electronics

Article

A Tree Structure Protocol for Hierarchical Deterministic Latency Name Resolution System

Wei Xie [1,2], Jiali You [1,2,*] and Jinlin Wang [1,2]

1 National Network New Media Research Center, Institute of Acoustics, Chinese Academy of Sciences, No. 21, North Ring Road, Haidian District, Beijing 100190, China
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: youjl@dsp.ac.cn

Abstract: Information-centric networking (ICN) shifts the communication model from a host-centric paradigm to an information-centric paradigm, and is promising for solving several problems on today's Internet. For more efficient information dissemination, most ICN architectures are based on the Identifier/Locator split design. Therefore, how to map an identifier to a routable locator is an important problem for efficient data transmission. Nowadays, many new network services such as industrial control and telemedicine are highly latency-sensitive and require deterministic service response latency. To meet such requirements, name resolution with a deterministic latency guarantee is needed, but less discussed. This paper proposes a tree-based resolution system structure for deterministic latency resolution, which can support the Local Name Mapping Resolution System (LNMRS) in the new ICN network architecture—SEANet—to provide deterministic name resolution service in latency-sensitive scenarios like industrial control and telemedicine. The correctness of such a structure is the key to achieving deterministic latency resolution. To ensure the structure's correctness in a distributed manner, a tree structure protocol based on delay measurement is also proposed for structure generation and maintenance. Simulation results show that the protocol is effective in generating a correct structure that has good performance in terms of service capability for deterministic name resolution and system scalability.

Keywords: ICN; name resolution system; tree structure; deterministic latency

check for updates

Citation: Xie, W.; You, J.; Wang, J. A Tree Structure Protocol for Hierarchical Deterministic Latency Name Resolution System. *Electronics* **2022**, *11*, 2425. https://doi.org/10.3390/electronics11152425

Academic Editors: Dongkyun Kim, Muhammad Azfar Yaqub and Muhammad Toaha Raza Khan

Received: 27 June 2022
Accepted: 28 July 2022
Published: 4 August 2022

1. Introduction

The widely used Internet protocol (IP) network architecture of today's Internet was designed for a host-to-host communication model with best-effort data transmission. Its data transmission relies on the endpoint hosts with limited performance and makes it difficult to meet the forwarding demands of emerging network services that require strict performance [1]. To improve the performance of the network, information-centric networking (ICN) [2,3] has emerged. It uses an information-centric communication model that separates content from its server location and can exploit the in-network storage capabilities to achieve more efficient information delivery.

One major cause of the limitations of today's Internet, such as scalability, flexibility, security, and mobility [4–6], is IP semantic overload. Most ICNs choose an Identifier/Locator split design to overcome this problem [7]. Based on whether name resolution is coupled with content routing, ICN networks can be classified into two categories: name-based routing (NBR) and standalone name resolution systems (SNR) [8]. The NBR approach uses identifier-based routing, such as a content-centric network (CCN) [9] and named data networking (NDN) [10]. The SNR approach uses the Identifier/Locator split design. Users first obtain the locator of the corresponding identifier from the resolution system, then use the locator for data routing. Such ICN architectures include data-oriented network architecture (DONA) [11], the publish–subscribe Internet routing paradigm (PURSUIT) [12,13], scalable and adaptive Internet solutions (SAIL) [14], and MobilityFirst [15]. The SNR approach avoids the problems caused by IP semantic overload and has the advantage of better scalability and compatibility with existing IP architecture.

In the SNR approach, accessing content requires mapping content names to locators first. Name resolution, as the first step of data transmission, is a key influence factor on the quality of service (QoS) provided by the network. Every ICN architecture that uses the SNR approach has a name resolution system (NRS) for name resolution. However, existing NRSs are having trouble meeting the QoS demand of emerging latency-sensitive applications. Emerging network services [16,17], such as autonomous driving, industrial control, smart medical, etc., have stringent requirements on network latency. These services not only require the end-to-end latency to be low (less than 10 ms) but also to have deterministic bounds [18,19]. Most existing NRSs cannot meet these requirements because their structure is based on DHT or AS-based inter-domain routing relations. In these systems, resolution requests are forwarded until they reach a node that can respond to the request. The forwarding path length and the single-hop delay between the resolution nodes are often not guaranteed, and therefore the resolution delay cannot be guaranteed. Inconstant query hops and unguaranteed single-hop latency are the key limitations of existing name resolution systems in providing resolution service with deterministically bounded latency. How to provide deterministic low-latency name resolution is a new challenge for ICN research in the face of the above-mentioned latency-sensitive scenarios.

SEANet [20] is a new ICN network architecture with on-site, elastic, and autonomous characteristics, which guarantees ultra-low service response latency through on-site services. It separates identifier and locator, each network entity is identified by a unique ID, and the mapping of IDs to network addresses is managed and maintained by a resolution system, which is crucial to its on-site data-processing capability. The SEANet resolution system consists of two parts, Global Name Mapping and Resolution System (GNMRS) and Local Name Mapping and Resolution System (LNMRS). The content accessibility is guaranteed by GNMRS, which provides global name resolution that ensures information retrieval, whereas the LNMRS provides resolution services on-site with guaranteed low response latency for latency-sensitive scenarios. LNMRS [21] aims to provide resolution services on-site for users within different deterministic time limits. This paper proposes a nested tree structure to support deterministic latency resolution for LNMRS, and a structure protocol to guarantee the correct generation of LNMRS in deployment. The proposed structure embeds latency relations of resolve nodes and users into the nested tree and achieves deterministic latency by limiting the query path and single-hop latency in the resolution process. Different layers of the tree have different latency guarantees, satisfying the differential needs of various applications. A nested tree protocol is proposed to ensure that the structure is correctly generated according to the latency relations and to maintain the structural validity when network conditions change.

The main contributions of this paper are as follows:

- This paper embeds latency constraints into the NRS structure and proposes a new nested tree structure for distributed resolution systems such as LNMRS, which can support deterministic latency by exploiting the embedded latency relations of resolvers and users.
- This paper proposes a protocol for nested tree generation and maintenance based on a node relation inference method and a virtual node mechanism. The proposed inference method ensures the desired latency characteristic of the generated structure. The virtual node mechanism generates placeholders when needed in the process of structure construction, enhancing the applicability of protocol in actual deployment.

The remainder of this paper is organized as follows: Section 2 introduces the related work on ICN resolution methods, Section 3 illustrates the LNMRS architecture and its protocol design requirements, Section 4 introduces the generation and maintenance protocol design for deterministic latency tree structure, Section 5 evaluates the performance of

LNMRS under the proposed protocol, and Section 6 concludes the work in this paper and discusses future works.

2. Related Work

The NBR approach [9,10] uses a name as the locator for routing. When the location of content changes, the adjustment to the routing table is expansive and slow. Although method [22] has been proposed to speed up the management of the routing table (e.g., Search, Add, and Delete prefixes) on a single router using hybrid naming and a Compact Trie, the flooding-based content publishing and request forwarding may produce high traffic overhead. The length of the forwarding path is also variable, and therefore unable to guarantee deterministic latency with a constant upper limit.

This paper focuses on the SNR approach. The SNR approach manages resolution entries in the network through an independent resolution system. Endpoints obtain the desired locator by querying the resolution system using the content name (identifier). Then the network will route data based on the locator. This identifier/locator split manner can support mobility better than the NBR approach. A lot of ICN architectures use the SNR approach for resolution, including DONA, PURSUIT, COMET [23], SAIL, MobilityFirst, etc. Their resolution systems differ in terms of system structure, query methods, etc. These ICN resolution systems can be divided into two categories, on-path resolution systems and query resolution systems. Prior works in both categories are insufficient in terms of deterministic latency guarantee.

2.1. On-Path Resolution

An on-path resolution system completes the resolution of identifiers to locators on the forwarding path of content requests between resolution servers. The resolution nodes in the system route requests to the content provider by content name, and the locator of the requester is in the request messages. When the request reaches the provider, the provider can use the locator within the request message to establish communication with the requestor. The resolution system of resolution handlers (RHs) [11] in DONA and the Content Resolution System (CRS) [23] in COMET is two typical on-path resolution systems. In the system of RHs, a resolution server is called a resolution handler and is placed in each autonomous system (AS). RHs are interconnected according to existing inter-domain routing relations, forming a hierarchical name resolution service. When an object is published, the publisher will register its name-locator binding in the local RH, and then the registration message is forwarded to the RHs in its parent and peering domains. The RH that receives the registration message stores a mapping between the content name and the address of the RH that forwarded the message, and continues the propagation of the registration message, all the way up to the RHs of the tier-1 ISPs; when resolving a name, it starts from the local RH, and each RH forwards the request to the next RH according to the registered mappings that it stores, or to peering and parent RHs when no mapping is found until the request reaches the content provider. RHs also cache content copies to support nearby copy lookup and fast data access for users. The CRS node in COMET is similar to the RH in DONA, except that the RH propagates data registration messages to the nodes in the parent and peering domains, whereas the CRS only propagates data registration messages to the parent domain to reduce the number of resolution entries maintained in a single node.

In the on-path resolution systems, the name entries of the lower-level AS will be aggregated to the top-level AS, which puts pressure on the RH of the top-level AS and is deficient in system scalability. Meanwhile, the nearest copy location is not necessarily close, and the caching of copies is determined by local RH policies with great uncertainty, resulting in an uncertain and possibly long resolution path. Therefore, deterministic and low-resolution latency is not guaranteed either.

2.2. Query Resolution

A query resolution system serves as a database for name–locator bindings, and maps content names to locators by searching for corresponding binding records that it stored. Most query resolution systems in ICN use DHT for scalable querying of flat name bindings, e.g., Dmap [24], Auspice [25], Ftree [26], DHT-NRS [27], HSkip [28], and MDHT [29].

