C2N: Bridging CAMARA Service APIs and 3GPP Core Network Exposure Function

Rafael Rodrigues Silva[1], **João Paulo Esper**[1], **Leandro C. de Almeida**[2], **Fábio L. Verdi**[3], **Kleber V. Cardoso**[1]

[1]Instituto de Informática (INF) – Universidade Federal de Goiás (UFG)

[2]Unidade Acadêmica de Informática – Instituto Federal da Paraíba (IFPB)

[3]Departamento de Computação (Dcomp) – Universidade Federal de São Carlos (UFSCar)

{silva_rafael, joaopauloesper}@discente.ufg.br, kleber@ufg.br

leandro.almeida@ifpb.edu.br, verdi@ufscar.br,

Abstract. *The exposure of 5G network capabilities to third-party applications is fundamental for realizing the Network-as-a-Service (NaaS) paradigm. However, the complexity of native 3GPP interfaces, particularly the Network Exposure Function, creates a significant barrier for application developers. The CAMARA project addresses this challenge by standardizing developer-friendly service APIs, yet the Transformation Function that is responsible for translating these APIs into network-native interfaces remains underspecified and typically implemented as a proprietary black box. This paper presents C2N, an open-source Transformation Function for the CAMARA Traffic Influence API. We propose a modular architecture comprising seven components that implement a three-phase semantic mapping strategy: direct mapping; logical transformation with topology enrichment; and structural mapping. This approach bridges the abstraction gap between CAMARA's developer-oriented interface and the 3GPP specification of the Network Exposure Function Northbound APIs. We validate C2N using a 5G testbed built with free5GC and UERANSIM, demonstrating successful traffic steering from a latency-degraded path to an alternative performance-satisfactory one. We make publicly available the necessary code and information for reproducibility and provide a reference for researchers investigating API translation strategies in the mobile networks NaaS ecosystem.*

1. Introduction

The telecommunications industry is undergoing a fundamental transformation, evolving from providing basic connectivity to becoming a platform for digital innovation. This evolution, often termed Network-as-a-Service (NaaS), is predicated on exposing granular network capabilities through Application Programming Interfaces (APIs), creating new monetization opportunities for 5G investments and enabling third-party developers to leverage network intelligence in their applications [Ordonez-Lucena and Dsouza 2022].

Historically, network APIs were complex and proprietary, creating a fragmented landscape that hindered developer adoption. To overcome this fragmentation, the GSMA Open Gateway initiative, in collaboration with the Linux Foundation's CAMARA project [GSMA 2023], is standardizing a set of global, developer-friendly APIs. The goal is to abstract the underlying network intricacies, allowing developers to integrate network capabilities without requiring deep expertise in telecommunications protocols.

The 5G Core (5GC), with its Service-Based Architecture (SBA), provides the technical foundation for the 5G network programmability. The standardized entry point for external applications is the Network Exposure Function (NEF), which offers a secure interface for authorized applications to request network information or influence network behavior. The NEF's capabilities are specified by 3GPP in [3GPP 2019b], including traffic influence, Quality of Service (QoS) control, and monitoring events. However, configuring these capabilities requires knowledge of parameters such as S-NSSAI (Single Network Slice Selection Assistance Information), DNN (Data Network Name), and flow descriptions using 3GPP-specific formats, i.e., a steep learning curve for developers not versed in telecommunications technologies.

The CAMARA architecture tackles this complexity gap with a component known as the Transformation Function, which translates developer-friendly service API calls into the corresponding network API operations. The Transformation Function is not limited to interfacing with the 3GPP NEF, encompassing integration with multiple backend platforms that include Operations and Management (OAM) systems via TMF Open APIs, cloud infrastructure through Cloud Native Computing Foundation (CNCF) and Network Functions Virtualization (NFV) interfaces, and other network exposure points. This multi-platform capability enables unified service API access across heterogeneous operator environments. Although the CAMARA specifications define the service API contracts and outline the Transformation Function's role at a conceptual level, they leave the concrete design of this translation layer largely open. As a result, each Communication Service Provider (CSP) creates its own proprietary implementation aligned with its particular network environment, leading to a lack of transparency that constrains reproducibility, obstructs interoperability research, and prevents the academic community from systematically exploring translation approaches and their performance trade-offs.

This paper addresses this gap by presenting C2N (CAMARA-to-NEF), an open-source modular architecture for implementing the Transformation Function, validated through the Traffic Influence API targeting the 3GPP NEF. The Traffic Influence API enables applications to request that user plane traffic be steered toward specific network locations, a capability essential for latency-sensitive applications that benefit from edge computing resources. We selected Traffic Influence as the validation case because it exercises all three categories of semantic transformation (direct, logical, and structural), providing comprehensive architectural validation. C2N realizes a seven-component gateway architecture designed to separate API-agnostic infrastructure from API-specific translation logic. While the current implementation targets the 3GPP NEF, the modular design accommodates extension to other Network API backends, enabling future integration with OAM platforms, cloud orchestrators, or hybrid environments. We validate C2N on an open-source 5G testbed and demonstrate that Traffic Influence subscriptions processed via our gateway successfully redirect data traffic between User Plane Functions (UPFs).

The main contributions of this paper are: **(1)** a detailed analysis of the semantic mapping requirements between CAMARA Service APIs and 3GPP NEF interfaces, identifying direct, logical, and structural transformation categories that generalize across different APIs; **(2)** the design of a modular, seven-component gateway architecture that deliberately separates reusable infrastructure from API-specific components, where architectural analysis indicates approximately 70% of the modules is API-agnostic; and **(3)** an open-source implementation of C2N validated through the Traffic Influence API, providing a reference that can be extended to additional CAMARA APIs and adapted to different Network API backends.

