List of Articles

## **1. 5G System Architecture**

### 1. *A softwarized perspective of the 5G networks*

* **Source:** Cardoso et al., arXiv, June 2020 [arxiv.org](https://arxiv.org/abs/2006.10409?utm_source=chatgpt.com)

#### **Key Points**

* + Overview of *Service-Based Architecture (SBA)* in 3GPP Rel-15 core.
  + Emphasizes cloud-native modularization and microservice-based NF implementation.
  + Highlights control-user plane separation within the core for flexibility.

#### **Key takeaways**

Service-Based Architecture (SBA) in 3GPP Rel‑15:

* 5G core is structured as an SBA, where Network Functions (NFs) expose services via APIs.
* NFs communicate directly using standard, language-agnostic protocols, easing interoperability and scalability.

Cloud-Native & Microservices Deployment:

* Core and RAN functions are fully software-controlled, decoupled from specialized hardware via virtualization, SDN, and SDR.
* This enables modular, composable NF design, facilitating independent development and deployment.

Control-Plane & Data-Plane Separation (CUPS):

* 5G embraces the CUPS model—the separation of control and user plane—both in core (e.g., SMF vs UPF) and in RAN (CU-CP vs CU-UP).
* This allows independent scaling and placement of NFs depending on their functional role.

Pillars of Softwarization: SDN, SDR & Virtualization:

a) SDN

* Decouples control and data planes in the network; centralized controller manages forwarding across infrastructure nodes.

b) SDR

* Radio functionality (modulation, MAC, PHY) becomes software-defined and programmable.

c) Virtualization

* Enables slicing and multi-tenancy through logical abstraction of network services over shared infrastructure.

Resource Abstraction & Cross-Layer Management (CML):

* Supports Converged Multi-Layer (CML) management: transmission, control, and compute resources across access and core.
* Enables dynamic resource optimization per service requirements (e.g. QoS), cross-domain coordination.

Functional Architecture Layers:

* **5G-SDWN-AL** (Application Layer): Defines service intent and slice-level resource requests.
* **5G-SDWN-NCL** (Control Layer): Acts as network controller/orchestrator (like SDN controller).
* **5G-SDWN-PLI** (Physical Layer): Contains RAN (small cells, macro-cells), core network elements.

RAN Disaggregation & Heterogeneous Access:

* RAN is virtualized and disaggregated: supports legacy (eNB, macro), small cells (mmWave), and 3G/4G nodes.
* Architecture supports both coordinated multi-AP (CM-5G-SDWN) and heterogeneous multi-tier (Het-5G-SDWN) deployments.

Network Slicing & Isolation:

* Virtualization extends to the creation of logical slices per tenant, with isolation across flows, protocols, and resources.
* Slices can exist at multiple layers (flow-, protocol-, resource-level) with strict isolation policies.

Centralized Hierarchical Resource Control Model:

Resource controller comprised of:

* **SD-VRM** — Virtual layer translating SLAs per slice
* **SD-CRM** — Coordinates pooled resources
* **SD-LRM** — Manages local device-level resources

Case Studies: Enhanced Coverage & QoS:

Demonstrated improvements in throughput, isolation, coverage via centralized control in both homogeneous and heterogeneous deployments.

#### Summary factoids

**Factoid 1**: The 5G Core Network introduced in 3GPP Release 15 adopts a Service-Based Architecture (SBA), where network functions expose services via standardized APIs (primarily HTTP/2 + JSON).

**Factoid 2**: SBA enables dynamic service discovery and direct NF-to-NF communication without requiring intermediaries or brokers.

**Factoid 3**: The SBA model decouples the service consumer from the service provider through the use of a Network Repository Function (NRF).

**Factoid 4**: SBA supports horizontal scalability and modularity by treating every function (e.g., AMF, SMF, PCF) as an independent service.

**Factoid 5**: 5G core and RAN functions are designed as cloud-native components that follow the principles of containerization, statelessness, and modularity.

**Factoid 6**: Network Functions (NFs) are implemented as microservices, enabling CI/CD deployment, independent updates, and fine-grained scaling.

**Factoid 7**: Each NF runs in isolated execution environments, typically containers orchestrated via Kubernetes or other NFVO frameworks.

**Factoid 8**: The use of microservices and container platforms supports vendor-agnostic and hardware-agnostic deployments in hybrid cloud environments.

**Factoid 9**: These cloud-native features allow for slicing, multi-tenancy, and elasticity in managing network resources.

**Factoid 10**: 5G continues the Control and User Plane Separation (CUPS) principle introduced in 3GPP Rel‑14, allowing independent placement and scaling of control-plane (AMF, SMF) and user-plane (UPF) functions.

**Factoid 11**: CUPS enables deploying UPF close to the network edge for latency-sensitive services, while control-plane functions remain centralized.

**Factoid 12**: Separation of control and user planes enhances fault isolation and resource utilization across the network.

**Factoid 13**: CUPS is a prerequisite for enabling flexible and performant network slicing and multi-access edge computing (MEC).

### 2. *Procedure-Aware Stateless Systems for 5G & Beyond Core Networks*

* **Source:** Goshi et al., IEEE, Jan 2024 [https://www.researchgate.net/publication/377840982\_Procedure-Aware\_Stateless\_Systems\_for\_5G\_Beyond\_Core\_Networks](https://www.researchgate.net/publication/377840982_Procedure-Aware_Stateless_Systems_for_5G_Beyond_Core_Networks?utm_source=chatgpt.com)

#### **Key Points**

* + SBA design introduced in 3GPP Rel‑15; CUPS concept from Rel‑14.
  + Stateless core functions store context externally via Unstructured Data Storage Function (UDSF).
  + Introduces piggyback and proactive-push methods to reduce latency in stateless NF state retrieval.

#### **Key takeaways**

Standards Context & Design Drivers:

5G SBA introduced in 3GPP Rel‑15; CUPS released in Rel‑14, enabling separation of control and user plane.

Introduction of UDSF for Stateless NFs:

* The **Unstructured Data Storage Function (UDSF)** acts as a central external database where stateless NFs (e.g., AMF, SMF, PCF, AUSF, UDM) store/retrieve their UE context
* Stateless functions are defined as those decoupled from their UE-state, enabling container-based scalability and dynamic orchestration

State Management Paradigms:

* Traditional stateful NFs bind local state and fail on instance loss; stateless architectures offload it externally
* Stateless strategies include:

a. **Transactional stateless**: state fetched before and saved after every request

b. **Procedure-aware stateless**: optimized for control procedures using request awareness.

Procedure-Aware Patterns Introduced:

**Piggyback-based**: single retrieval of global\_ue\_ctx at procedure start and piggyback in subsequent NF-to-NF HTTP calls. Reduces redundant fetches.

* Empirical results show ~44% & ~70% decrease in completion time for synchronous procedures, e.g. Registration/Deregistration.

**Proactive-push**: AMF instructs UDSF to proactively push relevant UE context to other NFs at procedure start (NFs must support new HTTP POST/GET endpoints).

* Provides ~13–22% improvements in asynchronous procedures (e.g. PDU session establishment/release)

Performance & Resource Impact:

* Both approaches reduce Procedure Completion Time (PCT) compared to the baseline (transactional stateless) with no significant CPU or bandwidth penalty.
* Piggyback approach yielded best results for synchronous procedures; proactive-push outperforms in asynchronous cases.
* A hybrid strategy combining both approaches yields optimal overall performance.

Control Procedures Covered:

* Evaluated on Registration, PDU Session Establishment, Release, and Deregistration

Deployment Context:

* Prototypes deployed in Kubernetes private cloud (containers, K8s orchestration) using Stateless Free5GC implementation with modified SBI endpoints.

#### Summary factoids

**Factoid 1**: In 5G SBA, core functions like AMF, SMF, AUSF, and UDM can operate statelessly by offloading UE context to an external Unstructured Data Storage Function (UDSF).

**Factoid 2**: Stateless NFs gain elasticity and failure resilience, enabling container-based deployments and fast restarts without local state loss.

**Factoid 3**: UDSF acts as a centralized, non-relational key-value store, enabling fast read/write of UE-related state.

**Factoid 4**: Stateful NFs are tightly coupled with internal UE context, causing failure risks and scaling limitations.

**Factoid 5**: The *piggyback-based approach* retrieves UE context once per procedure and embeds it in all NF-to-NF HTTP calls, reducing repeated fetches.

**Factoid 6**: Using piggybacking reduces procedure latency by ~44% for registration and up to ~70% for deregistration procedures.

**Factoid 7**: The *proactive-push approach* lets the AMF preemptively instruct UDSF to push context to downstream NFs via new SBI POST endpoints.

**Factoid 8**: Proactive-push improves asynchronous procedure completion times (like PDU session setup) by 13–22%.

**Factoid 9**: Piggybacking is optimal for synchronous flows; proactive-push works best for asynchronous flows.

**Factoid 10**: A hybrid of piggyback and proactive-push yields best end-to-end control procedure performance.

**Factoid 11**: These approaches require no major changes to core 5G architecture but depend on UDSF and SBI endpoint extension.

**Factoid 12**: Prototype deployments on Kubernetes with Free5GC confirmed stateless NFs with minimal CPU/memory overhead.

**Factoid 13**: Procedure-aware optimizations are backward compatible and non-intrusive to existing SBI-based service flows.

### 3. *5G networks: A review from the perspectives of architecture…*

* **Source:** Aranda et al., Novasinergia (open-access), June 2021<https://www.researchgate.net/publication/377840982_Procedure-Aware_Stateless_Systems_for_5G_Beyond_Core_Networks>

#### **Key Points**

* + Comprehensive survey on SBA: NFs, service discovery, SBA vs legacy EPC.
  + Discusses cybersecurity implications and standardized interface definitions.
  + Includes standalone vs. non-standalone architecture overview (SA vs NSA)

#### **Key takeaways**

Service-Based Architecture (SBA) vs EPC:

* The article confirms the shift from EPC’s reference-point model (3G/4G) to SBA in 5G, emphasizing NF-to-NF communication using HTTP/2 + REST
* SBA enhances modularity, interoperability, and poly-vendor deployments via standardized APIs and dynamic registries (NRF).

Service Discovery with NRF:

* The hierarchy of 5G SBA relies on dynamic registration and discovery through NRF, enabling NF instances to find each other at runtime

Standalone (SA) vs Non-Standalone (NSA) 5G Deployment:

* NSA: 5G NR radio connected to 4G EPC, using UPF, but control plane retained in LTE
* SA: Full deployment of 5G core (5GC), enabling URLLC, mMTC, slicing

Cybersecurity Emphasis:

* Highlights risk increases due to software-defined deployment; identifies attack surfaces across layers and functions.
* Encourages layered security: virtualization/hypervisor, NF APIs, RAN, slice penetration, testbed validation.

Standardized Interface Definitions:

* The article provides an overview of standardized interfaces in SBA, mapping NFs (e.g., AMF, SMF, UPF, UDM, PCF) to role and API specifications
* Also discusses roaming/security elements like SEPP, N3IWF, W-AGF.

5G Core Network Functions (NFs):

* Access and Mobility Management Function (AMF)
  + **Role**: Manages UE registration, mobility, and NAS signaling.
  + **Interfaces**:
    - N1 (UE-AMF): NAS over IP
    - N2 (AMF-gNB): NGAP over SCTP
    - N11 (AMF-SMF): SBI (HTTP/2+JSON)
* **Data**: UE context, mobility state, security context
* **Dependencies**: UDM (for subscription), AUSF (for auth), SMF (for session setup)
* **Protocols**: HTTP/2 (SBI), SCTP, NAS, NGAP

**Factoids**:

1. AMF authenticates UE using NAS and passes session control to SMF.
2. AMF is control-plane only; it does not handle user traffic.
3. AMF can redirect UEs to appropriate SMF/UPF combinations based on policies.