Dmap, Auspice, and Ftree use DHT for name–locator mapping distribution. DMap is a single-layer resolution system that consists of content routers (CR) from each AS. The identifier–locator mapping is mapped to k existing network addresses via k consistent hashing functions and stored in the CRs of the ASs that announce these addresses. When performing name resolution, local CR can reach the mapping in a single overlay hop using the same hashing functions. Because the geographical distribution is not considered when selecting the location of the k mappings, DMap cannot guarantee low latency resolution. Ftree is a tree-based resolution system that only stores name mappings on leaf nodes. It uses hash functions to distribute mappings to multiple leaf nodes to prevent a root-level bottleneck for hierarchical resolution systems. Requests are forwarded to the nearest leaf node that has the name mapping. As the number of hash functions used increases, it is more likely that the mapping would be placed in a nearby leaf node. However, the path between two leaf nodes must first go through their common ancestor, which can result in an inconstant and possibly long resolution path with no latency guarantee. Auspice takes the geographic distribution of name-binding replicas and name popularity into account and proposes an automatic name-binding replica placement method based on resolution demand to reduce resolution latency. It divides the network according to geographic regions and uses the demand-aware heuristic replica placement method to adaptively determine the replica number and locations. A name binding is mapped to several Auspice nodes using consistent hashing. These nodes are the replica controllers of the name binding that determine numbers and locations of the replicas periodically. Other nodes cache the replica locations from the replica controller when they receive the first request for the name so that their users can choose the closest replica to query. Auspice fully considers the spatiotemporal characteristics of name-binding distribution and greatly reduces the resolution latency. However, the network conditions of each region are different, and Auspice can only guarantee the acquisition of a close replica, but not the acquisition latency.

DHT-NRS, HSkip, and MDHT use DHT for structure generation. They also try to build the system hierarchy based on the hierarchical structure of the underlying inter-domain topology to achieve resolution locality and low latency. DHT-NRS is a hierarchical resolution system based on an enhanced DHT design named H-Pastry [30]. It deploys several rendezvous nodes (RNs) in each AS to store the name–locator bindings of locally published data, and the RNs form a hierarchical rendezvous network (RENE) according to the H-Pastry protocol. A resolution request that contains the requestor's locator is routed to the RN with the required binding using H-Pastry, and then the RN sends the request's locator and required content name to the publisher so that the publisher can initiate a transmission. HSkip is a resolution system based on the DHT architecture named SkipNet [31]. Every HSkip node is assigned a hierarchical string ID corresponding to the underlying network topology (e.g., country.provider.organization.node), and a flat numeric ID from the same namespace as the content. HSkip builds the system hierarchy base on the string ID and forwards resolution requests from the lowest layer up to achieving resolution locality. MDHT is a hierarchical resolution system that allows the coexistence of different DHT mechanisms among nodes. Each level represents a topological level of a network, e.g., the AS level and Point of Presence (POP) level. Each MDHT node can be in multiple resolution levels and has a different DHT routing table for each level, so MDHT can forward resolution requests between different DHT protocols traversing the resolution layer, giving different providers more autonomy in choosing their own DHT mechanism.

DHT-NRS, HSkip, and MDHT are naturally highly scalable, but the possible inter-domain forwarding of resolution requests, and the inconstant forwarding hops in DHT, may result in long resolution paths and high latency.

Table 1 summarizes the differences between the above-mentioned NRSs. As shown, in on-path resolution systems, name mappings from different ASs are aggregated at the root level. This puts a heavy load on the root level, creating a root-level bottleneck. Resolution paths to roots are also distant, resulting in a long resolution latency. In DHT-based NRSs, neighbors of nodes in a DHT ring can be physically scattered, and this would increase hop stretch. In addition, the searching process in a DHT ring may contain multiple hops, and therefore cannot guarantee resolution latency. Even though Dmap does not base its structure on DHT, the mappings scattered by hash functions can be too distant for most users. Auspice tries to address this problem using a proactive mapping placement method, but it requires extensive information gathering, and there is no guaranteed boundary on the latency from users to the closest location of mapping. Therefore, it is still unable to guarantee deterministic latency.

Table 1. The comparison of different ICN NRS.

NRS	Structure	Mapping Distribution	Resolution Latency
RHs	Tree based on AS relations	Propagates to all adjacent nodes	Suffers from long distance and heavy root-level load
CRS	Tree based on AS relations	Propagates to parent only	Suffers from long distance and heavy root-level load
Dmap	Flat structure based on AS relations	Distributed with consistent hash functions	Suffers from long distance
Ftree	Tree based on AS relations	With hush functions and only on leaf nodes	Suffers from possibly long forwarding path
Auspice	Tree based on geography	Demand-aware proactive replica placement	No guaranteed boundary
DHT-NRS	H-Pastry	Distributed with hash functions	Suffers from hop stretch
Hskip	SkipNet	Distributed with hierarchical naming and hash functions	Suffers from hop stretch
MDHT	Hierarchical heterogeneous DHT	Bottom-up propagation	Suffers from hop stretch

With the development of network service, the Internet tends to change from "best-effort" data delivery to "deterministic data transmission service." To meet this trend, this paper proposes a nested tree structure for a distributed resolution system to achieve deterministic latency. Different from prior solutions, our solution embeds latency constraints into the structure and keeps the forwarding of requests under latency limits.

The system model is illustrated in Section 3, and the protocol design is described in Section 4.

3. Local Name-Mapping Resolution System

To support the deterministic name resolution, this paper proposes a nested tree structure to implement LNMRS. Before presenting the structure proposed, this section first describes the system model of LNMRS in Section 3.1.

3.1. System Model

To achieve deterministic resolution latency, this paper proposes a tree-based system structure for LNMRS. In this section, the system model of LNMRS is introduced, and the proposed nested tree structure is illustrated. The notations used are summarized in Table 2.

On-site resolve node: The LNMRS nodes are placed at the locations that can be reached within a deterministic latency upper bound by users, and are chosen according to users' needs so that they can provide service with deterministic latency. Different applications have different latency requirements, so the latency demand of users is stratified from high to low into multiple levels, each with a latency upper bound T_l (l indicates the level). At each level, a resolve node (RN) is placed at the location that can reach the users within T_l to provide differentiated deterministic latency resolution. The user only chooses one node at each level as the main resolver, but a node can consist of one or more servers at the same location.

Table 2. Summary of notations.

Notation	Description
T_l	Latency upper bound in l level
G	The entire network
R or N	An LNMRS resolver
RA_R	The resolution area of the resolver R
RN^l	All resolvers of level l
$\Phi(R)$	The user set of resolver R
R_u^j	The resolver serving user u in level j
$D(u, R)$	The latency between u and R

Resolution area: When an LNMRS node R is placed, the network area that is within T_l distance to the node is its resolution area (RA). The resolution node can only guarantee deterministic latency resolution for queries from its RA (indicated as RA_R). With the gradual deployment of RNs, each part of the network should be contained in an RA to achieve full coverage of deterministic resolution service for the network, i.e., $G = \bigcup_{R \in RN^l} RA_R$ for all $l \in Levels$, where G indicates the entire network and RN^l indicates the RNs in level l.

Nested inter-level relation: The LNMRS node can only provide service within its RA, so the user set of node R, $\Phi(R)$, also resides in RA_R. To enable users to get different levels of resolution latency guarantee, the user set of the higher-level nodes with a higher latency upper bound and consequently larger RAs should contain the user set of the lower-layer nodes. We call this relationship a nested relationship, i.e., the RNs that both serve user u in levels j and k ($k > j$), indicated as R_u^j and N_u^k, respectively, should satisfy $\Phi\left(R_u^j\right) \subset \Phi\left(N_u^k\right)$.

Name registration and resolution: Users send registration and resolution requests to the node, whom they chose as their resolver, at the level that they need. The LNMRS node will only forward these requests to strictly chosen neighbors that can still satisfy the latency constraint. Name-binding propagation can be done by a proactive caching mechanism or taking the advantage of the GNMRS, or other methods. However, both neighbor selection and binding propagation are outside the scope of this paper, as this paper tries to focus on the structural part of the system. So, in this paper, the forwarding of requests to neighbors is not considered. The resolver only processes request locally, and will not forward the request to other resolvers. Li's paper [32] can be referred to as a possible neighbor selection method.

3.2. *Nested Tree Structure*

A specific structure is needed to realize the system model described above. The nested relationship between the upper and lower layers of the LNMRS fits naturally with the tree structure, which is scalable and has simple node relations that are easy to maintain, so we propose organizing LNMRS nodes based on a nested tree structure.

In the tree, each level has a guaranteed resolution latency T. The higher the level, the higher the T, i.e., $T_l > T_{l-1}$. Users only choose one RN at each level as their resolver. At level l, user u chooses RN R as a resolver only when $D(u, R) \leq T_l$; $D(u, R)$ is the latency between u and R. The nodes that are nested with the l-level node R in the $l-1$ level are the children of R, i.e., $\forall\ N \in RN^{l-1}, R \in RN^l$, if $\Phi(N) \subset \Phi(R)$, $N \in Children(R)$. Therefore, a node is nested with its entire sub-tree. Each node at the highest level with the highest T is a root of a tree. The system is a forest of nested trees.

Figure 1 shows the example of a three-layer LNMRS system, which contains a root manager (RM) and three layers of RNs, with each layer of RNs providing resolution services under different latency constraints.

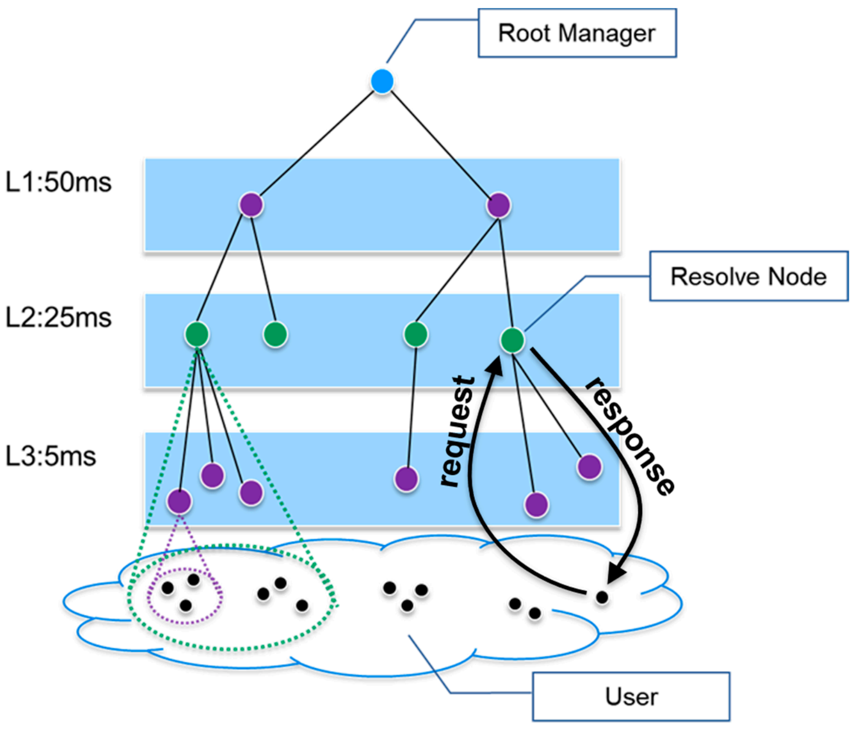


Figure 1. LNMRS implementation based on the tree structure.

