's Algo	Steiner tree	Source	Perform a tree search starting from the source. Message processing cost is high.
QoSMIC	Minimum spanning tree	Receiver	Perform local search firstly, if fails, perform tree search. It avoids blind search at the beginning, but still has a large message processing cost.
QMRP	Minimum spanning tree	Receiver	Perform single path search firstly, if fails, perform multi-path search. Message processing cost is high.
PIM-SM	Minimum spanning tree	Receiver	Shortest path tree algorithm.
Spanning-Join	Minimum spanning tree	Receiver	Broadcast join request to tree nodes to explore the path.

Our proposed methodology utilizes the ICN name resolution system to acquire a viable set of tree nodes, subsequently engaging in a single-path exploration process, thereby circumventing blind searches. When the detection message arrives at a tree node, the decision to either construct a multicast route or execute a constrained tree search process is determined based on the accumulated network status. This approach helps circumvent message processing overhead caused by blind exploration. Additionally, we facilitate routing decisions by upholding an accurate repository of neighborhood state information, thereby mitigating the potential risk of path construction failure resulting from inaccurate state information.

3. System Design

In addressing the challenges of delay-constrained multicast routing, this paper proposes a neighborhood state-aware delay-constrained multicast routing method in Information-Centric Networking (ICN). The method presented in this paper makes routing decisions and establishes multicast trees based on foundational routing information and neighborhood state information. We proposed a registration and resolution mechanism for selecting tree nodes. A mechanism for the announcement and maintenance of neighborhood state information is introduced, where dynamically changing link information is only transmitted

in real-time among neighboring nodes, thereby avoiding flooding of real-time updated path states throughout the network.

First, we provide an overview of the system's preparations. The design of our delay-constrained multicast routing system can be categorized into four key components: a system overview, management of local network status, a registration resolution mechanism, and an algorithm for constructing the multicast tree.

3.1. Preliminaries

The network topology is represented as a connected graph, *G(V, E)*, where the nodes in V correspond to network elements such as routers, and edges ℓ in E are defined as communication links between two network elements. Each link ℓ in E has two non-negative attributes: latency and cost, denoted as $D(\ell)$ and $C(\ell)$, respectively. $D(\ell)$ represents the delay experienced by communication between the two nodes connected by the edge, measured in milliseconds, while $C(\ell)$ represents the cost required for communication between the two nodes. In this paper, the formula for calculating communication link costs is as follows:

$$C(\ell) = \frac{K}{\text{Bandwidth}(\ell)} \tag{1}$$

In this context, K denotes the fixed reference bandwidth, set at 3000, while Bandwidth(ℓ) signifies the maximum interface bandwidth of the link ℓ, measured in Mbps.

Within a network, nodes maintain two distinct types of routing tables: the Least Delay Table (LDT) and the Least Cost Table (LCT). To determine the shortest paths, Dijkstra's algorithm is utilized, with delay serving as the metric for LDT and cost acting as the metric for LCT. Owing to the volatile nature of network conditions, resulting in frequent fluctuations in link delay information, the delay metric is based on average link delay statistics collected within a predetermined time window. To minimize the necessity for frequent LDT recalculations, this table undergoes refreshing at longer intervals, specifically set at once every 30 min in this study. This ensures that computational costs remain manageable, as observed in other routing protocols such as OSPF. Although the resulting least-delay routing information may not be entirely precise due to dynamic changes in link delays, it can function as a reference for routing decisions.

Each entry in the routing table includes the following information: the network address (NA) of the destination, path delay, path cost, Gateway (GW), two-hop gateway (GW2), GW Delay, and GW2 Delay. As shown in Figure 1, GW is the next-hop address from Source (S) to Destination, GW2 is the second-hop address from S to Destination (if it exists), and GW Delay and GW2 Delay represent the delay when S reaches nodes GW and GW2, respectively, in the route calculation process.

The network cost of multicast tree T is defined as follows:

$$TreeCost(T) = \sum_{\ell \in T} C(\ell) \tag{2}$$

Define *P(T,i)* as the set of links from the root of tree T to receiver i along the internal paths of the tree. The multicast transmission delay of receiver i is defined as follows:

$$TransmissionDelay(T, i) = \sum_{\ell \in P(T,i)} D(\ell) \tag{3}$$

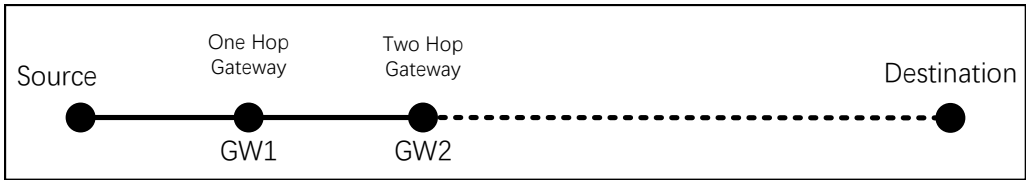


Figure 1. The information recorded in the routing table.

3.2. System Overview

The system is built on the ICN scheme, which relies on name-based lookup for content discovery, similar to MobilityFirst. In this framework, devices, data, and services are considered as Named Data Objects with globally unique Entity Identifiers (EID). Network Addresses (NA) serve as locators, similar to IP addresses in IP networks. The Name Resolution System (NRS) maintains dynamic mappings between identifiers and locators. Multicast services are treated as entities and assigned a globally unique Multicast Service Identifier (Multicast ID or MID).

Figure 2 illustrates a multicast system comprising routers, multicast sources, receivers, and a Name Resolution System (NRS). Initially, the receiver node consults the NRS to obtain the list of NAs associated with the MID. Select the appropriate NA and send a JOIN message to it. During the propagation of JOIN messages to tree nodes (TN), network status is collected. Upon receiving a JOIN message, the tree node constructs multicast routes to receiver nodes based on locally maintained information and the details recorded in the JOIN message. During multicast tree construction, routers create Multicast Forwarding Table (MFT) entries for the MID. The MFT maintained by the multicast TN records the MID, its corresponding predecessor node, and a list of forwarding nodes. Upon receiving multicast data packets, TN retrieves MFT entries using the recorded MID in the packet header, obtaining the forwarding nodes' list and forwarding the packets accordingly.

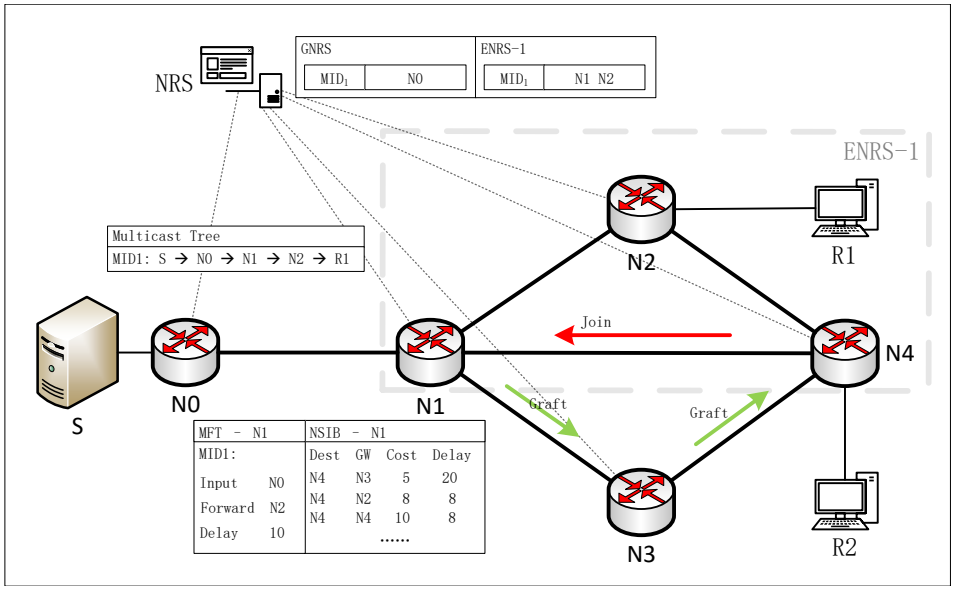


Figure 2. System overview of the distributed delay-constrained multicast routing construction approach.

3.3. Neighborhood State Management

To obtain neighborhood state information, two essential steps need to be taken. Firstly, it is necessary to calculate the forwarding delay for adjacent ports. Moreover, the neighbor state information is propagated to nodes directly connected to the network node. Each node is responsible for maintaining a neighborhood state information base (NSIB) that stores the status of nodes within a designated range.

To commence the process, each node will periodically transmit announcement messages to its adjacent nodes. These messages are designed explicitly for one-hop transmission and contain delay information, thereby avoiding excessive propagation throughout the network. This approach can seamlessly integrate with existing distributed routing protocols such as Open Shortest Path First (OSPF). The routing protocol transmits hello packets with timestamps at intervals (5 s). Upon receiving a hello packet, the node computes the transmission delay by comparing the timestamp enclosed in the packet with the local time.