Experimental results demonstrate that C2N effectively translates CAMARA requests while introducing modest overhead. The translation layer adds 3.2 ms of processing time, representing 28% of the synchronous API latency ($\sim$12 ms total), with the remaining 72% attributable to 5G Core operations. Functional validation confirms latency improvement when Traffic Influence redirects traffic from a degraded path to an optimized route. Analysis of the control plane signaling reveals that the complete end-to-end activation requires approximately 172 ms, dominated by an iterative user plane configuration loop (87% of asynchronous latency) where the Session Management Function programs the data plane. Additionally, the CAMARA Service API request (205 bytes) nearly doubles in size after translation to the 3GPP Network API format (402 bytes), illustrating the semantic expansion that the Transformation Function abstracts from developers.

The remainder of this paper is organized as follows. Section 2 presents the theoretical background on the network exposure ecosystem. Section 3 reviews related work. Section 4 details the C2N modular architecture. Section 5 describes our experimental evaluation through the Traffic Influence API. Finally, Section 6 concludes the paper and discusses future work.

2. Background

The 5G Core architecture adopts a Service-Based Architecture (SBA) where network functions expose their services through standardized interfaces. The Network Exposure Function (NEF) serves as the standardized entry point for external Application Functions (AFs) to interact with the 5G Core. Its capabilities, specified in [3GPP 2019b], include Traffic Influence for routing user plane traffic to specific Data Network Access Identifiers (DNAIs), Quality of Service (QoS) Control for session-level quality management, and Monitoring Events for network state observation. However, configuring these capabilities requires knowledge of telecommunications-specific parameters such as S-NSSAI (Single Network Slice Selection Assistance Information), DNN (Data Network Name), DNAI, and IP 5-tuple flow descriptions—presenting a significant barrier for developers unfamiliar with 3GPP standards.

CAMARA is a joint initiative between the GSMA and the Linux Foundation that develops open-source, developer-friendly APIs under the Open Gateway framework [Ordonez-Lucena and Dsouza 2022]. The project distinguishes between two API categories: *Network APIs*, which are interfaces implemented by network functions as defined by standards bodies (e.g., 3GPP NEF APIs in TS 29.522), closely tied to underlying technology; and *Service APIs*, which are developer-facing interfaces designed for third-party consumption, abstracting network complexity through intuitive parameters. For example, while the NEF requires explicit S-NSSAI, DNN, and DNAI values for traffic steering, the CAMARA Traffic Influence API allows developers to specify an `edgeCloudZoneId` representing the target edge location instead. This distinction between Network and Service APIs underpins the emerging paradigm of vertical services, where network applications serve as middleware leveraging exposed northbound APIs to deliver tailored services to vertical industries [Charpentier et al. 2025].

The CAMARA architecture bridges Service and Network APIs through two components: the API Gateway (implemented via CAPIF—Common API Framework for 3GPP Northbound APIs) and the Transformation Function. CAPIF, provides API publication, discovery, authentication, and access control through its Core Function (CCF) and API Exposing Function (AEF) [3GPP 2024, 3GPP 2020b]. The Transformation Function operates behind the AEF, translating validated Service API requests into Network API calls.

According to CAMARA documentation, the Transformation Function must fulfill several requirements: *parameter translation* to convert high-level identifiers into network-specific parameters; *topology resolution* to map abstract location identifiers (e.g., edge zones) into network identifiers (DNAIs, DNNs); *protocol adaptation* to handle differences in request/response formats; *state management* to maintain bidirectional mappings between Service and Network API resource identifiers; and support for *1-to-N mapping* scenarios where one Service API call requires multiple Network API operations. While CAPIF is well-specified by 3GPP, the Transformation Function's concrete implementation remains underspecified—motivating this paper's contribution.

3. Related Work

The GSMA Open Gateway initiative addresses the historical fragmentation of network APIs across operators [Sabella et al. 2021]. Prior to this initiative, each Communication Service Provider (CSP) exposed network capabilities through proprietary interfaces, hindering developer adoption. Open Gateway establishes a common framework enabling developers to write applications once and deploy them consistently across different networks, defining a catalog of standardized APIs covering device location, quality of service, number verification, and traffic influence [Charpentier et al. 2024]. Several research efforts have addressed API gateways for 5G networks. A microservice and event-driven NEF (MNEF) integrating CAPIF support with the 5G Core is presented in [Fernandes et al. 2024], demonstrating interoperability with Open5GS and OpenCAPIF through a cloud-native architecture featuring purpose-built microservices for Traffic Influence, QoS Control, and Analytics Exposure. While this work addresses scalability at the network API level with detailed performance evaluation under high-load scenarios, it does not address translation to developer-facing service APIs. Deep application-network integration architectures are explored in [Serracanta et al. 2024], identifying the need for abstraction layers without providing implementation details for the translation mechanisms.

Regarding CAMARA API implementations, the Quality on Demand (QoD) API is presented in [Ordonez-Lucena and Dsouza 2022] as the first CAMARA API validated in production, demonstrating the translation of Quality of Service (QoS) profiles into 3GPP parameters. This work provides conceptual foundations for our approach but focuses exclusively on the QoD API rather than Traffic Influence. Dynamic QoS optimization through CAMARA interfaces has been demonstrated in real-world 5G SA networks [Charpentier et al. 2024], where network applications modified QoS profiles via the CAMARA QoD API to adjust uplink data rates based on geofenced areas. This validation confirms the viability of CAMARA-based network programmability for vertical industries such as transport and logistics, though the internal translation mechanisms remain unexposed. More recently, the evolution toward 6G applications has been outlined [Charpentier et al. 2025], proposing that future network applications should leverage network exposure, AI/ML, and edge computing to deliver vertical services with features such as network awareness, location awareness, and intelligent decision-making. This vision reinforces the importance of transparent Transformation Function implementations that can evolve alongside advancing network capabilities.

