RESEARCH-ARTICLE

# H-NRF: A High-performance and Evolutive NRF Framework for Large-scale Mobile Core Network

**ZHUORAN MA**, Hunan University, Changsha, Hunan, China

**YANBIAO LI**, University of Chinese Academy of Sciences, Beijing, China

**XIN WANG**, Stony Brook University, Stony Brook, NY, United States

**XIAN YU**, University of Chinese Academy of Sciences, Beijing, China

**XINYI ZHANG**, University of Chinese Academy of Sciences, Beijing, China

**SHIYI LIU**, University of Chinese Academy of Sciences, Beijing, China

**View all**



**Published:** 07 August 2025



APNET 2025: The 9th Asia-Pacific Workshop on Networking
*August 7 - 8, 2025*
*Shang Hai, China*

APNET '25: Proceedings of the 9th Asia-Pacific Workshop on Networking (August 2025)
https://doi.org/10.1145/3735358.3735367
ISBN: 9798400714016

# H-NRF: A High-performance and Evolutive NRF Framework for Large-scale Mobile Core Network

Zhuoran Ma
Hunan University
Changsha, China
Computer Network Information Center, Chinese Academy of Science
Beijing, China
mazhuoran@hnu.edu.cn

Yanbiao Li
Computer Network Information Center, Chinese Academy of Science
Beijing, China
University of Chinese Academy of Sciences
Beijing, China
lybmath@cnic.cn

Xin Wang
SUNY Stony Brook
New York, USA
x.wang@stonybrook.edu

Xian Yu
Computer Network Information Center, Chinese Academy of Science
Beijing, China
University of Chinese Academy of Sciences
Beijing, China
yuxian@cnic.cn

Xinyi Zhang
Computer Network Information Center, Chinese Academy of Science
Beijing, China
University of Chinese Academy of Sciences
Beijing, China
xyzhang@cnic.cn

Shiyi Liu
Computer Network Information Center, Chinese Academy of Science
Beijing, China
University of Chinese Academy of Sciences
Beijing, China
syliu@cnic.cn

Kun Xie
Hunan University
Changsha, China
xiekun@hnu.edu.cn

Gaogang Xie
Computer Network Information Center, Chinese Academy of Science
Beijing, China
University of Chinese Academy of Sciences
Beijing, China
xie@cnic.cn

## Abstract

Fifth-generation mobile communication (5G) leverages Service Based Architecture (SBA) within the 5G Core (5GC) to achieve enhanced flexibility. The Network Repository Function (NRF) in 5GC plays a crucial role in managing Network Functions (NFs) through automated NF registration and discovery, thereby streamlining network configuration. NRF is essential for network automation; however, it can become a bottleneck in large-scale 5GC/6GC deployments. This is due to the increasing scale of distributed networks and NF deployments, which can lead to performance degradation.

To address this challenge, we introduce H-NRF, a lightweight and 3GPP-compliant hierarchical framework specifically designed to improve NRF's service discovery capabilities. At the heart of H-NRF is a novel Hierarchical NF Profile Index. This index facilitates efficient profile matching through a flexible tree structure that supports multiple matching methods and allows for selective attribute indexing. Building upon this index, we develop efficient NF registration and discovery algorithms, achieving constant-level discovery complexity. We have integrated H-NRF into the open-source Free5GC system and rigorously evaluated its performance against standard NRF variants. Our results convincingly demonstrate that H-NRF significantly enhances NF discovery efficiency, maintains consistent performance even as the network scales, and effectively supports the growing demands of 5G and future 6G networks.

## CCS Concepts

• **Networks** → **Mobile networks**; **Network control algorithms**; *Network design principles*; *Intermediate nodes*; *Network experimentation*; Application layer protocols.

## Keywords

Mobile Core, NRF, NF discovery, 5G, 6G

**ACM Reference Format:**
Zhuoran Ma, Yanbiao Li, Xin Wang, Xian Yu, Xinyi Zhang, Shiyi Liu, Kun Xie, and Gaogang Xie. 2025. H-NRF: A High-performance and Evolutive NRF Framework for Large-scale Mobile Core Network. In *9th Asia-Pacific Workshop on Networking (APNET 2025), August 07–08, 2025, Shang Hai, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3735358.3735367



*APNET 2025, Shang Hai, China*

ACM ISBN 979-8-4007-1401-6/25/08
https://doi.org/10.1145/3735358.3735367

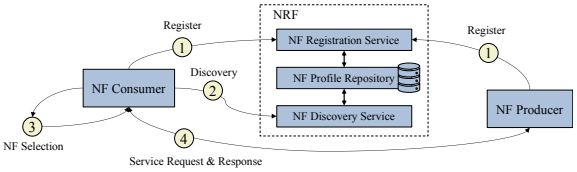


**Figure 1: NRF-based NF Management and NF Discovery**

## 1 Introduction

The fifth generation of mobile communication (5G) technology has achieved widespread adoption across industries[10]. Leveraging network function virtualization (NFV)[12], the 5G Core (5GC) attains significant flexibility and scalability.

