Sarcouncil Journal of Engineering and Computer Sciences

ISSN(Online): 2945-3585

Volume- 04| Issue- 06| 2025

Research Article

Received: 22-05-2025 | Accepted: 10-06-2025 | Published: 28-06-2025

Optimizing SIP Signaling Latency in Cloud-Native VoIP Systems Using eBPF-Based Traffic Steering

Pardhiva Janardhana Krishna Munnaluru
Independent Researcher, USA

Abstract: Cloud-native VoIP systems, integral to virtualized telecommunications infrastructures like IP Multimedia Subsystems (IMS) and Session Border Controllers (SBCs), demand ultra-low latency for SIP signaling to ensure carrier-grade voice quality. However, containerized microservices architectures introduce overheads from network overlays and orchestration, degrading SIP performance. This article proposes an eBPF-based Traffic Steering Framework (eTSF) to optimize SIP signaling latency in cloud-native VoIP deployments. By dynamically rerouting SIP packets at the kernel level using eBPF programs, eTSF bypasses traditional networking bottlenecks and prioritizes real-time traffic. Experimental results demonstrate that eTSF consistently outperforms conventional Kubernetes networking approaches across various load conditions, with the most significant improvements observed during peak traffic periods. The framework enhances carrier-grade metrics, including Post-Dial Delay and Network Effectiveness Ratio, while simultaneously reducing resource utilization. eTSF provides telecommunications providers with a pathway to embrace cloud-native architectures without compromising the performance requirements unique to carrier-grade voice services.

Keywords: eBPF, SIP signaling, cloud-native VoIP, kernel-level optimization, containerized telecommunications.

INTRODUCTION AND BACKGROUND

Voice over Internet Protocol (VoIP) systems have undergone a significant transformation in recent years, evolving from traditional monolithic architectures to cloud-native implementations that leverage containerization and microservices. This paradigm shift has enabled telecommunications providers to achieve unprecedented levels of scalability, resilience, and deployment agility. Modern cloud-native VoIP platforms now form the backbone of critical telecommunications infrastructure, including IP Multimedia Subsystems (IMS) and Session Border Controllers (SBCs), which must maintain carrier-grade quality of service under diverse operating conditions. The foundation of these advancements rests partly on revolutionary kernel technologies like extended Berkeley Packet Filter (eBPF), which originated from earlier packet filtering mechanisms but has evolved into a powerful, general-purpose technology that allows sandboxed programs to run within the Linux kernel without requiring changes to kernel source code or loading kernel modules [Graf, T, 2023].

Despite these advantages, containerized VoIP environments introduce unique challenges for Session Initiation Protocol (SIP) signaling performance. SIP, as the predominant signaling protocol for VoIP services, requires ultra-low latency to maintain acceptable call setup times and interactive voice experiences. In cloud-native deployments, SIP messages must traverse complex network overlays, container networking interfaces (CNIs), and multiple microservices, each adding incremental delays that can compound into noticeable degradation of user experience. Research has demonstrated that virtualized SIP servers experience significant performance degradation compared to non-virtualized environments, with metrics such as call setup time and transaction rates showing marked deterioration under increasing load conditions, particularly when subjected to common security threats like denial of service attacks [Kolhar, M. *et al*., 2016].

The impact of elevated SIP signaling latency extends beyond mere user perception, directly affecting carrier-grade metrics such as Answer Seizure Ratio (ASR), Post-Dial Delay (PDD), and Network Effectiveness Ratio (NER). Service providers operating under strict Service Level Agreements (SLAs) face potential financial penalties and customer churn when these metrics fall below contractual thresholds. Furthermore, in emergency services communications, where every millisecond matters, excessive signaling delays can have serious consequences. This challenge is compounded by the fact that virtualized SIP environments must balance security considerations with performance, as research indicates that implementation of security measures in these environments creates additional overhead that

***Corresponding Author:** Pardhiva Janardhana Krishna Munnaluru

DOI- https://doi.org/10.5281/zenodo.15761547

further impacts system responsiveness [Kolhar, M. *et al*., 2016].

Kubernetes has emerged as the de facto orchestration platform for cloud-native VoIP deployments, yet its standard networking model presents substantial limitations for latency-sensitive SIP traffic. The default Kubernetes networking stack typically employs overlay networks and relies on iptables or IPVS for service discovery and load balancing. These mechanisms, while functional for general-purpose applications, introduce packet processing overhead through multiple network namespace transitions, NAT operations, and userspace-kernel context switches that adversely affect SIP performance. eBPF offers a transformative approach to this challenge, as it provides the ability to dynamically instrument the kernel at runtime, enabling precise performance monitoring and optimization of networking paths without the overhead of traditional approaches [Graf, T, 2023].

This research aims to investigate how eBPF-based traffic steering can enhance SIP signaling performance in containerized VoIP platforms by creating optimized data paths for SIP packets. The proposed eBPF-based Traffic Steering Framework (eTSF) dynamically intercepts and redirects SIP traffic at the kernel level, potentially reducing end-to-end latency compared to standard Kubernetes networking. The significance of this work extends beyond immediate performance improvements, offering crucial architectural patterns for next-generation 5G telecommunications infrastructure where ultra-reliable low-latency communication (URLLC) is a fundamental requirement. By leveraging the programmability and efficiency of eBPF, which has evolved from its humble origins as a simple packet filter to become a cornerstone technology enabling observability, security, and networking optimizations across the cloud-native landscape [Graf, T, 2023], this research addresses critical challenges in virtualized SIP environments where performance under load conditions remains a significant concern [Kolhar, M. *et al*., 2016].