Recent work has extended CAMARA adoption beyond mobile networks to optical infrastructure. A NaaS platform for F5G-Advanced optical networks implementing CAMARA QoD and Edge Cloud APIs is presented in [Rahimi et al. 2025], demonstrating end-to-end slice orchestration across access (PON) and transport (fgOTN) domains using ETSI TeraFlowSDN and OSM. This platform features a hierarchical control architecture

with dynamic service differentiation between best-effort (IP) and premium (optical) traffic paths. While this work validates CAMARA adoption in optical networks with sophisticated multi-domain orchestration, the Transformation Function responsible for translating Service APIs to Network APIs remains encapsulated within the orchestration layer, without exposing implementation details or mapping methodologies.

In the domain of NEF-based network control, the Edge Enabler Layer (EEL) architecture is presented in [Kim et al. 2023], enabling edge applications to interact with the 5G Core through standardized interfaces. Vertical Application Enablers (VAEs) are investigated in [Fragkos et al. 2021], proposing cloud-native approaches but focusing primarily on network-facing components rather than developer-facing abstractions. QoS prediction for UAV optimization using 3GPP APIs is explored in [Makropoulos et al. 2023], demonstrating the potential of network exposure while highlighting the complexity barriers that developers face.

The literature reveals that while significant work exists on API gateways, CAMARA validation through commercial platforms, and NEF-based control across both mobile and optical domains, the Transformation Function's implementation details remain largely unexplored. Existing implementations rely on proprietary platforms or encapsulate the translation logic within orchestration components without exposing generalizable architectural patterns. C2N addresses this gap through three key differentiators: first, it provides an open-source reference implementation with detailed architectural decomposition and documented mapping methodology; second, it explicitly separates reusable infrastructure from API-specific logic, enabling systematic study of cross-API patterns; and third, it includes experimental validation using entirely open-source infrastructure, ensuring reproducibility.

4. Modular Architecture for CAMARA-to-NEF Translation

C2N implements a modular architecture designed to support multiple CAMARA APIs while separating reusable infrastructure from API-specific translation logic.The implementation is publicly available as open-source software[1] to enable reproducibility and community contribution. The current implementation validates this architecture through the Traffic Influence API, translating CAMARA requests into the NEF's interface specified in [3GPP 2019b]. Figure 1 illustrates C2N's position within the CAMARA ecosystem. C2N operates as the Transformation Function within the API provider domain, where external applications interact through CAMARA Service APIs authenticated via the CAPIF Core Function (CCF).

As depicted in Figure 1, the Transformation Function can interface with multiple Network API backends beyond the 3GPP NEF: Operations and Management (OAM) systems through TM Forum (TMF) Open APIs for service assurance and fault management; cloud infrastructure via CNCF-aligned interfaces and NFV orchestrators for resource provisioning; and the 3GPP network functions for core network exposure. This multi-backend capability reflects the heterogeneous nature of modern operator environments, where a single CAMARA Service API invocation may require coordination across network, cloud, and operations domains. The architecture follows four design principles: (i) modularity, where each concern is encapsulated in dedicated modules with well-defined interfaces; (ii) separation of concerns, with distinct components for validation, topology resolution, mapping, and state management; (iii) standards compliance, where input validation enforces

[1] https://github.com/LABORA-INF-UFG/C2N

CAMARA constraints and output adheres to backend-specific specifications; and (iv) extensibility, enabling new API-specific mappers and backend adapters to be implemented while reusing the common infrastructure.

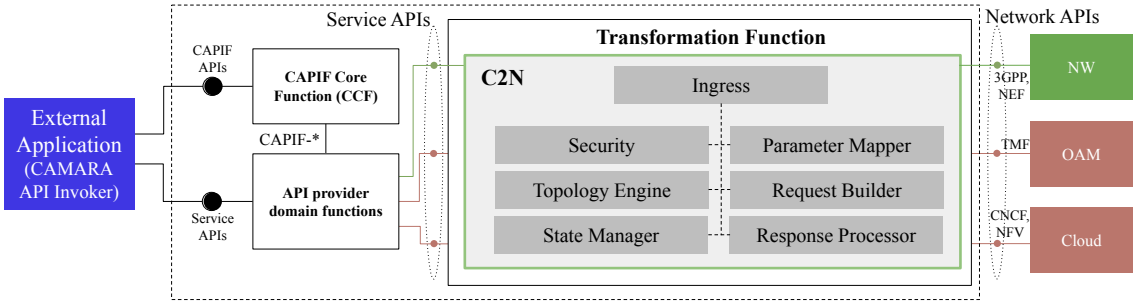


Figure 1. C2N architectural overview within the CAMARA ecosystem. The Transformation Function bridges Service APIs with Network APIs exposed by 3GPP NEF (NW), Operations and Management systems via TMF interfaces (OAM), and cloud infrastructure through CNCF/NFV interfaces (Cloud).

C2N comprises seven modules organized into four processing layers. The **Security Module** handles authentication and authorization, including token signature verification, OAuth 2.0 claim extraction, RBAC enforcement, rate limiting, CORS policy enforcement, and audit logging. The **Ingress Module** parses and validates incoming requests against OpenAPI specifications, ensuring required fields are present, identifiers conform to expected formats, and parameters are valid. Validation errors return compliant responses with appropriate HTTP status codes.

The **Topology Engine** resolves abstract CAMARA identifiers into network-specific parameters. For Traffic Influence, this involves mapping `edgeCloudZoneId` to DNAI, DNN, and S-NSSAI values. These mappings are configurable via JSON files, supporting policy-based resolution where different regions map to different network parameters. The **Parameter Mapper** implements core translation through a three-phase strategy: Phase 1 performs direct mapping for fields with direct correspondence such as `appId` to `afServiceId` and `notificationUri` to `notificationDestination`; Phase 2 executes logical transformation with topology-enriched resolution; Phase 3 handles structural mapping for complex conversions, including device information to UE indicators and traffic filters to `FlowInfo` structures with IP 5-tuple format.