* Session Management Function (SMF)
  + **Role**: Handles PDU session lifecycle: setup, modification, release.
  + **Interfaces**:
* N11 (SMF-AMF)
* N4 (SMF-UPF): PFCP over UDP
* N10 (SMF-UDM)
* **Data**: Session context, QoS rules, IP address allocation
* **Dependencies**: AMF, UPF, PCF
* **Protocols**: HTTP/2, PFCP

**Factoids**:

4. SMF allocates IP addresses and installs traffic steering rules in UPFs.

5. SMF enforces QoS via PCF policies.

6. SMF connects to UPFs via PFCP, a protocol designed for fast, stateless user-plane control.

* User Plane Function (UPF)
  + **Role**: Handles user traffic forwarding and QoS enforcement.
  + **Interfaces**:
    - N3 (UPF-gNB): GTP-U
    - N4 (UPF-SMF): PFCP
    - N6 (UPF-DN): IP forwarding
* **Data**: PDU sessions, QoS enforcement rules
* **Dependencies**: SMF
* **Protocols**: PFCP, GTP-U

**Factoids**:

7. UPF forwards traffic between RAN and external data networks.

8. Supports data buffering, DL packet marking, traffic shaping.

9. Multiple UPFs can be deployed regionally for MEC and slicing.

* Authentication Server Function (AUSF)
  + **Role**: Handles UE authentication using 5G AKA or EAP-AKA′
  + **Interfaces**:
* N12 (AMF-AUSF)
* N13 (AUSF-UDM)
* **Data**: Authentication vectors
* **Dependencies**: UDM (for subscription credentials)

**Factoids**:

10. AUSF performs challenge–response procedures with UEs during registration.

11. Stateless by design, using UDM to obtain auth data (e.g., K, OPc, SQN).

* Unified Data Management (UDM)
* **Role**: Central database for subscriber profiles and policies.
* **Interfaces**:
* N8 (UDM-AMF), N10 (UDM-SMF), N13 (UDM-AUSF)
* **Data**: Subscription data, access profiles, authentication keys
* **Dependencies**: May link with HSS (in interop scenarios)

**Factoids**:

12. UDM provides authentication data to AUSF and access policies to AMF.

13. Also stores SUPI, subscription profiles, and AM Policy Association.

* Policy Control Function (PCF)
  + **Role**: Provides policy decisions (QoS, charging) to other NFs.
  + **Interfaces**:
    - N7 (PCF-SMF), N15 (PCF-AMF)
  + **Data**: Policy rules, slicing info, QoS profiles
  + **Dependencies**: May link to UDR for policy storage

**Factoids**:

14. PCF enforces subscriber-specific policies via SMF and AMF.

15. Works with AF (Application Function) for application-level QoS.

* Network Repository Function (NRF)
* **Role**: Service discovery and registration for SBA.
* **Interfaces**:
* Nnrf (NF-NRF): HTTP/2
* **Data**: NF profiles, service status, availability
* **Dependencies**: All SBA-based NFs

**Factoids**:

16. NRF enables dynamic discovery of NF services using API lookup.

17. Ensures service-level routing and load-aware NF selection.

* Unified Data Repository (UDR)
* **Role**: Centralized database backend for UDM and PCF.
* **Interfaces**:
* N5 (UDM/PCF-UDR)
* **Data**: Subscriber data, policy data, configuration values

**Factoids**:

18. UDR decouples logic (UDM/PCF) from persistent storage.

19. It improves fault tolerance and scaling of data-centric functions

SEPP, N3IWF, W-AGF (Edge/Roaming Gateways)

* **SEPP**: Encrypts and proxies inter-PLMN control signaling across roaming borders.
* **N3IWF**: Gateway for non-3GPP access (e.g., Wi-Fi, IKEv2 tunnels).
* **W-AGF**: Wireline Access Gateway Function (for fixed/mobile convergence)

**Factoids**:

20. SEPP ensures end-to-end control-plane security via TLS/IPsec tunnels.

21. N3IWF bridges Wi-Fi clients to AMF securely without native RAN.

22. W-AGF enables FWA and fixed-line services to access 5GC.

#### **Summary factoids**

**Factoid 1**: “5G core employs SBA where NFs like AMF, SMF, UPF register/discover services via NRF using HTTP/2 + REST.”

**Factoid 2**: “SBA provides modularity, enabling dynamic NF registration and vendor interoperability.”

**Factoid 3**: “NSA combines 5G NR radio with 4G EPC core, while true SA uses 5GC supporting slicing, URLLC, mMTC.”

**Factoid 4**: “Cybersecurity in 5G requires layered defenses across virtualization, SBA APIs, RAN, and network slices.”

**Factoid 5**: “Standard SBA NFs include AMF, SMF, UPF, UDM, PCF; also roaming entities SEPP, N3IWF, W-AGF.”

### 4. *5G NR system design: a concise survey…*

* **Source:** Springer, Apr 2021<https://dl.acm.org/doi/10.1145/3708468.3711877?utm_source=chatgpt.com>

<https://link.springer.com/article/10.1007/s11276-021-02811-y?utm_source=chatgpt.com>

#### **Key Points**

* + Details overall 5GC (5G Core) and NG-RAN split (gNB‑CU vs gNB‑DU).
  + Highlights CUPS-rich architecture in gNB (CU‑CP vs CU‑UP).
  + Aligns architectural rationale between RAN and core separation principles.

**Key takeaways**

NG‑RAN Architecture & gNB Functional Split:

The NG‑RAN (Next Generation RAN) is disaggregated into:

* **gNB‑DU** (Distributed Unit): Handles real-time functions near the antenna (e.g., PHY, MAC).
* **gNB‑CU** (Central Unit): Manages higher-layer processing and interfaces with the core network

Within gNB‑CU, there’s further subdivision:

* **CU‑CP** (Control Plane): Performs connection setup, RRC, control signaling.
* **CU‑UP** (User Plane): Handles user data forwarding, buffering, QoS enforcement

Core‑RAN Architecture Alignment: CUPS:

* The architecture mirrors the core's **Control-User Plane Separation (CUPS)**: both include separate control and user plane entities (SMF–UPF vs CU‑CP–CU‑UP)
* This separation across both core and RAN supports **independent scaling**, **edge deployment**, and **fault isolation**.

Protocol Stack & Interface Mapping:

* Core and RAN layers align through well-defined interface protocols:
  + NG‑C (CU‑CP ↔ AMF): uses NG‑AP over SCTP
  + NG‑U (CU‑UP ↔ UPF): uses GTP‑U for data forwarding
* The **E1 interface** connects CU‑CP and CU‑UP using E1AP over SCTP.
* **F1‑C/F1‑U** interfaces allow CU↔DU split again via SCTP for signaling/data separation

Architectural Rationale: Flexibility & Performance:

* Built on **5G NR’s flexible architecture**, which includes:
* Scalable numerology (supporting eMBB, mMTC, URLLC)
* Ultra-lean, beam-centric PHY design
* Low latency and forward-compatibility goals
* Functional splits enable:
  + Distributed, cloud-based management
  + Multi-vendor deployment
  + Tailoring of deployment at edge vs centralized core, optimizing latency/performance

**Summary factoids**  
**Factoid 0**: NG‑RAN is split into gNB‑DU for real-time processing and gNB‑CU for centralized control and core interfacing.

**Factoid 1**: gNB‑CU is functionally divided into CU‑CP (control plane) and CU‑UP (user plane), mirroring CUPS design.

**Factoid 2**: CU‑CP connects to AMF over NG‑AP/SCTP (NG‑C link); CU‑UP transfers data to UPF via NG‑U using GTP‑U.

**Factoid 3**: E1 interface supports CU‑CP → CU‑UP communication via E1AP over SCTP for bearer/context management.

**Factoid 4**: F1‑C and F1‑U split handles control and user-plane data between CU and DU using SCTP.

**Factoid 5:** CUPS architecture enables horizontal scaling and placement flexibility across both RAN and core.

**Factoid 6**: 5G NR uses scalable OFDM numerology, ultra-lean design, and beamforming for diverse service support (eMBB, mMTC, URLLC).

**Factoid 7**: Separating control and data functions allows independent edge deployment—CU‑UP at edge, CU‑CP centralized—for low latency.

**Factoid 8**: Functional splits enhance vendor diversity, network slicing, and multi-tenant orchestration in cloud-native environments.

### 5. *5G core network control plane: Network security challenges…*

* **Source:** ScienceDirect, Jan 2025<https://arxiv.org/abs/2006.10409?utm_source=chatgpt.com>

#### **Key Points**

* + Focuses on SBA-inspired CP and its vulnerabilities.
  + Emphasizes NRF's role in service discovery via RESTful APIs.
  + Discusses potential mitigations for CP security issues inherent in SBA.

#### **Key takeaways**

SBA-Inspired Control Plane Vulnerabilities:

* The shift to Service-Based Architecture (SBA) increases attack surfaces across Control Plane (CP) due to its RESTful interface design and distributed nature.
* Susceptible to threats like HTTP/2-specific attacks such as header-bike or stream hijacking, as well as misconfiguration of API gateways.
* Traditional perimeter defenses are insufficient; new layered controls are necessary for each NF and control interface.

NRF’s Central Role & Risks:

* NRF (Network Repository Function) handles centralized registration, discovery, and load-balancing of network function.
* If compromised, NRF can be used for service poisoning, redirection of control flows, or man-in-the-middle (MITM) attacks.

* Requires strong mutual authentication mechanisms (e.g., mTLS) and integrity protections for registry data.

SBA-Specific Mitigation Techniques

* Proposed mitigations include:
  + **mTLS** for all NF-to-NF communication to ensure confidentiality and integrity.
  + Use of **API gateways** or **Service Communication Proxies** to detect anomalous requests, rate-limit flows, and prevent abuse
  + **Certificate-based PKI** with PLMN-scoped CAs to validate NF identities
  + RBAC enforcement for APIs along with runtime audit and anomaly detection systems

Expanded Threat Surface Due to Disaggregation:

* SBA’s microservices-based architecture and fragmentation across physical or container hosts increases lateral movement risks
* Necessitates micro‑segmentation and zero‑trust policies at NF and slice boundaries

**Summary factoids**

**Factoid 1**: “SBA control plane is more vulnerable due to REST over HTTP/2 interfaces between NFs, which enable new attack vectors like header manipulation and horizontal DDoS.”

**Factoid 2**: “NRF centralizes service registry and discovery; if compromised, it can route traffic to malicious or rogue NFs.”

**Factoid 3**: “End-to-end mutual TLS (mTLS) is required to secure NF-to-NF communication in SBA control plane.”

**Factoid 4**: “API gateways or service proxies can detect anomalous patterns, rate-limit flows, and enforce role-based access to NF services.”

**Factoid 5**: “Deploying PKI with PLMN-bound CAs allows strong NF identity verification and integrity within multitenant operator domains.”

**Factoid 6**: “NF deployments must include micro-segmentation and zero-trust policies to mitigate lateral threats in disaggregated environments.”

**Factoid 7**: “Control plane security must encompass confidentiality, integrity, authentication, segmentation, and runtime monitoring across all NF interfaces.”

## **2. Network Functions (NFs)**

### 1. *A softwarized perspective…* (Cardoso et al.)

#### **Key Points**

* + Introduces roles of core NFs: AMF, SMF, UPF, NRF, PCF, etc.
  + Outlines function responsibility boundaries and protocol stacks (HTTP/2, REST).
  + Discusses stateless vs. stateful NF designs as microservices.

#### **Key takeaways**

Core Network Functions (NFs) & Roles:

* **AMF**: Handles registration, connection, and mobility for UEs.
* **SMF**: Manages session lifecycle, IP allocation, and UPF programming.
* **UPF**: Carries user traffic, QoS enforcement, buffering, and routing.
* **NRF**: Manages service registration and discovery across NFs.
* **PCF**: Publishes policy decisions, especially for QoS and charging.
* Other contextual functions: **UDM**, **AUSF**, **NSSF**, etc.

Access and Mobility Management Function (AMF)

* **Role**: UE access control, registration, connection management, mobility anchoring.
* **Interfaces**:
  + N1: AMF ⇌ UE (NAS signaling)
  + N2: AMF ⇌ gNB (NGAP over SCTP)
  + N11: AMF ⇌ SMF (SBI, HTTP/2)
  + N12: AMF ⇌ AUSF
  + N15: AMF ⇌ PCF
* **Key Data**: UE context, registration state, mobility info, NAS security keys
* **Protocols**: NAS, NGAP, SCTP, HTTP/2 (SBA)
* **Stateful/Stateless**: Can be stateless with external context (e.g., UDSF)

Session Management Function (SMF)

* **Role**: PDU session lifecycle, IP allocation, QoS rules, packet routing control
* **Interfaces**:
  + N11: SMF ⇌ AMF
  + N4: SMF ⇌ UPF (PFCP)
  + N7: SMF ⇌ PCF (policy decisions)
  + N10: SMF ⇌ UDM (subscription & session policies)
* **Key Data**: IP address pools, traffic steering rules, UE session map
* **Protocols**: HTTP/2 (control), PFCP (user plane setup)
* **Stateful/Stateless**: Typically stateless (external context handling)

User Plane Function (UPF)

* **Role**: User data forwarding, packet routing, QoS enforcement, buffering
* **Interfaces**:
  + N3: UPF ⇌ gNB (GTP-U)
  + N4: UPF ⇌ SMF (PFCP)
  + N6: UPF ⇌ DN (data network)
* **Key Data**: Packet filters, flow descriptors, QoS enforcement policies
* **Protocols**: GTP-U, PFCP, IP
* **Stateful/Stateless**: Stateful (maintains user session forwarding state)

Network Repository Function (NRF)

* **Role**: NF instance registration, discovery, and load-balancing
* **Interfaces**:
  + Nnrf: NF ⇌ NRF (HTTP/2 REST API)
* **Key Data**: NF profiles, service availability, endpoint addresses
* **Protocols**: HTTP/2, JSON
* **Stateful/Stateless**: Stateful (NF registration database)

Authentication Server Function (AUSF)

* **Role**: Performs 5G AKA and EAP-AKA' authentication
* **Interfaces**:
  + N12: AUSF ⇌ AMF
  + N13: AUSF ⇌ UDM
* **Key Data**: Authentication vectors, session keys
* **Protocols**: HTTP/2 (SBA)
* **Stateful/Stateless**: Stateless (delegates to UDM)

Unified Data Management (UDM)

