

HyPR: A High-Performance Policy Registry for Scalable User Equipment Policy Management in 5G Core

Thanh Son Pham, Tien-Quang Nguyen, Quan Huan Vu, Duc Hai Nguyen

Payment Platform Development Center, Viettel High Technology Industries Corporation, Viettel Group
(sonpt26, quangnt22, huanvq1, haind13)@viettel.com.vn

Abstract—The advent of 5G Core networks has introduced network slicing and open APIs that enable diverse business-to-business (B2B) service opportunities. At the center of these capabilities lies the management of User Equipment Route Selection Policies (URSP), which allow fine-grained traffic control at the device level to guarantee Quality of Service (QoS). However, the current 3GPP-mandated approach of storing binary-encoded URSPs per UE in the Unified Data Repository (UDR) leads to severe scalability and redundancy issues. As the number of connected devices scales into very large deployments, redundant storage and cascading updates amplify operational bottlenecks and increase the risk of service-level agreement (SLA) violations. In this paper, we propose HyPR: a Hybrid Policy Registry that combines an object store for semantic policy definitions with a binary cache for frequently used, pre-encoded policies. Our evaluation demonstrates up to 60% storage reduction with negligible latency overhead (less than 0.1 milliseconds), even under large-scale device populations. By decoupling policy definitions from UE assignments, HyPR provides a scalable and dynamic solution that reduces UDR load and enables third-party-driven policy modification through network APIs.

Keywords—5G Core, Policy Data Management, Policy Control Function, Network Slicing, URSP, Scalability, High Performance

I. INTRODUCTION

A. Background and Motivation

The fifth-generation (5G) mobile networks have introduced revolutionary capabilities through network slicing, enabling operators to create multiple virtual networks on shared physical infrastructure. This architectural innovation supports diverse service categories including enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-Latency Communications (URLLC), and massive Machine-Type Communications (mMTC). Central to ensuring Quality of Service (QoS) guarantees across these heterogeneous services is the User Equipment Route Selection Policy (URSP), which governs how UE routes traffic to appropriate network slice as illustrated in Figure 1.

As operators deploy 5G SA and network slicing, this capability is reaching a rapidly expanding device base: Ericsson reports 2.6 billion 5G subscriptions worldwide by Q2 2025 [1], indicating a large—and growing—population of UE Policy-capable smartphones and Customer Premises Equip-



Figure 1: URSP usecase for smartphone

ment (CPE) that can consume URSP rules. Complementary industry guidance from GSMA [2] outlines UE requirements and operational practices for URSP, reinforcing its importance in commercial networks. Meanwhile, adjacent verticals such as V2X are aligning with 5G-V2X roadmaps defined by national programs and recent academic surveys [3, 4]. These developments demonstrate that policy-driven traffic steering and slice selection are becoming mainstream across both consumer and industrial domains.

The URSP mechanism, standardized in 3GPP from Release 16 and enhanced in subsequent releases [5], comprises a set of rules that include traffic descriptors (application identifiers, Fully Qualified Domain Names (FQDN), connection capabilities) and route selection descriptors (Single Network Slice Selection Assistance Information (S-NSSAI), Data Network Name (DNN) preferences, etc.). These policies are provisioned by the Policy Control Function (PCF) and stored as binary-encoded data structures in the Unified Data Repository (UDR). While this approach ensures efficient transmission to UEs, it creates significant challenges in large-scale deployments.