The Network Repository Function (NRF)[1] in 5GC provides dynamic service discovery capabilities, enabling NFs to register their profiles and discover suitable NF instances[2, 3]. When a User Equipment (UE) requires core network services, NFs must collaborate through NRF's discovery process. Fig. 1 illustrates the components of NRF and the NF registration and discovery processes.

With the advancement of 6G technology, applications such as satellite networks, Industry 4.0, and subnet collaboration are driving exponential growth in Network Functions (NFs)[5, 6, 15, 17]. In these large-scale deployment environments, while NRF simplifies 5GC implementation and management, it can become a performance bottleneck[4, 9, 14, 16]. This potential bottleneck arises because all NFs must query NRF for service discovery to identify NF producers for subsequent operational tasks.

To address these scalability challenges, we propose H-NRF, a lightweight and 3GPP-compliant hierarchical framework designed to enhance NRF's service capabilities. H-NRF employs a flexible hierarchical tree structure, termed the Hierarchical NF Profile Index, for efficient profile matching. This index supports multiple matching methods simultaneously and selectively indexes NF attributes based on specific requirements.

We further present NF registration and discovery algorithms grounded in the Hierarchical NF Profile Index. Subsequently, we integrate H-NRF into Free5GC, an open-source 5GC system, to validate its effectiveness. Our principal contributions are outlined below:

- We propose H-NRF, a novel framework to accelerate NF discovery while maintaining adherence to the 3GPP standard. To the best of our knowledge, H-NRF is the first study specifically focused on NRF acceleration.
- Based on the Hierarchical NF Profile Index, we have devised and implemented NF registration and discovery algorithms. The discovery algorithm achieves constant-level time complexity, enabling H-NRF to support a greater number of UEs and improve core network scalability.
- We implemented H-NRF on top of Free5GC's NRF and evaluated its performance against several NRF variants. The experimental results demonstrate that H-NRF achieves superior NF discovery efficiency and sustains a constant discovery speed as UE and NF scales increase.

| Message Flow | Operation Description |
|---|---|
| AMF->NRF* | Select AUSF by UE's PLMN, routing indicator, SUPI… |
| AMF->AUSF | Initiate UE authentication |
| AUSF->NRF* | Select UDM by UE's PLMN, routing indicator, SUPI… |
| AUSF->UDM | Get UE's authentication credentials |
| UDM->NRF* | Select UDR by UE's PLMN, routing indicator, SUPI… |
| UDM->UDR | Get UE's authentication credentials in repository |
| AUSF->AMF | Return UE authentication result |

**Table 1: UE Authentication Procedure Workflow**

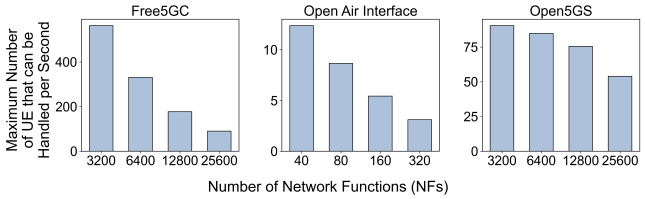


**Figure 2: Number of UEs Handled by NRF Under Different Numbers of NFs**

## 2 Background and Motivation

### 2.1 NF Discovery

In 5GC, each NF is defined by a dedicated NF Profile that characterizes its identity and functionality. This profile contains essential information such as capabilities, service scope for UEs, and operational status. Key attributes include `PLMN` (Public Land Mobile Network) for operator identification, `S-NSSAI` (Single Network Slice Selection Assistance Information) for network slice specification, and `SUPI` (Subscription Permanent Identifier) for UE identification.

In every 5GC procedure, an NF consumer initiates a request to NRF for a suitable NF producer. Due to variations in UE identity, locality, and other attributes, the NF Discovery procedure must be performed multiple times for different UEs to locate the appropriate NF producer. For example, in the UE Authentication workflow, a typical first discovery request might include parameters like: `[target-nf-type:"AUSF", requester-nf-type:"AMF", supi: "imsi-30593000000001", service-names: ["nausf-auth"]]`

NRF conducts a search within its repository of registered NF Profiles using the requested NF parameters as search criteria. The matching phase incorporates several types of matching operations: **Exact match** to verify if an attribute exactly matches the discovery parameter, **Intersection match** to determine if a parameter is present within a list of attributes, **Intersection or null match** to ascertain if a parameter exists in a list or if the list is empty (considered a match), and **Range matching** to evaluate if a value falls within a specified numerical range. Finally, upon successful matching, NRF serializes the matched NF profile into a JSON byte stream and transmits it back to the NF consumer.