* **Role**: Stores subscriber data, policies, access info
* **Interfaces**:
  + N8: UDM ⇌ AMF
  + N10: UDM ⇌ SMF  
    N13: UDM ⇌ AUSF
* **Key Data**: SUPI, session data, subscription profiles
* **Protocols**: HTTP/2
* **Stateful/Stateless**: Stateful (acts as DB frontend)

Unified Data Repository (UDR)

* **Role**: Storage backend for UDM, PCF, etc.
* **Interfaces**:
  + N5: UDR ⇌ UDM/PCF
* **Key Data**: Subscriber and policy information
* **Protocols**: HTTP/2 (REST)
* **Stateful/Stateless**: Persistent state store

Policy Control Function (PCF)

* **Role**: Centralized policy decisions (QoS, access, charging)
* **Interfaces**:
  + N7: PCF ⇌ SMF
  + N15: PCF ⇌ AMF
  + N5: PCF ⇌ UDR
* **Key Data**: QoS profiles, policy rules
* **Protocols**: HTTP/2
* **Stateful/Stateless**: Stateless (relies on UDR)

Network Slice Selection Function (NSSF)

* **Role**: Determines slice instance for incoming UE requests
* **Interfaces**:
  + N22: NSSF ⇌ AMF
* **Key Data**: Slice IDs, selection policies
* **Protocols**: HTTP/2
* **Stateful/Stateless**: Stateless

Security Edge Protection Proxy (SEPP)

* **Role**: Secures inter-PLMN SBA communication (roaming)
* **Interfaces**:
  + N32: SEPP ⇌ external SEPPs
* **Key Data**: Secure proxying rules
* **Protocols**: TLS/IPsec
* **Stateful/Stateless**: Stateful

Non-3GPP Interworking Function (N3IWF)

* **Role**: Gateway for non-3GPP UE (e.g., WiFi)
* **Interfaces**:
  + N2, N3, N1
* **Protocols**: IPsec, IKEv2, GTP-U
* **Stateful/Stateless**: Stateful (session tracking)

Protocol Stacks & NF Boundaries:

* NFs communicate via **HTTP/2 + RESTful APIs** across the SBA.
* Transport protocols include **SCTP**, **GTP-U**, and **PFCP**, depending on interface type.

Stateless vs. Stateful NF Design:

* NFs may be **stateful** (maintains UE context in-memory) or **stateless** (externalizes state to databases or UDSF).
* Stateless NFs enhance scalability and enable cloud-native microservices but require external state handling.
* Statelessness paves way for container-based deployment and resilience.

**Summary factoids**

**Factoid 1**: “AMF manages UE registration, mobility, and NAS signaling in 5G SBA deployments.”

**Factoid 2**: “SMF is responsible for PDU session management, IP assignment, and setting QoS enforcement in UPF.”

**Factoid 3**: “UPF processes and forwards user-plane traffic, applying buffering and QoS rules.”

**Factoid 4**: “NRF operates as the central service registry enabling dynamic NF discovery.”

**Factoid 5**: “PCF formulates and provides QoS and charging policies to network functions.”

**Factoid 6**: “SBA NFs expose HTTP/2 RESTful APIs and communicate over SCTP, GTP-U, and PFCP interfaces.”

**Factoid 7**: “Control and user plane separation is upheld not only in the core but also within RAN split architecture.”

**Factoid 8**: “Stateful NF designs maintain in-memory UE context, while stateless designs externalize this context.”

**Factoid 9**: “Stateless NFs are suitable for cloud-native microservice deployment, enabling resiliency and elasticity.”

**Factoid 10**: “Stateless NF implementations require additional state-management components like UDSF or central databases.”

### 2. *Procedure‑Aware Stateless Systems…* (Goshi et al.)

#### **Key Points**

* + Differentiates stateless (e.g. NRF, NSSF) and stateful NFs (e.g. AMF, SMF).
  + Explores how state is stored externally (UDSF) and caching strategies.
  + Highlights scaling benefits and trade-offs for stateless deployments.

#### **Key takeaways**

NF Statefulness Classification:

* Differentiates **stateless NFs** (e.g., NRF, NSSF) from **stateful NFs** (e.g., AMF, SMF, PCF).
* Stateless NFs do not hold UE-specific state; stateful NFs do.

External State Storage (UDSF):

* Stateful NFs offload context and metadata (e.g., UE-SUPI, security keys) to the **Unstructured Data Storage Function (UDSF)** .
* UDSF acts as a shared data service akin to a key-value store.

Caching Strategies & Access Patterns:

* System explores state retrieval styles:
  + **Piggybacked**: single retrieval per procedure, embedded in NF‑to‑NF calls.
  + **Proactive-push**: AMF pushes state to future NF participants before the procedure starts.

Performance Benefits vs Trade-offs:

* **Piggybacking** lowers latency by ~44–70% in synchronous control procedures (e.g., attach, detach)
* **Proactive-push** cuts latency by ~13–22% for asynchronous procedures (e.g., PDU session setup).
* The **hybrid** method yields optimal latency across all procedures without significantly increasing CPU or bandwidth overhead

Scalability & Elasticity Through Statelessness:

* Stateless design allows **horizontal scaling** of NF containers (e.g., AMF pods) without reshuffling state
* Enables **fast recovery and resilience**: worker instances can be added or removed dynamically in a Kubernetes environment.

Minimal Protocol Changes & Compatibility:

* Approach requires **no protocol changes** in core network specifications.
* Utilizes existing HTTP/2-based SBI; only modifications needed are:
  + UDSF interface for state storage
  + Optional state-push endpoints in select NFs

#### **Summary factoids**

**Factoid 1**: “NRF and NSSF are stateless; AMF and SMF are stateful and store UE context in UDSF.”

**Factoid 2**: “UDSF functions as a shared key-value store holding UE session context and security credentials.”

**Factoid 3**: “Piggyback-based retrieval retrieves all required UE context at procedure start, eliminating redundant external requests.”

**Factoid 4**: “Piggyback reduces synchronous procedure latency by 44–70%.”

**Factoid 5**: “Proactive-push preloads downstream NF state at procedure initiation, cutting asynchronous latency by 13–22%.”

**Factoid 6**: “The hybrid pattern (piggyback + proactive-push) optimizes latency across synchronous and asynchronous procedures without extra overhead.”

**Factoid 7**: “Container-based stateless deployment enables robust horizontal scaling and failure resilience.”

**Factoid 8**: “Stateless architecture maintains 3GPP compliance by preserving standard SBI interfaces and requiring only new UDSF endpoints.”

### 3. *The Cost of Stateless Network Functions in 5G*

* **Source:** ACM, Mar 2021<https://arxiv.org/abs/2309.14659?utm_source=chatgpt.com>

#### **Key Points:**

* + Empirically measures latency and resource overhead from NF statelessness.
  + Analysis of operational cost in cloud-native environment.
  + Guides capacity planning for scaling stateful vs. stateless NFs.

#### **Key takeaways**

Relationship Between Statefulness and CPU Usage:

* Surprisingly, **stateless NFs** may show **lower average CPU usage** than stateful ones due to increased **waiting periods while fetching state externally**: they spend more time idle than processing [researchgate.net+2cs.purdue.edu+2cs.purdue.edu+2](https://www.cs.purdue.edu/homes/fahmy/papers/21ancs-stateless.pdf?utm_source=chatgpt.com).
* A **queue buildup** occurs within stateless NF instances followed by CPU spikes downstream when state responses come back — this pattern repeats across AMF → SMF → UPF

Transactional vs Non-blocking Stateless Strategies:

* **Transactional stateless** NFs fetch state at each transaction (per request/response), triggering frequent serialization/deserialization.
* **Non-blocking stateless** methods do not wait for each DB response sequentially, improving throughput and reducing latency

Impact on Latency and Cos:

* Stateless designs increase **latency** due to state fetch/update operations.
* **Serialization overhead** of JSON-based key-value storage adds to response times.
* In a **cloud billing context**, increased latency can translate into **higher compute costs** due to longer instance runtime .

Capacity Planning Insights:

* Stateless NFs, while flexible, require **fewer CPU resources per instance** but more **instances** to maintain throughput.
* **Non-blocking stateless** models can **reduce overall CPU usage** by avoiding strict serialization/deserialization and enabling pipeline processing.

Optimizations via Shared Caching:

* Sharing “global\_ue\_ctx” between NFs (AMF & SMF) reduced database queries and improved performance by approximately **33%**.
* Embedding state in service chain messages can reduce 4×n DB read/write operations down to 2, saving ~**22%** overhead

#### **Summary factoids**

**Factoid 1**: “Stateless NF instances may exhibit lower average CPU usage than stateful ones, due to wait periods during external state retrieval.”

**Factoid 2**: “Queue buildup in AMF propagates to SMF and UPF as CPU spikes once state responses are processed.”

**Factoid 3**: “Transactional stateless NFs increase latency and cost by performing synchronous state fetches and JSON deserialization.”

**Factoid 4**: “Non-blocking stateless strategies improve latency and throughput by asynchronous state access.”

**Factoid 5**: “Working in cloud environments, stateless NF designs increase billing costs because longer request times increase runtime charges.”

**Factoid 6**: “Sharing global UE context between NFs reduces unnecessary database operations, improving performance by ~33%.”

**Factoid 7**: “Embedding user context into NF-to-NF messages cuts DB reads from 4×n to 2, reducing overhead by ~22%.”

**Factoid 8**: “Stateless NFs allow container-based scaling but require optimized caching strategies to avoid latency and cost penalties.”

### 4. *Stateless Paradigm for Resiliency in Beyond 5G Networks*

* **Source:** River Publishers, 2022 [sciencedirect.com+2dl.acm.org+2jwcn-eurasipjournals.springeropen.com+2](https://dl.acm.org/doi/10.1145/3493425.3502749?utm_source=chatgpt.com)[riverpublishers.com](https://www.riverpublishers.com/downloadchapter.php?file=RP_P9788770040020C5.pdf&utm_source=chatgpt.com)

<https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-021-01983-7?utm_source=chatgpt.com>

#### **Key Points**

* + Introduces quasi-local fetch-and-cache model for resilience.
  + Decouples processing and storage to improve redundancy.
  + Defines QoS and latency bounds relevant for NF distribution.

#### **Key takeaways**

Quasi‑Local Fetch‑and‑Cache Model:

* Proposes a **quasi‑local state model**, where NFs fetch required UE context from a remote store and cache it locally during active sessions to enhance resiliency.
* The NF caches state fetched at key checkpoints—such as after the initial attach—then uses it for subsequent processing until session end

Decoupling Compute from Storage Enhances Redundancy:

* Separates state storage (in UDSF-like stores) from processing functions to support resilience and rapid recovery after failures
* New NF instances can start mid-session by retrieving cached state, enabling quick rejoin and continued processing

Analyzes State Persistence and Session Workflow:

* Defines metrics: state volume, state size, and request frequency across procedures (e.g., registration, attach), aiding in storage performance modeling
* Auto-persist per-procedure state ensures minimized overhead while maintaining session awareness across shifted instances .

Maintains End-to-End Latency Budgets:

* Model ensures that E2E latency metrics relevant to 5G use cases are met—even under stateless operation—through caching and optimized design.

Supports Diverse Use Cases (URLLC, eMBB, IoT):

* Designed to support low-latency/error-sensitive traffic (e.g., V2X, telesurgery) via optimized local state retention and avoid excessive store-fetch cycles.

#### **Summary factoids**

**Factoid 1**: "Quasi-local model uses fetch-and-cache strategy, storing UE state locally after initial procedures."

**Factoid 2**: "Decoupling control compute from storage enables fast NF instance recovery mid-session."

**Factoid 3**: "Per-procedure caching optimizes storage accesses and controls DB IO per NF."

**Factoid 4**: "This approach maintains end-to-end latency budgets even during NF failover."

**Factoid 5**: "Quasi-local cache supports latency-critical use cases like V2X and telesurgery."

**Factoid 6**: "State metrics include volume, size, and frequency of operations; helps dimension datastore loads."

**Factoid 7**: "Quasi-local caching reduces network-wide datastore interactions during user-plane processing."

### 5. *PFCP (Packet Forwarding Control Protocol)*

* **Source:** Wikipedia (PFCP entry), Apr 2025<https://www.riverpublishers.com/pdf/ebook/chapter/RP_P9788770040020C5.pdf>

#### **Key Points**

* + PFCP is the standardized control-user plane interface (N4/Sx).
  + Governs session management and forwarding rule setup by SMF to UPF.
  + Critical for CUPS compliance.

#### **Key takeaways**

PFCP Defines Control–User Plane Interface:

* **PFCP** is the **standardized interface** between the Control Plane (SMF) and User Plane (UPF), conforming to N4 and Sx reference points in 3GPP 5G architecture .
* Uses **UDP transport** and is specified in **3GPP TS 29.244** .

Session Management & Forwarding Rule Setup:

* PFCP enables the SMF to manage PDU session lifecycles by sending **PFD (Packet Detection Rules)** and **FAR (Forwarding Action Rules)** to UPF .
* Supports session operations such as **establishment, modification, and release**.

CUPS Compliance & Required for Control/User Separation:

* PFCP is essential for enforcing the **Control-User Plane Separation (CUPS)** design, enabling the UPF to independently forward traffic under SMF instructions .