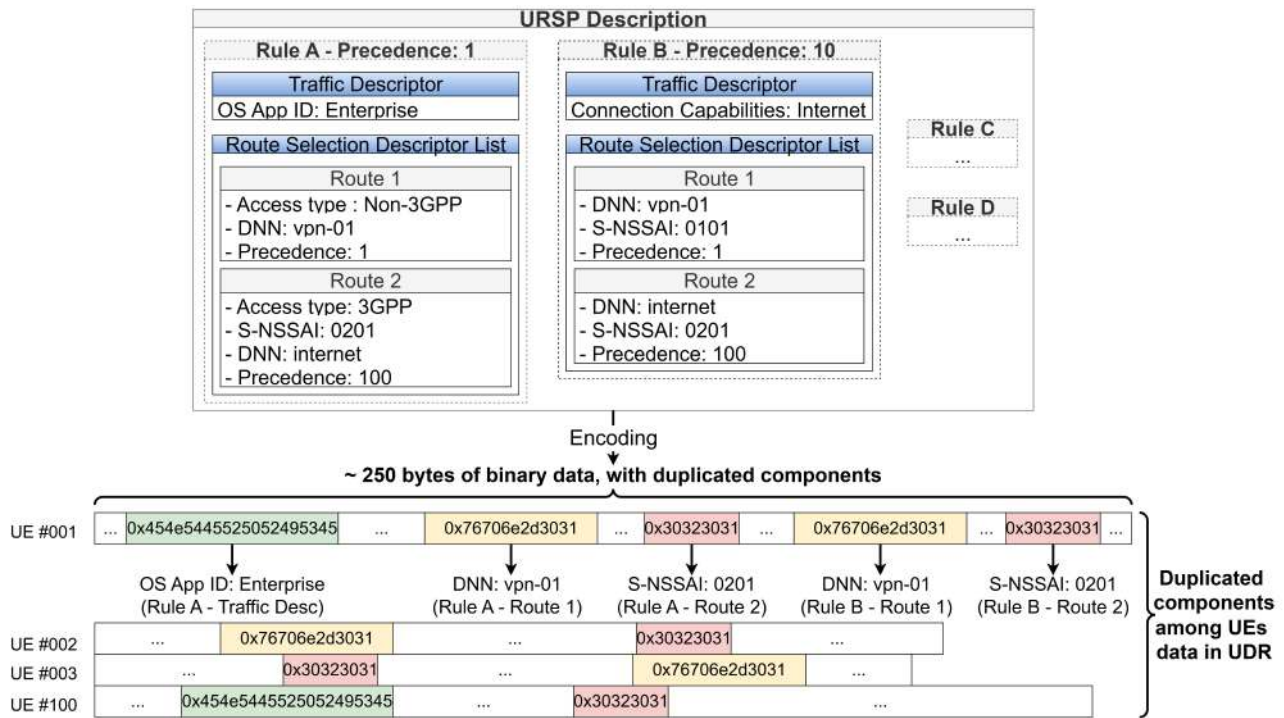


Figure 2: Example of URSP Storage in UDR following 3GPP Standard

B. Problem Statement

The current 3GPP architecture faces two critical challenges that limit its scalability and operational flexibility:

- 1) **Storage Redundancy Issue:** In the existing implementation, each UE's URSP is stored as an independent binary blob in the Unified Data Repository (UDR) [6], regardless of policy similarities across UEs. In practice, many users with similar usage patterns share identical or nearly identical policies, resulting in significant data redundancy. This issue becomes especially severe at large scale, where millions of devices may rely on the same policy configurations but still require separate storage allocations. For example, in Figure 2, significant redundancy exists at both inter-UE and intra-UE levels. The same 400-byte record may be repeated across millions of devices, while identical components such as the DNN ('vpn-01') or the S-NSSAI ('0201') are also duplicated within each UE. This duplication results in tens of gigabytes of wasted storage.
- 2) **On-demand Update Challenges:** The introduction of the Network Exposure Function (NEF) and northbound APIs enables third-party applications to request dynamic policy updates [7]. For example, enterprise customers may require time-based policies. During working hours (8 AM–6 PM), critical business traffic can be routed to premium slices to ensure performance. After hours, the traffic may be redirected to standard slices to reduce costs. Updating a single common rule that affects millions of UEs can trigger a cascade of individual UDR updates. This creates substantial system load and

increases the risk of Service Level Agreement (SLA) violations. For instance, a change in traffic steering policy for a popular service could simultaneously impact millions of subscribers. Similarly, policy updates for large-scale eMBB applications or slice reconfigurations for live sports streaming may trigger massive concurrent UDR write operations. Such bursts may overwhelm the system and degrade service quality.