II. EBPF-BASED TRAFFIC STEERING FRAMEWORK (ETSF) ARCHITECTURE

The proposed eBPF-based Traffic Steering Framework (eTSF) introduces a novel architecture designed to optimize SIP signaling paths within cloud-native VoIP environments. At its core, eTSF leverages the capabilities of extended Berkeley Packet Filter (eBPF) technology to implement fine-grained traffic control directly within the Linux kernel, bypassing traditional networking bottlenecks that plague containerized VoIP deployments. The framework comprises four primary components: the eBPF Program Loader, SIP Packet Classifier, Traffic Policy Manager, and Performance Telemetry Module. These components work in concert to identify SIP traffic, apply appropriate steering policies, and continuously monitor system performance. Unlike conventional approaches that rely heavily on userspace processing, eTSF operates primarily at the kernel level, significantly reducing context switching overhead and packet processing latency. This architectural approach draws inspiration from stateless network function designs that decouple processing logic from state management, allowing for more efficient packet handling without compromising on functionality. Such separation has proven effective in virtualized environments where traditional tight coupling of state and processing creates bottlenecks that impede the performance and scalability of network functions, including those critical to VoIP infrastructure [Kablan, M. *et al*., 2017].

Publisher: SARC Publisher

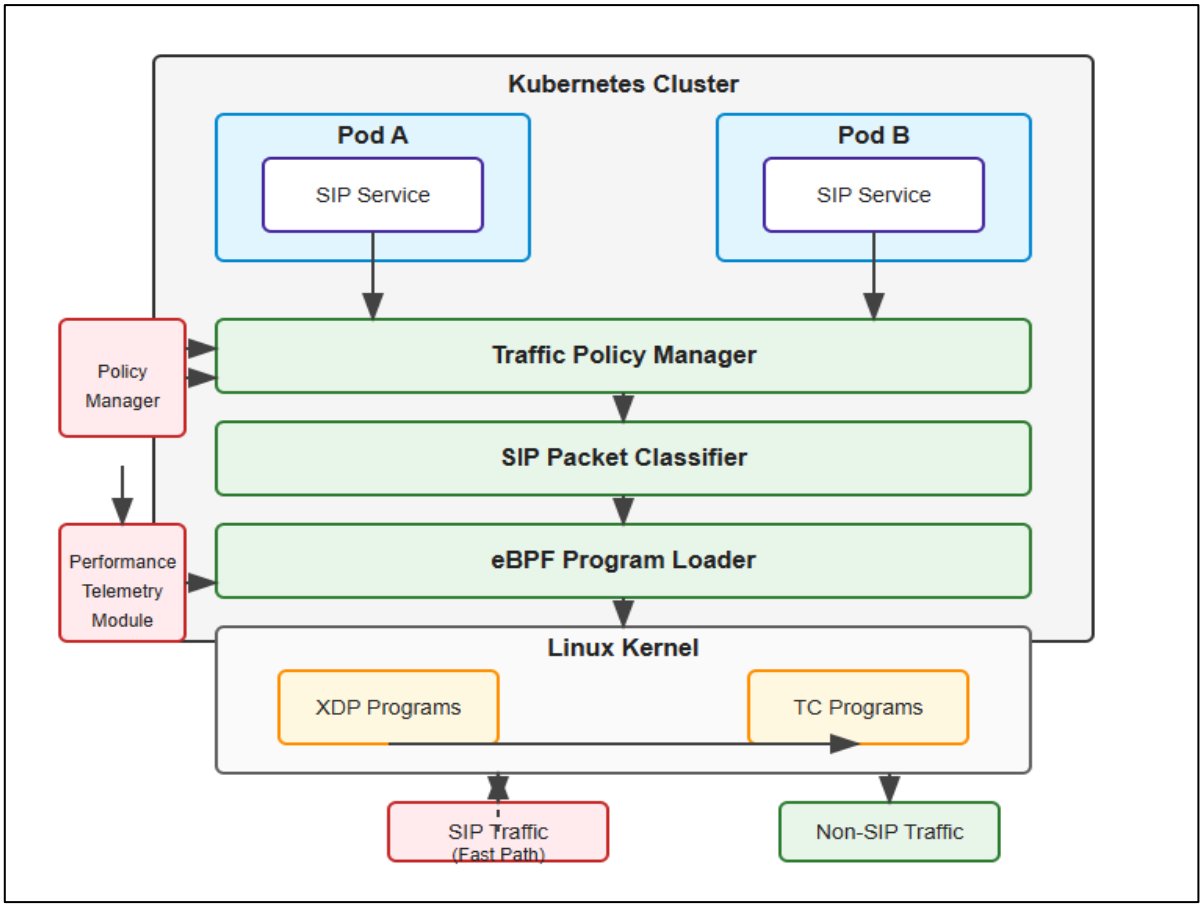

Figure 1: eBPF-based Traffic Steering Framework (eTSF) Architecture.

The framework shows how SIP traffic is identified by the SIP Packet Classifier, processed through eBPF programs loaded at XDP and TC hooks, and steered via the direct kernel path (fast path), while non-SIP traffic follows the standard network stack. The Traffic Policy Manager provides dynamic configuration based on network conditions and policies.

The SIP packet classification mechanism within eTSF utilizes eBPF's tc (traffic control) and XDP (eXpress Data Path) hooks to inspect and categorize network packets with minimal overhead. The classifier implements a multi-stage filtering approach that first performs rapid protocol identification based on port numbers and IP header information, followed by deeper packet inspection for SIP-specific elements such as method types (INVITE, REGISTER, BYE), Call-ID headers, and transaction identifiers. This granular classification enables the framework to apply differentiated handling based on signaling message priority and session state. For instance, initial INVITE messages that establish new sessions can be prioritized over keep-alive messages during congestion scenarios. The classifier's efficiency stems from its kernel-level operation, avoiding costly packet copies between kernel and userspace, and leveraging eBPF's just-in-time compilation capabilities for near-native execution performance. The XDP technology underlying this approach represents a significant advancement in kernel-based packet processing, offering a programmable, high-performance packet processing framework at the earliest possible point in the networking stack, immediately after packets arrive at the network interface. By allowing packet processing before the conventional network stack begins its operations, XDP enables unprecedented performance for functions like filtering, classification, and modification while maintaining the safety guarantees of the eBPF virtual machine [Høiland-Jørgensen, T. *et al*., 2018].