Advanced PFCP Features:

* Includes features like **Session and Keepalive** to ensure session continuity and heartbeats between SMF and UPF.
* Supports **BAR (Buffering Action Rule)** and **QER (QoS Enforcement Rule)** messages for traffic shaping and QoS.

#### **Summary factoids**

**Factoid 1**: “PFCP is the standardized protocol for SMF-to-UPF control, operating over UDP at the N4 and Sx reference points.”

**Factoid 2**: “SMF uses PFCP to install PFDs and FARs in UPF to govern user data forwarding rules.”

**Factoid 3**: “PFCP is fundamental for enabling Control-User Plane Separation (CUPS) in 5G core.”

**Factoid 4**: “PFCP includes mechanisms like Keepalive, Session, QER, and BAR to manage forwarding and buffering.”

**Factoid 5**: “PFCP is specified in 3GPP TS 29.244 and relies on UDP transport for control messaging.”

### Synthesis for Ontology & Knowledge Graph

| **Concept** | **Details** |
| --- | --- |
| **Architecture Layers** | SBA layers (NF as microservices), CUPS splits (control vs user), alignment in RAN/core separation. |
| **Key NFs & Roles** | AMF, SMF, UPF, NRF, PCF, UDM, UDSF; illustrate purpose and equivalence to EPC elements. |
| **State Management** | Distinct stateless NFs vs stateful; UDSF used externally; piggyback/proactive-push patterns for efficiency/resilience. |
| **Scaling & Redundancy** | Stateless designs support elasticity; empirical results quantify cost; fetch/cache architecture boosts resiliency and distributed NF. |
| **Protocols & Interfaces** | RESTful HTTP/2 APIs for SBA; PFCP on N4 interface; service discovery via NRF; RAN–core via NGAP, PFCP. |
| **Service Discovery** | NRF-based registry; service consumer–provider model; REST API-based discovery. |
| **Security Aspects** | PKI for inter-NF communication; SBA-specific vulnerabilities and mitigation strategies. |

## **3. Interfaces & Protocols**

### 1. ShareTechnote – *Core – N1 Interface – 5G*

* **Details**: Defines N1 (UE ↔ AMF) signaling path and variants via 3GPP TS 24.501<https://open5gs.org/open5gs/docs/?utm_source=chatgpt.com>

<https://www.sharetechnote.com/html/5G/5G_NetworkArchitecture_N1.html?utm_source=chatgpt.com>

* Crucial for establishing NAS-based relation in KG with directionality.

#### **Key takeaways**

Definition & Purpose of N1:

* **N1** is the reference point for **NAS (Non-Access Stratum) signaling** between the UE and AMF. It spans the entire control path: UE ↔ Access Network ↔ AMF
* NAS messages are **transparent to gNB** devices; they simply forward the messages between UE and AMF without interpreting their contents

Interfaces & Access Modes:

* **3GPP Access Mode**: N1 comprises RRC (Uu) between UE and gNB, followed by NGAP (N2) to the AMF
* **Non-3GPP Access**: The UE uses an **IPsec tunnel (NWu)** over non-3GPP RAT and continues via N2 to AMF

Functional Distinction: N1 Reference vs N1 Mode:

* **N1 Reference Point**: Architectural concept defining logical connectivity, including concatenated layers via underlying networks
* **N1 Mode**: Simplified access mode representing **Standalone (SA)** deployment with direct UE‑gNB‑AMF connectivity over NG interfaces. Opposed to **S1 Mode (NSA)**, which leverages 4G EPC

#### **Summary factoids**

**Factoid 1**: “N1 is the control-plane interface for NAS signaling between UE and AMF, across both gNB and access network.”

**Factoid 2**: “N1 NAS messages are transparently forwarded by gNB without processing.”

**Factoid 3**: “In 3GPP access, N1 uses RRC over Uu and NGAP over N2 between UE and AMF.”

**Factoid 4**: “In non-3GPP access, N1 includes an IPsec tunnel (NWu) to link UE to AMF securely.”

**Factoid 5**: “N1 mode denotes Standalone (SA) deployment with direct 5G Core connectivity to UE, whereas S1 mode uses NSA via 4G.”

### 2. ShareTechnote – *Core Architecture – 5G (N1–N16)*

* **Details**: Exhaustive mapping of reference points N1–N16, their endpoints (UE, AMF, SMF...), roles, plus SBA-related service interfaces (Namf, Nsmf…)<https://www.sharetechnote.com/html/5G/5G_NetworkArchitecture.html?utm_source=chatgpt.com> Perfect for ontology structure of interface → protocol → NF relations.

#### **Key takeaways**

Interface Definitions & Ontology Mappings:

* Full Reference-Point List (N1–N16)

Each interface maps two endpoints, protocols, and functional role:

| **Interface** | **Endpoints** | **Protocol(s)** | **Role** |
| --- | --- | --- | --- |
| N1 | UE ↔ AMF | NAS over RRC (UE–gNB), NGAP (gNB–AMF) / IPsec | UE registration & signaling |
| N2 | gNB ↔ AMF | NGAP over SCTP | RAN-to-core control signaling |
| N3 | gNB ↔ UPF | GTP‑U over UDP | User-plane traffic forwarding |
| N4 | SMF ↔ UPF | PFCP over UDP | UPF session control |
| N5 | PCF ↔ AF | HTTP/2 + REST | Policy indication to AF |
| N6 | UPF ↔ Data Network (DN) | IP / GTP‑U | External network access |
| N7 | SMF ↔ PCF | HTTP/2 + REST | Policy control messaging |
| N8 | UDM ↔ AMF | HTTP/2 + REST | AMF retrieves subscriber data |
| N9 | UPF ↔ UPF | GTP‑U | UL/DL branching and chaining |
| N10 | UDM ↔ SMF | HTTP/2 + REST | SMF queries subscription policies |
| N11 | AMF ↔ SMF | HTTP/2 + REST | Session control handover info |
| N12 | AMF ↔ AUSF | HTTP/2 + REST | Authentication orchestration |
| N13 | UDM ↔ AUSF | HTTP/2 + REST | Auth credential retrieval |
| N14 | AMF ↔ AMF | HTTP/2 + REST | UE context transfer for mobility |
| N15 | PCF ↔ AMF | HTTP/2 + REST | Policy for AMF (non-roaming) |
| N16 | SMF ↔ SMF (roaming) | HTTP/2 + REST | Cross-domain session continuity |

Design Philosophy & System Traits:

* **Modular flexibility**: Interfaces map clearly between NFs and components, aligning with 5G’s microservice architecture
* **Cloud-native & scalable**: Stateless interface use supports containerized scaling and NF orchestration
* **Secure by design**: Transport protocols use TLS/mTLS, IPsec (e.g., NAS vs non-3GPP access), and REST security frameworks

#### **Summary factoids**

**Factoid 1**: “N1 interface carries NAS signaling transparently via gNB, using RRC and NGAP for core-plane UE–AMF communication.”

**Factoid 2**: “Control-plane interfaces (N2, N11, N12, N14, N15) use HTTP/2 REST over SBA, while data-plane interfaces (N3, N6, N9) use UDP-based protocols.”

**Factoid 3**: “N4 interface (SMF–UPF) employs PFCP to manage user-plane sessions and forwarding behavior.”

**Factoid 4**: “Inter-NF service calls (e.g. N7, N8, N10) enable SMB workload routing and allow dynamic policy and data flow management.”

**Factoid 5**: “Roaming-related interfaces such as N16, N27, N32 facilitate control-plane continuity across administrative domains.”

**Factoid 6**: “NSSF selects network slices via N22 after UE–AMF registration for slice-specific NF association.”

**Factoid 7**: “N9 supports cascading of UPFs for scaling and multi-hop data-plane routing.”

**Factoid 8**: “The interface structure supports cloud-native scaling via clear separation and SLA isolation.”

**Factoid 9**: “Security features across interfaces include TLS for HTTP/2 SBA and IPsec tunneling for non-3GPP and RAN access paths.”

**Factoid 10**: “The complete N1–N16 mapping enables clear schema for modeling in RDF as Interface → Protocol → Function → Endpoints for KG use.”

| **N1** | “N1 carries NAS signaling between the UE and AMF and is transparent through the gNB, forming the control link for UE registration and session setup.” |
| --- | --- |

| **N2** | “N2 connects gNB to AMF using NGAP over SCTP and enables the transmission of access signaling and mobility control messages.” |
| --- | --- |

| **N3** | “N3 transports user-plane traffic between gNB and UPF using GTP-U over UDP, supporting high-throughput PDU sessions.” |
| --- | --- |

| **N4** | “N4 allows the SMF to control UPF behavior using PFCP, setting up and modifying forwarding rules and QoS enforcement.” |
| --- | --- |

| **N5** | “N5 enables application functions (AF) to communicate policy triggers to the PCF via HTTP/2, often for content-aware optimization.” |
| --- | --- |

| **N6** | “N6 links the UPF to external data networks (DN) over IP and handles routing, NAT, and service exposure for user-plane packets.” |
| --- | --- |

| **N7** | “N7 interface is used by the SMF to retrieve policy rules and QoS profiles from the PCF using RESTful HTTP APIs.” |
| --- | --- |

| **N8** | “N8 provides the AMF access to subscriber and authentication data by querying the UDM over HTTP/2.” |
| --- | --- |

| **N9** | “N9 enables inter-UPF communication via GTP-U for distributed user-plane routing, load balancing, or service chaining.” |
| --- | --- |

| **N10** | “N10 allows SMF to obtain subscriber policy data from the UDM, including PDU session authorization and QoS configuration.” |
| --- | --- |

| **N11** | “N11 connects the AMF and SMF, enabling control-plane interactions such as PDU session setup and mobility handling.” |
| --- | --- |

| **N12** | “N12 allows the AMF to forward authentication requests to the AUSF during UE registration or security mode command procedures.” |
| --- | --- |

| **N13** | “N13 connects AUSF to UDM for obtaining authentication vectors like AV/EAP for 5G-AKA procedures.” |
| --- | --- |

| **N14** | “N14 supports AMF-to-AMF communication for inter-region handover, UE context transfer, and mobility continuity.” |
| --- | --- |

| **N15** | “N15 is used by the PCF to supply policy rules directly to the AMF, influencing access and mobility behaviors.” |
| --- | --- |

| **N16** | “N16 links two SMFs (typically in roaming scenarios) to coordinate session continuity and policy across PLMN boundaries.” |
| --- | --- |

### 3. LinkedIn post – *5G Interfaces N1 N2 N3 N4*

* **Details**: Summaries of N1–N4, including directionality, key signaling messages, and protocol usage like SCTP, GTP-U, HTTP/2Excellent for protocol/type annotation in KG.
* LinkedIn summary by Abhijeet Kumar (Nov 5, 2024)

#### **Key takeaways**

N1 – UE ↔ AMF (NAS Signaling Path):

* **Purpose**: Primary control interface for UE registration, authentication, session management, and mobility via Non-Access Stratum (NAS).
* **Directionality**: Bidirectional between UE and AMF.
* **Message Types**:
  + Registration Request/Accept
  + Service Request
  + Security Mode Command
* **Protocols**: NAS messages encapsulated in RRC (UE–gNB), forwarded over NGAP (gNB–AMF).
* **Transparency**: NAS is transparent to gNB; no interpretation.

N2 – gNB ↔ AMF (Access to Core Signaling):

* **Purpose**: Transfers signaling messages between RAN and AMF; supports handovers, UE context handling.
* **Directionality**: Bidirectional (initiated by gNB or AMF).
* **Message Types**:
  + Initial UE Message
  + UE Context Setup/Release
  + Handover Request
* **Protocols**: NGAP over SCTP (TS 38.413).

N3 – gNB ↔ UPF (User-Plane Data Forwarding):

* **Purpose**: Carries user traffic from UE through RAN to UPF.
* **Directionality**: Primarily downlink from UPF to gNB and uplink from gNB to UPF.
* **Protocol**: GTP-U over UDP (TS 29.281).
* **Use**: Activated after session establishment; critical for data flow performance.

N4 – SMF ↔ UPF (Session Control Interface):

* **Purpose**: SMF controls UPF behavior: create, update, delete PDU sessions.
* **Directionality**: Bidirectional control commands.
* **Protocol**: PFCP over UDP (TS 29.244).
* **Messages**:
  + Session Establishment Request/Response
  + FARs (Forwarding Action Rules)
  + QERs (QoS Enforcement Rules)
  + BARs (Buffering Action Rules)

#### **Summary factoids**

**Factoid 1**: “N1 is the NAS signaling interface between UE and AMF, used for registration, authentication, and mobility procedures.”

**Factoid 2**: “N1 signaling is bidirectional, encapsulated in RRC at the UE side and transported via NGAP through gNB to AMF.”

**Factoid 3**: “N2 connects the gNB and AMF, using NGAP over SCTP to handle UE context and handover signaling.”

**Factoid 4**: “N2 transmits messages such as Initial UE Message and UE Context Release between RAN and Core.”

**Factoid 5**: “N3 uses GTP-U over UDP to forward user-plane traffic between gNB and UPF after PDU session establishment.”

**Factoid 6**: “N3 is unidirectional for data flow (uplink/downlink) and not involved in control signaling.”

**Factoid 7**: “N4 is the control interface between SMF and UPF, enabling session creation, QoS enforcement, and traffic routing via PFCP.”