C. Main Contribution and Paper Organization

This paper presents HyPR (**H**ybrid **P**olicy **R**egistry), a novel storage architecture that addresses these scalability challenges while maintaining full compatibility with existing 3GPP interfaces. Our key contributions include:

- A dual-storage mechanism that separates policy definition from UE assignment, eliminating redundancy through shared policy components.
- An intelligent caching layer that introduces only sub-millisecond latency overhead while reducing storage requirements by 50–60%.
- A drop-in PCF replacement design that maintains standard N15 and N36 interface compliance, ensuring seamless deployment in existing 5G Core System.
- Comprehensive performance evaluation demonstrating carrier-grade scalability for massive deployments.

The remainder of this paper is organized as follows: Section II presents the system model and proposed solution, Section III provides detailed performance evaluation and Section IV concludes with future research directions.

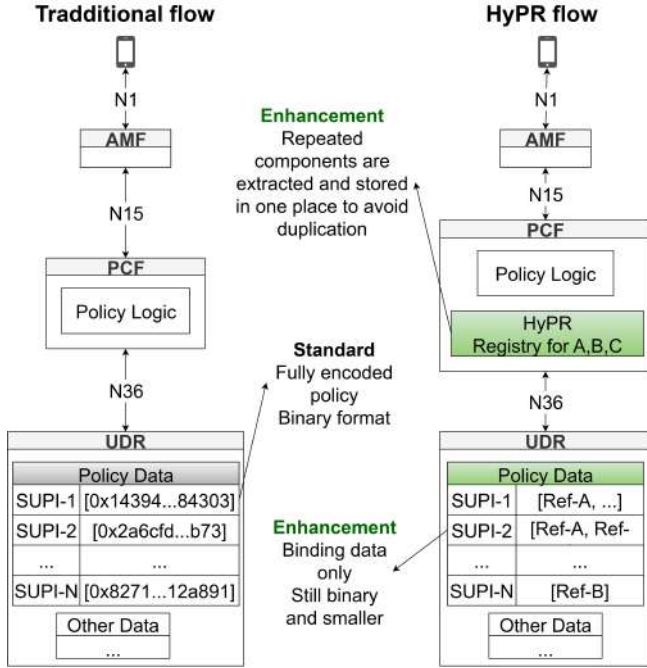


Figure 3: Overall architecture comparison

II. SYSTEM MODEL AND PROPOSED SOLUTION

A. Core Concept

HyPR introduces a fundamental architectural shift by separating common policy components from UE-specific assignments. The key innovation is representing policies through templates that act as blueprints for combining recurring components such as rules, traffic descriptors, and route descriptors. These components are also stored centrally in a registry to avoid redundancy. In the UDR, instead of storing complete binary policies for each UE, we maintain only lightweight bindings between UE identifiers and template identifiers. This approach eliminates the massive data redundancy inherent in current deployments where millions of UEs share identical or near-identical policies. Crucially, by maintaining binary format storage in the UDR and integrating HyPR directly within the PCF, all standard 3GPP interfaces (N15, N36) remain unchanged, ensuring seamless integration with existing network functions. Figure 3 presents comparison between the current 3GPP architecture and our proposed architecture, highlighting the introduction of HyPR.

While reducing storage through de-duplication, HyPR provides carrier-grade performance for policy retrieval and updates. Policies are retrieved or rebuilt quickly from templates and components using intelligent caching, maintaining sub-millisecond access times. Unlike traditional designs, where updating policies used by many UEs generates numerous writes to the UDR and slows down the system, HyPR applies updates efficiently without overloading the UDR. This approach enables the system to scale to support a large number of subscribers and applications.