### 2.2 Bottleneck in Large-Scale 5GC/6GC

As shown in Tab. 1, UE procedures require multiple NF Discovery requests. We define the NF Discovery latency as $t_d$ (starred items in Tab. 1) and NF Producer latency as $t_p$ (unstarred items). The total 5GC procedure latency is $t_a = t_p + t_d$.

While many studies focus on reducing $t_p$ through distributed deployments, this approach increases the number of NF Producers

and consequently the NF Profiles in NRF. This growth in profiles increases $t_d$, creating a new performance bottleneck. IMT-2030 research [11] suggests 6G networks may contain tens of thousands of NFs. Fig. 2 shows that when registered NFs increase eightfold, existing open-source NRF implementations [7, 8, 13] experience a 43-80% decrease in UE handling capacity, causing many requests to time out.

Our work focuses on accelerating $t_d$. To our knowledge, H-NRF is the first solution specifically addressing NRF acceleration.

## 3 Proposed Method

To accelerate $t_d$ while maintaining compatibility with current 5GC and future 6G standards, we propose the following approach:

**First**, a Hierarchical NF Profile Index is utilized to distribute NF Profiles across leaf nodes. This distribution strategy reduces the time complexity of matching operations and enhances NRF's capacity to support a greater number of UEs. **Second**, we categorize NF discovery matching methods into three fundamental types. This categorization allows different attributes to be strategically positioned within the Index for optimized performance. **Third**, by storing only NF Instance IDs within the Index itself, we decouple NRF from the underlying NF Profile repository. This decoupling enhances flexibility and simplifies deployment.

### 3.1 Overview of H-NRF Architecture

Fig. 3 presents the architecture of H-NRF, which enhances the base NRF functionality with a Hierarchical NF Profile Index and an independent storage engine. The operational workflow is as follows:

**NF Registration:** Step 1: Upon receiving a registration request, H-NRF extracts the NF Profile to obtain NF attributes and the NF instance ID. It then locates or creates a leaf node within the Index and appends the NF instance ID to this leaf node.

Step 2: H-NRF saves the complete NF Profile content into the storage engine, using the NF ID as an index for efficient retrieval.

**NF Discovery:** Step 3: In response to a discovery request, H-NRF queries the Hierarchical NF Profile Index using the NF Discovery parameters. This process retrieves a set of associated NF instance IDs that match the query.

Step 4: Utilizing the retrieved NF instance IDs, H-NRF fetches the corresponding NF Profile contents from the storage engine. For any NF Discovery parameters not covered by the index, NRF performs supplementary matching operations on the profiles retrieved from the storage engine.

### 3.2 Hierarchical NF Profile Index

This section initially categorizes NF discovery matching methods into three distinct types. Based on these categories, we design a tree-structured Hierarchical NF Profile Index. In this index, each layer is specifically designed to handle a particular matching method. This layered architecture enables the strategic placement of attributes to optimize and accelerate NF discovery. Following this, we detail the attribute placement strategy within the Index.

*3.2.1 Analysis of NF Discovery Match Methods.* To effectively design the Hierarchical NF Profile Index, we first analyzed the various NF discovery matching methods and categorized them into two fundamental types: exact matching and null-tolerant matching.

Matching methods specified in standards typically involve checking for equality between two elements, determining if an element or a list of elements intersects with another list, assessing if a list is empty, or iterating through objects in a list to verify if an attribute matches a given element. By generalizing lists and objects, these diverse methods can be simplified to two core operations: exact matching (requiring precise equality between elements) and null-tolerant matching (where a null value or absence of an attribute is also considered a valid match).

Range matching and regular expression (regex) matching also align with these categories. Instead of testing for direct equality, they verify if elements satisfy specified range criteria or match predefined patterns, respectively.

*3.2.2 Components of Hierarchical NF Profile Index.* To design the Hierarchical NF Profile Index, we define its key components. First, to accommodate different equality checking methods for various attributes, we define $S_A$ as the Data Structure used for different NF attributes. For example, a hash map can be employed if the attribute is a string, while a trie or interval tree can be used if the attribute involves numerous ranges.

Second, we group multiple $S_A$ data structures within an Attribute Node to minimize query time. This node is responsible for handling several attributes. By aggregating multiple instances of the same $S_A$ (e.g., multiple hash maps), we can enable parallel query processing.

Third, to support various matching methods, we introduce a Composite Node. This node manages a set of Attribute Nodes that share identical matching methods. Each $S_A$ structure points to either a Composite Node or directly to a leaf node.

A leaf node is the terminal node within the Index. It stores the NF Instance IDs of the NF profiles that match the combined criteria of the path leading to this leaf node from the root.