**Factoid 8**: “N4 manages forwarding rules through PFCP messages such as FAR, QER, and BAR to control UPF behavior dynamically.”

### 4. Enea – *Wi‑Fi and Cellular Convergence – What’s New in 5G*

* **Details**: Discusses N1/N2/N3 in 3GPP and non‑3GPP (Wi‑Fi) context, use of IPsec/EAP‑5G & SCTP, GTP‑U encapsulation [enea.com](https://www.enea.com/insights/wi-fi-and-cellular-convergence-whats-new-in-5g/?utm_source=chatgpt.com). Adds depth around multi-access and secure transport.

#### **Key takeaways**

Non‑3GPP (Wi‑Fi) Integration in 5G:

* 5G allows simultaneous connection over both cellular and Wi‑Fi using multiple NAS (N1) sessions, foundational for ATSSS
* Same UE‑level authentication methods (EAP-AKA′ / 5G‑AKA) are used across both access types

IPsec & EAP‑5G for Secure NAS Transport:

* EAP‑5G and IKEv2 protocols establish IPsec tunnels (NWu, NWt) between UE and non‑3GPP gateway functions (N3IWF, TNGF) for secure NAS over Wi‑Fi
* For *trusted Wi‑Fi*, IPsec may use NULL encryption to avoid redundant security since link layer is already protected

N1 over Wi‑Fi: NAS Signaling via IPsec:

* N1 over Wi‑Fi uses IPsec tunnels to transmit NAS messages to AMF through N3IWF or TNGF
* TWIF supports devices that don’t implement EAP‑5G by acting as an intermediary for NAS signaling

N2 over Non‑3GPP: NGAP over SCTP:

* Non‑3GPP control-plane (N2) uses the same NGAP over SCTP protocol between gateway (N3IWF/TNGF/TWIF) and AMF

N3 over Non‑3GPP: GTP-U User-Plane via Wi‑Fi:

* Non‑3GPP user-plane traffic uses GTP‑U over UDP between N3IWF/TNGF/TWIF and UPF, as with cellular access

#### **Summary factoids**

**Factoid 1**: "5G supports simultaneous NAS sessions over cellular and Wi‑Fi by establishing multiple N1 control-plane connections."

**Factoid 2**: "UEs authenticate to 5G core over Wi‑Fi using EAP-AKA′ or 5G-AKA prior to NAS signaling."

**Factoid 3**: "IPsec tunnels (NWu/NWt) using IKEv2 and EAP‑5G secure NAS traffic to N3IWF/TNGF gateways."

**Factoid 4**: "N1 over trusted Wi‑Fi uses IPsec with NULL encryption to avoid double encryption while preserving link-level security."

**Factoid 5**: "N2 control-plane messages over non-3GPP access use NGAP/SCTP between gateway functions and AMF."

**Factoid 6**: "TWIF enables NAS signaling for legacy Wi‑Fi devices lacking EAP‑5G support by acting as a NAS proxy."

**Factoid 7**: "N3 user-plane traffic between Wi‑Fi gateways and UPF uses GTP‑U over UDP, mirroring cellular data flow."

### 5. Ericsson blog – *Your Quick Guide to Network Functions in 5G Core*

* **Details**: Mentions that SBA covers only CP-NFs; interfaces N1, N2, N3, N4, N6, N9 are outside SBA. Clarifies protocol boundaries between SBA APIs (HTTP/2 REST) vs CP/UP plane<https://www.ericsson.com/en/blog/2019/2/your-quick-guide-to-network-functions-in-5g-core?utm_source=chatgpt.com> <https://telecompedia.net/5g-core-network-overview/?utm_source=chatgpt.com> <https://amslaurea.unibo.it/id/eprint/26454/1/Tesi%20Asma%20Noor.pdf?utm_source=chatgpt.com> Good source for interface–protocol distinctions.

#### **Key takeaways**

**SBA only applies to Control‑Plane NFs** — Service-Based Architecture (SBA) with HTTP/2 REST APIs is strictly for control-plane NFs (e.g., AMF, SMF, PCF, UDM, AUSF, etc.). All user-plane interfaces (N1, N2, N3, N4, N6, N9) and UE/RAN-facing control-plane interfaces are excluded from SBA.

**Core functional separation** — 5GC distinctly separates:

* **Access & Mobility Management** (AMF)
* **Session Management** (SMF)  
   This enhances specialization and modularity

**CUPS Evolution** — The architecture continues the trend from EPC: separating Control Plane and User Plane (CUPS), splitting gateway functions into logical control and user plane elements for flexish scaling and localization.

**Cloud-native modularity** — 5GC adopts cloud-native practices including containerized microservices, CI/CD automation, and programmable orchestration platforms.

**Interface and protocol delineation**:

* **Control-plane AFI (SBA)**: HTTP/2 + TLS between SBA NFs
* **UE/RAN/UP interfaces** (N1‑N4, N6, N9): Use traditional protocols (NAS, NGAP, GTP‑U, PFCP) with UDP/SCTP

#### **Summary factoids**

1. **Factoid 1**: “Service-Based Architecture (SBA) exists only between 5GC control-plane NFs via HTTP/2 REST APIs.”
2. **Factoid 2**: “Interfaces N1, N2, N3, N4, N6, and N9 are executed outside the SBA domain using traditional control/user-plane protocols.”
3. **Factoid 3**: “AMF and SMF represent separate functional domains in control-plane, enabling independent scaling and specialization.”
4. **Factoid 4**: “CUPS design separates control and user gateways for flexible placement and scalability in the 5G core.”
5. **Factoid 5**: “SBA-capable NFs are cloud-native microservices, orchestrated via container platforms with CI/CD pipelines.”
6. **Factoid 6**: “Control-plane APIs (e.g., Nsmf, Namf) are secured via TLS, whereas user-plane protocols (e.g., GTP‑U, PFCP) use UDP/SCTP.”
7. **Factoid 7**: “SBA’s RESTful interfaces cannot directly carry UE or user-data-bound control messages (e.g., RRC or NAS).”
8. **Factoid 8**: “In CUPS, user-plane functions like UPF scale and deploy separately from their control-plane counterparts (SMF, PCF).”

### **Key Takeaways for Ontology Expansion**

* **Interfaces (N1–N16)**: Define as RDF properties, linking NFs (e.g., amf --[N2 interface]--> gNB).
* **Protocols per Interface**: N1 uses NAS/UE↔AMF via RRC/IPsec; N2 via NGAP over SCTP; N3 uses GTP‑U; N4 uses PFCP; SBA uses HTTP/2+REST.
* **Directionality**: Each interface has defined endpoints—explicit directionality should be captured (e.g., UE→AMF or bidirectional).
* **Transport Layers**: Specify SCTP for NGAP, UDP for GTP‑U/N4, IPsec for non‑3GPP.
* **Security Layers**: IPsec/EAP‑5G for untrusted access and inter‑NF REST channels.

## **4. Open5GS–Specific Architecture**

### 1. Open5GS Quickstart Guide

* **Details**: Highlights configuration via YAML (e.g., upf.yaml, amf.yaml), logging paths (/var/log/open5gs/\*.log), and gNB integration via NGAP SCTP<https://open5gs.org/open5gs/docs/guide/01-quickstart/?utm_source=chatgpt.com>

<https://github.com/s5uishida/open5gs_5gc_ueransim_metrics_sample_config?utm_source=chatgpt.com>

<https://medium.com/networkers-fiit-stu/setting-up-open5gs-a-step-by-step-guide-or-how-we-set-up-our-lab-environment-5da1c8db0439>

<https://open5gs.org/open5gs/docs/guide/02-building-open5gs-from-sources/?utm_source=chatgpt.com>

#### **Key takeaways**

Configuration Files & YAML-based Setup:

* Open5GS defines each core NF via individual YAML files located in /etc/open5gs/ such as:
* amf.yaml, smf.yaml, upf.yaml, etc., each specifying bind addresses (e.g., NGAP, PFCP, GTP-U), PLMN IDs, TACs, and slices.
* Bind IP changes (e.g., from 127.0.0.5 to 192.x.x.x) must be synchronized with RAN/gNB and PLMN settings to ensure NGAP and GTP-U connectivity across hosts

Logging and Monitoring Paths:

* Each NF logs to /var/log/open5gs/\*.log (e.g., amf.log, upf.log). Logs show active protocols — for instance, gtp\_server() in SMF, ngap\_server() in AMF — confirming SCTP/GTP-PFCP/GTP-U operation

gNB Integration via NGAP SCTP:

* Open5GS core expects gNB connections over SCTP using NGAP on ports 38412 (AMF) and 36412 (MME). A successful SCTP handshake (SCTP INIT) must be visible in logs or packet capture

Firewall, TUN, and IP Forwarding Setup:

* Core setup includes firewall configurations (iptables) to:
  + Allow traffic on ogstun interface
* Block unwanted UE-originated traffic from subnet 10.45.0.0/16
* NAT and IP forwarding (sysctl) are enabled to bridge UPF’s subnet (e.g. 10.45.0.0/16 or 2001:db8:cafe::/48) to the external WAN
* A TUN interface (ogstun) is configured with IPv4/IPv6 subnets and brought up before running services

Configuration & Deployment Lifecycle:

* Core services run as systemd services: open5gs-amfd, -smfd, -upfd, etc.; these can be individually stopped or disabled to run selected NFs only
* In **containerized deployments (Docker/Kubernetes)**, YAML manifests must include host-bind addresses and exposed ports for NFs, along with correct network routing and firewall/NAT rules .
* Open5GS uses **MongoDB** as the database backend for NFs like NRF, PCF, and UDR (Mongo DB URI in pcf.yaml)

#### **Summary factoids**

Configuration:

**Factoid**: “Each Open5GS NF is configured via its own YAML file in /etc/open5gs/, specifying protocol ports, PLMN/TAC, and features.”

**Factoid**: “Changing bind addresses in NF YAML requires consistent updates across RAN configurations to establish NGAP/GTP‑U links.”

**Factoid**: “AMF listens on NGAP SCTP port 38412; UPF listens for GTP‑U on PFCP-assigned port, both reflected in NF logs.”

Logs & Protocol Verification:

**Factoid**: “Log entries such as ngap\_server() and gtp\_server() confirm the NF’s protocol stack initialization and port bindings.”

**Factoid**: “SCTP INIT/ABORT messages in logs reveal SCTP handshake status between gNB and AMF.”

Networking and Firewall:

**Factoid**: “Core setup creates ogstun TUN interface with IPv4/IPv6 subnets for UPF operations.”

**Factoid**: “IP forwarding (net.ipv4.ip\_forward) must be enabled to allow UE-originated packets to route to WAN.”

**Factoid**: “Firewall rules enforce subnet isolation (e.g., 10.45.0.0/16), blocking unauthorized access to NF services.”

Deployment Lifecycle:

**Factoid**: “Open5GS NFs run as systemd services and can be stopped or disabled individually (e.g., open5gs-amfd).”

**Factoid**: “Running only subsets of NFs is supported by stopping irrelevant services and editing YAML configs accordingly.”

Container & Cloud Deployments:

**Factoid**: “In Docker/Kubernetes mode, NF bind addresses, Kubernetes service IPs, and port mappings must align with core and RAN network settings.”

**Factoid**: “Open5GS uses Docker manifests that configure upfPublicIP and amfif.ip/port, binding container network to host services.”

Database Integration:

**Factoid**: “Open5GS uses MongoDB (e.g., mongodb://localhost/open5gs) for stateful NF data storage like NRF, PCF, and UDR.”

**Factoid**: “To support service discovery and policy control, Open5GS PCF includes dbi configuration pointing to a MongoDB URI.”

### 2. GitHub – sample config (Open5GS + UERANSIM + Prometheus)

* **Details**: Shows metrics config snippets in YAML for AMF, PCF, SMF, UPF; integration with Prometheus & Grafana[github.com](http://github.com).

#### **Key takeaways**

Metrics Export Configuration:

* Each NF (AMF, PCF, SMF instances, UPF instances) exposes Prometheus metrics via a built-in HTTP server.
* Configured in each NF’s YAML (amf.yaml, pcf.yaml, smf1.yaml, upf.yaml) under a metrics: section with host IP and port (e.g., 192.168.0.111:9090)

Prometheus & Grafana Setup:

* Prometheus scrapes each NF’s /metrics endpoint every 10 seconds as per prometheus.yml (jobs for open5gs-amfd, open5gs-pcfd, etc.)
* Grafana fetches data from Prometheus to build dashboards around these metrics .

NF Instances & Topology:

* Supports multiple SMF and UPF instances (e.g., smf1, smf2, upf1, upf2) with dedicated metric endpoints.
* Metrics setup adapts to multi-instance architecture indicating ability to monitor a geo-distributed or slice-specific deployment.

Metric Types Available:

* Examples include ues\_active, fivegs\_amffunction\_rm\_reginitreq, resource usage (memory, file descriptors, CPU)

Metrics Module (libogsmetrics):

* Open5GS integrates libogsmetrics (based on libprom + libmicrohttpd) at build time to expose Prometheus-formatted metrics.