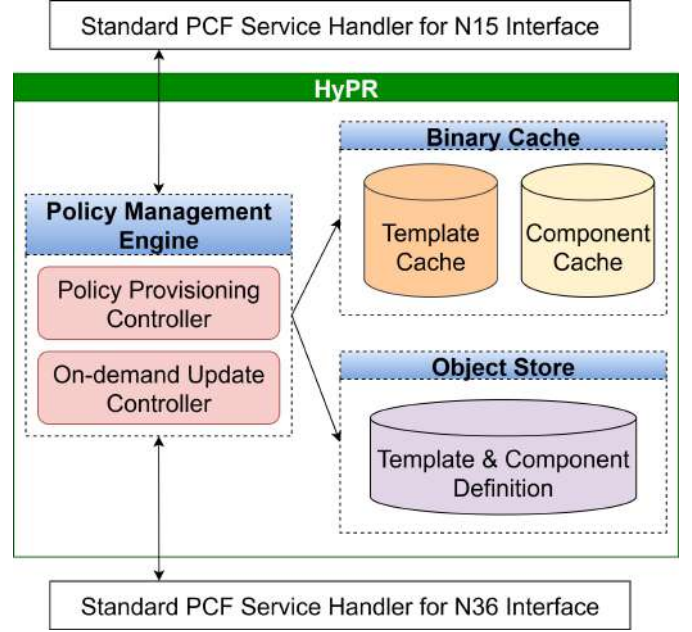


Figure 4: Internal PCF architecture with HyPR integration

B. Hybrid Policy Registry Design

Following the core concept of template-based policy storage, HyPR comprises two main architectural components: a dual storage system and a policy management engine, both integrated within the PCF to maintain standard 3GPP compliance. Figure 4 illustrates the complete HyPR architecture showing the interaction between dual storage components and the policy management engine within the enhanced PCF.

Dual Storage System: The heart of HyPR is its dual storage architecture that balances storage efficiency with access performance.

- 1) *Object Store:* This layer serves as the central repository for policy templates and components, storing the original semantic definitions of policy components according to 3GPP specifications. It maintains:
 - Template-to-component mappings managed in a relational structure to reduce redundancy while preserving 3GPP-defined relationships
 - Semantic components as defined in TS 24.526 [8] such as: rules, traffic descriptors (application IDs, FQDNs, IP filters, etc.), route selection descriptors (S-NSSAI, DNN, etc.)
- 2) *Binary Cache:* This layer ensures carrier-grade performance by maintaining pre-encoded binary representations of both complete policy templates and individual components. It consists of two main caches:
 - Template cache: Pre-encoded policy templates stored in memory, enabling delivery to the UE without additional build or encoding overhead.
 - Component cache: Pre-encoded building blocks (e.g., rules, traffic and route descriptors) stored in memory to speed up policy reconstruction based on templates.

Policy Management Engine: This component orchestrates all policy operations within the enhanced PCF, supporting two primary business functions:

- 1) *Policy Provisioning Controller:* handles policy retrieval requests when an UE attaches or a 3rd party application issues a demand. Its main operations are:
 - Receives policy requests from AMF via standard N15 interface
 - Queries UDR for UE-to-template bindings
 - Retrieves the corresponding policy information from HyPR storage, with potential rebuilding if required.

Key benefit: The policy retrieval process is driven by the UE-to-template binding: the system first searches in the template cache; if cache hits, the process finishes with the minimum latency. If no match is found, it falls back to the Object Store to retrieve the corresponding template definition. During the policy rebuilding process, components are also resolved against the component cache before querying the Object Store, reducing both latency and compute resources required for encoding.

- 2) *On-demand Update Controller:* Manages dynamic policy modifications triggered by NEF or operator requirements.
 - Receives update notification from UDR via N36 interface
 - Updates affected templates **in HyPR's internal storage only**
 - Identifies all UEs bound to modified templates through simple database query
 - Invalidates or updates relevant cache entries across all tiers

Key benefit: In conventional architecture, a policy update affecting a large number of UEs requires an equivalent number of write operations to the UDR, which creates substantial overhead and potential performance risks. HyPR addresses this by updating only the policy-related data within its internal storage, while preserving the existing UE-to-template bindings in the UDR. This design greatly reduces the operational load on the UDR and ensures consistency across UE records.