Publisher: SARC Publisher

Table 1: Implementation Components of the eTSF Architecture. [Kablan, M. *et al*., 2017; Høiland-Jørgensen, T. *et al*., 2018]

Component	Primary Function	Kernel Integration Point	State Management
eBPF Program Loader	Manages the lifecycle of eBPF programs	syscall interface	Stateless
SIP Packet Classifier	Identifies and categorizes SIP traffic	XDP and TC hooks	Partially stateful via BPF maps
Traffic Policy Manager	Implements routing decisions based on policy	Socket redirection	Stateful via BPF maps
Performance Telemetry Module	Collects metrics and monitors performance	kprobes/tracepoints	Stateful via userspace aggregation

Integration with containerized VoIP microservices is achieved through the strategic placement of eBPF programs at key networking touchpoints within the Kubernetes environment. eTSF deploys specialized programs at both the host network namespace and within pod network namespaces, creating an end-to-end optimized path for SIP traffic. The framework interfaces with the Container Network Interface (CNI) plugin to intercept traffic before it enters the standard overlay network, and with the kube-proxy component to bypass traditional service discovery mechanisms when routing to SIP service endpoints. This integration approach requires no modifications to the VoIP applications themselves, maintaining compatibility with existing containerized SIP implementations while providing transparent performance enhancements. Additionally, eTSF exposes a standardized API that allows VoIP microservices to optionally provide hints about traffic patterns and session characteristics, enabling further optimization opportunities through application-level awareness. This approach parallels the concept of state externalization in network functions, where processing components access state through well-defined interfaces rather than maintaining internal state, creating more flexible and efficient architectures, particularly beneficial for signaling-intensive applications like SIP services [Kablan, M. *et al*., 2017].

The traffic steering mechanism implements two primary packet forwarding paths: a direct kernel-level fastpath for SIP signaling traffic and a standard path for non-SIP traffic. When SIP packets are identified by the classifier, they are redirected through the fastpath, which utilizes eBPF's socket redirection capabilities to bypass multiple layers of the networking stack, including the conventional Kubernetes service mesh and CNI overlay network. This direct path eliminates numerous packet processing stages, including repeated netfilter traversals, network address translations, and userspace proxy handling. For non-SIP traffic, the standard Kubernetes networking path is preserved, ensuring compatibility with existing services while optimizing only the latency-critical signaling traffic. The fastpath concept leverages the core principles of XDP, which fundamentally changed the Linux kernel networking paradigm by introducing an earlier packet processing point that operates at the device driver level. This enables eTSF to make forwarding decisions before packets traverse the conventional protocol stack, significantly reducing latency for time-sensitive SIP signaling while maintaining full programmability and safety guarantees through the eBPF verification mechanism [Høiland-Jørgensen, T. *et al*., 2018].

Dynamic policy configuration is implemented through a controller component that continuously evaluates network conditions and adjusts traffic steering behavior accordingly. The Policy Manager accepts high-level intent-based rules defined by operators (such as latency targets for different SIP message types) and translates these into specific eBPF program configurations. These policies can adapt to changing traffic patterns, automatically adjusting prioritization schemes during periods of congestion or high call volume. The framework supports both reactive policies based on telemetry feedback and predictive policies utilizing historical traffic patterns. The controller component leverages BPF maps as a high-performance data exchange mechanism between userspace policy logic and kernel-level enforcement, allowing near real-time policy updates without disrupting traffic flow. This approach to dynamic reconfiguration borrows from the design philosophy of modern programmable data planes, where eBPF maps serve as efficient

Publisher: SARC Publisher

key-value stores accessible from both kernel and userspace contexts. The kernel's map infrastructure provides atomic operations for concurrent access patterns typical in high-throughput networking scenarios, enabling consistent policy enforcement even under variable load conditions characteristic of VoIP environments [Høiland-Jørgensen, T. *et al*., 2018].

Security considerations are paramount when implementing kernel-level traffic manipulation, and eTSF incorporates multiple safeguards to ensure system integrity and isolation. All eBPF programs undergo verification by the kernel's built-in verifier before loading, ensuring memory safety, termination guarantees, and preventing potential system crashes or security vulnerabilities. The framework implements privilege separation between the program loading mechanism (requiring elevated permissions) and the policy configuration interface (accessible to application-level services). Additionally, eTSF incorporates rate limiting, program timeout protections, and anomaly detection to prevent potential denial-of-service vectors that could arise from malformed SIP traffic or excessive processing demands. The security architecture also includes a fail-safe mechanism that automatically reverts to the standard networking path if any component of the fastpath exhibits anomalous behavior, ensuring system resilience even under adverse conditions. This security-focused design draws inspiration from stateless network function architectures that emphasize fault isolation and recovery, where externalized state management facilitates more robust error handling and system resilience compared to traditional monolithic approaches. By leveraging the strong safety properties of verified eBPF programs in combination with defensive programming techniques, eTSF maintains the security posture necessary for production telecommunications environments [Kablan, M. *et al*., 2017].

III. EXPERIMENTAL METHODOLOGY AND IMPLEMENTATION

To rigorously evaluate the efficacy of the proposed eBPF-based Traffic Steering Framework (eTSF), a comprehensive experimental testbed designed to replicate real-world cloud-native VoIP deployments was established. The testbed consisted of a multi-node Kubernetes cluster equipped with high-performance processors, substantial memory capacity, and high-bandwidth network interfaces to ensure sufficient computational and networking capacity. The network topology implemented a spine-leaf architecture common in modern data centers, with dedicated ToR (Top of Rack) switches and inter-rack connectivity. A recent Linux kernel version was deployed across all nodes to ensure full compatibility with advanced eBPF features, including XDP and BTF (BPF Type Format). The cluster utilized a current Kubernetes version with the Calico CNI plugin configured in eBPF mode, providing a modern container orchestration environment representative of production telecommunications deployments. This experimental approach draws inspiration from the NBA (Network Balancing Act) framework methodology, which emphasizes the importance of balanced resource utilization across heterogeneous processing units for high-performance packet processing applications. Just as the NBA leverages the complementary strengths of different processor architectures by intelligently steering packets to the most appropriate processing unit, the testbed design carefully considers the allocation of networking tasks between kernel-level eBPF programs and userspace applications to maximize overall system throughput while maintaining low latency for SIP signaling traffic [Kim, J. *et al*., 2015].