The **Request Builder** constructs NEF-compliant HTTP requests through JSON serialization, endpoint URL construction, and required headers including authentication and 3GPP-specific headers. It supports all CRUD operations with configurable timeouts and connection pooling. The **State Manager** maintains bidirectional mappings between CAMARA resource identifiers and NEF subscription identifiers for CRUD operations, implementing thread-safe storage, lifecycle tracking, correlation ID generation, and statistics collection. The **Response Module** transforms NEF responses to CAMARA format, extracting relevant identifiers and constructing compliant responses. Error responses are translated with relevant problem details.

Table 1 presents the field mappings between CAMARA Traffic Influence API and NEF interfaces. The most complex transformation involves traffic filters. CAMARA provides a simplified model with separate source and destination objects, while NEF requires FlowInfo structures with flow descriptions in IP filter rule format following the pattern `permit out <proto> from <src> to <dst>`.

Table 1. CAMARA to NEF Field Mappings for Traffic Influence.

CAMARA Field	NEF Field	Type
`appId`	`afServiceId`	Direct
`notificationUri`	`notifDestination`	Direct
`edgeCloudZoneId`	`dnn`	Logical
`edgeCloudZoneId`	`snssai.sst/sd`	Logical
`edgeCloudZoneId`	`trafficRoutes[].dnai`	Logical
`device` (absent)	`anyUeInd: true`	Structural
`device.phoneNumber`	`gpsi`	Structural
`device.ipv4Address`	`ipv4Addr`	Structural
`trafficFilters`	`FlowInfo[]`	Structural

Figure 2 illustrates the message flow for Traffic Influence subscription creation. The flow proceeds through four layers: first, the external application sends a POST request through the AEF (CAPIF), which validates the token and forwards to C2N; second, the Security and Ingress Layer performs token verification, claim extraction, RBAC validation, and payload parsing against the OpenAPI specification; third, the Enrichment and Mapping Layer where the Topology Engine resolves network context and the Parameter Mapper executes three-phase translation; fourth, the Request Building and State Layer where the Request Builder constructs the NEF payload, the State Manager generates correlation IDs, and the request is sent to NEF. NEF processes the subscription, configures policies, and returns `201 Created`. Finally, the Response Translation Layer stores ID mappings in the State Manager while the Response Module transforms the response to CAMARA format.

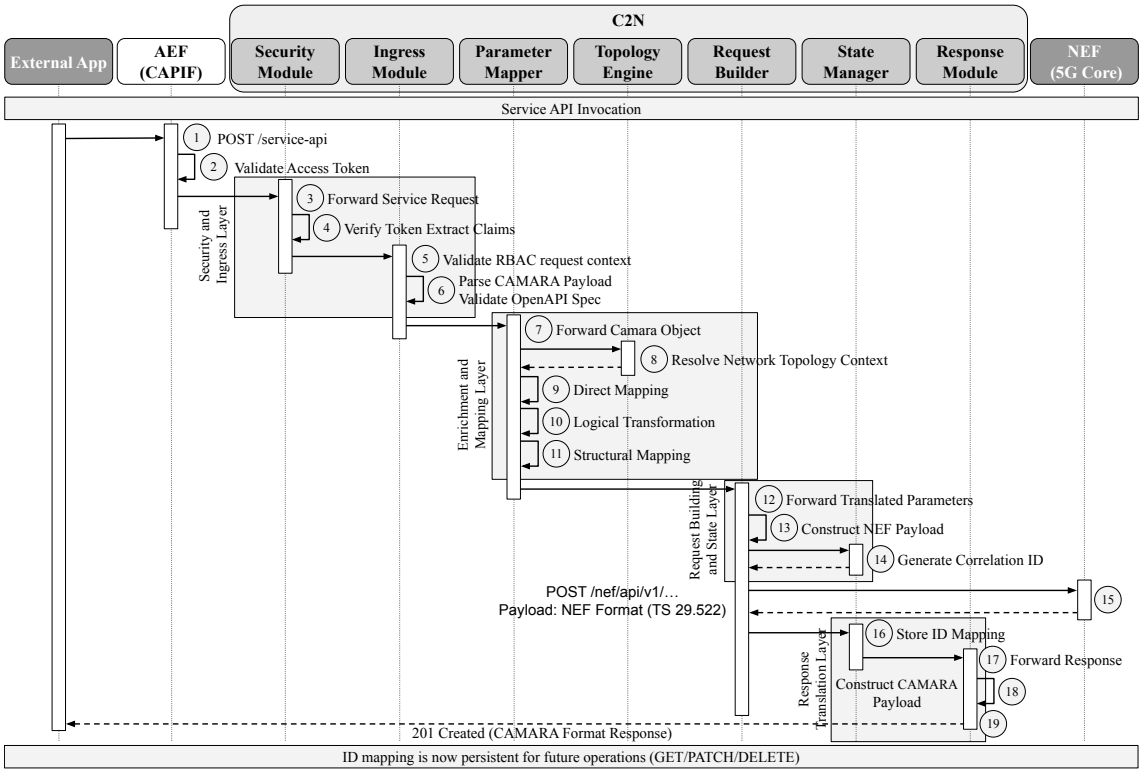


Figure 2. Sequence diagram for Traffic Influence subscription creation showing the external application's request traversing CAPIF authentication and C2N's internal processing modules.