*3.2.3 Structure of Hierarchical NF Profile Index.* Based on the two fundamental matching methods identified, we have designed a Hierarchical NF Profile Index with a three-layered structure, referred to as the Hierarchical Matching Tree.

As depicted in Fig. 3, the architecture comprises three layers: the Exact Matching Layer (first layer), the Null-Tolerant Layer (second layer), and the NF-Specific Layer (third layer). The NF-Specific Layer is intentionally separated from the initial two layers rather than being integrated with them. This separation is motivated by the fact that each NF Profile typically possesses only one instance of these NF-specific attributes. Incorporating NF-Specific attributes into the Exact Matching Layer would lead to redundancy, particularly with attributes like NF Type. This redundancy would result in numerous duplicated nodes and a consequent waste of storage space.

*3.2.4 Exact Matching Layer.* We designate two attribute sets for the exact matching layer: $e_m$, representing mandatory attributes in discovery parameters, and $e_o$, corresponding to optional discovery parameters.

For optional attributes, we define a set $E_o = \{e_{o1}, e_{o2}, \ldots, e_{oi}\}$ and utilize its power set $\mathcal{P}(E_o)$ to manage all possible combinations of optional parameters.

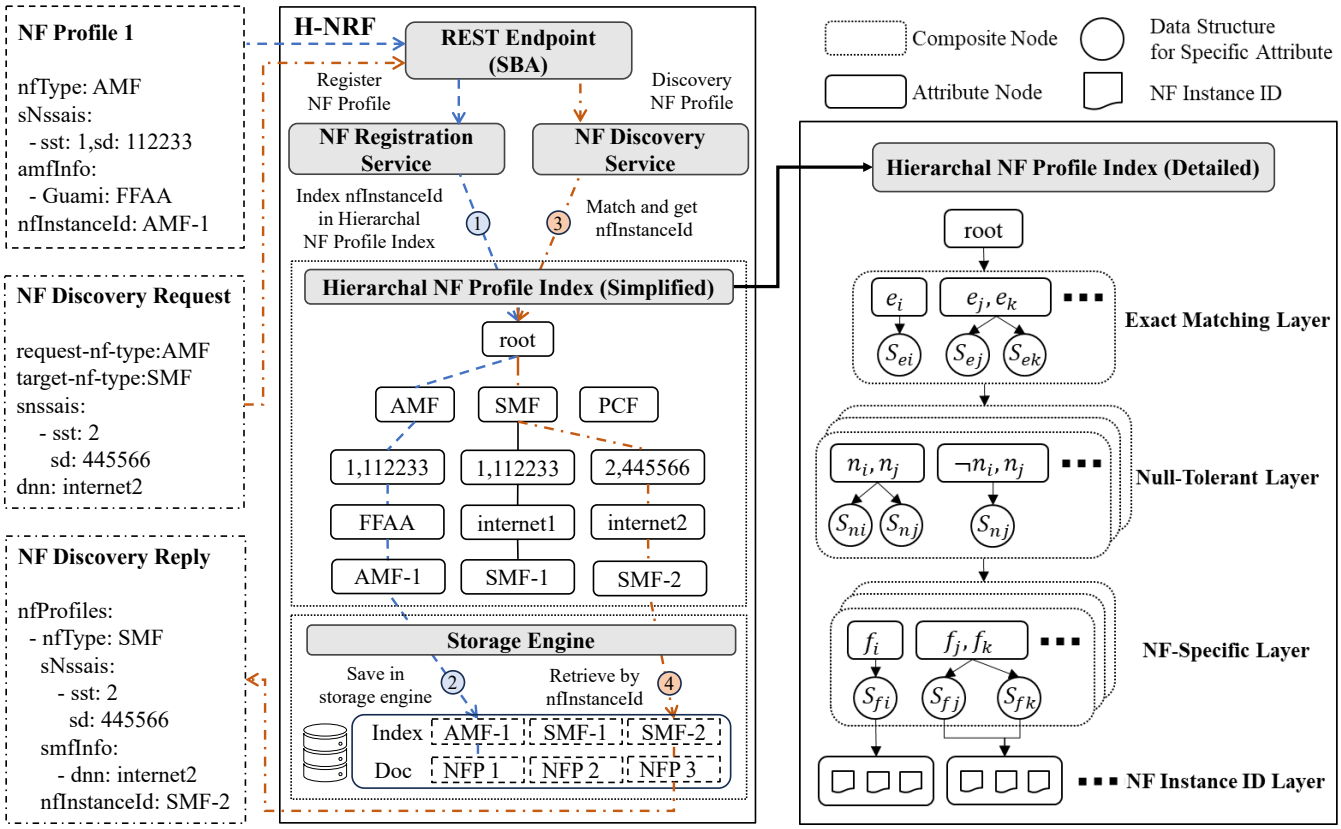


**Figure 3: H-NRF Architecture. The top-left corner shows an example of an NF Profile, which can be registered into H-NRF through the hierarchical NF Profile and stored in the storage engine. The left example of an NF Discovery Request can retrieve the corresponding NF Profiles based on the parameters. The right side illustrates the detailed structure of the Hierarchical NF Profile Index.**

For mandatory attributes, we define a set $E_m = \{e_{m1}, e_{m2}, \ldots, e_{mj}\}$. By employing $\mathcal{P}^*(E_m, E_o)$, we append $E_o$ to each subset of $E_m$ to derive the keys for attribute nodes in the Exact Matching Layer.