Delay calculating requires ensuring time synchronization between the two ports. Given the proximity of these two ports, time synchronization can be accomplished through the utilization of the Network Time Protocol (NTP). This method facilitates the seamless exchange of port delay between nodes in close proximity without significant overhead. Upon receiving the hello packets containing the announcement information from all local ports, the node has obtained the transmission delays of all neighbor nodes directly connected to the node. Subsequently, the node builds a neighbor status information table.

Next, nodes communicate the forwarding delay of their adjacent ports to neighboring nodes via hello packets. Suppose node N has neighbors N_1, N_2, …, N_i, where N_k is the node connected to port k. Node N sends a hello packet to Nk containing all the delay information of N_1, N_2, …, N_i, excluding N_k's. This enables N_k to obtain the delay information between N and all its neighboring nodes through node N. With this data, N_k can precisely calculate the delays for reaching N_1, N_2, …, N_i via node N, typically within a 10-second margin (depending on the specific configuration). Ultimately, N_k integrates this information into its neighborhood state information base (NSIB).

Throughout this process, each node within the network generates and maintains a repository of delay information for nodes located within a maximum range of two hops. This repository is consistently updated to ensure that accurate and up-to-date information is available.

To illustrate the process of establishing the Neighbor State Information Base (NSIB), Figure 3 is referenced. In Figure 3a, the network topology is depicted, where the neighbors of node N_3 periodically advertise their state information. Consequently, N_3 establishes a Neighbor State Information Table (NSIT), as depicted in Figure 3b. Over time, all nodes in the network build their respective NSITs. Following this, nodes begin disseminating their NSITs to neighbors. Utilizing the advertisement information from N_0, N_2, and N_3, Node N_1 constructs an NSIB that includes delay information for all nodes within two hops, as shown in Figure 3c. Utilizing this one-hop message propagation mechanism, nodes in the network acquire real-time status information within their two-hop neighborhoods, preventing the propagation of real-time updated statuses across the entire network. Subsequently, *NSIB*[N_i]. *Cost* and *NSIB*[N_i]. *Delay* are used to represent the cost and delay of reaching node N_i in the NSIB table, respectively. When integrated into real-world environments, switches may employ distinct Quality of Service (QoS) forwarding levels for various packet types. Different categories of data packets transmitted over the same link incur varying transmission delays. By redefining the method of delay computation, this issue can be effectively addressed. Specifically, the announcement messages record queuing and processing delays of packets at different QoS levels on the port. The network node computes the transmission delay based on the fixed propagation delay of the physical link and the queuing and processing delays on the ports recorded in the message.

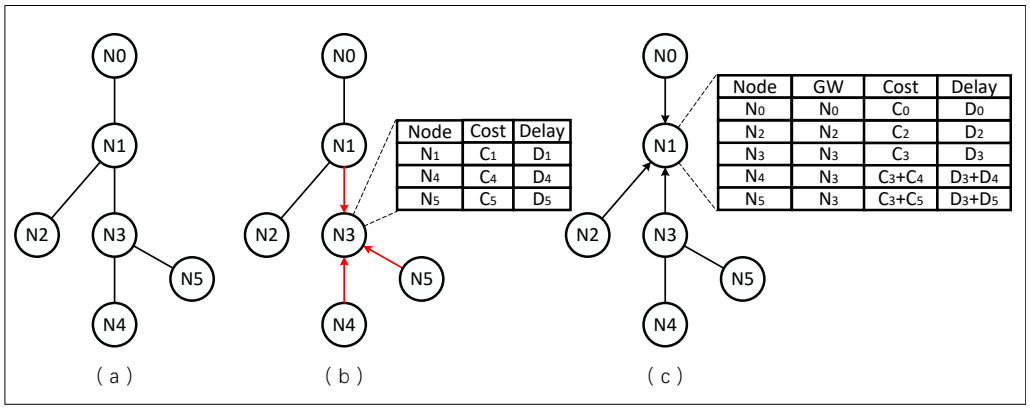


Figure 3. The Two-Step Construction Process of NSIB. (**a**) Example Topology; (**b**) Neighbor State Information Advertisement; (**c**) NSIT Advertisement.

3.4. Resolution System

At the initial stage of multicast routing, it is crucial to identify a suitable tree node for subsequent path construction from the multicast tree to the receivers. Both source-initiated and receiver-initiated routing methods entail the selection or search process for tree nodes. Aimless search processes may result in significant message overhead. The ICN NRS manages the mapping between multicast service IDs and the network addresses of tree nodes. Direct interaction with the resolution system facilitates convenient access to information about multicast nodes. By designing a registration and resolution mechanism to simplify the tree node search and selection process, it is possible to reduce the signaling cost associated with searching for suitable tree nodes.

We focus on the ICN network employing a name-based lookup mechanism. A flat naming approach is utilized to identify data entities, with the Multicast Service Identifier (MID) generated based on the content hash. Therefore, a resolution service is needed to map the MID to the NAs. Our proposed resolution system consists of a Global Name Resolution System (GNRS) and multiple Enhanced Name Resolution Systems (ENRS).

The GNRS allows all nodes to register information and respond to resolution query requests. Multiple ENRS are divided based on node latency. Within each ENRS, independent instances are deployed, allowing only nodes within the domain that satisfy latency constraints to register and respond to queries within those constraints. The structure of the resolution system is illustrated in Figure 4.

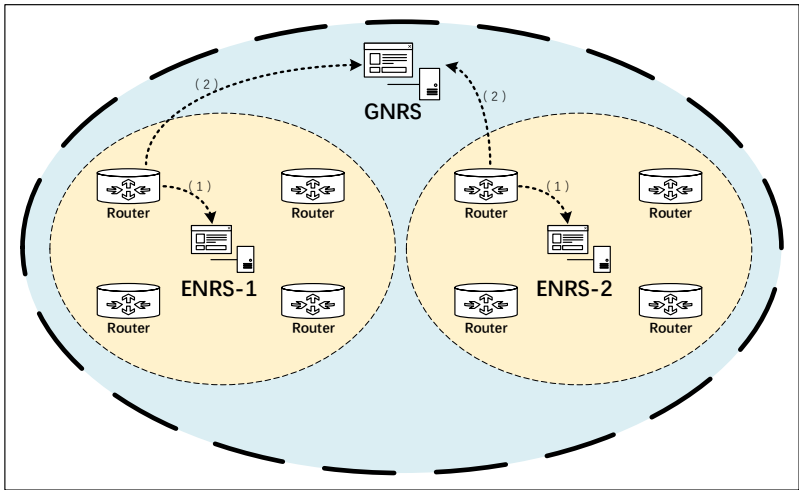


Figure 4. The process of interacting with the NRS. The node will initially interact with the ENRS within the yellow region, and subsequently, based on the outcome, engage with the GNRS within the blue region.

The root node (RN) of the multicast tree registers its information to the GNRS, while tree nodes (TN) within an enhanced resolution domain register their information to their respective Enhanced Name Resolution System (ENRS). As depicted in Figure 5, the left side illustrates a multicast tree with MID 3, comprising five tree nodes, including S and N_1-N_4, where S serves as the root node of the multicast tree. The table on the right offers a concise description of the entries within the resolution system. Root node S is registered in the GNRS, whereas the information of tree nodes N_3 and N_4 is recorded in Enhanced Name Resolution System 1 (ENRS-1). Each MID is associated with an NA List, which stores the network addresses NA_i of node N_i.

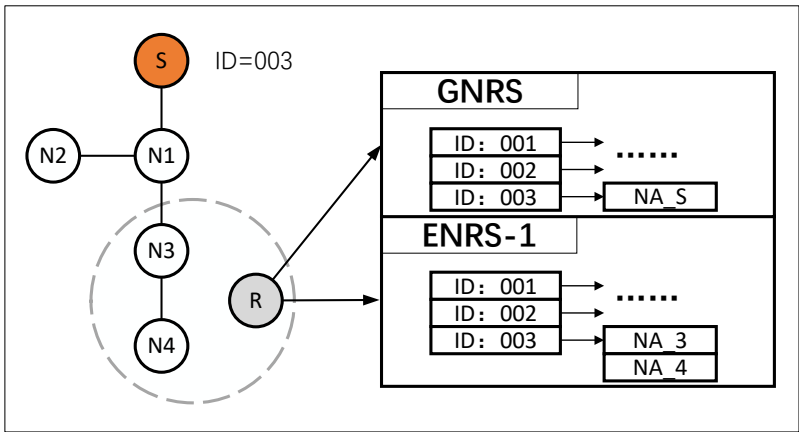


Figure 5. Multicast tree information storage format in the NRS.

3.5. Tree Construction Algorithm

The construction process of the delay-constrained multicast tree is divided into four phases: data source registration, tree node selection during multicast member joining, join message propagation, and grafting process. The source node initiates the multicast service by creating and registering a Multicast Identifier (MID). Receivers interested in specific multicast services acquire tree nodes by interacting with the resolution system and initiating join requests. The path delay is estimated during the join message propagation process, and the tree nodes are constructed hop by hop based on link states during the grafting process.