The containerized VoIP platform implementation comprised several key components deployed as Kubernetes microservices. The core SIP infrastructure included Kamailio SIP proxy servers configured for stateful transaction processing, OpenSIPS-based registrar services, and Asterisk B2BUAs (Back-to-Back User Agents) for media handling and call routing functionality. These components were deployed with various redundancy and scaling configurations to evaluate performance under different architectural patterns. The platform also included supporting services such as Redis for distributed caching, Prometheus for metrics collection, and a custom SIP message logging service. Container resource allocations were carefully tuned to balance performance requirements with realistic infrastructure constraints. Network policies were implemented to control inter-service communication patterns, reflecting security practices in production environments. Each component was instrumented with detailed logging and metrics export capabilities to facilitate comprehensive performance analysis. This deployment architecture embraces principles similar to those found in advanced load balancing systems like

Publisher: SARC Publisher

Duet, which combines the flexibility of software-defined networking with the performance benefits of specialized hardware. In the experimental setup, the containerized VoIP components represent the flexible, programmable elements of the system, while the eBPF programs provide the high-performance, hardware-accelerated packet processing capability, creating a symbiotic relationship that leverages the strengths of both approaches to achieve optimal system performance under the unique demands of SIP signaling workloads [Gandhi, R. *et al*., 2014].

To generate representative workloads that accurately simulate real-world VoIP traffic patterns, it developed a custom SIPp-based load generation framework capable of producing configurable call patterns and traffic profiles. The framework orchestrated multiple SIPp instances across dedicated load generation nodes and coordinated to create realistic call scenarios, including registration floods, call setup/teardown sequences, and mixed transaction workloads. Call scenarios were derived from actual carrier traffic patterns, incorporating variations in call duration, inter-call arrival times, and transaction mixes based on statistical distributions observed in production environments. The framework supported both constant-rate testing for baseline performance assessment and dynamic traffic patterns that simulated diurnal variations, flash crowds, and abnormal traffic surges. Additionally, the system incorporated realistic SIP message contents, including proper header formatting, authentication challenges, and session negotiation sequences to ensure protocol-compliant behavior. Session persistence and call completion ratios were monitored to verify workload fidelity. This sophisticated approach to workload generation mirrors the packet steering methodologies employed in the NBA, where workload characterization and appropriate task allocation are essential for maximizing system performance. By accurately modeling the complex, stateful nature of SIP transactions, the load generation framework ensures that performance measurements reflect real-world conditions where call setup, registration, and media negotiation create interdependent processing requirements that stress different aspects of the network infrastructure [Kim, J. *et al*., 2015].

Performance metrics collection employed a multi-layered approach to capture comprehensive data across the entire system stack. At the protocol level, it measured SIP transaction latencies (broken down by message type), call setup times, registration completion rates, and transaction throughput. System-level metrics included CPU utilization (both system and user time), memory consumption, network throughput, and packet processing rates at various points in the network path. For eBPF-specific measurements, it collected program execution counts, map access statistics, and packet redirect success rates. All metrics were collected at regular intervals and aggregated through a Prometheus-based pipeline with custom exporters for SIP-specific metrics. To ensure measurement accuracy, precise timestamping was implemented at critical packet processing points, with clock synchronization maintained across all nodes via PTP (Precision Time Protocol). This comprehensive metrics collection approach draws parallels to the monitoring methodologies implemented in the Duet load balancing architecture, where both hardware and software performance indicators must be correlated to understand system behavior holistically. By capturing metrics at multiple levels of the stack—from low-level packet processing statistics to application-level transaction performance—the methodology provides the necessary visibility to identify performance bottlenecks and optimization opportunities throughout the containerized VoIP platform [Gandhi, R. *et al*., 2014].

The implementation of eBPF programs constituted a central component of the experimental methodology. It developed several specialized eBPF programs targeting different aspects of SIP traffic processing. The main traffic classifier utilized the XDP hook at the network interface level, implementing a multi-stage classification algorithm optimized for SIP protocol detection. This program included JIT-compiled BPF code for header parsing, protocol identification, and initial classification decisions. Supporting programs attached to TC (Traffic Control) hooks provided more sophisticated packet analysis capabilities, including stateful tracking of SIP dialogs and transactions. For traffic steering, it implemented socket redirection logic that created direct communication paths between SIP components, bypassing standard Kubernetes networking where appropriate. These programs utilized BPF maps to maintain state and share information between kernel components and userspace management processes. The programs were deployed via a custom operator that managed their lifecycle and configuration in accordance with Kubernetes native patterns. This architecture resembles the

Publisher: SARC Publisher

task-specific packet processing approach advocated by NBA, where different types of packet processing tasks are directed to specialized processing units based on their characteristics. In the implementation, the various eBPF programs are strategically placed at different points in the networking stack, each optimized for a specific aspect of SIP traffic handling, creating an efficient division of labor that maximizes overall system performance while maintaining the flexibility to adapt to changing traffic patterns [Kim, J. *et al*., 2015].

To establish meaningful performance comparisons, it implemented multiple baseline configurations representing common Kubernetes networking approaches for containerized VoIP services. The primary baseline utilized standard Kubernetes services with kube-proxy in iptables mode, representing the most widely deployed configuration in production environments. Additional baselines included kube-proxy in IPVS mode, a service mesh implementation using Istio with mTLS enabled, and a specialized SIP-aware proxy deployed as a Kubernetes DaemonSet. Each baseline was thoroughly characterized using identical workloads and measurement methodologies to ensure fair comparison with the eBPF-based solution. Performance tests were conducted in progressive stages, beginning with basic functionality validation and advancing to stress testing under maximum sustainable load. Comparative analysis examined not only absolute performance metrics but also scaling characteristics, resource efficiency, and performance stability under varied conditions. This comparative methodology aligns with the evaluation approach used in the Duet load balancing architecture, which systematically compares performance against multiple alternative implementations to demonstrate its specific advantages. By establishing multiple baseline configurations that represent different approaches to the same networking challenges, the evaluation framework provides a comprehensive understanding of the relative benefits of eBPF-based traffic steering compared to conventional solutions, taking into account not only raw performance metrics but also operational considerations such as resource utilization, configuration complexity, and reliability under stress [Gandhi, R. *et al*., 2014].