RM is responsible for managing all the roots' information and serves as an entry of the forest, so users or new RNs can get the root node information from RM and search for the required RNs in the tree structure starting from the root.

Before using the LNMRS service, users will first determine its resolver at each layer by traversing the forest and performing latency measurements. The nodes that a user chooses are called a resolve node list (RNL). The user chooses which node to query in its RNL according to the differentiated latency requirements of applications. The RNL selection method can be implemented based on any tree-searching method, such as Deep First Search (DFS), Breadth First Search (BFS), or others. Due to the nested relationship between parent and child, the resulting RNL should be a path from the root to a leaf or an internal node of a tree in the system.

As mentioned in Section 3.1, an LNMRS resolver will not forward requests to other resolvers. After receiving the querying request, the RNs search the locally stored name–address bindings and return the results within a bounded latency to achieve deterministic latency resolution.

3.3. System Deployment Considerations

In practice, LNMRS nodes are autonomous and the deployment is done in a distributed manner rather than centralized global engineering. Each LNMRS node is managed autonomously but needs to be organized into one structure. For the deployment of the LNMRS, a protocol for structure construction is needed, but some challenges exist: (1) When constructing the system structure, it is possible that an RN cannot find an upper-layer RN that can satisfy the nested relation with it, for the qualified node has not yet been deployed—just left the system or failed—and therefore, it is unable to construct the nested tree; and (2) since all nodes are autonomous, a node has no knowledge of which existing node satisfies the nested relation with itself when joining. For the new node, how to determine whether a node is nested with itself and choose a qualified parent is another problem. For example, as shown in Figure 2, when nodes try to join the structure, they do not know the existing nodes' resolve area and need a method to determine which L2 node can satisfy the nested relation. However, in the situation shown, no L2 node satisfies the nested relation with nodes; therefore, node s is unable to choose a parent and join the system.

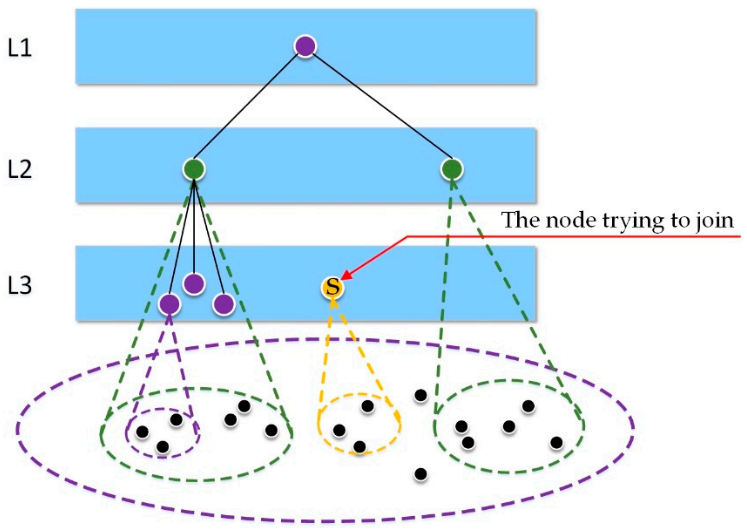


Figure 2. LNMRS implementation based on the tree structure.

For issue (1), a virtual node mechanism is proposed in Section 4.2. Virtual nodes are generated to fill in the place of the missing upper-layer nodes, so that smooth construction of the system structure can be achieved. For issue (2), a nested relation inference method is proposed in Section 4.1. Based on the virtual node mechanism and the inference method, our protocol is described in Section 4.

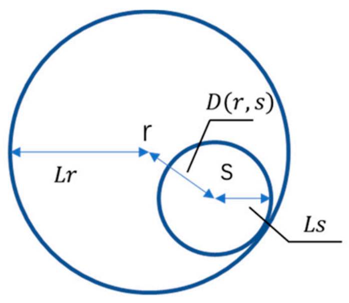
4. Tree Generation and Maintenance

An inference method is designed to determine nested relations of nodes in a distributed manner, and a virtual node mechanism is designed to enhance protocol applicability and ensure the consistency of the tree structure in various cases.

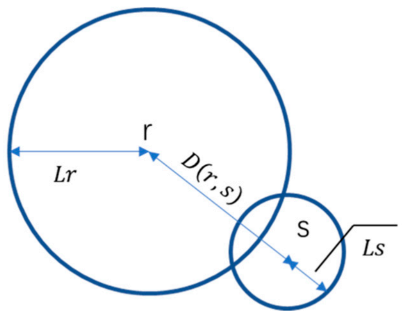
4.1. Nested Relationship Inference Method

The key to ensuring the nested relationship of resolution areas between nodes is correct inference. The network delay space is often modeled in network coordinate research. The most classical network coordinate algorithms such as GNP [33,34] and Vivaldi [35] map the network delay space into Euclidean space and use the coordinates of different network nodes in Euclidean space to estimate network latency. Based on this idea of representing network delay space with Euclidean space, we modeled the resolution area of nodes and designed a method to infer the relation of resolution areas. In this paper, the deterministic latency service area of the resolved nodes was also mapped into Euclidean space, and the nested relationship between two nodes was judged by their relations in the Euclidean space, as shown in Figure 3.

The node service area is mapped into a circle in the Euclidean space, with the node as the center and the guaranteed resolution latency as the radius. The guaranteed latency Lr and Ls of two resolved nodes r and s are known, and the distance $D(r,s)$ between them can be obtained by direct latency measurement, e.g., the average or maximum RTT of Ping in an IP network or the average or maximum respond latency of a designated measurement message during a period of time.



(**a**)



(**b**)

Figure 3. Nested relation inference method. (**a**) Same-tree constraint; (**b**) level constraint. r and s are two different nodes, Lr is the latency guarantee of r, Ls is the latency guarantee of s, $D(r,s)$ is the latency between s and r.

Theorem 1. *Level Inference Constraint (Possible Nested Inference): If*

$$D(r,s) > Ls + Lr, \tag{1}$$

Two circles do not intersect, i.e., there is no overlap in the service area of the two nodes, and there can be no nested relationship, otherwise, there may be full or partial nesting.

Theorem 2. *Same Tree Condition (Definite Nested Inference): If*

$$D(r,s) \leq |Ls - Lr|, \tag{2}$$

One circle is contained by the other, and there is a nested relationship between two nodes.

4.2. Virtual Node Mechanism

To address issue (1) discussed in Section 3.2, this section proposes a virtual node mechanism to enhance the applicability of the tree protocol and enable the smooth construction of nested tree structures. A virtual node is a logical placeholder in the tree. when a node cannot find an upper-layer node that is nested with itself, it will generate a virtual node at the logical location of the missing upper-layer node, ensuring smooth joining of nodes and tree structure integrity. The virtual node only provides the function of sustaining the logical tree and does not provide resolution service. It can be dynamically added or deleted according to the needs of real nodes.

1. Latency to Virtual Nodes

Because of the presence of virtual nodes, nodes also need to consider their latency to virtual nodes when joining the tree. A virtual node only provides structural connectivity logically and does not provide resolution services. It represents a node that can nest its children, so its position in the latency space should be determined by its real children. A virtual node may have multiple children or only virtual children, so the average latency of its real children in the nearest layer is used to represent its latency to other nodes.

2. Nested Relation Inference for Virtual Nodes

Because virtual nodes are merely placeholders, their real children are more important when doing nested relation inference, so their real children are used when inferencing whether a node is nested by the virtual node.

Theorem 3. *Level Inference Constraint for Virtual Node: if a child q of virtual node r exists whose latency to node s satisfies.*

$$D(q,s) > 2Lr + Ls - Lq, \tag{3}$$

Then q and s cannot both be r's children. There is no nested relation between s and r. As shown in (a) of Figure 4.

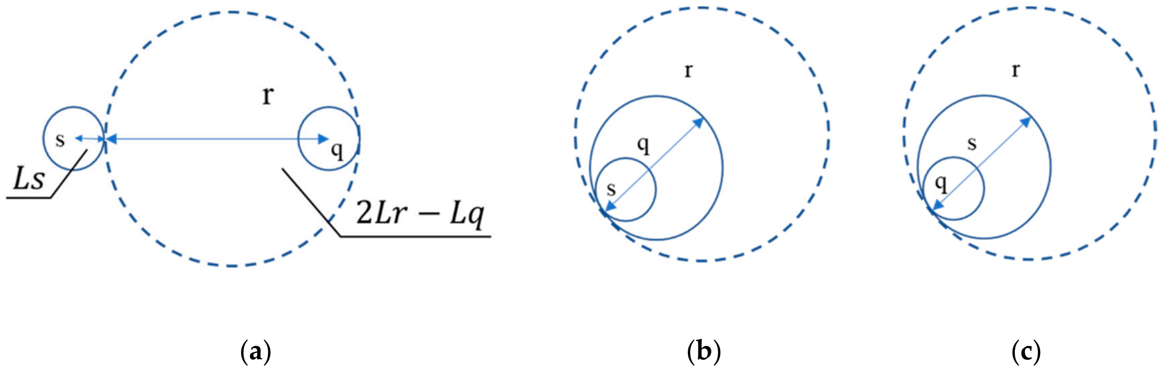


Figure 4. Nested relation inference for virtual node. (**a**) Virtual node level constraint; (**b**) Virtual node same-tree constraint with s on a higher level than q; (**c**) Virtual node same-tree constraint with s on a lower level than q. r is a virtual node, q is a real child of r, s is a real node, Lr is the latency guarantee of r, Lq is the latency guarantee of q.

When s is a possible child of r, its latency to r, $D(r,s)$, should satisfy Equation (1), and q's latency to r, $D(r,q)$, should satisfy Equation (2), so

$$D(r,s) + D(r,q) < 2Lr + Ls - Lq. \tag{4}$$

When s is at the farthest distance from q, these two nodes and r would be in the same line in Euclidean space, so

$$D(s,q) = D(s,r) + D(r,q), \tag{5}$$

Combining Equations (4) and (5), if s and q are both to be the children of r, they should satisfy Theorem 3.

Theorem 4. *Same-Tree Inference Constraint for Virtual Node: If a real child q exists in the nearest layer of the virtual node that satisfies Theorem 2 with s, then s is nested by the virtual node.*

If s is nested with the children of the virtual node, s should also be nested by the virtual node, as shown by (b) and (c) in Figure 4.