3.5.1. Registration of Data Sources

When a node is designated as the source node for a multicast group, it generates a MID and registers the association between the MID and its NA with both the GNRS and its corresponding ENRS (if exists). Each tree node TN_i maintains the accumulated delay, D_{accum}, for packets transmitted from the multicast source node to itself within the tree. The root node (RN), which also functions as a multicast tree node (TN), maintains an accumulated delay of 0.

3.5.2. Node Selection

Before a receiver initiates a specific multicast group join request, it determines its QoS Delay Requirement (D_{QoS}), specifying that the multicast transmission delay should be lower than this specified value. The multicast transmission delay is defined as the accumulated delay (D_{est}) of data transmitted from the multicast tree root to the receiver hop by hop, as shown in Figure 6a. For receivers (R) requesting to join a multicast group, the multicast delay can be divided into two parts: the delay from RN to TN (D_{accum}) and the delay from TN to R (D_{est}), as illustrated in Figure 6b.

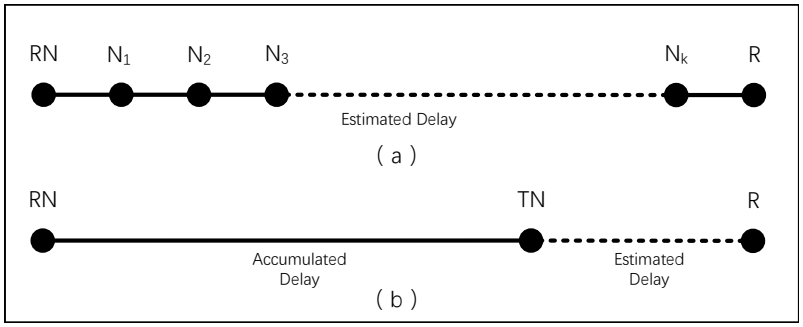


Figure 6. Multicast Transmission Delay. (**a**) Hop-by-hop Transmission Delay; (**b**) Estimated Delay.

The receiver must carefully select a specific TN (TN_{spec}) within the established multicast tree. The process begins with the receiver querying its affiliated Enhanced Name Resolution System (ENRS) for a comprehensive list of nodes associated with the multicast group MID. If the ENRS returns a non-empty list, the results returned by the ENRS are added to the candidate node set {TN}, and the TN_{spec} selection process begins.

If no entries corresponding to MID are found in the ENRS, the GNRS is queried to obtain the root node (RN) of the multicast tree.

The TN_{spec} selection process is as follows:

Step 1 If {TN} is empty, set TN_{spec} to RN, and skip to Step 4.

Step 2 For each TN_i node in the node set {TN}, retrieve the corresponding entry NSIB[TN_i] from the NSIB of the receiver node. If relevant information can be retrieved, select the node with the least cost, NSIB[TN_i].cost, as TN_{spec}, and skip to Step 4.

Step 3 For each TN_i node in the node set {TN}, query the LCT, and select the node with the minimum cost LCT[TN_i].cost as TN_{spec}.

Step 4 Based on the information recorded in the routing entry LCT[TN_{spec}], send a JOIN message, as shown in Figure 7, to LCT[TN_{spec}].Gateway. Set the Request Node to the receiver's network address (NA), set the Branch Node to TN_{spec}, and set the QoS Delay Requirement field to D_{QoS}.

Figure 7 depicts the message format utilized in this paper. The ID Header records the MID for the join request in Figure 7. The Message Type field can have three types: JOIN, GRAFT, and PRUNE. The Branch Node represents the selected tree node and serves as the destination address for JOIN messages. Request Node represents the receiving node initiating the JOIN message, while Previous Delay represents the accumulated delay from the root node to the intermediate node. The link status between the receiver and TN_{spec} will be probed during the JOIN message propagation. The Link State Table (LS Table) records the measured link state information. The process of building the LS Table is described in the following section.

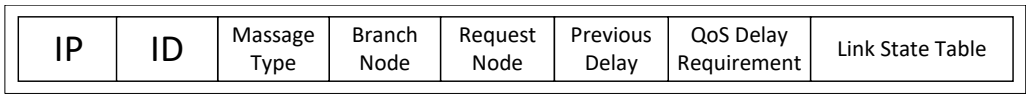


Figure 7. Header format of multicast signaling messages.

3.5.3. Joining Process and Path Delay Estimation

The JOIN message includes an LS Table to record link information, storing the network addresses (NAs) of nodes N_i probed by the JOIN operation, along with the corresponding accumulated path cost (N_i.*Cost*) and delay (N_i.*Delay*) from each N_i to the Receiver Node.

Upon receiving a JOIN message, nodes in the network calculate the accumulated delay and cost from their node to the Receiver Node using the LS Table, subsequently appending this information to the table. The specific construction process of the LS Table is depicted in Figure 7. Upon receiving the JOIN message forwarded by node N_1, the non-tree node N_2 parses the LS Table to retrieve the table length (L) and information about the tail node (N_1). Subsequently, N_2 looks up N_1 in the NSIB and calculates the measured values D_{meas} and C_{meas} for delay and cost using Equations (4) and (5):

$$D_{meas} = \begin{cases} NSIB[N_1].Delay, & L = 0 \\ NSIB[N_1].Delay + N_1.Delay, & L > 0 \end{cases} \quad (4)$$

$$C_{meas} = \begin{cases} NSIB[N_1].Cost, & L = 0 \\ NSIB[N_1].Cost + N_1.Cost, & L > 0 \end{cases} \quad (5)$$

If N_2 is a non-tree node, append its own NA, D_{meas}, and C_{meas} to the LS Table. Query the LCT based on the recorded NA_{Branch} in the Branch Node field. Then, forward the configured JOIN message to LCT[NA_{Branch}]. Gateway according to the routing entry in the LCT.

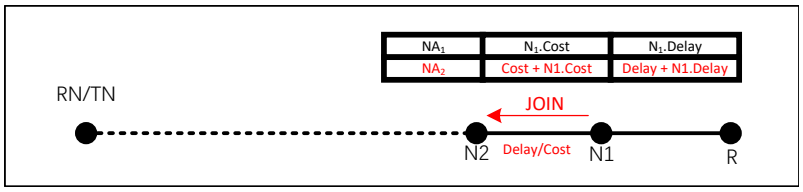


Figure 8. Construction of LS Table During JOIN Message Propagation.

If TN_{spec} is RN, the complete link from RN to R must be detected during the JOIN message propagation process, as illustrated in Figure 8. Otherwise, only the link between TN and R needs to be probed.

If N_2 is a tree node, calculate the minimum possible delay using Equation (6), where *LDT*[i].*Delay* represents the delay recorded in *LDT* for reaching node i:

$$Delay_{min} = D_{meas} + LDT[N_2].Delay \quad (6)$$

If $Delay_{min}$ exceeds the delay required by QoS, the path constructed from this node does not meet the delay constraint. Thus, the JOIN message must be forwarded to the predecessor node recorded in the local MFT[MID]. On the other hand, if the delay is within the constraint, the grafting process can proceed accordingly.

3.5.4. Grafting Process

Following the method described in Section 3.5.2, the tree node extracts information recorded in the tail entry of the LS Table within the JOIN message and calculates D_{meas} and C_{meas}. Additionally, TN maintains the accumulated delay, D_{accum}[TN].

As illustrated in Figure 9, the multicast delay for R under the current join path is calculated as Formula (7):

$$Delay[TN] = D_{meas} + D_{accum}[TN] \quad (7)$$

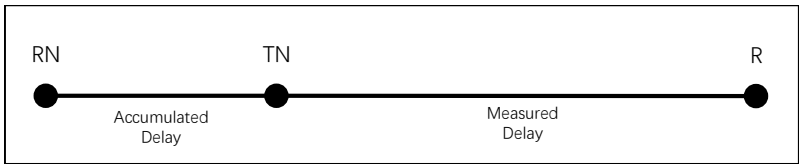


Figure 9. The delay measured upon arrival of the join message at the tree node.

If *Delay*[TN] meets the delay requirement of QoS, update the information maintained by the node and the fields in the message accordingly. Append the NA recorded in the tail entry of the LS Table to the forwarding node list of MFT[MID]. The Message Type and Request Node field in the message is modified to GRAFT and TN's NA, respectively. Forward the message to the NA recorded in the tail entry of the LS Table, and then remove the tail entry. Subsequently, the node forwards multicast data packets by utilizing entries in the multicast replication forwarding table.

If TN's delay does not meet the QoS requirement, the relay node selection Algorithm 1 is employed to choose a relay node from the LS table. This selection is based on the already detected link information and the neighborhood state information maintained by the local node. Subsequently, a new path is established from TN to the relay node.

At this time, add the Gateway returned by the algorithm to the forwarding node list of MFT[MID].