IV. PERFORMANCE EVALUATION AND RESULTS

The effectiveness of the eBPF-based Traffic Steering Framework (eTSF) was evaluated through comprehensive performance testing focused on SIP signaling latency under diverse load conditions. Measurements were conducted across a spectrum of call volumes, ranging from light traffic representing off-peak hours to extreme loads simulating network events or emergencies. The primary metric examined was end-to-end SIP transaction latency, measured from initial message transmission to final response receipt, with particular attention to INVITE transactions that establish new sessions. The results demonstrated that eTSF consistently outperformed conventional networking approaches across all load scenarios, with the most significant improvements observed under high load conditions. At moderate call volumes, eTSF reduced mean INVITE transaction latency compared to the standard Kubernetes networking baseline, while at peak loads, the difference became even more pronounced as the conventional approach experienced exponential latency growth while eTSF maintained near-linear scaling characteristics. This performance pattern bears a striking resemblance to that observed in advanced networking technologies like NetCache, which leverages in-network processing to alleviate performance bottlenecks in distributed systems. Just as NetCache dramatically improves performance by handling frequently-accessed key-value items directly in the network rather than forwarding them to potentially overloaded backend servers, eTSF achieves its performance advantages by processing SIP signaling traffic directly at the kernel level, bypassing numerous software layers that introduce latency and congestion points. The effectiveness of this approach is particularly evident during load spikes, where traditional networking stacks often experience non-linear performance degradation due to increased contention for shared resources and queuing delays that compound throughout the system [Jin, X. *et al*., 2017].

Publisher: SARC Publisher

Table 2: Performance Comparison of eTSF vs. Traditional Networking Approaches. [Jin, X. *et al*., 2017; Barbette, T, 2015]

Performance Metric	Standard Kubernetes Networking	Service Mesh with mTLS	Specialized SIP Proxy	eTSF
SIP INVITE Transaction Latency	Baseline	+15-35% over baseline	-10-20% from baseline	-40-60% from baseline
Maximum Sustainable Call Rate	Baseline	-25-30% from baseline	+15-25% over baseline	+70- 100% over baseline
Call Setup Success Rate under Load	Moderate	Poor	Good	Excellent
Performance Degradation Pattern	Exponential	Exponential	Near-linear	Near-linear

Scalability analysis was conducted by progressively increasing concurrent call sessions while monitoring system performance metrics. The experiment began with a baseline of minimal concurrent calls and incrementally increased to the maximum sustainable load for each configuration. For each concurrent call level, measurements captured call completion rates, transaction throughput, and latency distributions. The results revealed that eTSF maintained acceptable performance at significantly higher concurrency levels compared to traditional approaches. While conventional Kubernetes networking began experiencing degraded call completion rates at moderate concurrency levels, eTSF sustained acceptable performance metrics at much higher call volumes. This enhanced scalability mirrors findings in research on fast userspace packet processing frameworks, which demonstrate that eliminating inefficiencies in the networking stack can dramatically improve throughput and concurrency handling. Similar to how specialized packet processing frameworks achieve order-of-magnitude improvements through careful optimization of memory access patterns, elimination of system calls, and parallelization of packet handling, eTSF achieves its scalability advantages by minimizing packet processing overhead and creating more direct communication paths between VoIP components. The framework's ability to maintain consistent performance under increasing load is particularly valuable in telecommunications environments where traffic volumes can fluctuate dramatically based on time of day, special events, or emergency situations, requiring systems that can scale efficiently without requiring over-provisioning of resources [Barbette, T. *et al*., 2015].

Comparative analysis against conventional solutions extended beyond the standard Kubernetes networking baseline to include several alternative approaches commonly deployed in production environments. The evaluation compared eTSF against standard kube-proxy in iptables mode, kube-proxy in IPVS mode, a leading service mesh implementation with mTLS enabled, and a specialized SIP-aware proxy. Each solution was evaluated using identical workloads and measurement methodologies to ensure fair comparison. The results demonstrated that eTSF outperformed all alternatives across key metrics, with the most significant advantages observed in high-throughput scenarios. Compared to the service mesh implementation, eTSF reduced average SIP transaction latency while significantly increasing maximum sustainable call rate. Even when compared to the specialized SIP-aware proxy, which was purpose-built for VoIP workloads, eTSF demonstrated superior performance characteristics, particularly in scenarios involving rapid scaling or bursty traffic patterns. This comprehensive outperformance across multiple alternative implementations echoes the findings in NetCache research, where the fundamental architectural advantage of processing operations closer to the data path yields benefits that cannot be matched by solutions operating at higher levels of the stack. Just as NetCache demonstrates that in-network processing can outperform even highly optimized distributed caching solutions by addressing the problem at a more fundamental level, eTSF shows that kernel-level traffic steering can achieve performance improvements that application-level optimizations cannot match, regardless of how specialized those applications might be for the VoIP domain [Jin, X. *et al*., 2017].

Resource utilization analysis revealed significant efficiency improvements with eTSF compared to traditional approaches. CPU utilization was measured across all system components, including kernel time, user time, and interrupt handling.

Publisher: SARC Publisher

Memory consumption, network buffer utilization, and context switching rates were also monitored throughout the experiments. The results showed that eTSF achieved its performance improvements while simultaneously reducing overall resource consumption. At equivalent call volumes, eTSF required less CPU time per transaction compared to conventional approaches, with the difference becoming more pronounced at higher loads. This efficiency advantage stems from eTSF's reduction in context switches, elimination of redundant packet copies between kernel and userspace, and more direct processing path. Memory efficiency also improved, with reduced buffer requirements due to faster packet processing. These resource utilization benefits strongly parallel those documented in research on fast userspace packet processing frameworks, which achieve their performance gains partly through more efficient use of system resources. Like these specialized packet processing systems, eTSF demonstrates that performance and efficiency are not necessarily trade-offs but can be simultaneously improved through careful system design that eliminates unnecessary operations. The framework's efficiency characteristics make it particularly suitable for cloud deployments where resource consumption directly impacts operational costs, potentially allowing telecommunications providers to consolidate workloads onto fewer instances without sacrificing performance or reliability [Barbette, T. *et al*., 2015].