For example, given $i = 2, j = 1$, we have $E_o = \{e_{o1}, e_{o2}\}$, the power set is: $\mathcal{P}(E_o) = \{\emptyset, \{e_{o1}\}, \{e_{o2}\}, \{e_{o1}, e_{o2}\}\}$ Then, we can obtain derived attribute node keys: $\mathcal{P}^*(E_o, E_m) = \{\{e_{m1}\}, \{e_{m1}, e_{o1}\}, \{e_{m1}, e_{o2}\}, \{e_{m1}, e_{o1}, e_{o2}\}\}$

*3.2.5 Null-Tolerant Layer.* For attributes employing null-tolerant matching, we define $\neg n$ to represent the attribute node storing NF Profiles where attribute $n$ is absent or empty, thus $n_{\text{state}} \in \{n, \neg n\}$.

Let $p_n$ be the value of a discovery parameter and $\neg p_n$ denote the absence of the parameter in the discovery request. Consequently, $p_{\text{state}} \in \{p_n, \neg p_n\}$.

The null-tolerant match function (NTMatch) is defined as:

$$\text{NTMatch}(n_{\text{state}}, p_{\text{state}}) = \begin{cases} \text{False} & \text{if } (n_{\text{state}} = n \wedge p_{\text{state}} = p_n \\ & \wedge n \neq p_n) \\ \text{True} & \text{otherwise} \end{cases}$$

Due to the consideration of null states, we utilize a Cartesian product instead of a power set to avoid generating invalid combinations. For each null-tolerant attribute $n_i$, we define $N_i = \{n_i, \neg n_i\}$.

The set of attributes in the null-tolerant layer is determined by the Cartesian product $\prod_{i=1}^{n} N_i$.

For example, with $n = 2$:

$$\prod_{i=1}^{2} N_i = N_1 \times N_2 = \{\{n_1, n_2\}, \{\neg n_1, n_2\}, \{n_1, \neg n_2\}, \{\neg n_1, \neg n_2\}\}$$

*3.2.6 NF-Specific Layer.* The attributes within the NF-Specific Layer are categorized into two types: exact match attributes $F_e$ and null-tolerant match attributes $F_n$. The final result from this layer is the intersection of NF instance IDs that satisfy both exact and null-tolerant matching criteria.

## 3.3 NF Registration and Discovery within the Index

This section provides a detailed description of how NF Instance IDs are stored and retrieved within the Hierarchical NF Profile Index, consistent with the overview presented in Sec. 3.1.

**Algorithm 1** Hierarchical Matching Tree Insertion Process

**Require:** NF Profile with attributes $A = \{a_n \mid a_n \in (E_m \cup E_o \cup N \cup F_e \cup F_n)\}$
**for all** exact attribute combinations $C \subseteq \mathcal{P}^*(E_o, E_m)$ **do**
  **append** $get\_C\_nodes(EC\_node, C)$ **to** $NTC\_nodes$
**end for**
**for all** null-tolerant node $node \in NTC\_nodes$ **do**
  **for all** null-tolerant attribute combinations $C \subseteq O(N)$ **do**
    **append** $get\_C\_nodes(node, C)$ **to** $NFS\_nodes$
  **end for**
**end for**
**for all** $node \in NTC\_nodes$ **do**
  // Perform the above operations separately for $F_e$ and $F_n$ attributes
**end for**
**for all** NF Instance Node $node \in leaf\_nodes$ **do**
  $save\_id(node, nfInstanceId)$
**end for**

**Algorithm 2** Hierarchical Matching Tree Retrieval Process

**Require:** NF Discovery request with Parameters $P = \{p_n \mid p_n \in (E_m \cup E_o \cup N \cup F_e \cup F_n)\}$
**Ensure:** All NF Profile IDs that meet the NF discovery request
$match\_key \leftarrow \{p_n \in (E_m \cup E_o)\}$
$NTC\_node \leftarrow Match(EC\_node, match\_key)$
$match\_key \leftarrow \{p_n \in N\}$
$NFSC\_node \leftarrow NTMatch(EC\_node, match\_key)$
$ids \leftarrow []$
$match\_key_e \leftarrow \{p_n \in F_e\}$
$match\_key_n \leftarrow \{p_n \in F_n\}$
$id_e \leftarrow Match(\{F_e | F_e \in NFSC\_node\}, match\_key_e)$
$id_n \leftarrow NTMatch(\{F_n | F_n \in NFSC\_node\}, match\_key_n)$
**append** $(id_e \wedge id_n)$ **to** $ids$
**return** $ids$

For conciseness in pseudocode, we use notations [EC/NTC/NFSC] _node to represent Exact Matching/Null-Tolerant/NF-Specific Layer Composite Node, respectively.