The multicast delay from RN to the receiver node (R), depicted in Figure 10, is determined by the accumulated delay from RN to TN, the delay from TN to Relay, and the delay from Relay to R. The multicast delay can be represented as Formula (8):

$$Delay = D_{accum}[TN] + D_{est}[Relay] + D_{meas}[Relay] \quad (8)$$

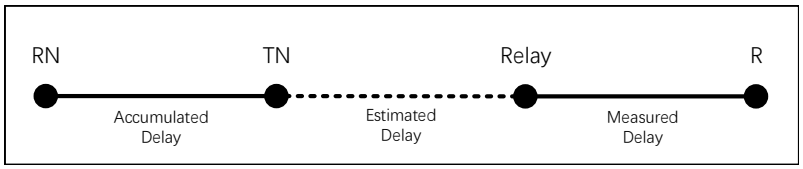


Figure 10. Delay estimation during the grafting process.

Advancing to the Relay Node Selection phase, the delay of the minimum-cost path from the current tree node to the receiver has been precisely calculated. If this path fails to meet the delay constraints, adaptation occurs by selecting a relay node along the path. After this selection, adjustments are made to the forwarding path from TN to the relay node to align with the minimum delay route. The relay node selection process involves access to precise delay information, covering the accumulated delay at the TN node, the delay from the relay node to the receiver, and neighborhood delay information in NSIB. The only imprecise information lies in the delay from TN (Tree Node) to the relay node. Evaluate the recorded delays of various nodes in the multicast message, treating them as potential relays, and select the node most likely to satisfy delay constraints while minimizing costs.

The relay node selection Algorithm 1 utilizes symbols listed in Table 2, and the detailed execution steps of the algorithm are provided below:

Table 2. Summary of the notations.

Symbol	Meaning
LS_i	The i-th entry in LS Table.
$LDT[NA]$	Least delay routing entry with NA as destination.
$NSIB[NA]$	Neighborhood state information base entry with NA as destination.
$ProbeDelay$	Latency estimated based on inaccurate information.
$CalibratedDelay$	Delay calibrated by neighborhood information.

Algorithm 1 Relay Node Selection Algorithm 1.

Input: $LS\ Table,\ D_{\text{QoS}},\ LDT,\ NSIB$
Output: $Gateway,\ Cost_{\text{Candidate}},\ Delay_{\text{Candidate}}$
$NA_{\text{Candidate}} = nil,\ Cost_{\text{Candidate}} = \infty,\ Delay_{\text{Candidate}} = \infty$
for each LS_i **in** $LS\ Table$ **do**
    $ProbeDelay = LS_i.Delay + D_{accum} + LDT[LS_i.NA].Delay$
    $CalibratedDelay = ProbeDelay - LDT[LS_i.NA].towHopDelay + NSIB[NA].Delay$
    if $CalibratedDelay <= D_{\text{QoS}}$ **or** $CalibratedDelay < Delay_{\text{Candidate}}$ **then**
        $Gateway = LDT[LS_i.NA].Gateway$
        $NA_{\text{Candidate}} = LS_i.NA$
        $Cost_{\text{Candidate}} = LS_i.Cost + LDT[LS_i.NA].Cost$
        $Delay_{\text{Candidate}} = LS_i.Delay + LDT[LS_i.NA].Delay$
    end if
end for
$Erase\ entry\ in\ LS\ Table\ from\ NA_{\text{Candidate}}\ to\ tail$
return $Gateway,\ Cost_{\text{Candidate}},\ Delay_{\text{Candidate}}$

When a non-tree node receives a GRAFT message, it calculates the accumulated delay using Formula (9) based on the neighbor delay recorded in NSIT (Neighborhood State Information Table) and the Previous Delay recorded in the GRAFT message:

$$D_{accum} = PreviousDelay + NSIT[RequestNode] \cdot delay \tag{9}$$

This D_{accum} is then added to the local accumulate delay table. The Relay node selection process mentioned earlier is executed to obtain the NA of Gateway. Add Gateway to the forwarding node list of MFT[MID], and set the NA recorded in the Request Node field

as the predecessor node. Finally, modify the Message Type field to GRAFT, change the Request Node field to the NA of the non-tree node, update the Previous Delay to D_{accum}, and forward the message to the Gateway. Lastly, register the mapping of MID to its own NA in the corresponding ENRS.

As the Graft message is transmitted to the receiver node hop by hop, the construction of the path from the multicast tree to the receiver is completed, concluding the grafting process.

3.6. Algorithm Analysis

This paper assesses the scalability of the distributed routing algorithm from two key perspectives: message quantity and message processing frequency. Our evaluation focuses on examining the algorithm's performance in terms of the number of messages exchanged among nodes and the frequency of message processing events. These metrics serve as crucial indicators to gauge the algorithm's efficiency and load-handling capabilities in a distributed environment. Concerning message quantity, our algorithm requires two messages for each receiver: one JOIN message for probing status information and one GRAFT message for constructing the path. Consequently, each join request from a receiver necessitates the processing of two messages. If the number of receivers is defined as 'm', the message complexity is O(2 m). Regarding message processing frequency, given that each routing node is required to handle messages, the processing frequency for each message depends on the number of hops the message propagates in the network. Assuming the average number of hops a data packet experiences in the network is represented by 'H'. Under normal circumstances, one request corresponds to 2H message processing events. The message processing complexity is O(2 mH). In a worst-case scenario, where messages require feedback from tree nodes to the root node, additional H message processing events are required. Thus, in the worst-case scenario, the message processing complexity is O(3 mH).

4. Simulation

In the process of evaluating the performance of our proposed routing approach, we conducted a comprehensive series of experiments with the primary objective of assessing its effectiveness in terms of success ratio, message processing overhead, and network cost. In the subsequent discussion, we will delve into the details of the experimental design and methodologies employed to achieve these objectives.

$$Success\ Ratio = \frac{Number\ of\ New\ Members\ Accepted}{Total\ Number\ of\ Join\ Requests} \tag{10}$$

$$Message\ Processing\ Overhead = \frac{Total\ Number\ of\ Message\ Processing}{Total\ Number\ of\ Join\ Requests} \tag{11}$$

$$Network\ Cost = \frac{Total\ Cost}{Total\ Number\ of\ Join\ Requests} \tag{12}$$

The success ratio, denoted by the ratio of the total number of successful members to the overall number of requests, is mathematically expressed in Formula (10). Message processing overhead, as quantified in Formula (11), is characterized by the average number of message processing times during the construction of a route for a join request. This definition implies that if a message is processed n times during route construction, its overhead is considered as n. The network cost is defined as the average cost per receiver, with the Total Cost in Equation (12) representing the sum of the Tree Costs of all multicast trees in the network.

In addition, our algorithm was compared with the PIM-SM routing protocol, Jia's Algo-based routing protocol, and QMRP. In the case of PIM-SM, the source node was designated as the tree root. Jia's Algo, a source-driven routing approach, initiates a search for potential optimal paths within nodes of the multicast tree starting from the source.

Upon reaching the leaf node, the results are consolidated and returned to the source, which then determines the optimal path and communicates this information to the leaf nodes, instructing them to construct paths to the receiver nodes. QMRP, as a receiver-driven routing algorithm, QMRP adheres to the recommended parameters, with the maximum fallback limit set at 2.

In scenarios where the QoS requirements of the request cannot be met during the tree-building process, ours algorithm will continue to construct the multicast tree along the path with the minimum delay; PIM-SM will continue building the multicast tree along the unicast routing; Jia's algorithm will proceed by selecting the minimum delay tree node to build the path to the receiver and QMRP will terminate the routing construction process.

We implemented the aforementioned algorithms in NS-3 [30], a discrete-event simulation platform. We used the Waxman [13] (generalized linear preference) model to generate network topologies in our simulation experiments. The Waxman model, proposed by Waxman in 1988, is a classic random graph topology model. Nodes are positioned randomly on a rectangular coordinate grid, with each node located at integer coordinates. The likelihood of establishing a connection between any two nodes is contingent on their distance, as defined below:

$$P(u, v) = \alpha \cdot \exp\left(-\frac{d}{\beta L}\right) \tag{13}$$

The parameters α and β take values within the range (0,1]. Here, d represents the Euclidean distance between node u and v, and L signifies the maximum distance within the graph. A higher value of β results in a denser network with more edges, whereas a lower value of α leads to a higher density of short edges.

We generated 15 topologies, each consisting of 200 nodes, using the same parameters. The specific parameters are as follows: $\alpha = 0.063$, $\beta = 2.2$, $L = 10{,}000$.

We employed a consistent random number seed across all sets of comparative experiments with identical parameters to minimize the interference of random processes on the experimental results. This seed was used to generate data sources, determine the behavior of receivers, and establish the timing for receivers to initiate join requests. Therefore, in comparative experiments with the same parameters, different simulation experiments using different topologies and routing algorithms would use the same source nodes, receiver nodes, and the same join request time sequence, even though these parameters were randomly generated. Source nodes and receiver nodes are selected using a uniform random distribution. The timing of join requests sent by receivers follows an exponential random distribution. Variations in link latency values are determined by a Gaussian random distribution, whereas the time intervals for changes in link latency follow an exponential random distribution.