4.3. Node Join and Leave

Node Join: Combined with the above mechanism, the node-joining process of LNMRS is designed as shown in Figure 5 and Algorithm 1.

A new node joining the tree will request the list of roots from RM, measure latency to real roots in the root list, and use Theorem 2 to find the root that can nest the new node as its ancestor. If no root satisfies Theorem 2, the new node will select the closest node among the real roots that satisfies Theorem 1 as its ancestor. If there is no real root satisfying the condition, then the virtual root is considered. If neither real nor virtual roots exist that satisfy the nested relationship with the new node, a virtual node is generated to fill in as the missing ancestor. After selecting a root as the ancestor, the new node will continue searching for an ancestor in the next layer among the chosen root's children. This process is repeated until a parent is found one layer above the new node, and the new node finds its position in the tree. If the new node is in the highest layer, it will directly report its information to the RM and join the structure as a root node.

Algorithm 1 Node Joining

```
Input: $n$: the node joining tree; $level$: node join level; $rootManager$: address of Root Manager
1:  $\boldsymbol{parent} \leftarrow rootManager$
2:  $l = 1$
3:  while $l < level$ do
4:      get $realChildren, virtualChildren$ from $\boldsymbol{parent}$
5:      find $possibleAncestors$ in $realChildren$ with Theorem 2
6:      if $\boldsymbol{possibleAncestors}$ exist then
7:          find $definiteAncestors$ in $possibleAncestors$ with Theorem 1
8:          if $\boldsymbol{definiteAncestors}$ exist then
9:              $\boldsymbol{parent} \leftarrow$ chose one of $definiteAncestors$
10:         else
11:             $\boldsymbol{parent} \leftarrow$ closest in $possibleAncestors$
12:         end if
13:     else
14:         find $possibleAncestors$ in $virtualChildren$ with Theorem 3
15:         if $\boldsymbol{possibleAncestors}$ exist then
16:             find $definiteAncestors$ in $possibleAncestors$ with Theorem 4
17:             if $\boldsymbol{definiteAncestors}$ exist then
18:                 $\boldsymbol{parent} \leftarrow$ chose one of $definiteAncestors$
19:             else
20:                 $\boldsymbol{parent} \leftarrow$ closest in $possibleAncestors$
21:             end if
22:         else
23:             generate virtual node $vn$
24:             REGISTER ($vn$, $\boldsymbol{parent}.virtualChildren$)
25:             $\boldsymbol{parent} \leftarrow vn$
26:         end if
27:     end if
28:     $l$++
29: end while
30: REGISTER ($n$, $\boldsymbol{parent}.realChildren$)
```

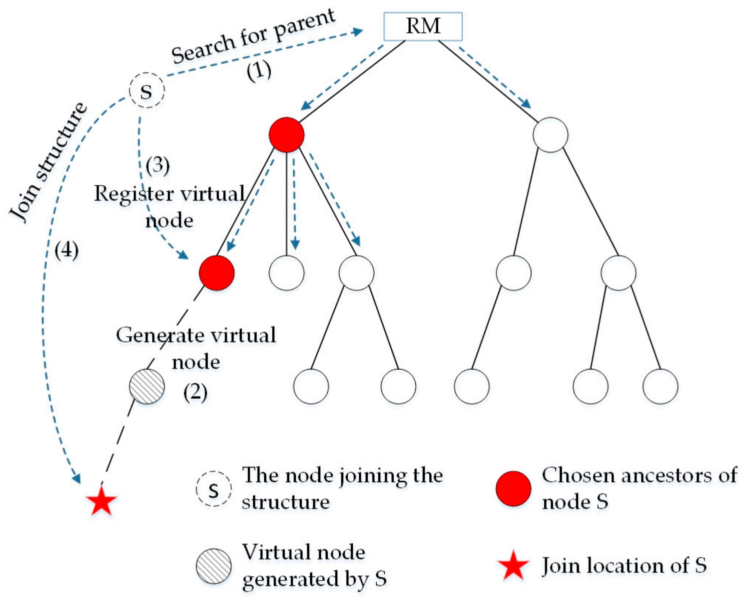


Figure 5. Node-joining process.

Replacement of virtual node: After joining the structure, a node will judge whether it can replace its virtual sibling. A virtual node is only a placeholder and does not have resolution capability. When a real node that can replace the virtual node in the tree appears, the virtual node will no longer be of use and needs to be replaced. Replacement is decided by judging whether it can nest the sub-tree of its virtual sibling.

The replacement process is as shown in steps (1)–(6) in Figure 6. Step (1): Request the list of real children in the nearest layer from all virtual siblings. Step (2): Delete the nodes that do not satisfy Theorem 1, and select the nodes that satisfy Theorem 2 to issue the adoption invitations. If no node satisfies Theorem 2, invitations will be sent to the remaining children in ascending order of latency distance. Step (3): The invited nodes can accept the adoption and change their parents or reject it according to their situation. Steps (5) and (6): When the virtual node finds itself with no children, it will notify its parent and delete itself.

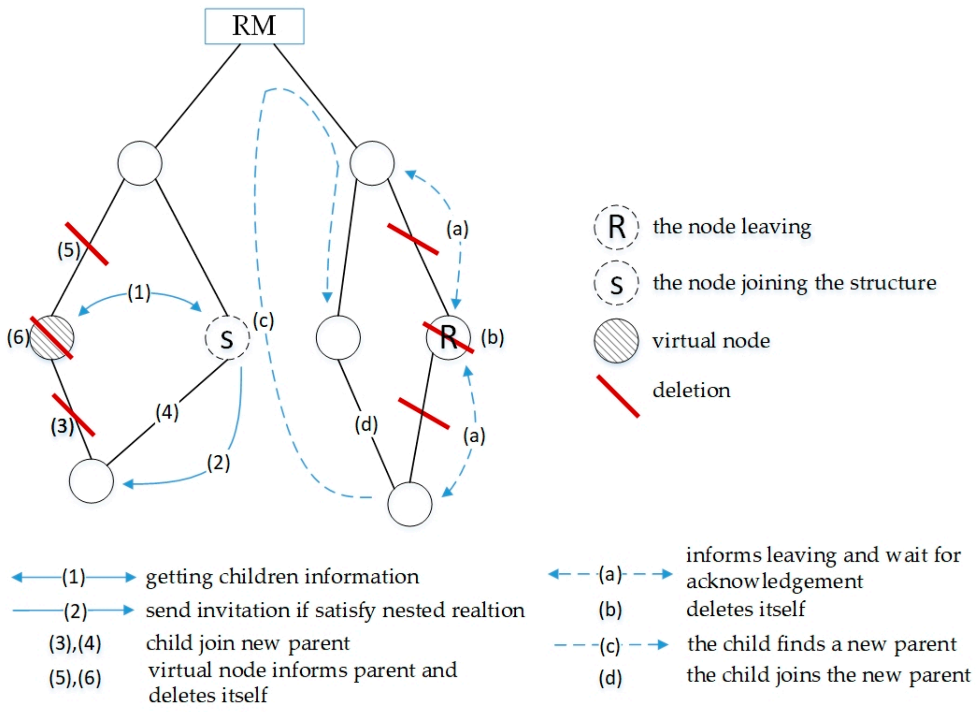

RM
(1)
(2)
(3)
(4)
(5)
(6)
S
(c)
R
(b)
(a)
(a)
(d)
R the node leaving
S the node joining the structure
virtual node
deletion
(1) getting children information
(2) send invitation if satisfy nested realtion
(3),(4) child join new parent
(5),(6) virtual node informs parent and deletes itself
(a) informs leaving and wait for acknowledgement
(b) deletes itself
(c) the child finds a new parent
(d) the child joins the new parent

Figure 6. Virtual node replacement and node leaving.

Node leaving: The node-leaving process is as shown in steps (a)–(d) in Figure 6**.** Step (a): When a node exits, the exiting node sends an exit message to its parent and children. Step (b): The parent/children will remove the exiting node from its children/parent list after receiving the message. Steps (c) and (d): The child node will re-execute the node-joining process after receiving the exit message and find a new parent to join the structure.

4.4. Node Failure

In the LNMRS structure, both machine failures and latency changes between nodes can cause node failures.

Machine failure: When a node goes offline due to failure caused by a program crash, hardware damage, etc., other nodes will not be notified of its leaves, and a node that has gone offline will remain in the structure. Therefore, a heartbeat detection mechanism is used to detect the crashed nodes. As shown in Figure 7, the online node sends heartbeat messages to its parent and children at regular intervals to prove that it is online, and if the heartbeat messages are not received from the node after a certain time limit, the node is considered to be offline. Then, the parent will remove the node from its children list, and its children will re-execute the node-joining process to find a new parent, as shown by steps (a) and (b) in Figure 7.

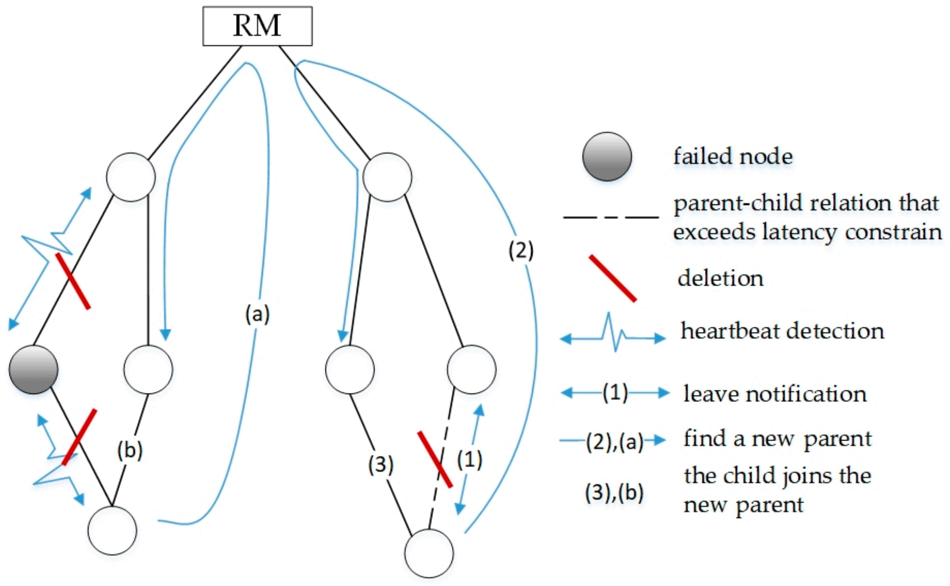

RM
(a)
(b)
(2)
(3)
(1)
failed node
parent-child relation that exceeds latency constrain
deletion
heartbeat detection
(1) leave notification
(2),(a) find a new parent
(3),(b) the child joins the new parent

Figure 7. Example of node failure recovery.