*3.3.1 NF Registration.* Algorithm 1 outlines the NF Profile insertion process into the Hierarchical Matching Tree during NF registration. For each NF Profile, we need to generate all relevant attribute combinations, specifically subsets of $P^*(E_o, E_m)$ within the EC_node's attribute nodes. These attribute combinations are then used as keys to retrieve the corresponding NTC_nodes.

In the Null-Tolerant layer processing, we define the operation $O(N)$ to generate a family of attribute sets that contain $N$ along with all combinations of its negations. Formally, it is defined as: $O(N) = \bigcup_{k=0}^{|N|} \{N \setminus \{n_i\} \cup \{\neg n_i\} \mid n_i \in N\}$. For instance, given $N = \{n_1, n_2, \neg n_3\}$, $O(N) = \{N, \{\neg n_1, n_2, \neg n_3\}, \{n_1, \neg n_2, \neg n_3\}, \{\neg n_1, \neg n_2, \neg n_3\}\}$. These generated attribute combinations are utilized to retrieve NFS_nodes in the subsequent step.

For the NF-Specific layer, we apply the same process separately to the sets of exact match attributes $F_e$ and null-tolerant match attributes $F_n$ to obtain leaf nodes and subsequently store the NF instance IDs within these nodes.

Algorithm 2 illustrates the retrieval process for NF Profiles during NF discovery. By using the appropriate combinations of discovery parameter types for each layer, we can effectively navigate to the composite node of the subsequent layer. When processing the NF-Specific layer, we obtain the intersection of the results from exact matching and null-tolerant matching to collect all NF instance IDs that satisfy the discovery request.

*3.3.2 Time Complexity.* For NF registration, the time complexity of Algorithm 1 is determined to be $O(2^{(E_m+E_o+N+F_e+F_n)})$, which simplifies to $O(2^a)$, where $a$ is the total number of attribute types incorporated within the hierarchical matching tree. While this shows exponential time complexity, it remains acceptable because the frequency of NF registrations is significantly lower compared to discovery operations.

For NF discovery using Algorithm 2, the time complexity is $O(n \times o)$, where $n$ is the number of NF discovery parameters and $o$ is the average time complexity for matching each parameter.

Given that most matching conditions are based on equality, hash maps are the commonly used data structure for attribute nodes. By consolidating different hash maps within each attribute node and considering that hash map lookups have an average time complexity of O(1), we can reduce the overall time complexity to $O(n \times o_{specialMatch})$. Here, $o_{specialMatch}$ represents the time complexity of specific matching operations, such as range matching or regex matching, which might be required for certain parameters.

## 4 Evaluation and Analysis

### 4.1 Experimental Setup

We implemented H-NRF based on the Free5GC NRF, employing the following attributes: `nfInstanceId`, `nfType`, `locality`, `plmn`, `allowedNfType`, `serviceName`, `snssais`, `supi`, `guami`, and `dnn`. Specifically, `supi` was implemented using a trie data structure, whereas other attributes utilized hash maps.

We assessed the performance of H-NRF with two different storage engines: MongoDB (H-NRF-M) and Redis (H-NRF-R), and compared it against the original Free5GC NRF (O-NRF) and its variants that incorporate MongoDB indexes (O-NRF-ID, O-NRF-ALL). MongoDB index is a feature to accelerate queries on specific attributes. H-NRF-M requires an index for NF Instance ID, thus O-NRF-ID was used as a control. O-NRF-ALL is O-NRF with every attribute indexed, serving as a performance baseline. Experiments were conducted on virtual machines equipped with AMD EPYC 7662@2.0GHz processors. The SBI Emulator was used to simulate NF interactions and accurately measure NF discovery time ($t_d$).

### 4.2 Metrics

Performance was evaluated based on the following metrics:

- **Number of UEs Served:** The maximum number of concurrent UEs that NRF can effectively handle in a steady state.