We conducted the simulation on a Ubuntu 22.04 server equipped with 192 GB of RAM and dual 16-core 32-thread processors. A comprehensive and systematic assessment was performed on 15 distinct topologies for each set of comparative experiments. The outcome was derived by averaging the results across these 15 topologies. The evaluation setting is shown in Table 3.

Table 3. Summary of experimental parameters.

Parameter	Value
CPUs	64 Intel(R) Xeon(R) Silver 4216 CPU @ 2.10 GHz
Memory	192 GB
Operating system	Ubuntu 22.04
Topology parameters	Waxman, $\alpha = 0.063$, $\beta = 2.2$, $degree = 4$
Number of nodes	200
Number of groups	200
Members per group	5, 10, 15, 20, 25, 30, 35, 40
QoS requirement (ms)	50, 60, 70, 80, 90, 100, 110, 120

4.1. Success Ratio

Firstly, we analyze the success ratio trends associated with varying delay constraints. The experiments were conducted across 15 independent topologies. Within each topology, 200 groups were generated, each comprising 20 members. The average success ratio was calculated using Equation (10). As depicted in Figure 11, the success ratios of all tested algorithms exhibit an increasing trend as the delay constraints extend from 50 ms to 100 ms. Specifically, our proposed algorithm consistently maintains the highest success ratio. Under stringent delay constraints (QoS Requirement < 100 ms), our algorithm significantly outperforms other algorithms. Particularly under a 50ms constraint, its performance superiority is most pronounced, surpassing QMRP, PIM-SM, and Jia's Algo by 223.8%, 113.7%, and 86.5%, respectively. This validates the path computation capability of our algorithm, as it endeavors to search for paths in the network that offer smaller delays when constraints are stringent. With the relaxation of delay constraints, Jia's Algo, PIM-SM, and QMRP exhibit similar increasing trends in success ratios. This outcome can be attributed to the fact that as delay constraints become more lenient, routes can be constructed using links that failed under stringent constraints. However, despite the similar growth trends, the success rate and rate of increase of QMRP are notably lower than those of other algorithms. This is due to the blind exploration of feasible paths to the multicast tree during QMRP's join process. Upon discovering that the current path is unsatisfactory, QMRP backtracks and explores alternative paths that may be available. To curb the flooding of join messages in the network, the detection process is terminated after a limited number of attempts, contributing to a reduced success ratio for QMRP. Jia's algorithm initiates a tree search process during the join phase, where it selects an appropriate tree node, and utilizes the underlying routing information to establish a path from the tree node (TN) to the receiver. However, if a path exists between TN and the receiver that satisfies the delay constraint but is not provided by the underlying routing, Jia's algorithm fails to detect it, resulting in a lower success rate compared to our algorithm. Our algorithm addresses the limitations of both approaches by executing a direct tree search within the ICN NRS mechanism and subsequently detecting the path from the tree node to the receiver.

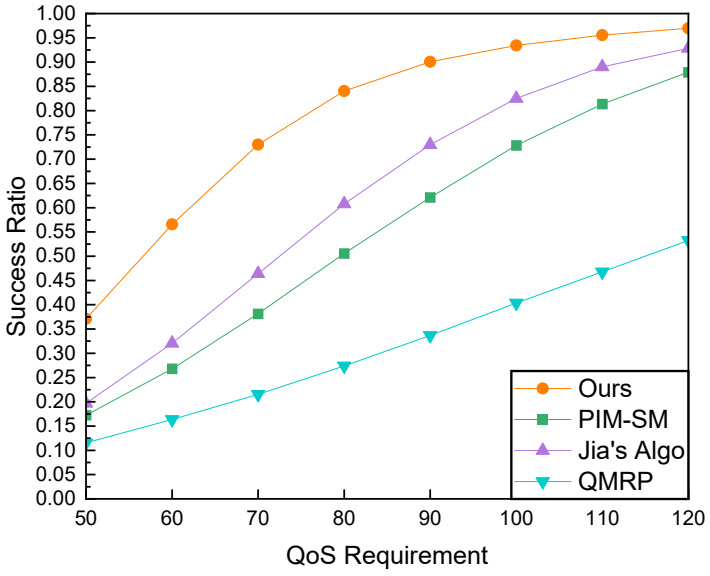


Figure 11. Comparison of Success Ratio and QoS Requirement with a Fixed Group Size of 20 Members.

Subsequently, we analyze the impact of group size on the success ratio under 50 ms and 100 ms delay constraints, as illustrated in Figure 12a,b. With the increase in the group size, our algorithm, PIM-SM, and Jia's Algo exhibit overall stable performance, albeit with minor fluctuations. Our algorithm demonstrates excellent stability, and its performance does not deteriorate with changes in the number of group members. However, the QMRP

algorithm exhibits a significant decline in performance. QMRP stops searching when it reaches a tree node, regardless of whether it finds a suitable path. Therefore, as the number of group members increases, the number of tree nodes also increases, leading to a higher probability of the algorithm stopping prematurely and resulting in a decrease in success rate.

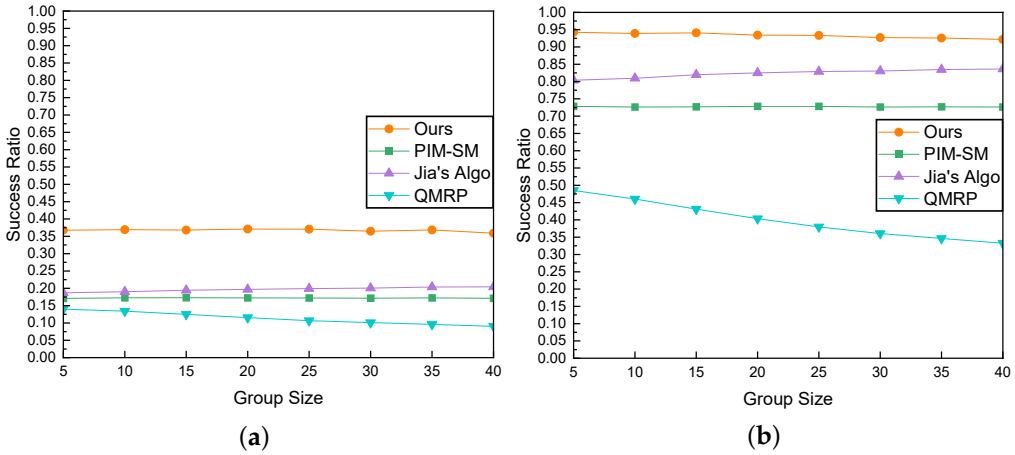


Figure 12. Comparison of Success Ratio and Group Size. (**a**) QoS Requirement is 50 ms; (**b**) QoS Requirement is 100 ms.

4.2. Message Processing Overhead

We analyzed the impact of delay constraints on message processing overhead, and the results are illustrated in Figure 13. Our algorithm, Jia's Algo, and PIM-SM demonstrate overall stable trends, with mean values of 7.8, 34.0, and 4.9, respectively.

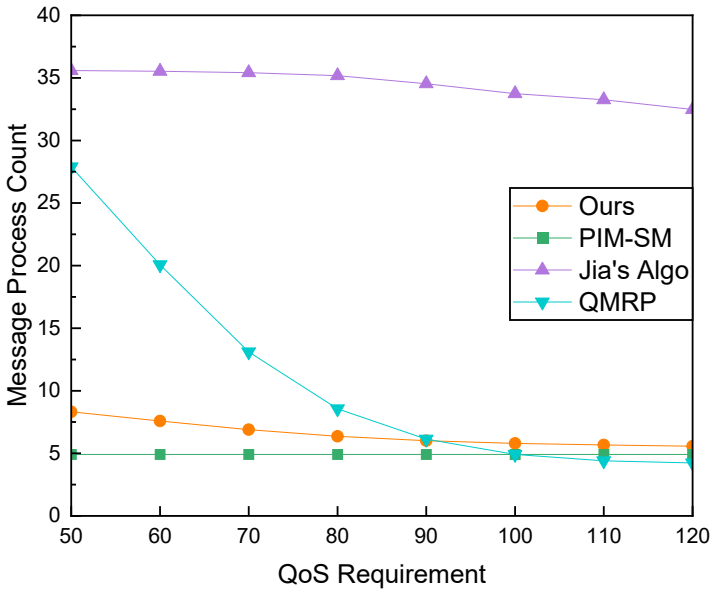


Figure 13. Comparison of Message Processing Overhead and QoS Requirement with a Fixed Group Size of 20 Members.

The message processing overhead of our algorithm is slightly higher than that of PIM-SM, this can be attributed to the need for path replanning when planned links fail to satisfy constraints, along with additional overhead from lookup operations in the resolution system. The message processing overhead of our algorithm is significantly lower than that of Jia's Algo and QMRP under strict constraints. This is because the search process in our algorithm does not involve message flooding.

While the performance of Jia's Algo remains relatively smooth, its average processing overhead is significantly higher than that of other algorithms. This is attributed to the algorithm's necessity to search for feasible solutions across the entire multicast tree for each new join request.