Change of latency between nodes: The nested relationship is the key for the deterministic latency resolution system to provide proper service, while the actual network situation is constantly changing. With a change in routing policy, physical network topology, or network traffic, network latency between two points may change, which often means a change in their deterministic latency resolution area and their nested relation. To keep providing deterministic services to users, the two nodes need to be restructured. A periodic latency-checking mechanism is used to ensure the correctness of the nested relationship between nodes. An example is shown in Figure 7. Each node periodically checks the latency to its parent, and if they no longer satisfy the nested relationship constraint, it will notify the parent and re-execute the node-joining process to find a new parent, as shown by steps (1)–(3) in Figure 7.

5. Evaluation

In order to evaluate the performance of the designed protocol, we developed a simulator to simulate the resolution system. The simulator is based on Icarus [36]. Icarus is a well-designed caching simulator for ICN. We used its cache simulation ability and add some extensions to support the simulation of a resolution system. The simulated underlying network topologies were modeled and analyzed using NetworkX [37] and FNSS [38]. The code is available at https://gitee.com/Phajuer/nmrsim (accessed on 11 July 2022). The simulator reads a network topology and selects the deployment location of the resolution server on the topology, and uses the server nodes to verify the structured protocols.

Considering the different latency requirements of major application scenarios in 5G-IoT, the simulated LNMRS was divided into three layers, denoted as L1, L2, and L3, with one-way deterministic latency requirements of 50 ms, 25 ms, and 5 ms, respectively [39]. The underlying network topologies were generated on scales of 2500, 5000, 7500, 10,000 by BRITE [40] using the top-down approach. For the delay settings between nodes in the topology we referred to the paper by Rajahalme et al. [41], who used the averages over published path latencies and lengths [42], with an average value of 2 ms for inter-domain single-hop delay and 34 ms for inter-domain single-hop delay. For the server placement, the node that could meet the delay requirements with the most network nodes was selected as the resolution node (RN) in a greedy manner at each level until all nodes were in the deterministic service area of at least one RN. The selected nodes generated the system according to the designed protocol and join the system layer by layer from the top. The nodes in the topology other than the RNs were viewed as user access points, and the users chose the longest qualified path in the forest structure as their RNL. A path was considered qualified only when the latency from the user to every real node on the path was under the guaranteed service latency. The experimental environment was Ubuntu 14.04LTS with 16GB RAM, created and run in Python 3.8.

We counted metrics such as service area nested ratio, service coverage, message overhead, etc., to analyze the effectiveness of the node relationship inference method, the service capability of the generated structure, and the scalability of the protocol; we also compared the impact of different generation strategies on the service capability of the structure.

The performance of the LNMRS generated under the proposed protocol was compared with the Ftree system. We distributed 2.5×10^4 contents evenly across the network and set up 5×10^4 name resolution requests as workloads for the experiment. These requests are initiated by the network nodes in a stable distribution, where the content of the requests follows a Zipf [43] distribution with parameter $\lambda = 0.9$, and the request event happens following a Poisson distribution, with a default rate of 1000 requests per second. We assumed the message is forwarded by cache-supporting ICN routers with the same cache sizes and enforces a cache replacement strategy of least recently used (LRU) [44]. To address the cold-start problems of network states, such as caching and NRS registration, we treated 10^4 requests of the workload as warm-up traffic and did not record these request events; the remaining 4×10^4 requests were recorded. The basic parameters of our experiments are summarized in Table 3.

Table 3. Summary of parameters.

Parameters	**Value**
LNMRS service levels	L1, L2, L3
Latency bounds	50 ms, 25 ms, 5 ms
Network topology	BRITE: top-down
Topology scale	2500, 5000, 7500, 10,000
Content number	2.5×10^4
Request number	5×10^4 (10^4 $warm - ups$)
Request rate	1000 req/s
Request distribution	Poisson
Distribution of content source	Random
Distribution of requested content	Zipf with $\lambda = 0.9$
Ftree hash function number	10, 30, 50

5.1. Structure Analysis

The generated structure was analyzed to prove the correctness of the nested relations inference method, the usability of the generated structure, and the overhead of the structure protocol.

5.1.1. Service Area Nested Ratio

The nested relationship means that the service area of the upper-layer node contains that of the lower-layer node and is the key for the system to meet the differentiated needs of users. The service area of the node is represented by the node's user set, and the ratio of the intersection of the two nodes' user sets to the lower-layer node's user set is defined as the nested ratio. The nested ratio indicates how much of the lower-layer node service area can be contained by the upper-layer node and reflects the accuracy of the nested relationship inference method. The higher the nested ratio, the larger the common service area there is between the nodes of the upper and lower layer, and the more users can get differentiated services from both nodes.

In order to evaluate whether the designed nested relation inference method can effectively infer the nested relationship of the service area between two nodes, the service area nested ratio was calculated between all resolved nodes and their ancestor in the forest structure generated under 2500, 5000, 7500, and 10,000 scale topologies in simulation. Among them, for the node pairs that satisfied the same tree condition, the percentage of the nested ratio at 100% is shown in Figure 8. In Figure 6, the nested ratio of all L1 nodes to L2 nodes and L2 nodes to L3 nodes reached 100%, and more than 99% of the node pairs between L1 and L3 had a nested ratio of 100%. The high proportion of 100% nested ratio among node pairs indicates that the definite nested inference using the same tree condition can accurately infer the complete nested relationship of service area between nodes with high probability.

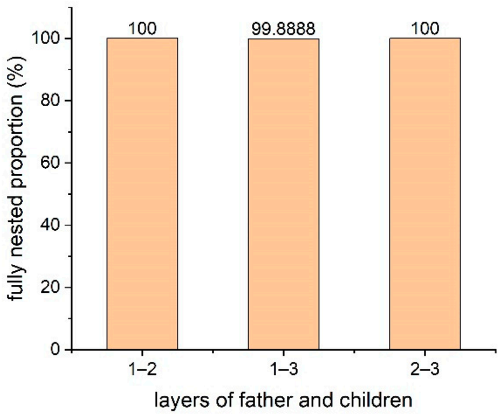


Figure 8. Accuracy for definite nested inference.

For ancestors and children that did not satisfy the same-tree constraint but satisfied the level constraint, the cumulative distribution function of the nested ratio between different layers is shown in Figure 9. For all node pairs satisfying the level constraint, no more than 30% of 1–2 layer and 2–3 layer node pairs had a nested ratio of less than 70%, and only about 7% of 1–3 layers node pairs had nested ratios less than 100%. The high proportion of high nested ratios indicates that under the possibly nested inference, the service area of the upper layer node could still nest most of the service area of the lower layer node with a high probability, and the differentiated needs of most users could be guaranteed even if strict same tree constraint could not be satisfied.

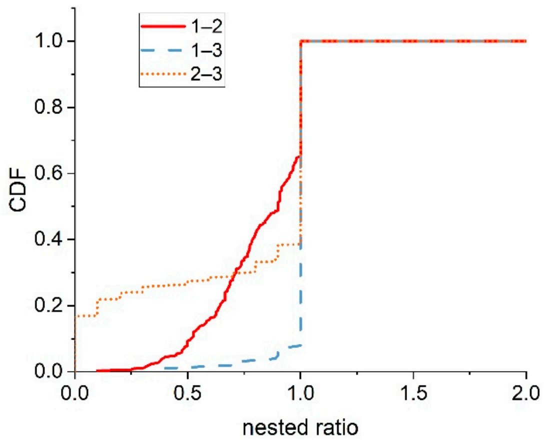


Figure 9. CDF for nested ratio in possible nested inference.

5.1.2. Service Coverage

The percentage of users getting deterministic latency services from a certain layer among all users is defined as the service coverage of that layer. A higher service coverage means that the generated structure can guarantee resolution latency for more users and has better service capability. The nodes in the topology other than the RN are considered the user locations. Each user selects a path in the forest as its RNL. Latency from each node in the RNL to the user should be lower than the latency limits of the corresponding level. The selected path is the longest path among all qualified paths. If a user has a real node of a certain layer in its RNL, it is covered by the resolution service. Service coverage is calculated in simulated networks of different scales.

As shown in Figure 10, among different topological scales, all coverage ratio of the L1 layer was above 99%, the lowest of L2 was 91%, and the lowest of L3 was 86%, which indicates that the structure generated by our protocol can guarantee deterministic resolution service for most users with different latency limits, and has good enough service capability.

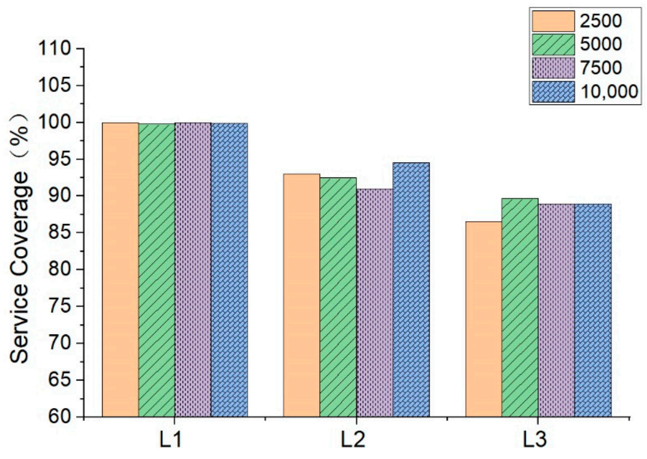


Figure 10. Service coverage.

5.1.3. Message Overhead

(1) Node-joining overhead: The number of messages generated by the 1441 simulated nodes in the 10,000-scale topology is shown in Figure 11. Nodes joined the structure in top-down order. As shown in Figure 11, there was a significant difference in the join message overhead between different layers and some similarities between the same layer. In general, from the top down, the message overhead gradually increased, and the difference in overhead within the layer increased, and this difference mainly came from the specific structures of different sub-trees. The message overhead of newly joined nodes in one layer did not change significantly with the expansion of the system scale, indicating that the protocol has good scalability, which is attributed to the tree-like hierarchy and the nested relationship constraints of parents and children. It does not take many children to fully cover the parent's service area, and the placement of the server is based on user demand, so a parent will not have many children. Therefore, the message overhead for joining the system will not expand drastically.