A key architectural contribution is the deliberate separation between API-agnostic

and API-specific components. Architectural analysis of the Traffic Influence implementation indicates that five of the seven modules—Security, Ingress, Request Builder, State Manager, and Response Module—are fully reusable across different CAMARA API implementations, representing approximately 70% of the modules. These modules handle concerns common to any Service-to-Network API translation: authentication, request validation, HTTP client operations, identifier correlation, and response formatting. Only the Topology Engine and Parameter Mapper require API-specific implementations, as they encode the semantic mappings particular to each Service API. This separation is by design: when implementing additional APIs such as Quality on Demand or Device Location, developers need only implement these two components while reusing the remaining infrastructure. While this reusability claim is based on architectural analysis rather than empirical validation across multiple APIs, the modular boundaries were specifically designed to maximize cross-API reuse.

5. Experimental Evaluation

This section presents the experimental validation of C2N through the Traffic Influence API. The evaluation addresses two complementary objectives: (1) *functional validation*, demonstrating that C2N correctly translates CAMARA requests into NEF subscriptions that effect actual traffic steering; and (2) *performance characterization*, quantifying the processing overhead introduced by the translation layer and analyzing the end-to-end signaling flow in the 5G Core.

C2N is implemented in Go using the Gin web framework. The implementation uses minimal dependencies, relying primarily on Gin for HTTP routing and the standard library for JSON processing and HTTP client operations. Configuration is managed through environment variables for deployment parameters and JSON files for topology mappings, enabling adaptation to different network environments without code modifications.

5.1. Testbed Environment

Figure 3 illustrates the testbed architecture and the API translation process. The experimental infrastructure consists of a complete 5G Core network based on free5GC[2], comprising the NEF, Unified Data Repository (UDR), Policy Control Function (PCF), Session Management Function (SMF), and two UPFs with distinct roles: UPF1 serves as the PDU Session Anchor for default routing (Path 1), while UPF2 acts as a branching point for local breakout to the App server (Path 2). RAN emulation uses UERANSIM[3] for gNB and UE simulation. The C2N Gateway runs as a containerized service, translating CAMARA Service API requests into 3GPP Network API calls.

The message snippets in Figure 3 illustrate the complexity gap that C2N bridges. The CAMARA Service API request (top) contains 205 bytes with intuitive fields such as `appId`, `edgeCloudZoneId`, and `device.ipv4Address`. The corresponding 3GPP Network API request (bottom) nearly doubles in size to 402 bytes, requiring network-specific parameters including `dnn`, `snssai` (with SST and SD components), `trafficRoutes` with DNAI, and `flowDescriptions` in IP filter rule format. Even in this simple scenario, with a single device requesting traffic steering to one edge zone, the structural and semantic expansion is substantial. This asymmetry underscores the value of the Transformation Function: developers specify intent in approximately half the payload size, while C2N handles the verbose 3GPP parameter construction.

[2]https://free5gc.org/

[3]https://github.com/aligungr/UERANSIM

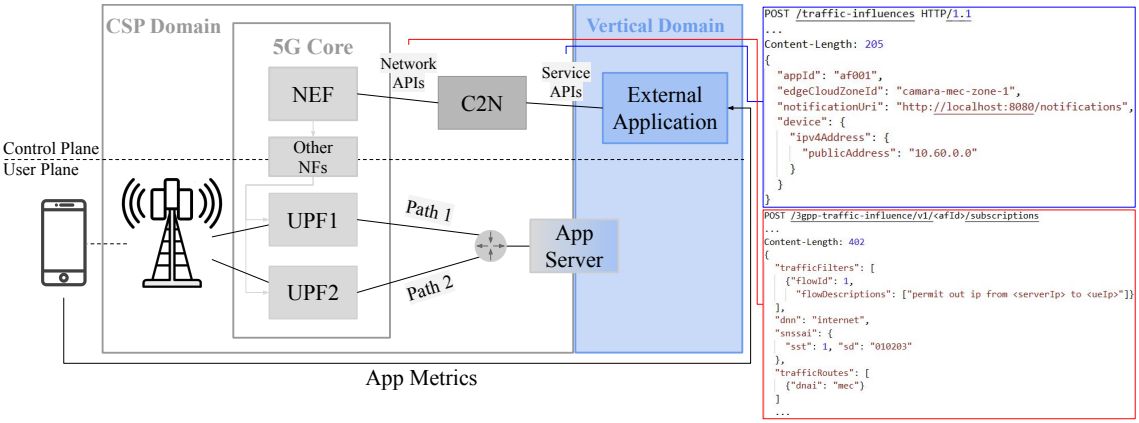


Figure 3. Experimental testbed architecture.

The default route through UPF1 (Path 1) includes an artificial baseline latency of approximately 10 ms, introduced by Linux traffic control (tc) to represent traditional metropolitan network conditions. During the experiments, the latency receives an additional 40 ms to represent an event such as a heavy traffic condition. Table 2 summarizes the configuration parameters and software versions.

In the experimental setup, CAPIF components (CCF and AEF) were not instantiated; the external application submitted requests directly to C2N, with token validation performed using a locally configured key to simulate the post-AEF authenticated context assumed by the Transformation Function.

Table 2. Testbed Configuration.

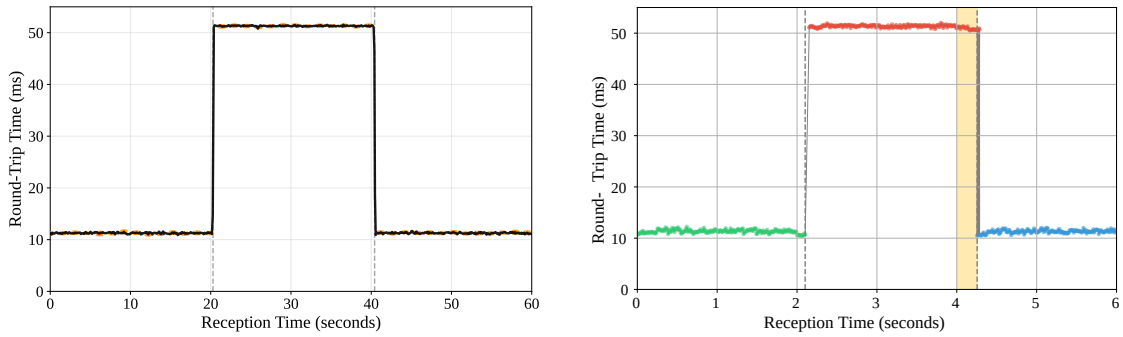
Component	Configuration
5G Core	free5GC v4.1.0
RAN Simulator	UERANSIM v3.2.7
C2N Gateway	Go 1.21
Baseline latency	10 ms (a common metropolitan delay)
Additional delay	+ 40 ms (a heavy traffic event)
Traffic generation	fping (100 pps)