C. Template and Component Design

The design of policy templates significantly impacts both storage efficiency and system performance. We identify three primary approaches for template construction, each offering different trade-offs.

Level 1 - Atomic Component Granularity is the most granular level, where components are individual URSP elements as defined in 3GPP, such as OS App ID, FQDN, S-NSSAI, DNN, etc. Templates become detailed instructions for assembling these atomic elements into complete policies, which is quite similar to the original instructions from 3GPP. While this approach achieves maximum deduplication by storing each unique element only once, it requires complex assembly logic with multiple database lookups and intricate dependency management during policy construction.

Level 1 - Atomic Component Granularity

Components:

C1: OSAppId="com.netflix.app",
C2: FQDN="*.netflix.com",
C3: S-NSSAI="01-000001",
C4: S-NSSAI="01-000002",
C5: DNN="internet.apn",

Template

T1: $1=[C1 \Rightarrow (C4+C5)]$, $100=[C2 \Rightarrow (C3+C5)]$
(1 and 100 are rule precedences)

Level 2 - Descriptor Granularity uses traffic descriptors and route selection descriptors as components, with templates providing instructions for combining these descriptors into complete rules. Templates specify how to pair traffic descriptors with appropriate route descriptors to form rules. This approach achieves moderate deduplication while maintaining more manageable complexity than Level 1, as descriptors are logical groupings that align with 3GPP structure.

Level 2 - Descriptor Granularity

Components:

TD1: [FQDN="*.netflix.com"],
TD2: [OSAppID="com.netflix.app"],
RD1: [DNN="internet", SNSSAI="01-000001"]
RD2: [DNN="internet", SNSSAI="01-000002"]

Template

T1: $1=[TD1 \Rightarrow RD1]$, $100=[TD1 \Rightarrow RD2]$

Level 3 - Rule Granularity. At this level, components are complete URSP rules containing all necessary traffic matching and routing instructions, while templates are collections rule references. Although this approach provides less deduplication, it offers significant deployment advantages through operational simplicity and fast policy reconstruction with minimal processing overhead.

Level 3 - Rule Granularity

Components:

NetlixHD: [traffic="netflix", routes="internet, slice 1"],
Netlix4K: [traffic="netflix", routes="vpn, slice 2"],

Template

T1: $1=Netlix4K$, $100=NetlixHD$

For eMBB deployments, we adopt Level 3 based on several practical considerations. A single rule can effectively represent the complete traffic control policy for one application, making this level naturally aligned with how operators manage services. This application-centric approach significantly simplifies subscriber behavior analytics and network planning as it's easier to analyze patterns based on application usage rather than using atomic components or descriptors. Furthermore, the simple concatenation-based assembly eliminates complex logic and reduces latency during policy reconstruction.

III. PERFORMANCE EVALUATION

To validate the effectiveness of HyPR, we conduct comprehensive experiments focusing on two critical aspects: storage efficiency and operational performance. Our evaluation simulates realistic 5G eMBB deployment scenario to demonstrate the system's behavior under different cache configurations.

A. Storage Efficiency Test

For simplicity and reproducibility, the construction of URSP rules is limited to fundamental components, namely OS App ID, S-NSSAI, and DNN. The policy for each application is represented by a single rule, as illustrated in the example below.