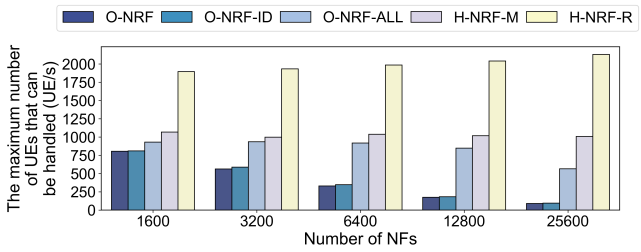


**Figure 4: Number of UEs Handled by O-NRF and H-NRF Under Different Numbers of NFs**

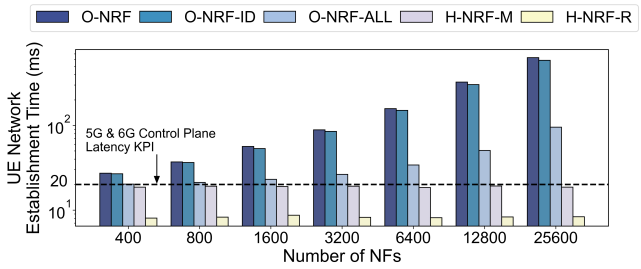


**Figure 5: UE Network Establishment Time**

- **UE Network Establishment Time:** $t_d$ represents the time required for a single UE to establish a network connection. This includes the NF discovery time during UE registration and PDU Session establishment.
- **NF Registration Time:** The duration needed for an NF to complete the registration process with NRF.
- **Memory Overhead:** The memory footprint of both the NRF service and its associated storage engine.

## 4.3 Experimental Results

*4.3.1 Scalability.* Fig. 4 demonstrates that H-NRF maintains a consistent number of served UEs irrespective of the increasing NF count. In contrast, O-NRF shows a significant performance degradation as the number of NFs increases. H-NRF-R supports approximately 90% more UEs than H-NRF-M. Although O-NRF-ALL shows improvement over O-NRF, it still exhibits performance degradation with a higher number of NFs.

*4.3.2 UE Network Establishment Time.* Fig. 5 illustrates that H-NRF consistently achieves lower UE network establishment times compared to O-NRF across varying numbers of NFs. The network establishment time for O-NRF increases exponentially with the NF count, whereas H-NRF maintains a relatively constant time. H-NRF-R is approximately four times faster than H-NRF-M in network establishment time. O-NRF-ALL shows faster establishment times than O-NRF but still performs slower than H-NRF.

*4.3.3 NF Registration Time.* Fig. 6 presents the NF registration time for various NRF implementations under different NF counts. NRF implementations based on MongoDB (O-NRF, O-NRF-ID, O-NRF-ALL, H-NRF-M) exhibit comparable NF registration times without

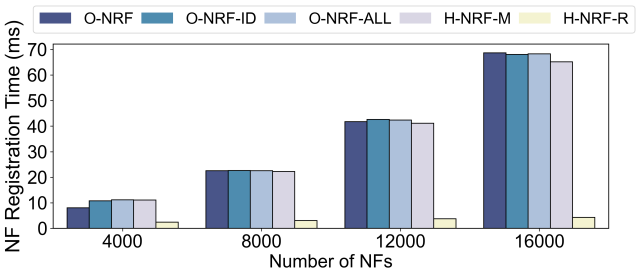


**Figure 6: NF Registration Time**

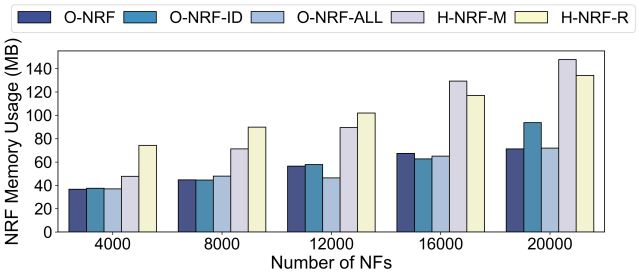


**Figure 7: NRF Service Memory Overhead**

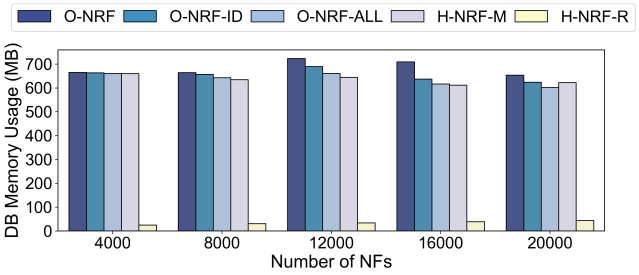


**Figure 8: Database Memory Overhead**

significant variance across different NF numbers. However, H-NRF-R demonstrates a considerably lower average NF registration time and a less pronounced increase in registration time as the number of NFs grows, in comparison to MongoDB-based NRFs.

*4.3.4 Memory Overhead.* H-NRF shows a higher memory usage in the NRF service itself (Fig. 7) due to the overhead of the added index structures. H-NRF-M exhibits a higher growth rate in memory usage, likely due to encoding overhead associated with MongoDB. However, employing Redis-based storage in H-NRF-R significantly reduces database memory consumption (Fig. 8). The increased memory usage in the NRF service is considered acceptable in typical server deployment scenarios given the performance gains achieved.