5.2. Functional Validation

We designed a three-phase experimental protocol to evaluate C2N's capability to translate CAMARA Traffic Influence requests into effective traffic steering decisions. Phase 1 (Baseline) routes traffic through the default path via UPF1 (Path 1), establishing baseline latency measurements. Phase 2 (Network Degradation) introduces additional delay to the default route, representing a network event (e.g., a congestion). Phase 3 (Traffic Influence Activation) invokes the C2N Traffic Influence API to redirect traffic from the degraded path to a route similar to the default under normal conditions via UPF2 (Path 2).

Two complementary experiments were conducted using ICMP probes at 100 packets per second (10 ms interval): a 60-second experiment (6,000 packets, 20 seconds per phase) providing statistical validation as shown in Figure 4(a); and a 6-second experiment (600 packets, 2 seconds per phase) offering detailed visibility into phase transition behavior as presented in Figure 4(b). The 60-second experiment shows mean RTT with 95% confidence intervals calculated over 10-packet sliding windows, while the 6-second

experiment displays individual packet RTT values in reception order. The results confirm that C2N successfully translates CAMARA Traffic Influence requests into NEF subscriptions that effect actual routing changes. The near-identical RTT between Phase 1 and Phase 3 illustrates the successful redirection from the degraded path to the better route (Path 2). As illustrated in Figure 4(b), during the transition (highlighted by the orange stripe) from Phase 2 (in red) to Phase 3 (in blue), there is a delay of approximately 200 ms (corresponding to the control plane signaling latency detailed in Section 5.3), followed by simultaneous packet flows through both paths for a short period of time. This happens because some packets already in flight through the old route continued arriving while more recent packets used the new route through Path 2. It is an expected behavior during traffic redirection and demonstrates the transient system behavior while the Traffic Influence policy is effectively applied.



(a) 60-second experiment showing mean RTT and 95% confidence intervals over 10-packet windows.

(b) 6-second detailed experiment showing individual packet RTT values.

Figure 4. RTT measurements demonstrating Traffic Influence API effectiveness.

5.3. Control Plane Signaling Analysis

To characterize the control plane behavior and measure C2N's processing overhead, we instrumented the testbed to capture high-resolution timestamps from all network function logs. Figure 5 illustrates the whole signaling sequence initiated when an external application invokes the CAMARA Traffic Influence API. The behavior follows the procedure specified in [3GPP 2019a] for processing AF requests to influence traffic routing.

The procedure spans three administrative domains and unfolds across three distinct phases. *Phase 1 (Subscription Creation – Synchronous)* begins when the external application sends an HTTP POST to C2N's `/traffic-influences` endpoint. C2N performs the three-stage translation, including the mappings specified in [3GPP 2019a]: from AF-Service-Identifier to DNN/S-NSSAI, from zone identifiers to DNAI, and from device identifiers to UE addressing formats. Next C2N forwards the request to the NEF via the `Nnef_TrafficInfluence_Create` service operation. The NEF persists the AF request information in the UDR (Data Set: Application Data; Data Subset: AF traffic influence request information) and returns a `201 Created` response. This synchronous phase completes in nearly 12 ms, representing the latency perceived by the application.

Phase 2 (Policy Control – Asynchronous) follows the 3GPP publish-subscribe mechanism: PCFs that have subscribed to modifications of AF requests receive a `Nudr_DM_Notify` notification from the UDR. The PCF processes the influence data and

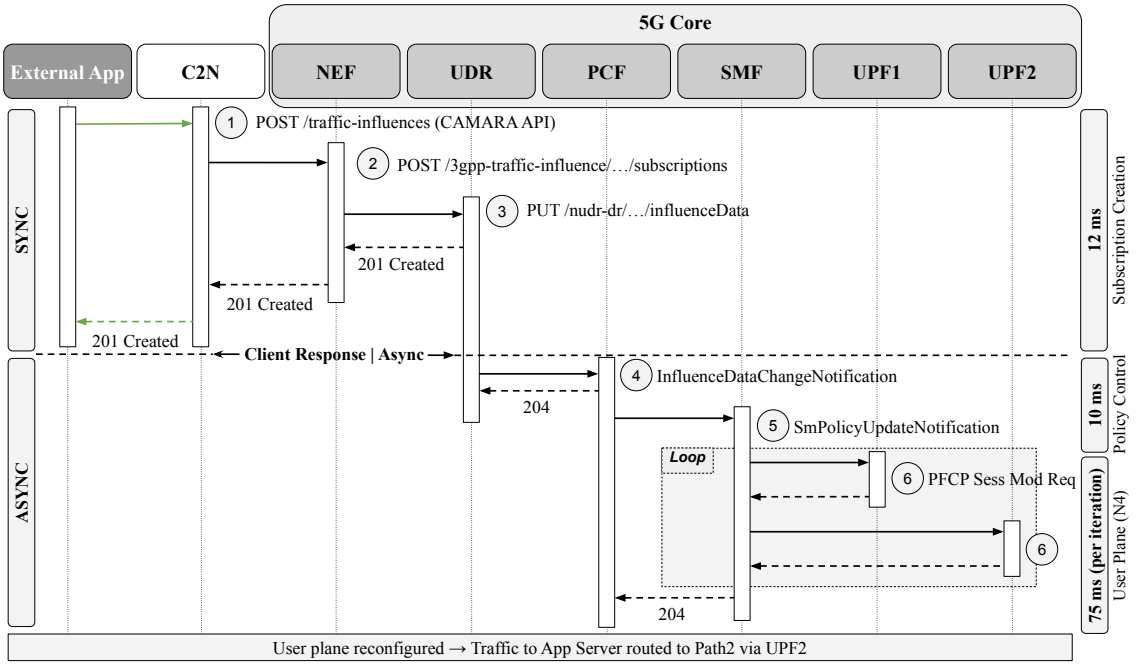


Figure 5. 5G Core signaling flow for Traffic Influence activation.