URSP Rule Encoding Example	
• Traffic Descriptor: 30 bytes	
– Application ID (OSId + OSAppId):	
* Type/Length fields: 2 bytes	
* Type/Length fields: 1 byte	
* OS Id: “97a498e3-fc92-5c94-8986-0333d06e4e47”	
(UUID represents Android OS) - 16 bytes	
* Type/Length fields: 1 byte	
* OS App Id: “ENTERPRISE” - 10 bytes	
• Route Selection Descriptors: 62 bytes	
– Main route: 29 bytes	
* Type/Length fields: 2 bytes	
* Route precedence: 1 byte	
* Type/Length fields: 2 bytes	
* S-NSSAI: Premium Slice - 4 bytes	
* Type/Length fields : 2 bytes	
* DNN string: “vpn01.mnc01.mcc001” - 18 bytes	
– Backup route: 33 bytes	
* Type/Length fields: 2 bytes	
* Route precedence: 1 byte	
* Type/Length fields: 2 bytes	
* S-NSSAI: Default Slice - 4 bytes	
* Type/Length fields : 2 bytes	
* DNN string: “internet.mnc01.mcc001” - 22 bytes	
Total bytes per rule: 30 + 62 = 92 bytes	

We simulate deployment scenarios at three scales: 10M UEs with 500 applications, 50M UEs with 3,000 applications, and 100M UEs with 5,000 applications. These configurations capture both subscriber growth and increasing application diversity. For each scenario, policy data is generated per UE. Following GSMA guidelines [2], every UE is assigned three basic rules: one for IP Multimedia Subsystem (IMS) traffic (voice/video call), one for general Internet traffic, and one match-all rule that covers all remaining traffic. In addition, one to five application-specific rules are added using a truncated normal distribution ($\mu = 3$, $\sigma = 1.2$), while adhering to the 3GPP limit of 8 concurrent slices per UE [7].

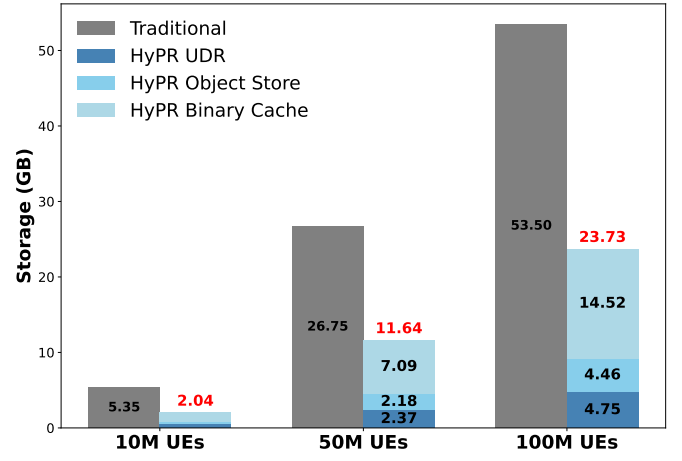


Figure 5: Storage Efficiency Comparison

TABLE 1: STORAGE EFFICIENCY TEST SUMMARIZATION

Scale	Traditional	HyPR	Reduction
10M UEs 500 apps	5.35 GB	2.04 GB total UDR: 0.47 GB Object store: 0.37 GB Binary caches: 1.20 GB	61.87% 3.31 GB
50M UEs 3K apps	26.75 GB	11.64 GB total UDR: 2.37 GB Object store: 2.18 GB Binary caches: 7.09 GB	56.49% 15.11 GB
100M UEs 5K apps	53.50 GB	23.73 GB total UDR: 4.75 GB Object store: 4.46 GB Binary caches: 14.52 GB	55.64% 29.77 GB

Application assignment for a single UE is simulated based on statistics from mobile app marketplaces [9, 10]. In each category, only a small number of applications account for most downloads. This skewed distribution is modeled according to Zipf's law [11]. It also explains the duplication observed in UE Policy data, since many UEs converge on rules for the same dominant applications.

The UE Profiles Analysis (Table 2) indicates that the simulated datasets effectively capture both duplication and diversity. The template reuse rate remains consistently above 75%, while no single template dominates the UE population (each accounts for less than 10% coverage). Building on this observation, HyPR achieves substantial storage efficiency compared to the traditional approach. Empirical results from Table 1 and Figure 5 show storage reductions of 55.64% for 100M UEs and up to 61.87% for 10M UEs, saving tens of gigabytes of storage. The UDR storage footprint was also significantly reduced, with binding data occupying only a few bytes per UE. These findings confirm HyPR's capability to deliver a scalable and cost-effective solution for large-scale telecommunication deployments.