#### **Summary factoids**

**Factoid**: “Each NF (AMF, PCF, SMF, UPF) in Open5GS can expose Prometheus metrics via an HTTP server configured under metrics: in its YAML file.”

**Factoid**: “Metrics endpoints are individually defined per NF instance (e.g., open5gs-amfd, open5gs-smfd1, open5gs-upfd2) with distinct IP and port.”

**Factoid**: “Prometheus scrapes /metrics endpoints every 10 seconds using job definitions matching NF names in prometheus.yml.”

**Factoid**: “Grafana connects to Prometheus as a data source to visualize NF-specific metrics like ues\_active or amf\_session.”

**Factoid**: “Multiple SMF and UPF instances can be monitored in parallel, supporting slice- or region-specific deployments.”

**Factoid**: “Open5GS’s build process includes libogsmetrics to compile libprom and libmicrohttpd support for metrics output.”

**Factoid**: “Example metrics include counters for NAS registration requests (rm\_reginitreq) and memory/resource usage (process\_resident\_memory\_bytes).”

**Factoid**: “Using distinct job\_name entries in Prometheus enables selective scraping and dashboarding per NF type.”

**Factoid**: “The sample config binds metrics HTTP servers to host IPs, facilitating external observability for containerized NFs.”

**Factoid**: “Metrics integration demonstrates how Open5GS can operate as 'Prometheus-enabled' microservices in Kubernetes or Docker environments.”

### 3. Medium – *Setting Up Open5GS*

* **Details**: Covers gNB search config (gnbSearchList), PLMN/MCC/MNC alignment, and running commands for UERANSIM integration<https://medium.com/networkers-fiit-stu/setting-up-open5gs-a-step-by-step-guide-or-how-we-set-up-our-lab-environment-5da1c8db0439>

#### **Key takeaways**

Prerequisites & Installation:

* Installs Open5GS via PPA on Ubuntu (e.g., sudo apt install open5gs), requiring MongoDB for subscriber storage
* Recommended hardware includes Intel i5 boards due to MongoDB compatibility issues on Celeron systems.

Core Configuration (amf.yaml, mme.yaml, sgwu.yaml):

* Default bind addresses use loopback IPs; to connect UERANSIM/gNB, you must update configs to the host’s LAN IP (e.g., amf.yaml: ngap.addr = 192.168.50.5)
* PLMN configuration uses test operator PLMN 001/01 or private PLMN 999/99. Changes must align in both Open5GS and gNB/UERANSIM configs

Configuring gNB & UE in UERANSIM:

* open5gs-gnb.yaml in UERANSIM requires linkIp, ngapIp, gtpIp, and amfConfigs.address set to networked IPs
* UE uses gnbSearchList pointing to RAN host IP (e.g., 192.168.0.131) to locate the gNB and attempts attach via TUN interface (e.g., uesimtun0).

PLMN & TAC Alignment:

* PLMN ID (mcc/mnc) and TAC values must match between core and RAN configurations to enable UE connectivity.

External Connectivity & Testing:

* After PDU session establishment, UE creates a TUN interface (uesimtun0) with IP (e.g., 10.45.0.3), enabling external Internet access via curl or ping through the core.
* Optional TCP proxy setups illustrate multi-machine network test scenarios over simulated public IPs.

#### **Summary factoids**

**Factoid**: “Open5GS is installed via Ubuntu PPA and requires MongoDB for subscriber context storage.”

**Factoid**: “Hardware with Intel i5 is preferred for MongoDB compatibility over Celeron-based systems.”

**Factoid**: “Default Open5GS configs use loopback IPs; to integrate with external RAN simulators, host LAN IPs must replace loopback addresses in NF YAML.”

**Factoid**: “PLMN ID and TAC must match across core (Open5GS) and RAN (UERANSIM/gNB) configurations for successful connectivity.”

**Factoid**: “UERANSIM open5gs-gnb.yaml must specify linkIp, ngapIp, gtpIp, and core AMF address to enable N2 and N3 connectivity.”

**Factoid**: “UE uses gnbSearchList in UERANSIM config to locate gNB IP for network attachment.”

**Factoid**: “Upon PDU session setup, the UE forms a TUN interface (e.g., uesimtun0) to receive IP routes via the UPF.”

**Factoid**: “UE external Internet access can be validated using tools like curl or ping over the TUN interface.”

**Factoid**: “Test environments may employ TCP proxies on public IPs for multi-machine end-to-end network validation.”

**Factoid**: “Open5GS core services integrate with RAN simulators through manual IP alignment in both core and RAN YAML configurations.”

### 4. Nick vs Networking blog – *My First 5G Core: Open5GS + UERANSIM*

* **Details**: Details configuring amf.yaml for ngap.addr, binding address for external RAN access; describes N2 handling via SCTP<https://nickvsnetworking.com/my-first-5g-core-open5gs-and-ueransim/?utm_source=chatgpt.com>

#### **Key takeaways**

AMF N2 Binding Configuration:

* The AMF (Access and Mobility Function) initially binds to loopback IP for NGAP/SCTP by default.
* To integrate with external RAN (e.g., UERANSIM gNB on another host), the ngap.addr field in /etc/open5gs/amf.yaml must be set to the host's LAN IP (e.g., 10.0.1.207), and the open5gs-amfd service needs restarting.

N2 Interface Usage (NGAP over SCTP):

* NGAP (N2) used by AMF to handle 5G NAS messaging from UE and run RAN procedures like attachment and handovers.
* SCTP is the underlying transport, with proper binding essential for cross-host RAN communication.

Logical Separation of Core and RAN Hosts:

* Running UERANSIM (UE + gNB simulator) on a separate server than Open5GS core requires explicit IP binding in configs to allow SCTP communication over N2.

Service Restart Required for Config Reload:

* After updating ngap.addr, restarting the open5gs-amfd service is necessary to apply the new binding and allow external RAN access.

Simulator Setup Details:

* The article walks through the installation of UERANSIM with prerequisites (libsctp-dev, cmake, snap) and build via git clone and make, preparing the gNB/UE simulator.

#### **Summary factoids**

**Factoid**: “Open5GS AMF by default binds NGAP/SCTP to loopback; must set ngap.addr in amf.yaml to LAN IP to enable RAN connectivity.”

**Factoid**: “N2 interface uses NGAP over SCTP to deliver UE NAS signaling and handover events from gNB to AMF.”

**Factoid**: “For multi-host setups (Core and RAN on separate servers), explicit IP binding is required for N2 to function across hosts.”

**Factoid**: “Running sudo systemctl restart open5gs-amfd applies the ngap.addr binding change to the AMF service.”

**Factoid**: “UE RAN simulation with UERANSIM requires SCTP libraries (libsctp-dev) and CMake for building the gNB module.”

**Factoid**: “Proper AMF binding allows remote UERANSIM-created gNB to successfully attach and exchange NGAP over SCTP.”

**Factoid**: “N2 SCTP handshake logs confirm successful gNB-AMF connectivity, validating multi-host network setup.”

### 5. GitHub Issue – *Metrics Monitoring System #1559*

**Details**: Describes Open5GS integration with Prometheus: libprom, libpromhttp, HTTP server per NF for exporting counters<https://github.com/open5gs/open5gs/issues/1559?utm_source=chatgpt.com>

#### **Key takeaways**

Requirement for Metrics Export:

The Open5GS project requested a built-in system to expose runtime metrics (counters, gauges) to monitoring tools such as Prometheus.

Use of Prometheus Client Libraries:

Integration relies on the C Prometheus client libraries: **libprom** for formatting metrics and **libpromhttp** for providing an HTTP endpoint via libmicrohttpd.

Generic Metrics API Architecture Design:

* A core **generic metrics API** in Open5GS under lib/metrics/ to define and update metrics
* **Void (“no-op”) implementation** for when monitoring is disabled
* **Prometheus implementation** built conditionally against libprom and libmicrohttpd

HTTP Metrics Endpoint per NF Process:

* A core **generic metrics API** in Open5GS under lib/metrics/ to define and update metrics
* **Void (“no-op”) implementation** for when monitoring is disabled
* **Prometheus implementation** built conditionally against libprom and libmicrohttpd

HTTP Metrics Endpoint per NF Process:

* Each NF (e.g., open5gs-smfd, open5gs-amfd) runs its own HTTP server to serve /metrics, exposing counters and gauges.

#### **Summary factoids**

**Factoid**: “Open5GS metrics system was added to export performance counters and gauges for monitoring active PDP contexts and NF activity.”

**Factoid**: “Metrics in Open5GS are implemented via libprom and libpromhttp, built on top of libmicrohttpd, enabling an embedded HTTP server.”

**Factoid**: “The metrics subsystem uses a generic API in lib/metrics/, with a conditional Prometheus backend and a no-op fallback.”

**Factoid**: “Each NF (SMF, AMF, UPF, etc.) hosts its own /metrics endpoint, allowing individual scraping by Prometheus.”

**Factoid**: “Metrics definitions include counters, gauges, and potentially histograms to capture NF performance and load.”

**Factoid**: “The HTTP metrics server is embedded in the NF process, so exposing metrics doesn’t require external exporters.”

**Factoid**: “Prometheus scraping is enabled by building Open5GS with the Prometheus backend; otherwise, the metrics API is a stub.”

**Factoid**: “Example metrics include active session counts and internal NF resource telemetry.”

**Factoid**: “The conditional build approach allows operators to disable metrics support by omitting libprom-related dependencies.”

**Factoid**: “The architecture supports containerized deployments where each NF can be independently monitored.”

### **Key Takeaways for Ontology Expansion**

* **NFs & Binaries**:  
  + Binaries include open5gs-amfd, -smfd, -upfd, -pcfd, -nrfd, etc.
  + Each corresponds to a core function: AMF, SMF, UPF, PCF, NRF.
* **Configuration Formats**:  
  + Managed via YAML files (\*.yaml) under /etc/open5gs/; some CLI/web UI for subscriber entry.
  + JSON might be used for CLI/API (via Python open5gsapi SDK) [github.com+1open5gs.org+1](https://github.com/s5uishida/open5gs_5gc_ueransim_metrics_sample_config?utm_source=chatgpt.com)[enea.com+4sharetechnote.com+4telecompedia.net+4](https://www.sharetechnote.com/html/5G/5G_NetworkArchitecture_N1.html?utm_source=chatgpt.com)[pypi.org](https://pypi.org/project/open5gsapi/?utm_source=chatgpt.com).
* **RAN Integration**:  
  + Uses UERANSIM: nr-gnb connects to amf over N2 via SCTP; GTP-U on N3.
* **Logging & Troubleshooting**:  
  + Log files at /var/log/open5gs/\*.log.
  + Journalctl duplication settings.
* **Metrics & Monitoring**:  
  + Expose per‑NF HTTP endpoints for Prometheus metrics.
  + Config done via metrics: stanza in YAML.
  + Grafana dashboards visualize performance over time.

### Mapping to Your Pipeline

* **Ontology / Knowledge Graph**:  
  + **Nodes**: Interfaces (N1–N16), Protocols (SCTP, GTP‑U, PFCP, etc.), Configuration formats, Metrics endpoints, NFs and their binaries.
  + **Relations**: interface –uses→ protocol, NF –has binary→ open5gs-amfd, configuration –in format→ YAML.
* **RDF and Triples**:  
  + Example: :AMF :listensOnInterface :N2 . :N2 :usesProtocol :SCTP .
  + :UPF :exportsMetricsVia :PrometheusEndpoint .
* **Vector DB Context**:  
  + Store rich textual definitions, code snippets, summaries of interface behaviors, config YAML stanzas, metrics schemas.

## **5. Deployment Models**

### 1. Google Cloud Blog – *Deploying and operating cloud-based 5G networks*

* **Focus:** CSPs using cloud infrastructure to deploy UPF, DU/CU at edge and core. Discusses microservices in Kubernetes, latency optimization, location-aware deployments<https://techdocs.broadcom.com/content/dam/broadcom/techdocs/us/en/pdf/sde/telco-cloud/telco-cloud-platform/telco-cloud-platform-2-5/telco-cloud-platform-5G-edition-reference-architecture-guide-25.pdf?utm_source=chatgpt.com>

<https://cloud.google.com/blog/topics/telecommunications/how-csps-can-use-cloud-networks-to-deliver-5g?utm_source=chatgpt.com>

<https://www.nas.ewi.tudelft.nl/Publications/2021_cloud.pdf?utm_source=chatgpt.com>

<https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/building-robust-critical-networks-with-the-5g-system?utm_source=chatgpt.com>

<https://amslaurea.unibo.it/id/eprint/26454/1/Tesi%20Asma%20Noor.pdf?utm_source=chatgpt.com>

<https://arxiv.org/abs/2207.11936?utm_source=chatgpt.com>

<https://www.cisco.com/c/en/us/td/docs/wireless/ucc/smf/2025-01/config-and-admin/b_ucc-5g-smf-config-and-admin-guide_2025-01/m_smf-redundancy-support.html?utm_source=chatgpt.com>

<https://arxiv.org/html/2501.17964v2?utm_source=chatgpt.com>

<https://arxiv.org/abs/1911.03600?utm_source=chatgpt.com>

<https://www.etsi.org/deliver/etsi_ts/129500_129599/129500/17.08.00_60/ts_129500v170800p.pdf?utm_source=chatgpt.com>

<https://www.cisco.com/c/en/us/td/docs/wireless/ucc/smf/2025-01/config-and-admin/b_ucc-5g-smf-config-and-admin-guide_2025-01/m_smf-redundancy-support.html?utm_source=chatgpt.com>

#### **Key takeaways**

Cloud-Native Transition in Telco:

* CSPs are decoupling hardware and software via disaggregation: virtualized network functions (VNFs) evolved into containerized network functions (CNFs), reducing vendor lock-in and resource inefficiency associated with VMs and PNFs.
* Kubernetes and containers are the core cloud-native platform for CSPs to deploy both core and RAN elements across infrastructure.