generates PCC rules encoding the traffic steering decision, delivering them to the SMF via `Npcf_SMPolicyControl_UpdateNotify`. This phase completes in nearly 10 ms.

Phase 3 (User Plane Configuration – Asynchronous) translates PCC rules into data plane programming commands. The SMF communicates with the UPFs over the N4 interface through the PFCP (Packet Forwarding Control Protocol), sending Session Modification Requests to both UPFs. As specified in [3GPP 2019a], the SMF may add or replace a UPF to act as a Branching Point and update traffic steering rules at the target DNAI. This SMF-UPF signaling occurs as an *iterative loop* rather than a single transaction. Across all test executions, we consistently observed two iterations of the PFCP exchange cycle. Each iteration comprises: (i) the PCF sending an `SmPolicyUpdateNotification` to the SMF; (ii) the SMF issuing parallel PFCP Session Modification Requests to both UPFs; (iii) the UPFs responding with Session Modification Responses; and (iv) the SMF acknowledging completion to the PCF. Each iteration requires approximately 75 ms, resulting in a total *Phase 3* duration of 150 ms.

Table 3 presents the C2N overhead extracted from the logs of our instrumented testbed. The translation overhead is just over 3 ms, representing less than 28% of the Phase 1 latency and less than 2% of the total latency of the three phases. It is worth highlighting that this time frame considered the complete operation of C2N, encompassing input processing (authentication, validation, three-phase translation) and output processing (transformation of the response).

5.4. Discussion

The occurrence of multiple iterations of the PFCP exchange cycle is attributable to the PCC rule processing logic within the PCF. When the influence data triggers modifications to multiple PCC rule groups, the PCF generates separate notification messages for each group, causing the SMF to execute distinct PFCP transaction batches. In our test configuration, the Traffic Influence subscription consistently resulted in two notification cycles, suggesting that the free5GC PCF implementation partitions the generated rules into two groups.

Table 3. Synchronous phase latency decomposition.

Component	Duration	Proportion
C2N ingress (authentication + translation)	2.85 ms	24.6%
5G Core (NEF + UDR)	8.36 ms	72.2%
C2N egress (response handling)	0.37 ms	3.2%
C2N overhead (total)	**3.22 ms**	**27.8%**
Synchronous latency	**12 ms**	100%

Regarding this SMF-UPF signaling behavior, the observed two-iteration pattern is characteristic of free5GC's PCC rule grouping strategy, and commercial 5G Core implementations may exhibit different iteration counts. Nevertheless, the fundamental signaling flow is mandated by 3GPP specifications and thus expected to be consistent across compliant implementations, with variations primarily in timing rather than in the overall procedure. While validated through Traffic Influence, the architectural principles are designed to apply to other CAMARA APIs.

The 172 ms total end-to-end latency is dominated by the N4 loop (87% of asynchronous latency), with the two-iteration pattern being characteristic of free5GC's PCC rule grouping strategy. This suggests that latency-critical applications should activate Traffic Influence proactively, upon session establishment or geofence entry, rather than reactively when low-latency routing is immediately required. The 12 ms synchronous latency enables applications to receive rapid confirmation without blocking on user plane configuration, supporting real-time decision-making workflows.

The evaluation used ICMP traffic only, although 3GPP Traffic Influence supports TCP/UDP flow steering through Session and Service Continuity (SSC) modes and Uplink Classifier (ULCL)/Branching Point (BP) mechanisms according to [3GPP 2020a]. Our testbed configuration did not allow us to observe TCP/UDP traffic being redirected in real time, limiting the experimental scope to connectionless traffic for validating the steering mechanism.

The dual-UPF topology represents a minimal branching point configuration sufficient to validate the Traffic Influence mechanism because production networks typically feature additional UPFs, hierarchical topologies, and geographic distribution. However, since C2N operates exclusively at the control plane level (translating API requests and receiving responses), the data plane topology complexity does not affect translation correctness, and additional UPFs would increase only the PFCP configuration phase duration, not the C2N overhead or the validation of the mapping methodology.

6. Conclusion

This paper presented C2N, a modular open-source architecture for implementing the CAMARA Transformation Function, validated through the Traffic Influence API. The work makes three contributions: (i) a systematic three-phase mapping methodology (direct, logical, and structural transformations) that provides a conceptual framework for Service-to-Network API translation; (ii) a seven-component architecture that deliberately separates API-agnostic infrastructure from API-specific and backend-specific logic; (iii) an open-source implementation that enables reproducibility and community extension and a comprehensive experimental validation demonstrating successful traffic steering with detailed characterization of the translation overhead and control plane signaling behavior.