TABLE 2: UE PROFILES ANALYSIS

Scale	10M	50M	100M
Rules count	503	3,003	5003
Templates count	1,964,583	11,766,713	24,131,501
Templates reuse rate	80.4%	76.5%	75.9%
Top 100 templates coverage	8.01%	1.59%	0.81%
Top 100 app coverage	95.6%	93.7%	93.4%

B. Operational Performance Evaluation

We used the 50M UEs dataset to emulate large-scale policy provisioning. Measurements covered the entire process from UDR query to the generation of a fully encoded URSP. The testbed comprised PCF and UDR modules, communicating via the N36 interface, deployed on servers equipped with Intel Xeon Platinum 8260 CPUs and 20 Gbps network links. Redis served as the UDR database. Within the PCF, template definitions and pre-encoded rules were fully cached in memory operating at 2,933 MT/s. Different proportions of pre-encoded templates were randomly selected for caching to evaluate coverage across low to high levels. The setup emulated a sustained load of 1,000 TPS, with the 99th-percentile (P99) latency adopted as the primary performance metric.

TABLE 3: LATENCY UNDER DIFFERENT CACHE COVERAGE (P99)

	Traditional	HyPR-H	HyPR-M	HyPR-L
Cache coverage (%)	-	100	60	10
Cache size (GB)	-	9.23	6.43	2.89
Total latency (μ s)	1,426	1,408	1,414	1,438
UDR query	1,426	1,368	1,361	1,356
Template fetch	-	36	30	22
Template query	-	-	32	45
Rules fetch	-	-	15	18
Rules assembly	-	-	7	15
Improvement (%)	-	1.26	0.42	-2.08

As summarized in Table 3, the UDR query latency accounts for over 90% of the total processing time. HyPR slightly reduced UDR query time (6%) by minimizing the payload size as described above. The overhead of cache-related operations including template fetch from cache (22–36 μ s), template definition query (32–45 μ s), rule fetch (15–18 μ s), and rule assembly (7–15 μ s), aggregated to less than 0.1 ms. Even with the lowest cache coverage (10%), the HyPR method resulted in only a 2.08% degradation in latency compared to the baseline, indicating robustness against constrained cache conditions.

A closer examination demonstrates the sensitivity of latency to cache coverage. With full coverage (HyPR-H), latency was slightly reduced (1.26%), while limited coverage (HyPR-L) caused a modest increase (2.08%). This trend highlights the trade-off between memory allocation and performance: higher cache coverage reduces reliance on additional stages at the cost of greater memory consumption. These results highlight the role of cache hit rates in reducing latency, suggesting future

work on smarter caching strategies (e.g., ARC [12] or machine learning based models) to improve efficiency.

An additional advantage of the HyPR design lies in its architectural decoupling between UE-specific data and template definitions. As a result, policy content updates can be applied solely through cache refreshes or object store updates within HyPR, without requiring modifications to the UDR. This separation not only reduces the operational burden on UDR but also enhances system scalability and maintainability.

IV. CONCLUSION AND FUTURE WORK

This paper introduced HyPR, a hybrid policy registry that decouples policy definitions from UE-specific data while leveraging intelligent caching to ensure both storage efficiency and operational scalability. Evaluation results demonstrated that HyPR can reduce storage requirements by more than 50% while adding sub-millisecond overhead, even under large-scale deployments with high transaction loads. The proposed architecture also simplifies policy updates, alleviating UDR bottlenecks and improving maintainability.

Looking ahead, the caching strategy can be enhanced through frequency-aware replacement policies or machine learning based prediction models to maximize hit rate with less memory consumption. HyPR can be extended beyond eMBB to support a broader range of services, e.g. mMTC with billions IoT devices, URLLC for V2X, A2X applications, enabling seamless support for future 5G and 6G network innovations as specified in the 3GPP standard.