Edge‑Core Workload Placement:

* Latency-sensitive components (gNB RU/DU/CU‑UP, UPF, ML/AI workloads) are best placed at the edge (≤ 5 ms RTT), while centralized functions (AMF, SMF, model training) fit in regional or public clouds.
* Google Distributed Cloud + Anthos provides a single control plane spanning edge, private, and public cloud, simplifying deployment orchestration and policy consistency.

Microservices & Orchestration:

* CNFs (e.g., UPF, CU‑UP) follow cloud-native design patterns using microservices, CI/CD pipelines, and managed Kubernetes platforms.
* Common infrastructure APIs, security, and lifecycle management are shared across edge and core workloads.

Latency Optimization & Location Awareness:

* Strategic placement of workloads based on latency, throughput, and service-level expectations. Example: UPF and CU‑UP deployed at far edge (<5 ms), while CU‑CP and AMF reside centrally.
* Hybrid deployment—leveraging private datacenters, telco edge, and public cloud—enables dynamic scaling and workload migration.

#### **Summary factoids**

**Factoid**: “CSPs transition from PNFs to CNFs using Kubernetes to gain vendor-agnostic, scalable infrastructure.”

**Factoid**: “Disaggregated network functions (RU, DU, CU‑UP, UPF) run at the edge under 5 ms RTT for optimal performance.”

**Factoid**: “Control-plane NFs like AMF and SMF are deployed in centralized cloud regions, not latency-critical.”

**Factoid**: “Anthos/GDC provides unified orchestration and policy across edge, private, and public 5G network deployments.”

**Factoid**: “CNFs adhere to microservice design and CI/CD lifecycle models for fast feature deployment and upgrades.”

**Factoid**: “Edge deployment enables CSPs to host both 5G-CNFs and third-party edge applications (e.g., AR/VR) on shared infrastructure.”

**Factoid**: “Use of hybrid clouds allows dynamic workload placement—edge for real-time tasks, cloud for batch or training.”

**Factoid**: “Telco workloads use infrastructure-as-code pipelines for security, scaling, and orchestration across distributed sites.”

**Factoid**: “Latency-aware deployment ensures UPF/CU-UP services are co-located near user for ultra-low latency applications.”

**Factoid**: “Container-based CNF approach enables feature-rich experiences while keeping costs and TCO under control.”

### 2. Linux Foundation OSS – *Kubernetes Native Infrastructure and Operator Framework for 5G*

* **Focus:** VNF → CNF migration; microservice containerization; Kubernetes Operators for edge/core orchestration. Distinguishes microservices vs monolithic CNFs<https://events19.linuxfoundation.org/wp-content/uploads/2019/07/OSS2019-HS-k8sNativeInfra-OperatorFor5Gedge.pdf?utm_source=chatgpt.com>

#### **Key takeaways**

VNF → CNF Migration & Containerization:

* Telco industry is migrating from VM-based **Virtual Network Functions (VNFs)** to Kubernetes-managed **Cloud-Native Network Functions (CNFs)** using containers and microservice patterns
* CNFs run in lightweight **Pods** rather than full VMs, supporting faster scaling and smaller resource footprints.

Kubernetes Operators & CRD-Based Lifecycle Automation:

* Operators (built with Operator SDK or via Helm/Ansible) manage Day‑2 lifecycle tasks: scaling, upgrades, backups, health checks.
* CustomResourceDefinitions (CRDs) enable declarative control over CNF instances, handling complex dependency orchestration.

Monolithic vs Microservice CNF Architectures:

* Presentation contrasts **monolithic container CNFs** (single Pod with many functions) vs **microservice CNFs** (each function in its own Pod with independent scaling)
* Microservice CNFs allow granular scaling and fault isolation; monolithic ones are simpler but less flexible .

Kubernetes Native Infrastructure (KNI) for Edge:

* KNI integrates Kubernetes and Linux networking optimizations (e.g. DPDK, SR-IOV, GPU plugins) for real‑time PN & CNF workloads .
* Kubernetes clusters on bare-metal edge sites provide low-latency host environments for real-time RAN and UPF services

Service Mesh Integration for SBA:

* Integrates **Istio/Envoy** for control-plane CNFs to provide service-to-service secure communication, tracing, and policy enforcement.
* Service mesh aids observability, traffic routing, and secure NF-to-NF API enforcement in SBA architectures.

Site Reliability (SRE) + Operator Model:

* K8s Operators enable **SRE practices**, automating monitoring, self-healing, and lifecycle management across Telco CNFs
* The architecture supports multi-site scalability, edge deployments, and cloud environments under a unified operational framework .

#### **Summary factoids**

**Factoid**: “Telco workloads are migrating from VNFs in VMs to Kubernetes-managed CNFs deployed as Pods.”

**Factoid**: “Kubernetes Operators, via CRDs, support complex lifecycle tasks like upgrades and scaling for CNFs.”

**Factoid**: “Microservice-based CNFs run each function in separate Pods, enabling independent scaling and failure isolation.”

**Factoid**: “Monolithic CNFs simplify deployment but lack scalability granularity compared to microservice designs.”

**Factoid**: “KNI combines Kubernetes with DPDK, SR‑IOV, and GPU networking to support telco-grade performance.”

**Factoid**: “Service mesh tools (Istio/Envoy) secure and manage APIs between control-plane CNFs in SBA.”

**Factoid**: “Operators enable Site Reliability Engineering for CNFs by automating observability, self-healing, and dynamic scaling.”

**Factoid**: “KNI supports edge and core deployments using unified Kubernetes control plane for CNF orchestration.”

### 3. Red Hat Blog – *Edge computing: How to architect distributed scalable 5G…*

* **Focus:** Multi-tier edge deployments; service mesh and observability; self-scaling 5G core CNFs<https://www.redhat.com/en/blog/5g-core-observability-edge?utm_source=chatgpt.com>

#### **Key takeaways**

Edge-to-Core Multi-Tier Architecture:

* 5G Core CNFs are deployed across **multiple tiers**: central regional data centers (“hub”) and distributed edge clusters (“spokes”) nearer to RAN infrastructure
* Use cases often involve **UPF and CU-UP at edge**, while control-plane CNFs (e.g., AMF/SMF) remain centralized

Service Mesh & Observability:

* Red Hat OpenShift Service Mesh (Istio/Envoy) provides critical **service discovery**, **secure mTLS**, tracing, and policy enforcement across CNFs
* Edge observability uses a **hub/spoke model** where logs, metrics, and traces are aggregated centrally (e.g., via Loki) from remote edge CNFs

Self-Scaling CNFs via Automation:

* Automation with **zero-touch provisioning (ZTP)** and **GitOps** supports cluster bursting and multi-site CNF scalability
* Red Hat Advanced Cluster Management (ACM) facilitates dynamic management and scaling of 5G CNF clusters via policy-based placement and lifecycle workflows

Operational Fabrics for Distributed Deployments:

* Three network fabrics are necessary:
  + **Cluster management fabric** (hub ↔ spokes)
  + **Inter-cluster connectivity** (spoke ↔ spoke)
  + **Access fabric** (connectivity to RAN or core APIs)

Scalable Deployment Patterns:

* Two deployment models outlined::
  + **Distributed UPF only**, with SMF controlling remote UPF instances
  + **Fully distributed CNF bundles** (e.g., SMF+UPF) per location
* Workloads are placed based on **DNN, TAC, cell\_id**, via NRF-assisted service discovery

#### **Summary factoids**

**Factoid**: “5G core CNFs are deployed in multi-tier hub-and-spoke architecture with control-plane centralized and user-plane at edge.”

**Factoid**: “Edge CNFs (UPF/CU-UP) are co-located with RAN, while CNFs like AMF/SMF remain in central cloud.”

**Factoid**: “Istio-based service mesh provides mTLS, secure discovery, tracing and policy across distributed CNFs.”

**Factoid**: “Logs, metrics, and traces from edge sites are aggregated centrally (e.g., via Loki) for full-stack observability.”

**Factoid**: “Zero-touch provisioning and GitOps enable on-demand scaling and burst deployments of 5G CNF clusters.”

**Factoid**: “ACM placement rules ensure CNFs are deployed to appropriate clusters based on location and capacity policies.”

**Factoid**: “Network fabric layers include hub management, inter-cluster connectivity, and RAN access paths.”

**Factoid**: “UPF selection is determined by SMF via NRF lookup using DNN, TAC, and cell\_id metadata.”

**Factoid**: “Partial edge deployment supports only UPF instances, while fully distributed bundles include SMF+UPF CNFs in remote sites.”

**Factoid**: “Service mesh federation allows secure cross-cluster communication and traffic splitting across hub and edge.”

**Factoid**: “Observability fabric leverages centralized policy enforcement across all distributed edge clusters.”

### 4. AWS Whitepaper – *5G Network Evolution with AWS*

* **Focus:** Deployment evolution from NSA to SA; cloud-native microservices; stateless architecture on AWS; orchestration with Kubernetes<https://d1.awsstatic.com/whitepapers/5g-network-evolution-with-aws.pdf?utm_source=chatgpt.com>

#### **Key takeaways**

NSA → SA Deployment Evolution:

* **Initial deployment** of 5G often begins in **NSA mode** (Options 3/4/7) where 5G RAN is anchored on 4G EPC for control-plane functions; CSPs later transition to **fully standalone (SA) core** architecture utilizing 5G core NFs (Option 2)

Cloud-Native, Microservices & Stateless Design on AWS:

* 5G core NFs benefit most from **cloud-native microservices** in **container-based, stateless architectures**, leveraging AWS managed services for data storage (ElastiCache, DynamoDB, Aurora), networking (EKS, ECS), service mesh (App Mesh), and discovery (Cloud Map)
* Stateless NFs externalize subscriber and session data (e.g., SUPI, context) to UDSF via AWS managed stores to maximize elasticity and failure recovery

CUPS & Edge Deployment:

* AWS enables Control-User Plane Separation (CUPS) by centralizing control-plane NFs in AWS Regions and deploying UPF (user-plane) functions at the edge via AWS Outposts, Wavelength, or Local Zones for ultra-low latency processing.

Orchestration with Kubernetes & DevOps:

* Kubernetes (EKS) is the primary orchestration layer, combined with AWS DevOps tooling (CodePipeline, CloudFormation, CDK) for CI/CD-enabled lifecycle management of NFs
* Automated blue/green or canary deployments align with 12‑factor microservice design, ensuring resilience and minimal downtime

Infrastructure Considerations:

* AWS offers performance-enhanced instances (SR-IOV, DPDK, ENA, huge pages, bare metal) to satisfy NF requirements for throughput (up to 100 Gbps) and latency-sensitive packet processing
* Network slicing is enabled through programmable infrastructure (API Gateway, App Mesh, CDK, Step Functions) that integrates with NF-level slice awareness (DNN, PCF policies)

#### **Summary factoids**

**Factoid**: “Initial 5G deployments use NSA (RAN on 5G, control on 4G EPC); evolution to fully standalone (SA) core follows in later phases.”

**Factoid**: “AWS supports stateless microservices for 5G NFs via containers and managed databases like DynamoDB, Aurora, and ElastiCache.”

**Factoid**: “CUPS is implemented using AWS Outposts and Local Zones to deploy UPF at the edge, while control-plane NFs remain in central regions.”

**Factoid**: “Kubernetes (EKS) orchestrates CNFs on AWS, with CI/CD pipelines enabling rapid lifecycle management and blue/green deployments.”

**Factoid**: “Stateless design externalizes NF state to UDSF-likes stores, enabling resilience and container-based scalability.”

**Factoid**: “Network slicing is orchestrated through AWS service mesh (App Mesh) and programmable infrastructure via CDK and Step Functions.”

**Factoid**: “AWS offers enhanced instances with SR-IOV, DPDK, huge pages, and bare-metal to meet packet-processing demands of CNFs.”

**Factoid**: “Container-based NF design on AWS aligns with the ‘12-factor app’ model, ensuring process isolation, observability, and lifecycle automation.”

**Factoid**: “AWS Direct Connect and Global Accelerator deliver dedicated, high-performance network paths for 5G workload placement.”

**Factoid**: “DevOps automation (CodePipeline, CloudFormation) allows NF operators to treat infrastructure as code, enabling reproducible, version-controlled deployments at scale.”