Subsequently, we analyzed the trends in message processing overhead for various methods as the group size varied, with delay constraints set to 50 ms and 100 ms, as illustrated in Figure 14a,b.

With the relaxation of delay constraints to 100 ms, the overhead curves for all algorithms, except for Jia's Algo algorithm, closely align. Compared to the scenario with strict constraints, the differences in overhead among the algorithms can be attributed to the additional cost incurred in searching for more optimal paths.

Jia's Algo exhibits substantial overhead under both latency constraints due to its consistent need for a tree search, and the associated flooding in the tree incurs significant overhead.

We observed that under diverse QoS requirements, as the group size increases, the processing overhead of our algorithm does not display linear growth but approaches a horizontal straight line. This indicates that our algorithm demonstrates good scalability in this context.

Under strict constraints, QMRP is more likely to trigger the search mechanism, resulting in many messages. Therefore, the significant difference in message processing overhead for QMRP under these two different constraints is evident. The message processing overhead of Jia's algorithm increases significantly with the increase in group size. As the group size increases, the size of the multicast tree also expands, leading to a proportional increase in the required number of detection nodes for Jia's Algorithm.

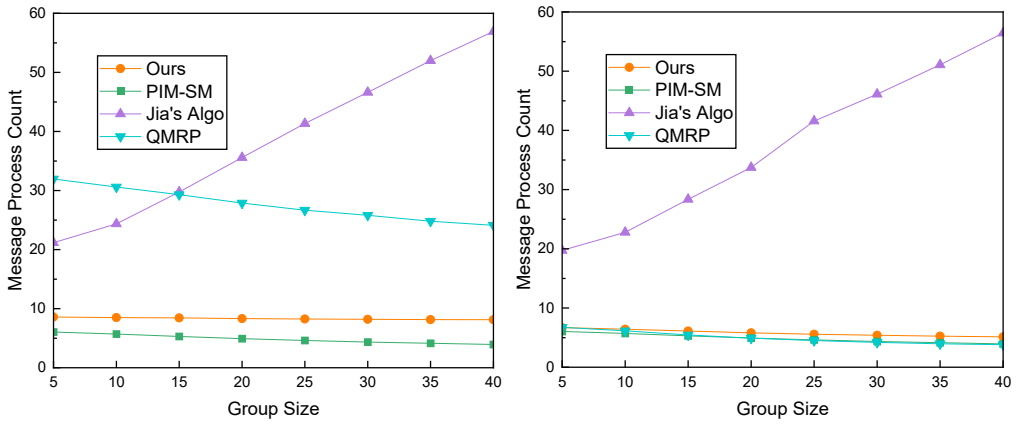


Figure 14. Comparison of Message Processing Overhead and Group Size. (**a**) QoS Requirement is 50 ms; (**b**) QoS Requirement is 100 ms.

4.3. Network Cost

Finally, we examined the impact of delay constraints on network cost, as illustrated in Figure 15. Under strict constraints, our algorithm incurs higher network costs compared to others. However, with the relaxation of QoS constraints, our algorithm exhibits a substantial reduction in cost, whereas the cost variations in other algorithms are less pronounced. Combining this outcome with our earlier exploration of success ratios, it can be inferred that the higher costs in our algorithm primarily arise from the addition of links to the multicast tree that satisfy the relaxed delay constraints but come with higher associated network costs, which are not discovered by other algorithms. As the delay constraints are relaxed, paths with higher latency but lower costs become feasible for constructing multicast trees, leading to the observed reduction in network costs for our algorithm.

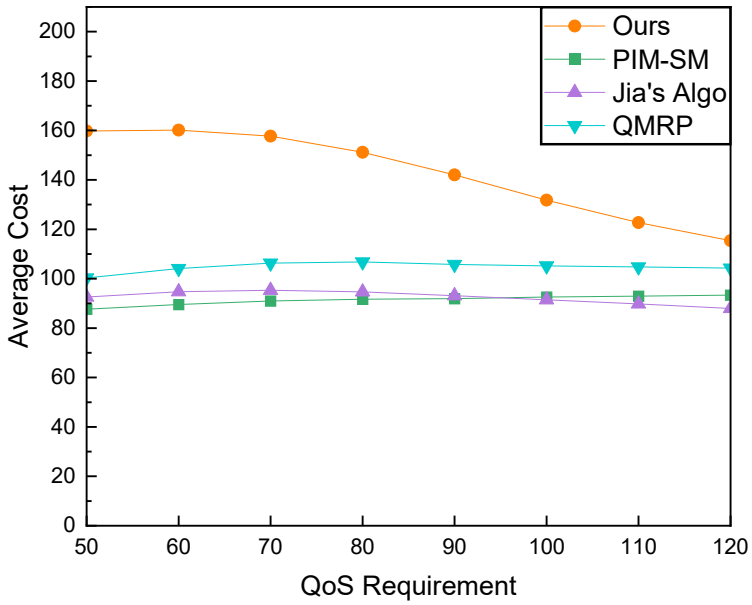


Figure 15. Comparison of Average Cost and QoS Requirements with a Fixed Group Size of 20 Members.

We further investigated the trend of network cost as group size changed, with delay constraints set to 50 ms and 100 ms, as depicted in Figure 16a,b. As the QoS constraint delay increased from 50 ms to 100 ms, our algorithm exhibited a significant reduction in cost, with costs gradually decreasing as the group member counts increased. Similar cost trends were observed in other algorithms, showing a gradual decrease. The increase in group size results in more multicast tree nodes distributed throughout the network, making it easier for routing algorithms to select tree nodes closer to the receivers. This contributes to the observed decrease in average cost. This outcome further substantiates the earlier inference. The primary objective of our algorithm is to meet the specified delay constraints. This drives our algorithm to seek paths characterized by lower latency, even if it results in higher costs. Upon the relaxation of constraints, it becomes apparent that paths with lower costs can still adhere to the stipulated delay constraints, resulting in a notable reduction in overall network cost. This further confirms our algorithm's capability to intelligently select the optimal path based on different delay constraints.

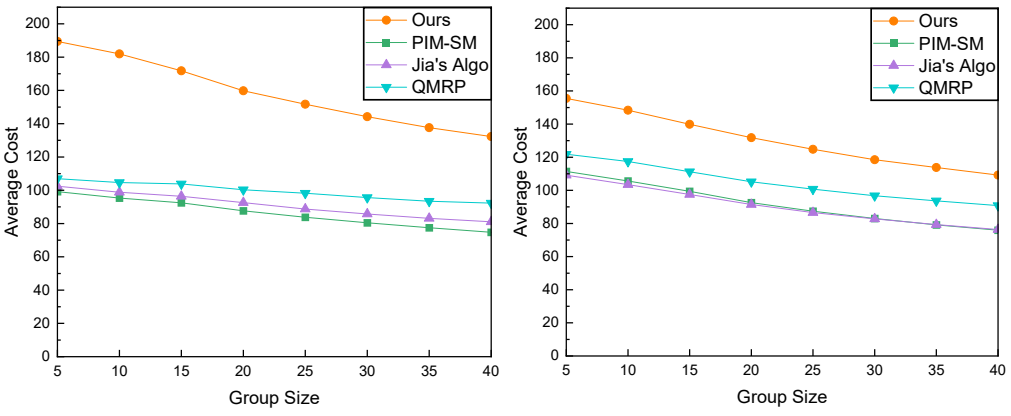


Figure 16. Comparison of Average Cost and Group Size (**a**) QoS Requirement is 50 ms; (**b**) QoS Requirement is 100 ms.

Based on the above test results, we observed that our method exhibits remarkable success ratios compared to other algorithms, particularly in scenarios with stringent delay

constraints. Our approach comes at a higher cost than other algorithms. For instance, when the QoS requirement is set to 50 ms, the cost of our method is 71.8% higher than the average cost of other algorithms. However, our success rate is also 128.4% higher than the average success rate of other algorithms. In relaxed delay constraint scenarios, the cost of our method decreases significantly. This is because, when the search of JOIN message fails to find a path that meets the requirements, we choose lower-latency links to construct the path, resulting in higher costs.

Additionally, our method keeps the message search scope relatively small, making it superior in terms of message-processing overhead and stability compared to Jia's Algo, which is source-initiated, and QMRP, which blindly expands the search scope while receiver-initiated.

4.4. Scalability

To assess the scalability of our proposed approach, we conducted a series of experiments involving larger network topologies and group sizes. We generated larger-scale network topologies through the Waxman model, containing 200, 250, 300, 350, 400, 450, and 500 nodes, respectively. Additionally, we augmented the number of multicast group members to investigate the impact of scalability on our approach.

Our experiment involved two main scenarios: firstly, maintaining a fixed group size of 20 while increasing the network topology size, and secondly, keeping the topology size constant at 500 while incrementing the group size. We observed the trends in success rates and signaling processing overhead under these varying conditions.