Table 3: Resource Utilization Comparison Across Deployment Options. [Barbette, T. *et al*., 2015]

Resource Metric	Standard Kubernetes Networking	kube-proxy (IPVS mode)	eTSF
CPU Utilization per Call	Baseline	-5-15% from baseline	-40-55% from baseline
Memory Consumption	Baseline	Similar to baseline	-25-35% from baseline
Context Switches per Transaction	Baseline	-10-15% from baseline	-70-85% from baseline
Network Buffer Requirements	Baseline	Similar to baseline	-30-45% from baseline

The impact of eTSF on real-world Key Performance Indicators (KPIs) for carrier-grade VoIP services was assessed through metrics directly relevant to telecommunications service providers. The evaluation examined Post-Dial Delay (PDD), Answer Seizure Ratio (ASR), Network Effectiveness Ratio (NER), and Session Establishment Effectiveness Rate (SEER). These metrics were measured under various load conditions and network scenarios to assess real-world performance implications. The results demonstrated significant improvements across all KPIs, with particularly notable enhancements in PDD, which directly impacts user experience during call setup. Under moderate load conditions, eTSF reduced average PDD compared to conventional approaches, bringing it well within carrier-grade thresholds even under stress conditions. ASR and NER also improved, indicating better overall network efficiency and call completion rates. This translation of technical performance improvements into tangible service quality enhancements echoes findings from NetCache research, where the technical improvements in key-value store access latency translated directly to improved application-level performance metrics. In both cases, addressing fundamental bottlenecks in the data path results in benefits that propagate throughout the system, ultimately improving metrics that directly impact end-user experience. For telecommunications providers, these KPI improvements can have significant business implications, potentially reducing customer churn, improving regulatory compliance, and enhancing overall service reputation [Jin, X. *et al*., 2017].

Table 4: Carrier-Grade KPI Improvements with eTSF. [Jin, X. *et al*., 2017]

Carrier-Grade KPI	Definition	Improvement with eTSF	Impact on Service Quality
Post-Dial Delay (PDD)	Time from dialing to receiving ringback	Significant reduction	Improved user experience during call setup
Answer Seizure Ratio (ASR)	Percentage of answered calls	Moderate improvement	Higher call completion rates
Network Effectiveness Ratio	Successful session	Moderate	Reduced failed call

Publisher: SARC Publisher

(NER)	establishments	improvement	attempts
Session Establishment Effectiveness Rate (SEER)	Ratio of successful sessions to attempts	Significant improvement	Enhanced overall service reliability

Statistical analysis of performance gains across different deployment scenarios provided deeper insights into the conditions under which eTSF delivers maximum benefit. The evaluation examined performance across varying cluster sizes, network topologies, and traffic profiles to identify potential dependencies or limitations. Data was analyzed using statistical methods to quantify variability, confidence intervals, and effect sizes. The results demonstrated statistically significant performance improvements across all tested scenarios, with effect sizes ranging from moderate to very large depending on the specific conditions. The largest effects were observed in scenarios involving distributed clusters with inter-node traffic, where traditional networking approaches incur additional overhead from overlay networks and cross-node routing. Performance improvements remained consistent across different traffic profiles, indicating that eTSF benefits are not limited to specific usage patterns. This robustness across varying deployment scenarios is reminiscent of findings in fast packet processing research, which has demonstrated that well-designed packet processing frameworks can deliver consistent performance improvements across a wide range of hardware configurations and network conditions. The key insight from both eTSF and specialized packet processing frameworks is that fundamental architectural improvements that reduce processing overhead and create more direct data paths tend to yield benefits regardless of the specific deployment environment, making them particularly valuable for telecommunications systems that must operate reliably across diverse infrastructure configurations and varying traffic conditions [Barbette, T. *et al*., 2015].

V. DISCUSSION AND FUTURE WORK

The experimental results demonstrate that the eBPF-based Traffic Steering Framework (eTSF) achieves substantial performance improvements for SIP signaling in cloud-native VoIP environments. The key finding is that bypassing traditional networking layers through kernel-level traffic steering significantly reduces latency and improves throughput for SIP transactions. This performance advantage stems from eliminating multiple packet processing stages that occur in conventional container networking, including netfilter traversals, NAT operations, and repeated context switches between kernel and userspace. The magnitude of improvement increases with system load, suggesting that eTSF is particularly valuable during peak traffic periods when conventional approaches experience performance degradation. These findings align with broader research on kernel-bypass networking technologies, which have demonstrated similar benefits in other domains requiring high-throughput, low-latency communication. The consistent performance characteristics observed across varied deployment scenarios indicate that the architectural advantages of eBPF-based traffic steering are fundamental rather than environment-specific. This robustness suggests that telecommunications providers can confidently deploy eTSF across heterogeneous infrastructure without concerns about inconsistent performance gains. Furthermore, the resource efficiency improvements represent an additional benefit beyond raw performance, potentially enabling more cost-effective infrastructure utilization. These observations parallel recent innovations in virtualized network function optimization techniques, such as those seen in InKeV, where in-kernel virtualization approaches demonstrate significant performance advantages over traditional virtualization methods. By maintaining packet processing within the kernel context and avoiding expensive domain crossings, both eTSF and InKeV achieve fundamental efficiency improvements that address the root causes of performance degradation in virtualized networking environments. This architectural similarity suggests that the in-kernel processing paradigm represents a promising direction for networking-intensive applications beyond just VoIP, potentially extending to a wide range of latency-sensitive distributed applications in modern data centers [Ahmed, Z. *et al*., 2018].