REFERENCES

- [1] Ericsson. 5g to account for one-third of mobile subscriptions in 2025. [Online]. Available: <https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-subscriptions-outlook>
- [2] GSMA. Ue requirements related to network slicing using ursp. [Online]. Available: <https://www.gsma.com/newsroom/wp-content/uploads/TS.62-V1.0-UE-Requirements-related-to-network-slicing-using-URSP-1.pdf>
- [3] X. Zhang, J. Li, J. Zhou, S. Zhang, J. Wang, Y. Yuan, J. Liu, and J. Li, "Vehicle-to-everything communication in intelligent connected vehicles: A survey and taxonomy," *Automotive Innovation*, vol. 8, 02 2025.
- [4] NHTSA. Report to congress: Research on connected vehicle technology. [Online]. Available: <https://www.nhtsa.gov/sites/nhtsa.gov/files/2024-02/research-connected-vehicle-technology-report-to-Congress-021524.pdf>
- [5] ETSI. Policy and charging control framework for the 5g system (5gs). [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123503/17.05.00_60/ts_123503v170500p.pdf
- [6] —. Usage of the unified data repository service for policy data, application data and structured data for exposure. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/129500_129599/129519/17.07.00_60/ts_129519v170700p.pdf
- [7] —. System architecture for the 5g system (5gs). [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/17.05.00_60/ts_123501v170500p.pdf
- [8] —. User equipment (ue) policies for 5g system (5gs). [Online]. Available: https://www.etsi.org/deliver/etsi_ts/124500_124599/124526/17.07.00_60/ts_124526v170700p.pdf
- [9] H. Wang, Z. Liu, J. Liang, N. Vallina-Rodriguez, Y. Guo, L. Li, J. Tapiador, J. Cao, and G. Xu, "Beyond google play: A large-scale comparative study of chinese android app markets," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 293–307. [Online]. Available: <https://doi.org/10.1145/3278532.3278558>
- [10] A. A. R. R., V. E., and S. B., "Exploring google play store apps using predictive user context: A comprehensive analysis," *International*

Journal of Advanced Computer Science and Applications, vol. 15, no. 7, 2024. [Online]. Available: <http://dx.doi.org/10.14569/IJACSA.2024.01507125>

- [11] J. Kanwal, K. Smith, J. Culbertson, and S. Kirby, "Zipf's law of abbreviation and the principle of least effort: Language users optimise a miniature lexicon for efficient communication," *Cognition*, vol. 165, pp. 45–52, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0010027717301166>
- [12] N. Megiddo and D. S. Modha, "Arc: A self-tuning, low overhead replacement cache," in *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, ser. FAST '03. USA: USENIX Association, 2003, p. 115–130.



Thanh Son Pham received his B.Sc. degree in Mathematics and Informatics Engineering from Hanoi University of Science and Technology, Vietnam in 2018. He is currently a software engineer in Viettel High Technology, Viettel Group. He has industry experience with micro-service and high-performance computing, especially for telecommunications.



Tien-Quang Nguyen received his Bachelor's degree in Information Technology from University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam, in 2024. He is currently a software engineer at Viettel High Tech, focusing on automated software engineering.



Quan Huan Vu received his B.Sc. degree in Mathematics and Informatics Engineering in 2014 from Hanoi University of Science and Technology, Vietnam. He is currently as Head of Research and Development Department of Payment Platform Development Center, Viettel High Technology Industries Corporation, Viettel Group. He has industry experience in developing online charging system in telecom.



Duc Hai Nguyen received his B.Sc. in Mathematics and Informatics Engineering from Hanoi University of Science and Technology, Vietnam in 2009. He is currently director of the Payment Platform Development Center, Viettel High Technology Industries Corporation, Viettel Group. He has many years of experience in research and development of 5G Core systems such as online charging system. His research interests include telecommunications core networks, cloud Computing.