The experimental evaluation addressed both functional validation and performance

characterization. The signaling analysis, aligned with [3GPP 2019a], revealed that the C2N translation layer introduces negligible overhead in comparison to the other necessary procures, mainly user plane configuration. Architectural analysis indicates that five of seven modules are designed to be reusable across different CAMARA API implementations and backend platforms. These modules handle concerns common to any translation scenario, while only the Topology Engine and Parameter Mapper require API-specific and backend-specific implementations. Although the current validation targets the 3GPP NEF, the architecture accommodates extension to other Network API backends, including TMF OAM systems and cloud infrastructure via CNCF/NFV interfaces, reflecting the multi-platform nature of the CAMARA ecosystem. By making the Transformation Function transparent and proper documented, C2N enables researchers to analyze translation strategies, compare approaches, and build upon this foundation.

We consider some directions as promising future work. Implementing additional CAMARA APIs would extend the validated architecture to cover a broader API portfolio, potentially revealing additional shared patterns across APIs. For example, Quality on Demand, Device Location, and Network Slice Booking are potential priority candidates. Extending the implementation to alternative backend platforms (TMF, CNCF) would demonstrate the multi-platform capabilities already designed into the architecture. From an evaluation perspective, application-level scenarios such as DASH/HLS streaming would complement the current validation by exercising Traffic Influence in conjunction with application session lifecycle management. From an operational perspective, Kubernetes-native deployment, service mesh integration, and ETSI ZSM alignment would enable automated network configuration in production environments. Full CAMARA compliance requires integration with OpenCAPIF or similar frameworks for complete API lifecycle management. Performance optimization through load testing with multiple concurrent UEs would assess scalability characteristics.

Acknowledgments

This work was partially supported by MCTIC/CGI.br/FAPESP under grant no. 2020/05127-2 (SAMURAI project), by CNPq under grant no. 306283/2025-5, and by RNP/MCTIC under grant no. 01245.020548/2021-07 (Brasil 6G project). The first author also thanks INATEL for the institutional support during this work.

References

3GPP (2019a). TS 23.502: Procedures for the 5G system (5GS). Technical report, 3rd Generation Partnership Project (3GPP). Release 15.

3GPP (2019b). TS 29.522: 5G system; network exposure function northbound APIs; stage 3. Technical report, 3rd Generation Partnership Project (3GPP). Release 15.

3GPP (2020a). TS 23.501: System architecture for the 5g system (5GS). Technical report, 3rd Generation Partnership Project (3GPP). Release 16.

3GPP (2020b). TS 29.122: T8 reference point for northbound APIs. Technical report, 3rd Generation Partnership Project (3GPP). Release 16.

3GPP (2024). TS 29.222: Common API framework for 3GPP northbound APIs; stage 3. Technical report, 3rd Generation Partnership Project (3GPP). Release 18.

Charpentier, V., Landi, G., Giannopoulou, E., Brenes, J., Camelo, M., Marquez-Barja, J. M., and Slamnik-Kriještorac, N. (2025). Advancing Vertical Services for 6G: Future Directions and Innovations. *IEEE Network*.

Charpentier, V., Slamnik-Kriještorac, N., Akintola, A., Gasparroni, M., Obasola, B., Bruynseels, L., Gjorgjievski, B., Kia, G., Limani, X., Pinheiro, J. F. N., et al. (2024). Next-Gen Connectivity: Dynamic QoS Optimization for 5G Standalone and Vertical Integration. In *IEEE INFOCOM Workshops*.

Fernandes, J., Rosa, L., Gameiro, J., Valente, P., Ramos, E., Oliveira, J., Raposo, D., Ferrer, A. T., Portela, J. M., Rito, P., et al. (2024). Pushing the Boundaries of Scalable 5G Core Networks: Cloud-Native NEF and CAPIF Interplay. In *Proc. 3rd International Conference on 6G Networking (6GNet)*, pages 201–205.

Fragkos, D., Makropoulos, G., Sarantos, P., Koumaras, H., Charismiadis, A.-S., and Tsolkas, D. (2021). 5G Vertical Application Enablers Implementation Challenges and Perspectives. In *IEEE Int. Mediterranean Conf. Communications and Networking (MeditCom)*, pages 117–122.

GSMA (2023). Camara project: Network apis for developers. https://www.gsma.com/solutions-and-impact/technologies/networks/camara/.

Kim, H., Featherstone, W., Jeong, S., Lee, J., Pattan, B. J., Chitturi, S., Kim, D., and Han, J.-K. (2023). Mobile Edge Computing Enabler Layer: Edge-native APPLICATION Architecture For Mobile Networks. *IEEE Communications Standards Magazine*, 7(4):50–59.

Makropoulos, G., Fragkos, D., Koumaras, H., Alonistioti, N., Kaloxylos, A., Koumaras, V., Dounia, T., Sakkas, C., and Tsolkas, D. (2023). *5G Network Programmability Enabling Industry 4.0 Transformation*. IGI Global.

Ordonez-Lucena, J. and Dsouza, F. (2022). Pathways towards network-as-a-service: The CAMARA project. In *Proc. ACM SIGCOMM Workshop on Network-Application Integration (NAI)*, pages 53–59.

Rahimi, H., Gifre, L., Hajipour, S., Vilalta, R., Muñoz, R., Armingol, P., González, O., Fernández-Palacios, J. P., Yu, H., Wang, Y., et al. (2025). Premium and Best-Effort Connectivity Services for Cloud Immersive Applications with F5G-Advanced Networks. In *25th Int. Conf. Transparent Optical Networks (ICTON)*.

Sabella, D., Thakkar, P., Baltar, L. G., Sharma, J., Harmand, A., Laganà, A., Querio, R., De Paola, M., Castagno, M., Das, S., et al. (2021). Global MEC supporting automotive services: from multi-operator live trials to standardization. In *IEEE Conf. Standards for Communications and Networking (CSCN)*, pages 7–13.

Serracanta, B., Gao, K., Ros-Giralt, J., Rodriguez-Natal, A., Contreras, L., Yang, R., and Cabellos, A. (2024). Toward Deep Application-Network Integration: Architectures, Progress and Opportunities. *IEEE Communications Magazine*, 62(12):128–134.