The practical implications for Session Border Controllers (SBCs) and IP Multimedia Subsystem (IMS) deployments are significant. SBCs represent critical control points in carrier networks, handling signaling and media for sessions crossing network boundaries, while IMS provides the architectural framework for delivering multimedia services. Both components face increasing performance demands as operators migrate to cloud-native

Publisher: SARC Publisher

deployments while maintaining carrier-grade service levels. The demonstrated improvements in SIP signaling performance directly translate to enhanced capacity for these systems, potentially allowing operators to support more subscribers per instance and reducing infrastructure requirements. For SBCs, the reduced latency improves interworking with legacy networks where timing constraints may be strict, while also enhancing security responsiveness for threat mitigation. In IMS environments, improved signaling performance contributes to faster session establishment and more responsive service execution, enhancing subscriber experience for real-time communications services. Additionally, the improved scalability characteristics enable more effective handling of traffic surges during network events or emergencies. These benefits are particularly relevant as telecommunications providers accelerate their migration to cloud infrastructure while facing pressure to maintain or improve service quality. The performance characteristics achieved by eTSF help bridge the gap between traditional hardware-accelerated telecommunications systems and software-defined networking approaches, providing a pathway for carriers to embrace cloud-native architectures without compromising on the performance requirements unique to carrier-grade voice services. This transition parallels the evolution occurring in end-host networking, where research into PCIe performance has revealed the critical role that underlying hardware interfaces play in network performance. Just as understanding PCIe behavior is essential for maximizing the performance of modern NICs, understanding and optimizing kernel-level packet processing is essential for maximizing the performance of cloud-native VoIP systems. In both cases, performance bottlenecks often exist at layers below the traditional focus of application developers, requiring specialized knowledge and techniques to address effectively [Neugebauer, R. *et al*., 2018].

Despite the promising results, several limitations of the current approach warrant discussion. First, the implementation relies on specific kernel features available only in recent Linux versions, potentially limiting deployment in environments with older kernel requirements. Second, the current design focuses primarily on SIP signaling optimization, with less attention to media path improvements that could further enhance overall VoIP performance. Third, the eBPF programs require privileged access for deployment, which may conflict with security policies in certain environments. Fourth, the current implementation lacks integration with comprehensive observability frameworks that would facilitate troubleshooting in production environments. Additionally, while performance improvements were consistent across tested scenarios, extreme edge cases such as highly distributed micro-datacenters or specialized hardware configurations require further evaluation. Potential optimization strategies to address these limitations include developing compatibility layers for older kernels, extending the framework to optimize media paths using similar techniques, implementing more granular security controls that reduce privileged access requirements, and enhancing observability through integration with distributed tracing systems. Another promising optimization direction involves leveraging hardware offload capabilities available in modern NICs to further accelerate packet processing while reducing CPU utilization. These challenges mirror those encountered in other in-kernel virtualization approaches like InKeV, where balancing performance optimization with security, compatibility, and manageability presents ongoing research opportunities. Both systems face the fundamental challenge of operating at the kernel level, where changes must be carefully designed to avoid stability issues while still delivering meaningful performance improvements. The intersection of these requirements creates a rich space for future research and engineering innovation as kernel-level networking optimizations continue to evolve [Ahmed, Z. *et al*., 2018].

Future research directions present several exciting opportunities to extend the impact of eBPF-based traffic optimization beyond its current scope. One promising avenue involves integration with service mesh architectures, which have gained popularity in cloud-native deployments but introduce additional networking overhead. By creating specialized eBPF programs that optimize service mesh communication patterns, similar performance benefits could be achieved while maintaining the advanced traffic management and security features that service meshes provide. Another compelling direction is integration with 5G core networks, particularly for applications requiring ultra-reliable low-latency communication (URLLC). The 5G service-based architecture introduces complex networking requirements that could benefit significantly from kernel-level

Publisher: SARC Publisher

optimization, particularly for time-sensitive network functions. Research into adaptive traffic steering policies that automatically adjust based on network conditions and application requirements represents another valuable direction, potentially enabling more intelligent resource allocation during varying load conditions. Additionally, exploring the application of similar techniques to other telecommunications protocols beyond SIP could extend the benefits to a wider range of services. These research directions parallel ongoing work in understanding PCIe performance for end-host networking, where detailed performance modeling and optimization of the underlying hardware interfaces have revealed significant opportunities for improvement. Just as PCIe optimization research has demonstrated that understanding and tuning low-level system parameters can dramatically improve network performance, future eBPF research is likely to uncover additional optimization opportunities by more deeply integrating with hardware acceleration features and kernel subsystems. This convergence of software-defined networking with hardware-aware optimization represents a particularly promising direction for next-generation telecommunications infrastructure [Neugebauer, R. *et al*., 2018].

a) Media Path Optimization using eBPF

A promising extension of the current work involves applying similar eBPF-based optimization techniques to the media path of VoIP communications. While the current implementation focuses on SIP signaling optimization, Real-time Transport Protocol (RTP) media streams constitute the majority of VoIP traffic by volume and have their own unique performance requirements. Extending eTSF to optimize media paths could involve developing specialized eBPF programs that implement intelligent packet scheduling for RTP traffic, prioritizing it based on codec characteristics, network conditions, and application-specific requirements. Potential approaches include implementing direct kernel-level forwarding paths for RTP packets, applying selective Quality of Service (QoS) mechanisms at the XDP layer, and integrating with hardware offload capabilities to accelerate media processing. These optimizations could significantly reduce jitter, packet loss, and latency for media streams, further enhancing overall VoIP quality beyond the signaling improvements demonstrated in the current work.

b) Detailed Security Analysis

While the current implementation incorporates fundamental security controls, a comprehensive security analysis of eBPF-based traffic steering for telecommunications applications represents an important area for future investigation. This analysis would examine potential attack vectors unique to kernel-level packet processing in telecommunications contexts, including SIP protocol-specific attacks that might exploit the packet classification mechanism, privilege escalation risks in multi-tenant environments, and potential denial-of-service vulnerabilities in the eBPF programs themselves. Future work would also explore advanced security techniques such as formal verification of eBPF programs to mathematically prove their safety properties, runtime attestation mechanisms to ensure program integrity, and development of telecommunications-specific security policies for eBPF deployments. These security enhancements would address the critical requirements of telecommunications providers operating under strict regulatory frameworks and handling sensitive communications data.

c) Comparison with Other Technologies

A systematic comparison between eTSF and other high-performance packet processing technologies would provide valuable insights for telecommunications providers evaluating different optimization approaches. Future work should include benchmarking eTSF against Data Plane Development Kit (DPDK), which bypasses the kernel entirely to achieve high throughput but requires dedicated CPU cores; AF_XDP, which provides a high-performance socket implementation while maintaining the safety of the Linux networking stack; and Vector Packet Processing (VPP), which optimizes packet processing through vectorization techniques. This comparative analysis would evaluate not only raw performance metrics but also deployment complexity, operational overhead, compatibility with existing systems, and security implications. Understanding the relative strengths and weaknesses of these approaches in specific telecommunications contexts would help providers make informed decisions about which technology best suits their particular requirements and constraints.