## 5 Conclusion and Future Work

This paper introduces the H-NRF architecture, which employs a hierarchical data structure combined with differentiated matching methods to reduce NF discovery time. This approach effectively transitions the scaling of NF discovery time from linear with the

number of NFs to a constant level. Future research directions include refining the 5G NF discovery procedure to better meet the performance demands of the 5GC control plane in industrial applications, and further optimization of the proposed H-NRF framework.

## Acknowledgments

We would like to thank the anonymous reviewers for their insightful and constructive suggestions. This work was partially supported by the National Natural Science Foundation of China (Grant Nos. 62025201 and 62321166652). The corresponding authors of this paper are Yanbiao Li and Kun Xie.

## References

[1] 3GPP. 2024. *5G System; Network function repository services; Stage 3.* Technical Specification (TS) 29.510. 3rd Generation Partnership Project (3GPP). http://www.3gpp.org/DynaReport/29510.htm

[2] 3GPP. 2024. *Procedures for the 5G System (5GS).* Technical Specification (TS) 23.502. 3rd Generation Partnership Project (3GPP). http://www.3gpp.org/DynaReport/23502.htm

[3] 3GPP. 2024. *System architecture for the 5G System (5GS).* Technical Specification (TS) 23.501. 3rd Generation Partnership Project (3GPP). http://www.3gpp.org/DynaReport/23501.htm

[4] Carlos Hernan Tobar Arteaga, Armando Ordonez, and Oscar Mauricio Caicedo Rendon. 2020. Scalability and Performance Analysis in 5G Core Network Slicing. *IEEE Access* 8 (2020), 142086–142100. https://doi.org/10.1109/ACCESS.2020.3013597

[5] James Blackman. 2024. *Private 5G in Industry 4.0 Report.Pdf.* Technical Report. RCR Wireless.

[6] Shuping Dang, Osama Amin, Basem Shihada, and Mohamed-Slim Alouini. 2020. What Should 6G Be? *Nature Electronics* 3, 1 (Jan. 2020), 20–29. https://doi.org/10.1038/s41928-019-0355-6

[7] EURECOM. 2024. OpenAirInterface - 5G Software Alliance for Democratising Wireless Innovation.

[8] free5GC Team. 2024. free5GC - Open-Source Project for 5th Generation (5G) Mobile Core Networks.

[9] Endri Goshi, Michael Jarschel, Rastin Pries, Mu He, and Wolfgang Kellerer. 2021. Investigating Inter-NF Dependencies in Cloud-Native 5G Core Networks. In *2021 17th International Conference on Network and Service Management (CNSM).* IEEE, Izmir, Turkey, 370–374. https://doi.org/10.23919/CNSM52442.2021.9615565

[10] IBM. 2024. What Is 5G? | IBM. https://www.ibm.com/topics/5g.

[11] IMT-2030. 2022. *6G Distributed Network Technology Application Scenarios and Requirements Research Report.* Technical Report. IMT-2030. Chinese Version.

[12] Ultan Mulligan. 2024. Network Functions Virtualisation (NFV). https://www.etsi.org/technologies/nfv.

[13] Open5GS Team. 2024. Open5GS - Open Source Implementation for 5G Core and EPC.

[14] Quang Tung Thai and Namseok Ko. 2022. Towards Realizing a Cloud-Native B5G Mobile Core Architecture. In *2022 13th International Conference on Information and Communication Technology Convergence (ICTC).* IEEE, Jeju Island, Korea, Republic of, 1018–1023. https://doi.org/10.1109/ICTC55196.2022.9952761

[15] Cheng-Xiang Wang, Xiaohu You, Xiqi Gao, Xiuming Zhu, Zixin Li, Chuan Zhang, Haiming Wang, Yongming Huang, Yunfei Chen, Harald Haas, John S. Thompson, Erik G. Larsson, Marco Di Renzo, Wen Tong, Peiying Zhu, Xuemin Shen, H. Vincent Poor, and Lajos Hanzo. 2023. On the Road to 6G: Visions, Requirements, Key Technologies, and Testbeds. *IEEE Communications Surveys & Tutorials* 25, 2 (2023), 905–974. https://doi.org/10.1109/COMST.2023.3249835

[16] Guang Yang. 2021. *5G Signaling and Control Plane Traffic Depends on Service Communications Proxy (SCP).* Technical Report. Strategy Analytics.

[17] Volker Ziegler, Harish Viswanathan, Hannu Flinck, Marco Hoffmann, Vilho Raisanen, and Kimmo Hatonen. 2020. 6G Architecture to Connect the Worlds. *IEEE Access* 8 (2020), 173508–173520. https://doi.org/10.1109/ACCESS.2020.3025032