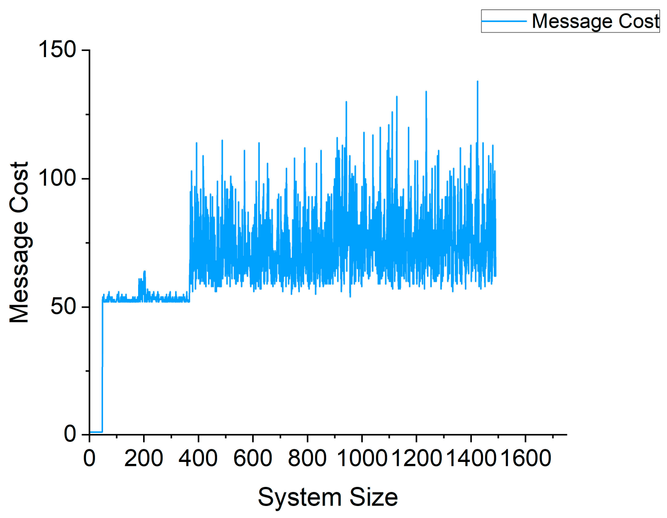


Figure 11. Node-joining overhead at different system size.

(2) Maintenance overhead: The structure maintenance mechanism mainly checks the latency relationships and online status of adjacent nodes (parent and children), and the maintenance overhead is the same for each adjacent node, so the number of relationships that nodes need to maintain can reflect the structure maintenance overhead.

The number of relationships that nodes needed to maintain in L1 and 2 at different system scales (an L3 node has only one relationship with its parent) is shown in Figure 12. The total number of relationships that needed to be maintained in one layer increased with the expansion of the system scale, but the average number of relationships maintained by nodes remained relatively stable, indicating that the structure maintenance pressure of a single node did not rise significantly with the expansion of the system and the maintenance mechanism was rather scalable.

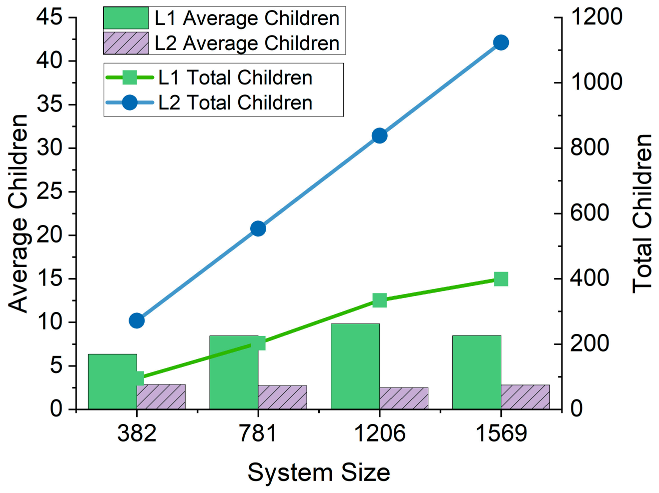


Figure 12. Maintenance overhead at different system sizes.

While ensuring the consistency of the tree structure, the virtual node mechanism also preserves the benefits of the tree structure in terms of scalability. An L1 node that could nest the most L3 nodes was selected as the root in a 10,000-scale topology and the system maintenance overheads of it with and without the virtual node were compared. We used the selected root as the only L1 node and made all the nodes in the topology that could be nested by the selected root join the structure as L3 nodes, then recorded the number of relationships maintained in both cases. The results are shown in Figure 13.

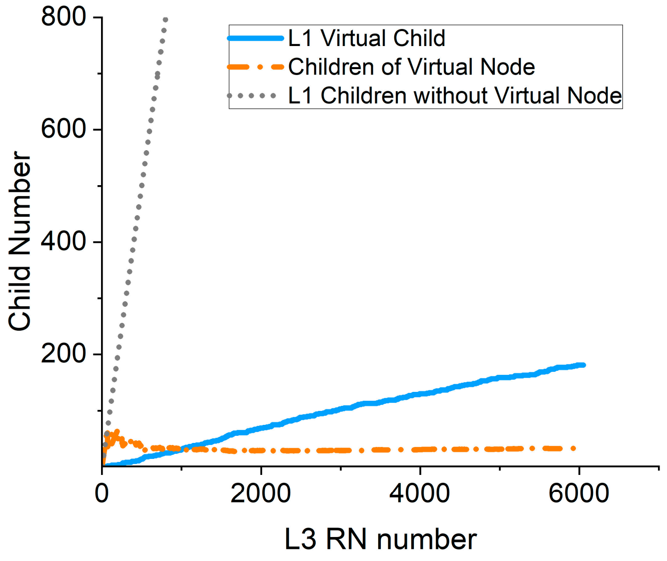


Figure 13. Maintenance overhead with and without virtual node design.

In the case without the virtual node mechanism, the L3 nodes were put directly under the L1 node. It can be seen from Figure 11 that the number of relationships maintained by L1 nodes was significantly reduced and the growth rate slowed down significantly after adopting the virtual node mechanism, whereas the average number of relationships maintained by the virtual nodes was stable. This shows a great scalability advantage in adopting the virtual node mechanism.

5.2. Protocol Analysis

5.2.1. Number of Search Paths

The protocol designed in this paper chooses ancestors layer by layer, and the upper-layer decisions have a great influence on the lower layer. Generally, more search paths can reduce the influence of upper-layer decisions on the final decision and find a better solution than a single search path. The effect of different numbers of search paths on the service coverage of the generated structure was analyzed by letting the RNs search one, two, and three paths to determine their position when joining the system under the 10,000-scale topology. The results are shown in Figure 14. There was no significant difference in service coverage between the systems generated with different search paths, whereas a higher number of search paths implied a higher message overhead. From the experiments, it is clear that the node-joining strategy with a single search path is effective enough to build a system structure with high service coverage.

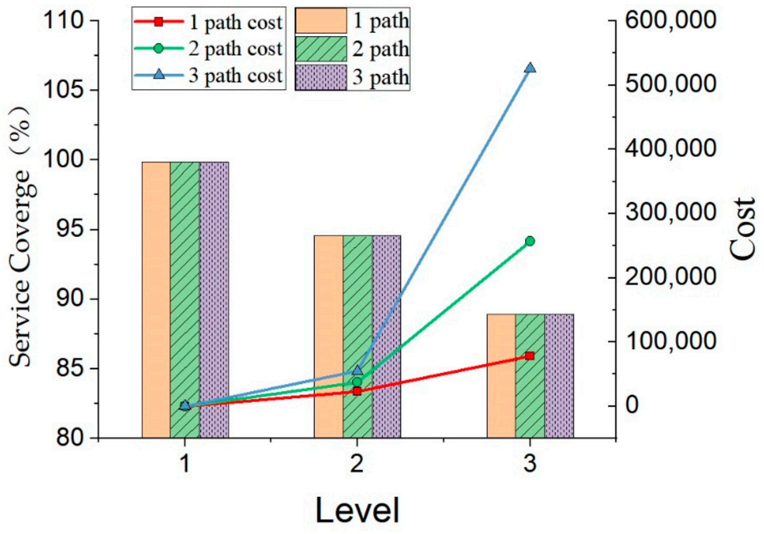


Figure 14. Impact of search path number on structure.

5.2.2. Inference Strategy

Two constraints are proposed for nested relationship inference: same-tree constraint and level constraint. The same-tree constraint requires that the service area of one node fully contain the other in Euclidean space, whereas the level constraint only requires that the service area of two nodes overlap. The strategy that only judges two nodes that satisfy the same-tree constraint (i.e., Theorem 2 in Section 4.1) to be nested is referred to as the strict strategy, and the strategy that uses the level constraint (i.e., Theorem 1 in Section 4.1) is referred to as the relaxed strategy. Since RNs are deployed in a decentralized manner, there is no guaranteed nested relationship between them. A strict relation inference strategy for nodes may ensure a more complete nesting of service areas among RNs, but it may also cause users to miss services on certain layers because it would be hard for RNs to find ancestors, and the accuracy of the relaxed strategy is limited. In order to determine the impact of the two inference strategies on the system structure, the tree-generation process was simulated on 10,000-scale topology under two strategies. One is that the RNs can only choose ancestors from nodes that satisfied the same-tree constraint (strict strategy), and the other is that they can also choose the nodes that satisfy the level constraint when none satisfy the same-tree constraint (relaxed strategy).

The results are shown in Figure 15. Under the two strategies, the service coverage of the system in L1 was not much different, but the service coverage of the relaxed strategy in other layers was much higher than that of the strict strategy. Under the strict condition, a large number of lower layer nodes could not find eligible parents, so a large number of virtual nodes was generated and users were missing services, whereas under the relaxed condition they could find eligible parents, so some of the users could still get deterministic service from lower layers, thus achieving a higher service coverage. The message overhead of latency measurement and maintenance of virtual nodes was much larger than that of real nodes, and the strict strategy was thus more costly, as shown in Figure 15. Based on the results, the relaxed relationship inference strategy has advantages in both system service coverage and overhead.

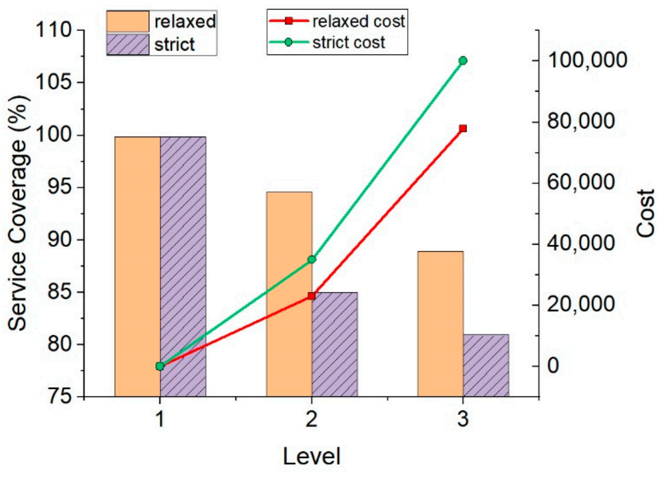


Figure 15. Impact of inference strategy on structure.

5.3. Comparative Study

To evaluate the performance of the LNMRS generated under the proposed protocol, we conducted a comparative study between LNMRS and Ftree with the same set of resolution nodes. The response latency, the success rate of resolution requests, and the request message overhead were compared.