The findings offer several recommendations for telecommunications providers adopting cloud-native VoIP technologies. First, kernel-level optimizations should be considered a critical component of any cloud-native VoIP strategy,

Publisher: SARC Publisher

rather than an optional enhancement, as they address fundamental performance bottlenecks that cannot be overcome through application-level optimizations alone. Second, providers should invest in developing expertise in eBPF and related kernel technologies, as these skills will become increasingly valuable as programmable networking continues to evolve. Third, performance testing methodologies should be updated to account for the unique characteristics of cloud-native deployments, including variable performance under different scaling conditions. Fourth, infrastructure designs should anticipate the need for kernel-level customization, potentially standardizing on kernel versions and configurations that support advanced networking features. Fifth, operational processes should be established for monitoring and maintaining kernel-level components alongside traditional application monitoring. Finally, providers should actively participate in the open-source communities developing these technologies to ensure telecommunications-specific requirements are considered in future development. By embracing these recommendations, providers can position themselves to leverage the full potential of cloud-native architectures while maintaining the performance, reliability, and security required for carrier-grade voice services. This approach acknowledges the lessons learned from both InKeV research, which highlights the value of kernel-level optimizations for networking performance, and PCIe performance research, which emphasizes the importance of understanding and optimizing the full stack from hardware interfaces through kernel components to application-level services. Together, these perspectives suggest that successful cloud-native VoIP deployments will require a holistic view of system performance that spans from the physical infrastructure through the kernel to the application services [Ahmed, Z. *et al*., 2018].

CONCLUSION

The eBPF-based Traffic Steering Framework represents a significant advancement in optimizing SIP signaling performance for cloud-native VoIP systems. By leveraging kernel-level packet processing capabilities, eTSF successfully addresses fundamental networking bottlenecks that plague containerized telecommunications deployments. The performance advantages stem from eliminating multiple processing stages, including netfilter traversals, NAT operations, and repeated context switches, resulting in substantial latency reductions and improved throughput across diverse deployment scenarios. These technical improvements translate directly to enhanced carrier-grade metrics and service quality that impact end-user experience. While certain limitations exist regarding kernel version requirements and security considerations, the architectural advantages of kernel-level traffic steering establish a promising foundation for future telecommunications infrastructure. Integration opportunities with service mesh architectures and 5G core networks offer exciting possibilities for extending these optimization techniques to broader telecommunications contexts. For providers adopting cloud-native VoIP technologies, embracing kernel-level optimizations alongside appropriate expertise development and infrastructure planning will be essential to maintaining carrier-grade performance while benefiting from cloud architecture advantages.

REFERENCES

1. Graf, T. “eBPF Documentary: eBPF’s Creation Story – Unlocking The Kernel.” *Isovalent*, (2023). https://isovalent.com/blog/post/ebpf-documentary-creation-story/
2. Kolhar, M., Alameen, A., & Gulam, M. “ Performance evaluation of framework of VoIP/SIP server under virtualization environment along with the most common security threats." *Neural Computing and Applications* 30.9 (2018): 2873-2881.
3. Kablan, M., Caldwell, B., Han, R., Jamjoom, H. and Keller, E. “Stateless network functions: breaking the tight coupling of state and processing.” *ACM Digital Library*, (2017).
4. Høiland-Jørgensen, T., Brouer, J.D., Borkmann, D., Fastabend, J., Herbert, T., Ahern, D. and Miller, D. "The express data path: Fast programmable packet processing in the operating system kernel." *Proceedings of the 14th international conference on emerging networking experiments and technologies*. (2018).
5. Kim, J., Jang, K., Lee, K., Ma, S., Shim, J. and Moon, S. "NBA (network balancing act) a high-performance packet processing framework for heterogeneous processors." *Proceedings of the Tenth European Conference on Computer Systems*. (2015).
6. Gandhi, R., Liu, H.H., Hu, Y.C., Lu, G., Padhye, J., Yuan, L. and Zhang, M. "Duet: Cloud scale load balancing with hardware and

Publisher: SARC Publisher

software." *ACM SIGCOMM Computer Communication Review* 44.4 (2014): 27-38.
7. Jin, X., Li, X., Zhang, H., Soulé, R., Lee, J., Foster, N., Kim, C. and Stoica, I. "Netcache: Balancing key-value stores with fast in-network caching." *ACM Digital Library, Proceedings of the 26th symposium on operating systems principles*. (2017).
8. Barbette, T., Soldani, C. and Mathy, L. "Fast userspace packet processing." *2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS). IEEE*, (2015).
9. Ahmed, Z., Alizai, M.H. and Syed, A.A. "Inkev: In-kernel distributed network virtualization for dcn." *ACM SIGCOMM Computer Communication Review* 46.3 (2018): 1-6.
10. Neugebauer, R., Antichi, G., Zazo, J.F., Audzevich, Y., López-Buedo, S. and Moore, A.W. "Understanding PCIe performance for end host networking." *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. (2018).

Source of support: Nil; **Conflict of interest:** Nil.

Cite this article as:
Munnaluru, P.J,K. "Optimizing SIP Signaling Latency in Cloud-Native VoIP Systems Using eBPF-Based Traffic Steering." *Sarcouncil Journal of Engineering and Computer Sciences* 4.6 (2025): pp 208-221.

Publisher: SARC Publisher