As shown in Figures 17a and 18a, our results indicate that as topology size and group size increased, the success ratio remained consistently high. The algorithm demonstrated robustness in maintaining efficient multicast routing, highlighting its effectiveness in accommodating a larger population of multicast group members.

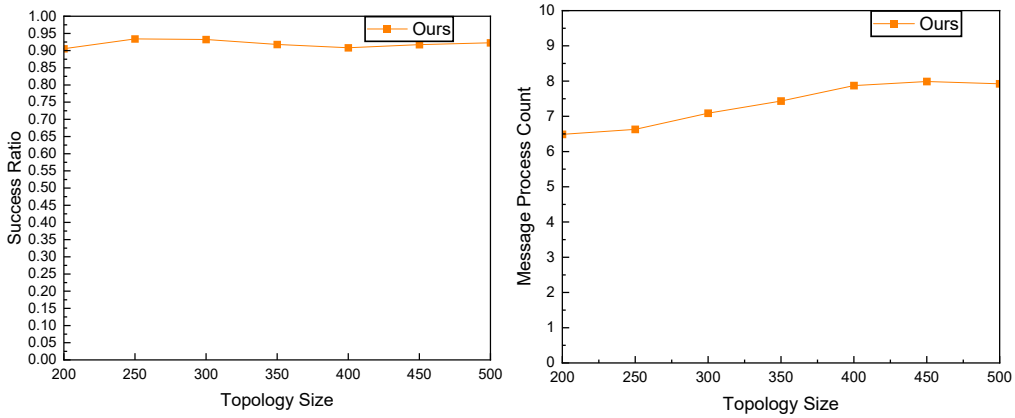
An integral aspect of scalability is the impact on message processing overhead. As depicted in Figure 17b, our experiments demonstrate a manageable and gradual increase in message processing overhead with the growth of the network topology, emphasizing the scalability of our approach. The trend in overhead demonstrates the adaptability of our method to varying scales.

In Figure 18b, as the group size increases, the processing overhead exhibits a gradual decreasing trend. This observation indicates a favorable scalability aspect of our approach. The manageable reduction in processing overhead with the expansion of the group size underscores the efficiency and scalability of our proposed method. This trend suggests that our algorithm maintains reasonable performance even as the multicast group scales up, showcasing its potential for scalability in handling larger network scenarios.

In conclusion, the results of the experiments showcase the scalability of the proposed approach. As the network topology increased, we observed a manageable growth in signaling processing overhead, demonstrating that our method scales effectively. Furthermore, the success rate remained stable, indicating that our approach maintains a high level of performance even in scenarios with an expanded network and increased multicast group complexity.

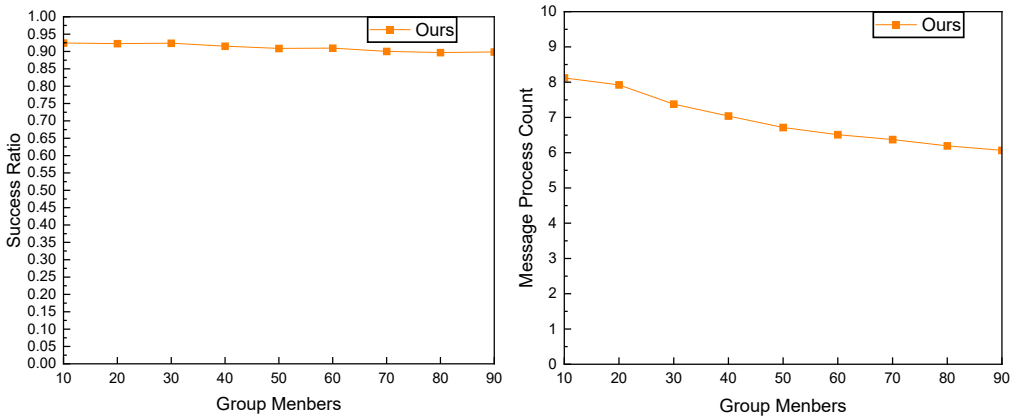
While our approach demonstrates notable strengths, it comes with certain limitations that merit consideration. Firstly, in comparison to alternative methods, our algorithm incurs higher network costs. This is attributed to our strategy of constraining the search scope for feasible paths to mitigate unnecessary signaling overhead, potentially resulting in locally optimal solutions rather than always achieving the minimum-cost paths. Secondly, our primary focus has been on addressing how to meet application latency requirements during the tree construction phase. However, when network conditions change, the already established multicast routes may no longer satisfy the constraints. Lastly, our algorithm currently emphasizes the latency metric, acknowledging that application Quality of Service (QoS) requirements encompass multiple constraints. It is important to note that future research can explore the algorithm's performance in handling multiple constraint

conditions, providing a more comprehensive solution to meet the demands of different application scenarios.



(**a**) (**b**)

Figure 17. The impact of topology size on success ratio and message processing overhead. (**a**) Success Ratio; (**b**) Message Processing Overhead.



(**a**) (**b**)

Figure 18. The impact of Group Member on success ratio and message processing overhead. (**a**) Success Ratio; (**b**) Message Processing Overhead.

5. Conclusions

This paper introduces a distributed method for constructing delay-constrained multicast trees in Information-Centric Networking (ICN). The ICN system utilized in this work operates on a name-based lookup mechanism, using globally unique names to identify multicast services. The Name Resolution Service (NRS) is responsible for managing the mappings between multicast identifiers and the Network Addresses (NAs) of the tree nodes.

We proposed a receiver-initiated distributed multicast path construction mechanism. The receiver node retrieves the nodes in the multicast tree through NRS, detects real-time delay information along the path with a join message, and constructs a multicast path from the tree node to the receiver in real-time based on the network status. To circumvent acquiring accurate information regarding the status of the complete network, we design a path construction algorithm that makes decisions based on precise path state information and imprecise routing information. Furthermore, to facilitate the selection of suitable tree nodes, we have designed a tree node registration and resolution mechanism based on NRS. This enables receivers to obtain a feasible node set within the multicast tree via NRS. Additionally, we introduced a cost-effective method to build a Neighborhood State Information Base (NSIB) to achieve more accurate path delay estimation. Simulation results

demonstrate that our approach outperforms some other methods in terms of both success rates and low message-processing overhead.

Moreover, we have observed that fluctuations in network state may lead to violations of QoS metrics. To address this challenge, it is imperative to consider the dynamic adjustment of multicast trees and the establishment of fault-tolerant paths. This issue resides within the research domain of multicast tree fault recovery, which surpasses the scope of this paper. Prospective solutions to this matter will be pursued in forthcoming research endeavors.

Additionally, our future research will delve deeper into the prospect of minimizing the cost of multicast trees while concurrently maintaining a high success rate in QoS multicast tree construction. Concurrently, our focus will shift towards extending the proposed methodology to encompass a broader range of constraint parameters. These explorations will form the basis for our forthcoming research endeavors.

Author Contributions: Conceptualization, J.S., H.N. and X.Z.; Funding acquisition, H.N.; Methodology, J.S., H.N. and X.Z.; Project administration, X.Z.; Software, J.S.; Supervision, H.N.; Writing—original draft, J.S.; Writing—review & editing, J.S., H.N. and X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by Frontier Exploration Project Independently Deployed by Institute of Acoustics, Chinese Academy of Sciences: Research on transmission technology for information center network (Project No. QYTS202112).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in the article.