5.3.1. Resolution Response Latency

The resolution response latency is the time between the user sending a request and receiving a response from the NRS. We compared the response latency of LNMRS and Ftree under different topology sizes. The one-way latency requirement was set to 5 ms, 25 ms, and 50 ms for L3, L2, and L1, respectively, in LNMRS, and the required deterministic response round trip times (RTT) were, therefore, 10 ms, 50 ms, and 100 ms. The number of hash functions used in Ftree was set to 10, 30, and 50, respectively. For each hash function number and each level of LNMRS, 4×10^4 requests were simulated. Figure 16 shows the distribution of the response latency for the requests. As can be seen from Figure 16, the overall response latency of LNMRS was much lower than Ftree and had a deterministic guarantee. LNMRS could guarantee the max resolution response latency to be within the set response RTT at each level, and achieved deterministic resolution response. For Ftree, although it could reduce its average response latency by increasing the number of hash functions used, the upper bound of the response latency was hardly affected. Therefore, it could not guarantee deterministic response latency.

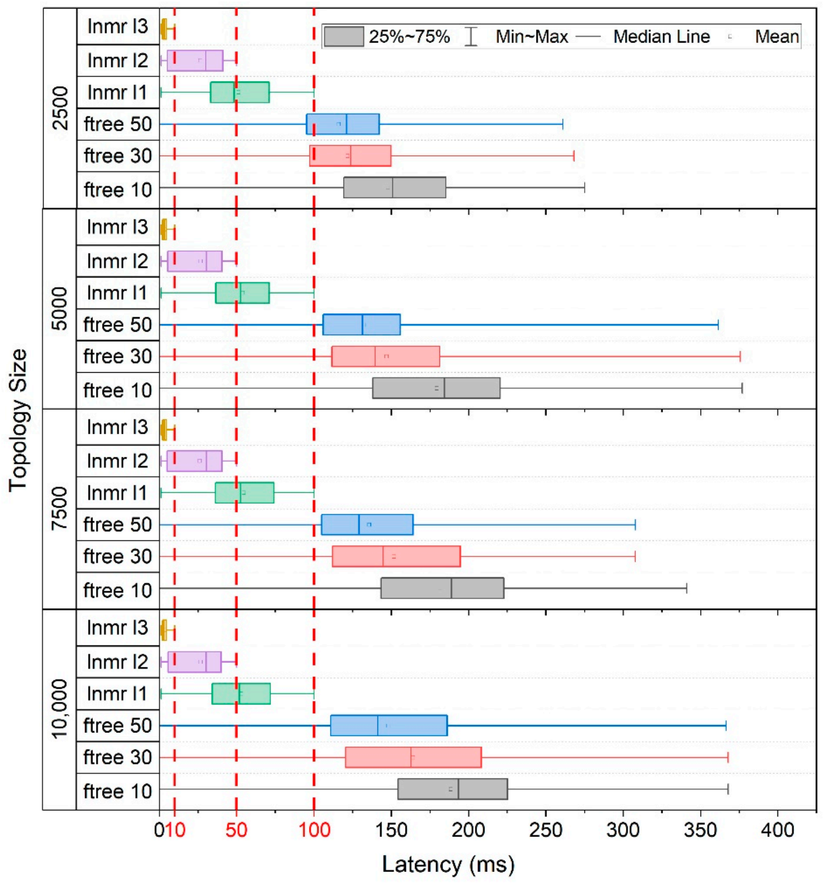


Figure 16. Response latency of LNMRS and Ftree.

5.3.2. Success Resolution Rate

The success resolution rate is the ratio of the number of resolution requests successfully resolved in an NRS to the total number of requests received from users. Successfully "resolved" means that the user received the correct locator corresponding to the requested name from the NRS.

The success resolution rates within different limits of resolution time are shown in Figure 17. They indicate how many requests are successfully resolved within the time limits. The total 4×10^4 request is denoted as S_{Req} and the time limit as T, and the success resolution rate within the time limit T is defined as in Equation (6):

$$SRate_T = \sum_{r \in S_{Req}} I_T(r) / \left| S_{Req} \right| \quad (6)$$

where $I_T(r)$ indicates whether the request r is successfully resolved within T, as shown in Equation (7):

$$Ir = \begin{cases} 1, \, if \; successfully \; resolved \; within \; T \\ 0, if \; resolution \; failed \; or \; exceeds \; T \end{cases} \quad (7)$$

As shown in Figure 17, LNMRS had higher success resolution rates than Ftree within all set latency requirements. It shows that LNMRS is better suited for applications with low and deterministic latency requirements. LNMRS emphasizes timely response over successful retrieval. The successful retrieval is guaranteed by the other part of the resolution scheme (GNMRS) in SEANet, so the success rate shown in Figure 17 is tolerable, but is certainly not ideal. Future works are needed on the improvement of the success rate, which we will discuss in Section 6.

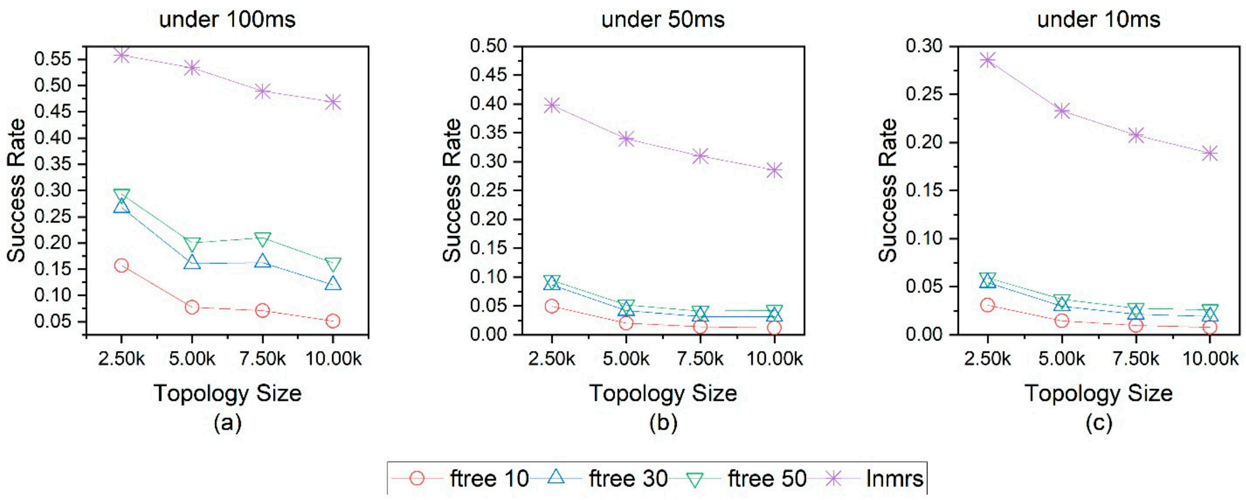


Figure 17. Success resolution rate under different time limits.

5.3.3. Resolution Overhead

The number of query messages generated during the resolution process of the 4×10^4 request is shown in Figure 18. For Ftree, the message number increased with the network size, because the size of the NRS also increased, resulting in longer forwarding paths for requests. LNMRS, on the other hand, did not forward requests to other nodes, and therefore the message overhead during resolution was lower and did not increase with the network size. This means that the LNMRS could effectively reduce the network traffic generated by name resolution.

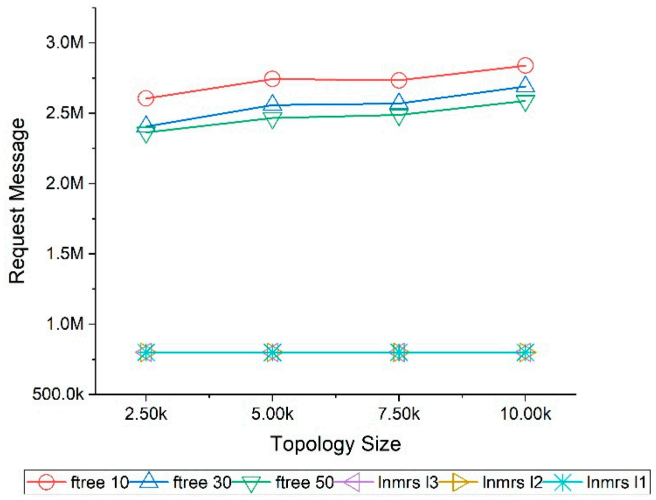


Figure 18. Number of messages generated during the resolution process.

6. Conclusions and Future Work

This paper proposes a tree structure for LNMRS that guarantees deterministic resolution latency and a protocol to generate the structure. The protocol enables the generation and maintenance of a nested tree structure, each layer of which provides users with a different latency guarantee. In order to ensure the generation of correct nested relationships between LNMRS nodes, this paper also proposes a nested relationship inference method based on latency measurement. A virtual node mechanism is designed to ensure the smooth joining and exiting of nodes and enhance protocol applicability.

Based on the simulation results the accuracy of definite nested inference reached more than 99%, and less than 30% of the node pairs satisfying possible nested inference had a service area nested ratio below 70%, which indicates that the proposed inference method can effectively inference node relations. The structure generated under the protocol had good service capability regardless of the network topology scale and could achieve high service coverage at each layer. The node-joining and structure maintenance overheads were relatively stable at each layer, with no significant positive correlation with the system scale, showing good scalability for the system. Comparative experiments with the Ftree showed that LNMRS generated with the proposed protocol could support the deterministic latency name resolution well and that the generated resolution traffic was low.

Our solution embeds latency constraints into the structure of NRS by limiting the forwarding of requests to achieve deterministic latency resolution. Since the forwarding is limited, the successful resolution depends highly on the locally stored mappings of the queried resolver. Although desired information is often published locally for scenarios like auto-driving or industry control, the limited forwarding may still result in a low hit ratio for name mappings. Once local resolution fails, users have to access the global NRS, and the QoS degrades. Thus, for future work, further research on the neighbor mechanisms and the mapping distribution method can be done to improve the success rate of local lookups. A neighbor mechanism can also improve mobility support for the system. For mobile scenarios like auto-driving, the user needs a fast transfer from the old resolver to the new one that serves its current area to maintain the deterministic service. So, a cooperation mechanism for neighbors is important. Another direction of future work worth considering is the optimization of the system structure. The deterministic latency is achieved by exploiting the latency constraints embedded in the system structure; therefore, the structure is important for deterministic resolution. Although the protocol proposed in this paper ensures a structure that correctly embeds the latency constraint, it does not guarantee the structure to be optimal, especially when the member nodes change. Future work on how to optimize the system structure based on member changes can be further studied to improve system usability.