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

References

1. Li, J.; Li, Z.; Lu, R.; Xiao, K.; Li, S.; Chen, J.; Yang, J.; Zong, C.; Chen, A.; Wu, Q.; et al. LiveNet: A Low-Latency Video Transport Network for Large-Scale Live Streaming. In Proceedings of the ACM SIGCOMM 2022 Conference, Amsterdam, Netherlands, 22–26 August 2022; Reply: Changes confirmed. The modifications are reasonable. SIGCOMM '22, Reply: They can be deleted. pp. 812–825. [CrossRef]
2. Singh, P.K.; Bhargava, B.; Hong, W.C.; Angin, P. 1174: Futuristic trends and innovations in multimedia systems using big data, IoT and cloud technologies (FTIMS). *Multimed. Tools Appl.* **2022**, *81*, 34439–34445. [CrossRef]
3. Tsoukaneri, G.; Condoluci, M.; Mahmoodi, T.; Dohler, M.; Marina, M.K. Group Communications in Narrowband-IoT: Architecture, Procedures, and Evaluation. *IEEE Internet Things J.* **2018**, *5*, 1539–1549. [CrossRef]
4. Bull, D.R.; Zhang, F. Chapter 11—Communicating pictures: Delivery across networks. In *Intelligent Image and Video Compression*, 2nd ed.; Bull, D.R., Zhang, F., Eds.; Academic Press: Oxford, UK, 2021; pp. 385–434. [CrossRef]
5. Frank, A.J.; Wittie, L.D.; Bernstein, A.J. Multicast Communication on Network Computers. *IEEE Softw.* **1985**, *2*, 49–61. [CrossRef]
6. Deering, S.E. Multicast Routing in Internetworks and Extended LANs. In Proceedings of the Symposium Proceedings on Communications Architectures and Protocols, Stanford, CA, USA, 16–18 August 1988; SIGCOMM '88, pp. 55–64. [CrossRef]
7. Campbell, A.; Coulson, G.; Hutchison, D. A Quality of Service Architecture. *SIGCOMM Comput. Commun. Rev.* **1994**, *24*, 6–27. [CrossRef]
8. Sahasrabuddhe, L.; Mukherjee, B. Multicast routing algorithms and protocols: A tutorial. *IEEE Netw.* **2000**, *14*, 90–102. [CrossRef]
9. Huang, T.L.; Lee, D. A distributed multicast routing algorithm for real-time applications in wide area networks. *J. Parallel Distrib. Comput.* **2007**, *67*, 516–530. [CrossRef]
10. Zhang, L.; Deering, S.; Estrin, D.; Shenker, S.; Zappala, D. RSVP: A New Resource ReSerVation Protocol. *IEEE Commun. Mag.* **1993**, *40*, 116–127. [CrossRef]
11. Yuan, X.; Zheng, W.; Ding, S. A comparative study of QoS routing schemes that tolerate imprecise state information. In Proceedings of the Eleventh International Conference on Computer Communications and Networks, Miami, FL, USA, 16 October 2002; pp. 230–235. [CrossRef]
12. Sanguankotchakorn, T.; Perera, N. Hybrid Multi-constrained Optimal Path QoS Routing with Inaccurate Link State. In Proceedings of the 2010 Ninth International Conference on Networks, Menuires, France, 11–16 April 2010; pp. 321–326. [CrossRef]
13. Waxman, B. Routing of multipoint connections. *IEEE J. Sel. Areas Commun.* **1988**, *6*, 1617–1622. [CrossRef]
14. Winter, P. Steiner problem in networks: A survey. *Networks* **1987**, *17*, 129–167. [CrossRef]

15. Garey, M.R.; Johnson, D.S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*; W. H. Freeman & Co.: New York, NY, USA, 1990.
16. Oliveira, C.A.S.; Pardalos, P.M., Metaheuristics for Multicast Routing. In *Mathematical Aspects of Network Routing Optimization*; Springer: New York, NY, USA, 2011; pp. 77–94. [CrossRef]
17. Zhang, L.; Afanasyev, A.; Burke, J.; Jacobson, V.; Claffy, K.; Crowley, P.; Papadopoulos, C.; Wang, L.; Zhang, B. Named Data Networking. *SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 66–73. [CrossRef]
18. Ahlgren, B.; Dannewitz, C.; Imbrenda, C.; Kutscher, D.; Ohlman, B. A survey of information-centric networking. *IEEE Commun. Mag.* **2012**, *50*, 26–36. [CrossRef]
19. Xylomenos, G.; Ververidis, C.N.; Siris, V.A.; Fotiou, N.; Tsilopoulos, C.; Vasilakos, X.; Katsaros, K.V.; Polyzos, G.C. A Survey of Information-Centric Networking Research. *IEEE Commun. Surv. Tutorials* **2014**, *16*, 1024–1049. [CrossRef]
20. Din, I.U.; Asmat, H.; Guizani, M. A review of information centric network-based internet of things: Communication architectures, design issues, and research opportunities. *Multimed. Tools Appl.* **2019**, *78*, 30241–30256. [CrossRef]
21. Bari, M.F.; Chowdhury, S.R.; Ahmed, R.; Boutaba, R.; Mathieu, B. A survey of naming and routing in information-centric networks. *IEEE Commun. Mag.* **2012**, *50*, 44–53. [CrossRef]
22. 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; CoNEXT '09, pp. 1–12. [CrossRef]
23. Zeng, L.; Ni, H.; Han, R. An Incrementally Deployable IP-Compatible-Information-Centric Networking Hierarchical Cache System. *Appl. Sci.* **2020**, *10*, 6228. [CrossRef]
24. Venkataramani, A.; Kurose, J.F.; Raychaudhuri, D.; Nagaraja, K.; Mao, M.; Banerjee, S. MobilityFirst: A Mobility-Centric and Trustworthy Internet Architecture. *SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 74–80. [CrossRef]
25. Wang, J.; Cheng, G.; You, J.; Sun, P. SEANet: Architecture and Technologies of an On-site, Elastic, Autonomous Network. *J. Netw. New Media* **2020**, *6*, 1–8.
26. Liao, Y.; Sheng, Y.; Wang, J. Survey on the Name Resolution Technologies in Information Centric Networking. *J. Netw. New Media* **2020**, *9*, 1–9.
27. Crawley, E.; Nair, R.; Rajagopalan, B.; Sandick, H. RFC 2386: A Framework for QoS-Based Routing in the Internet. Available online: https://www.rfc-editor.org/rfc/rfc2386 (accessed on 1 August 1998).
28. Apostolopoulos, G.; Kama, S.; Williams, D.; Guerin, R.; Orda, A.; Przygienda, T. RFC 2676: QoS Routing Mechanisms and OSPF Extensions. Available online: https://www.rfc-editor.org/rfc/rfc2676 (accessed on 1 August 1999).
29. Wang, B.; Hou, J. Multicast routing and its QoS extension: Problems, algorithms, and protocols. *IEEE Netw.* **2000**, *14*, 22–36. [CrossRef]
30. Riley, G.F.; Henderson, T.R. The ns-3 Network Simulator. In *Modeling and Tools for Network Simulation*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 15–34. [CrossRef]
31. Deering, S.E. RFC 1112: Host Extensions for IP Multicasting. Available online: https://www.rfc-editor.org/rfc/rfc1112 (accessed on 1 August 1989).
32. Schollmeier, G.; Winkler, C. Providing sustainable QoS in next-generation networks. *IEEE Commun. Mag.* **2004**, *42*, 102–107. [CrossRef]
33. Deering, S.; Estrin, D.; Farinacci, D.; Jacobson, V.; Liu, C.G.; Wei, L. An Architecture for Wide-Area Multicast Routing. In Proceedings of the Conference on Communications Architectures, Protocols and Applications, London, UK, 31 August–2 September 1994; SIGCOMM '94, pp. 126–135. [CrossRef]
34. Kou, L.; Markowsky, G.; Berman, L. A fast algorithm for Steiner trees. *Acta Inform.* **1981**, *15*, 141–145. [CrossRef]
35. Prim, R.C. Shortest connection networks and some generalizations. *Bell Syst. Tech. J.* **1957**, *36*, 1389–1401. [CrossRef]
36. Zhu, Q.; Parsa, M.; Garcia-Luna-Aceves, J. A source-based algorithm for delay-constrained minimum-cost multicasting. In Proceedings of the INFOCOM'95, Boston, MA, USA, 2–6 April 1995; Volume 1, pp. 377–385. [CrossRef]
37. Jia, X. A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks. *IEEE/ACM Trans. Netw.* **1998**, *6*, 828–837. [CrossRef]
38. Huang, T.L.; Lee, D. Comments and an improvement on "A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area Networks". *IEEE/ACM Trans. Netw.* **2005**, *13*, 1410–1411. [CrossRef]
39. Kompella, V.P.; Pasquale, J.C.; Polyzos, G.C. Optimal multicast routing with quality of service constraints. *J. Netw. Syst. Manag.* **1996**, *4*, 107–131. [CrossRef]
40. Carlberg, K.; Crowcroft, J. Building Shared Trees Using a One-to-Many Joining Mechanism. *SIGCOMM Comput. Commun. Rev.* **1997**, *27*, 5–11. [CrossRef]
41. Yan, S.; Faloutsos, M.; Banerjea, A. QoS-aware multicast routing for the Internet: The design and evaluation of QoSMIC. *IEEE/ACM Trans. Netw.* **2002**, *10*, 54–66. [CrossRef]
42. Chen, S.; Nahrstedt, K.; Shavitt, Y. A QoS-aware multicast routing protocol. *IEEE J. Sel. Areas Commun.* **2000**, *18*, 2580–2592. [CrossRef]
43. Koponen, T.; Chawla, M.; Chun, B.G.; Ermolinskiy, A.; Kim, K.H.; Shenker, S.; Stoica, I. A Data-Oriented (and beyond) Network Architecture. *SIGCOMM Comput. Commun. Rev.* **2007**, *37*, 181–192. [CrossRef]

44. 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, Athens, Greece, 25–27 October 2010; Tomkos, I.; Bouras, C.J.; Ellinas, G.; Demestichas, P.; Sinha, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 1–13.
45. Mukherjee, S.; Bronzino, F.; Srinivasan, S.; Chen, J.; Raychaudhuri, D. Achieving Scalable Push Multicast Services Using Global Name Resolution. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–6. [CrossRef]
46. Yang, B.; Chen, X.; Xie, J.; Li, S.; Zhang, Y.; Yang, J. Multicast Design for the MobilityFirst Future Internet Architecture. In Proceedings of the 2019 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 18–21 February 2019; pp. 88–93. [CrossRef]