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

Funding: This research was funded by Strategic Leadership Project of Chinese Academy of Sciences: SEANET Technology Standardization Research System Development (Project No. XDC02070100).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to express our gratitude to Jiaqi Li and Yanxia Li for their meaningful support in this work.

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

References

1. Li, R.R. Network 2030 a Blueprint of Technology, Applications and Market Drivers towards the Year 2030 and beyond. 2019. Available online: https://www.itu.int/en/ITU-T/focusgroups/net2030/Documents/White_Paper.pdf (accessed on 26 June 2022).
2. Barakabitze, A.A.; Xiaoheng, T.; Tan, G. A Survey on Naming, Name Resolution and Data Routing in Information Centric Networking (ICN). *IJARCCE* **2014**, *3*, 8322–8330.
3. Xylomenos, G.; Ververidis, C.N.; Siris, V.A.; Fotiou, N.; Polyzos, G.C. A Survey of Information-Centric Networking Research. *IEEE Commun. Surv. Tutor.* **2013**, *16*, 1024–1049. [CrossRef]
4. Gurtov, A.; Komu, M.; Moskowitz, R. Host identity protocol: Identifier/locator split for host mobility and multihoming. *Internet Protoc. J.* **2009**, *12*, 27–32.
5. Kafle, V.P.; Otsuki, H.; Inoue, M. An ID/locator split architecture for future networks. *Commun. Mag. IEEE* **2010**, *48*, 138–144. [CrossRef]
6. Menth, M.; Hartmann, M.; Klein, D. Global Locator, Local Locator, and Identifier Split (GLI-Split). *Future Internet* **2013**, *5*, 67–94. [CrossRef]

7. Ohlman, B. From ID/locator split to ICN. In Proceedings of the 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, 9–12 January 2015; pp. 256–261.
8. 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.
9. Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.F.; Braynard, R.L. Networking named content. *Commun. ACM* **2009**, *55*, 117–124. [CrossRef]
10. Zhang, L.; Afanasyev, A.; Burke, J.; Jacobson, V.; Papadopoulos, C.; Claffy, K.; Lan, W.; Crowley, P.; Zhang, B. Named data networking. *Acm Sigcomm Comput. Commun. Rev.* **2014**, *44*, 66–73. [CrossRef]
11. 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.
12. Fotiou, N.; Nikander, P.; Trossen, D.; Polyzos, G.C. Developing information networking further: From PSIRP to PURSUIT. In Proceedings of the International Conference on Broadband Communications, Networks and Systems, Athens, Greece, 25–27 October 2010; pp. 1–13.
13. Lagutin, D.; Visala, K.; Tarkoma, S. Publish/subscribe for internet: Psirp perspective. In *Towards the Future Internet*; IoS Press: Amsterdam, The Netherlands, 2010; pp. 75–84.
14. Edwall, T.; Tremblay, B. SAIL Project. Available online: https://sail-project.eu/wp-content/uploads/2011/10/SAIL-Newsletter_4.pdf (accessed on 26 July 2022).
15. Venkataramani, A.; Kurose, J.F.; Raychaudhuri, D.; Nagaraja, K.; Mao, M.; Banerjee, S. Mobilityfirst: A mobility-centric and trustworthy internet architecture. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 74–80. [CrossRef]
16. Li, S.; Da Xu, L.; Zhao, S. 5G Internet of Things: A survey. *J. Ind. Inf. Integr.* **2018**, *10*, 1–9. [CrossRef]
17. Agiwal, M.; Saxena, N.; Roy, A. Towards connected living: 5G enabled internet of things (IoT). *IETE Tech. Rev.* **2019**, *36*, 190–202. [CrossRef]
18. Nasrallah, A.; Thyagaturu, A.S.; Alharbi, Z.; Wang, C.; Shao, X.; Reisslein, M.; Elbakoury, H. Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 88–145. [CrossRef]
19. Parvez, I.; Rahmati, A.; Guvenc, I.; Sarwat, A.I.; Huaiyu, D. A Survey on Low Latency Towards 5G: RAN, Core Network and Caching Solutions. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3098–3130. [CrossRef]
20. 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.
21. ITU-T Y. ICN-NMR Framework of Locally Enhanced Name Mapping and Resolution for Information Centric Networking in IMT-2020. Available online: https://www.itu.int/md/T17-SG13-C-1319/ (accessed on 26 July 2022).
22. Souk, S.H.; Ahmed, S.H.; Kim, D. Hierarchical and hash based naming with Compact Trie name management scheme for Vehicular Content Centric Networks. *Comput. Commun.* **2015**, *71*, 73–83.
23. García, G.; Beben, A.; Ramón, F.J.; Maeso, A.; Psaras, I.; Pavlou, G.; Wang, N.; Śliwiński, J.; Spirou, S.; Soursos, S. COMET: Content mediator architecture for content-aware networks. In Proceedings of the 2011 Future Network & Mobile Summit, Warsaw, Poland, 15–17 June 2011; pp. 1–8.
24. Vu, T.; Baid, A.; Zhang, Y.; Nguyen, T.D.; Fukuyama, J.; Martin, R.P.; Raychaudhuri, D. Dmap: A shared hosting scheme for dynamic identifier to locator mappings in the global internet. In Proceedings of the 2012 IEEE 32nd International Conference on Distributed Computing Systems, Macau, China, 18–21 June 2012; pp. 698–707.
25. Sharma, A.; Tie, X.; Uppal, H.; Venkataramani, A.; Westbrook, D.; Yadav, A. A global name service for a highly mobile internetwork. In Proceedings of the 2014 ACM Conference on SIGCOMM, Chicago, IL, USA, 17–22 August 2014; pp. 247–258.
26. Louati, W.; Ben-Ameur, W.; Zeghlache, D. A bottleneck-free tree-based name resolution system for Information-Centric Networking. *Comput. Netw.* **2015**, *91*, 341–355. [CrossRef]
27. Katsaros, K.V.; Fotiou, N.; Vasilakos, X.; Ververidis, C.N.; Tsilopoulos, C.; Xylomenos, G.; Polyzos, G.C. On inter-domain name resolution for information-centric networks. In Proceedings of the International Conference on Research in Networking, Prague, Czechia, 21–25 May 2012; pp. 13–26.
28. Dannewitz, C.; D'Ambrosio, M.; Vercellone, V. Hierarchical DHT-based name resolution for information-centric networks. *Comput. Commun.* **2013**, *36*, 736–749. [CrossRef]
29. D'Ambrosio, M.; Dannewitz, C.; Karl, H.; Vercellone, V. MDHT: A hierarchical name resolution service for information-centric networks. In Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking, Toronto, ON, Canada, 19 August 2011.
30. Fotiou, N.; Katsaros, K.; Vasilakos, X.; Tsilopoulos, C.; Ververidis, C.N.; Xylomenos, G.; Polyzos, G.C. H-Pastry: An Adaptive Multi-Level Overlay Inter-Network. *Athens Univ. Econ. Bus. Athens Greece Tech. Rep.* **2012**. Available online: https://www.eurecom.fr/~{}vasilako/pubs/techRep/2011-MMLAB-TR-003.pdf (accessed on 26 June 2022).
31. Harvey, N.; Jones, M.B.; Saroiu, S.; Theimer, M.; Wolman, A. SkipNet: A Scalable Overlay Network with Practical Locality Properties. In Proceedings of the Usenix Symposium on Internet Technologies & Systems, Seattle, WA, USA, 26–28 March 2003.
32. Li, J.; You, J.; Deng, H. Adjacency-Information-Entropy-Based Cooperative Name Resolution Approach in ICN. *Future Internet* **2022**, *14*, 68. [CrossRef]

33. Ng, T.E.; Zhang, H. Predicting Internet network distance with coordinates-based approaches. In Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, New York, NY, USA, 23–27 June 2002; pp. 170–179.
34. Ng, T.E.; Zhang, H. Towards global network positioning. In Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement, San Francisco, CA, USA, 1–2 November 2001; pp. 25–29.
35. Dabek, F.; Cox, R.; Kaashoek, F.; Morris, R. Vivaldi: A decentralized network coordinate system. *ACM SIGCOMM Comput. Commun. Rev.* **2004**, *34*, 15–26. [CrossRef]
36. Saino, L.; Psaras, I.; Pavlou, G. Icarus: A caching simulator for information centric networking (icn). In Proceedings of the SimuTools, Lisbon, Portugal, 17–19 March 2014; pp. 66–75.
37. Hagberg, A.; Swart, P.; S Chult, D. *Exploring Network Structure, Dynamics, and Function Using NetworkX*; Los Alamos National Lab. (LANL): Los Alamos, NM, USA, 2008.
38. Saino, L.; Cocora, C.; Pavlou, G. A toolchain for simplifying network simulation setup. *SimuTools* **2013**, *13*, 82–91.
39. Schulz, P.; Matthe, M.; Klessig, H.; Simsek, M.; Fettweis, G.; Ansari, J.; Ashraf, S.A.; Almeroth, B.; Voigt, J.; Riedel, I. Latency Critical IoT Applications in 5G: Perspective on the Design of Radio Interface and Network Architecture. *IEEE Commun. Mag.* **2017**, *55*, 70–78. [CrossRef]
40. Medina, A.; Lakhina, A.; Matta, I.; Byers, J. BRITE: An Approach to Universal Topology Generation. In Proceedings of the International Workshop on Modeling, Rzeszów, Poland, 17–20 September 2001; pp. 346–353.
41. Rajahalme, J.; Särelä, M.; Visala, K.; Riihijärvi, J. On name-based inter-domain routing. *Comput. Netw.* **2011**, *55*, 975–986. [CrossRef]
42. Zhang, B.; Ng, T.E.; Nandi, A.; Riedi, R.; Druschel, P.; Wang, G. Measurement based analysis, modeling, and synthesis of the internet delay space. In Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, Rio de Janeriro, Brazil, 25–27 October 2006; pp. 85–98.
43. Breslau, L.; Pei, C.; Li, F.; Phillips, G.; Shenker, S. Web caching and Zipf-like distributions: Evidence and implications. In Proceedings of the Infocom 99 Eighteenth Joint Conference of the IEEE Computer & Communications Societies IEEE, New York, NY, USA, 21–25 March 1999.
44. Fricker, C.; Robert, P.; Roberts, J. A versatile and accurate approximation for LRU cache performance. In Proceedings of the 2012 24th International Teletraffic Congress (ITC 24), Krakow, Poland, 4–7 September 2012.