### 5. NGMN – *Experience on Cloud Native Adoption*

* **Focus:** UPF deployment Forms: hardware vs virtualized; edge platform guidelines; deployment suited to application and location<https://www.ngmn.org/wp-content/uploads/220128-Experience-on-Cloud-Native-Adoption-v1.1-Final.pdf?utm_source=chatgpt.com>

#### **Key takeaways**

UPF Deployment Strategies:

* UPF can be deployed either as **hardware appliances** or **virtualized instances** (VNFs or CNFs). Deployment choices depend on application scenario requirements (SLA, performance), vendor maturity, and infrastructure constraints
* Traffic steering capabilities at the edge rely on UPF to dynamically direct flows toward cloud or local applications using standardized APIs

Edge Platform Infrastructure:

* Edge platforms support a mix of **virtual machines, containers, and PaaS components** to host telco NFs near RAN infrastructure
* Edge IaaS commonly uses OpenStack-managed VMs; PaaS layers offer services like NAT, vFW, DNS, load-balancing, RNIS, bandwidth mgmt, user ID, and location data

Cloud–Edge Collaboration & Workload Placement:

* Workloads are placed based on network requirements, policy, and resource availability, managed by an automated deployment orchestrator that considers application needs, location, and platform capabilities
* UPF traffic-steering APIs enable application-level decisions—e.g., sending video stream traffic to edge for lower latency

**Summary factoids**

**Factoid**: “UPF deployment may be hardware-based or virtualized; choice depends on SLA, performance needs, and vendor maturity.”

**Factoid**: “Hardware UPF suits high-performance needs; virtualized UPF aids flexibility and geo-distribution.”

**Factoid**: “UPF steers traffic at edge using standardized APIs to route flows toward local or cloud-hosted applications.”

**Factoid**: “Edge IaaS platforms host VMs and containers managed by OpenStack and other platforms.”

**Factoid**: “Edge PaaS layers provide networking, RNIS, location, user identity, firewall, DNS, and load balancing services.”

**Factoid**: “An orchestrator deploys NFs and applications based on service requirements, policies, and resource templates mapped to location.”

**Factoid**: “Traffic steering APIs allow flow decisions to be made dynamically per application context (e.g., video, IoT).”

## **6. Network Planning & Topology**

### 1. Cisco – *UCC 5G SMF Configuration and Administration Guide*

* **Focus:** UPF redundancy: active/passive model, GR instances, N4 control sharing; Kubernetes spine/leaf hardware guidance<https://www.cisco.com/c/en/us/td/docs/wireless/ucc/smf/2025-01/config-and-admin/b_ucc-5g-smf-config-and-admin-guide_2025-01/m_smf-redundancy-support.html?utm_source=chatgpt.com>

#### **Key takeaways**

1:1 UPF Active/Standby Redundancy:

* **Model Structure**: Each **Active UPF** has a dedicated **Standby UPF**, connected via **Service Redundancy Protocol (SRP)**, using ICSR for session state sync.
* **Session Continuity**: The **Standby UPF takes over the same Sx/N4 address** during switchover, making the transition **transparent to the SMF**.

Control Interface Switchover (N4/Sx):

* **Monitoring Mechanism**: BFD and SRP monitor **Sx/N4 heartbeat**; failure triggers **active-to-standby switchover**.
* **Sx/N4 Checkpointing**: Active UPF replicates session and IP pool info to Standby during each association setup and periodic updates.
* **Switchover Handling**: Standby inherits **same IP/port**, so SMF doesn't detect failure; Sx/N4 heartbeat timeout must exceed SRP’s switchover delay.

Protocols & Diagnostic Control:

* **SRP Subsystems**: UPF includes VPP health and BGP monitoring (optionally with BFD) to support proactive switchovers.
* **Manual Recovery**: CLI commands (srp reset-sx-fail, force-pactv-to-actv-timeout) allow manual handling of exception cases.

#### **Summary factoids**

**Factoid**: “Cisco UPF implements 1:1 Active/Standby redundancy via SRP with ICSR-based state sync.”

**Factoid**: “Standby UPF assumes the same Sx/N4 address during switchover, making the transition transparent to the SMF.”

**Factoid**: “Sx/N4 control-plane heartbeat monitoring with BFD triggers fast UPF switchover in failure events.”

**Factoid**: “Active UPF replicates IP-pool and session context to Standby during Sx/N4 association and checkpoint cycles.”

**Factoid**: “Standby UPF starts in ‘Pending-Active’ until SRP elections and manual timeout configurations finalize switchover.”

**Factoid**: “VPP health and BGP monitoring are integrated into SRP for multi-layered UPF redundancy triggering.”

**Factoid**: “Manual CLI controls (e.g., srp reset-sx-fail) allow operators to override automatic failover conditions.”

**Factoid**: “SMF is unaware of standby UPF and always interacts with the active endpoint via stable Sx/N4 address.”

**Factoid**: “SRP Active/Standby redundancy is supported without dual-active scenarios due to address takeover and heartbeat control.”

**Factoid**: “Proper timing between SMF heartbeat and UPF SRP timeout is crucial to avoid false session drop detection.”

### 2. Ericsson Technology Review – *Building Robust Critical Networks with the 5G System*

* **Focus:** Generic NF Set concept: grouping NF instances for geo-redundancy and scaling; context-sharing across NF Set<https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/building-robust-critical-networks-with-the-5g-system?utm_source=chatgpt.com>

#### **Key takeaways**

Generic NF Set Concept:

* An **NF Set** is defined as a **group of interchangeable NF instances** of the same type (e.g., AMF Set, SMF Set) that share context and behave as a collective
* These NF Sets enable **geo-redundancy**, allowing instances distributed across regions to survive localized failures without disrupting service
* Membership in the NF Set supports **context-sharing**: session/user state is accessible by any active instance within the set, enabling failover continuity

Stateless NF Architecture Facilitates Set Operation:

* NFs within the same set are typically **stateless**, with the state stored externally in systems like UDSF — a key enabler for context mobility
* Stateless architecture simplifies upgrade workflows: NF instances within a set can run **different software versions**, allowing smooth rolling upgrades and service continuity

Resilience and Scalability Benefits:

* NF Sets support **N+M redundancy**, which avoids overprovisioning while ensuring seamless failover
* NF Sets reduce “signaling storms” during instance failover, as alternative instances pick up context without requiring UE re-registration
* Context-sharing across geographically distributed instances fosters **regional scaling**—instances can absorb localized increases in traffic or failure load

Operational Transparency:

* NF Sets are logically transparent to clients; e.g., SMF Set presents a single IP interface to UPF or other querying NFs, even though multiple instances are active
* This is managed via network-layer abstractions—e.g., load-balancers or anycast IP—for routing requests to any context-aware instance

#### **Summary factoids**

**Factoid**: “An NF Set groups interchangeable NF instances of the same type (e.g., AMF, SMF), enabling shared context and failover capability.”

**Factoid**: “NF Sets support geo-redundancy via distributed instances and externalized session state.”

**Factoid**: “Stateless NFs in an NF Set externalize their state to UDSF, enabling context retrieval by any instance.”

**Factoid**: “NF instances in a set can have different software versions, enabling seamless rolling upgrades.”

**Factoid**: “NF Set uses N+M redundancy, reducing overprovisioning while enabling fast failover.”

**Factoid**: “Context-sharing in NF Sets prevents signaling storms by avoiding re-registration during failover.”

**Factoid**: “An NF Set presents a single virtual interface (e.g., anycast IP) to clients despite multiple active instances.”

**Factoid**: “Geographically distributed NF Set instances enable localized scaling and resilience per region.”

### 3. 5G-SMART Report – *Second Report on 5G Network Architecture Options*

* **Focus:** Dual UE/device redundancy, disjoint UPF paths; redundancy only for user plane; relevance to industrial/private deployments<https://5gsmart.eu/wp-content/uploads/5G-SMART-D5.4-v1.0.pdf?utm_source=chatgpt.com>

#### **Key takeaways**

Disjoint UPF Paths for Redundancy:

* The report proposes **dual user-plane (UPF) paths** from UE to UPF, enabling **fully-disjoint redundancy** and interrupt-free session continuity
* Example: Two PDU sessions are each assigned to different UPFs via independent RAN and core paths, improving end-to-end availability beyond 99.999%

Dual‑Connectivity for Parallel Paths:

* Dual‑connectivity in RAN allows simultaneous connections to two gNBs. If these connect to separate UPFs, the UE maintains **parallel data flows** across disjoint user-plane routes
* The end-to-end model considers parallel redundancy of UE → gNB → UPF, significantly increasing system availability

Redundancy Focused on User Plane:

* The redundancy strategy applies **only to user-plane paths**; control-plane remains single-path, simplifying system design while ensuring data continuity

High Availability for Industrial/Private Deployments:

* Dual-path redundancy is particularly suited for **Non-Public Networks (NPN)** in industrial settings requiring ultra-high availability, such as manufacturing and automation
* Demonstrated model achieves calculated **E2E availability of 99.99996%** using redundancy across gNBs and UPFs

Reliability Modeling for System Design

* The report emphasizes building **Reliability Block Diagrams** to model series and parallel S/W & H/W components (e.g., gNB, UPF), enabling precise evaluation of E2E availability
* Both **Mean Time To Repair (MTTR)** and parallel systems are critical for hitting "5-nines" SLAs

**Summary factoids**

**Factoid**: “Disjoint UPF deployment with dual PDU sessions increases end‑to‑end availability by enabling parallel user‑plane paths.”

**Factoid**: “Dual‑connectivity through two separate gNBs allows simultaneous data flows to distinct UPFs, forming a redundant user‑plane chain.”

**Factoid**: “User-plane redundancy is achieved via dual UPFs; control-plane remains single-path to maintain management simplicity.”

**Factoid**: “Industrial/private 5G deployments can reach availability levels above 99.9999% using dual‑path UPF redundancy.”

**Factoid**: “E2E availability gains are calculated using parallel reliability models like Reliability Block Diagrams.”

**Factoid**: “Mean Time To Repair (MTTR) and path redundancy are the two primary enablers of achieving telecom-grade availability targets.”

**Factoid**: “Redundancy strategy leverages dual PDU sessions mapped across disjoint UPFs and RAN connections for ultra-reliable industrial use cases.”

### 4. ArXiv – *Packet Level Resilience for the User Plane in 5G Networks*

* **Focus:** Edge-level redundancy via PREOF (1+1 packet replication); UPF latency, jitter minimization; private 5G focus<https://arxiv.org/html/2501.17964v2?utm_source=chatgpt.com>

#### **Key takeaways**

URLLC Requirements & 5G Resilience Context:

* URLLC applications (industrial control, self-driving cars, remote surgery) need ultra-low packet loss and bounded latency; standard FRR mechanisms may not suffice

PREOF-Based 1+1 Packet Replication:

* The Packet Replication, Elimination, and Ordering Function (**PREOF**) supports **1+1 path protection** by replicating packets across two disjoint paths and eliminating duplicates at the receiver

Integration in 5G User Plane:

* PREOF can be integrated at either the **gNB** or the **UE** as a Protection Tunnel Ingress (PTI), replicating packets, while Protection Tunnel Egress (PTE) handles elimination and ordering

Disjoint Redundant UPFs:

* The replicated packets traverse two separate UPFs on disjoint paths to ensure packet delivery even if one path fails

Ordering Offload Mechanisms:

* Complex in-order delivery (ordering function) is offloaded from programmable switches to an external server (PTE-O) or implemented using DPDK/eBPF

GTP-U Encapsulation Compatibility:

* PREOF works transparently with GTP-U encapsulation, sending packets through standard tunnels used by UE → gNB → UPF paths

Trade-offs: Latency vs Resilience:

* The design balances reduction in disruption with a small increase in latency and complexity due to packet duplication and reordering overhead

#### **Summary factoids**

1. **Factoid**: “URLLC applications demand sub-millisecond latency and negligible packet loss, requiring stronger resilience than standard Fast ReRoute mechanisms.”
2. **Factoid**: “PREOF enables 1+1 path protection by duplicating packets over two disjoint UPF paths and eliminating redundant packets at the receiver.”
3. **Factoid**: “The PREOF mechanism can be deployed at the gNB or UE as a Protection Tunnel Ingress (PTI), replicating packets for redundancy.”
4. **Factoid**: “Replicated packets traverse two synchronized UPFs before reaching a Protection Tunnel Egress (PTE) node for elimination.”
5. **Factoid**: “Ordering of packets is offloaded to an external PTE-O server or implemented via eBPF/DPDK, reducing complexity in programmable hardware switches.”
6. **Factoid**: “PREOF is fully compatible with existing GTP-U tunnels, encapsulating replicated packets transparently to UPFs.”
7. **Factoid**: “1+1 path protection increases system resilience at the cost of slightly higher latency and resource usage.”
8. **Factoid**: “PREOF aligns with 3GPP Release 18’s support for packet duplication mechanisms to enhance URLLC resilience.”
9. **Factoid**: “Disjoint UP paths with replicated UPFs ensure session survival even if one UPF or path experiences a failure.”
10. **Factoid**: “Choosing between PTI placement at gNB versus UE reflects trade-offs in performance, complexity, and protection scope.”

### Key Insights for Ontology & Knowledge Graph

#### Deployment Models

* **Architectural Style**:  
  + *Monolithic* NFs vs *microservice* CNFs.
  + Define properties: :hasDeploymentType :Monolithic|:Microservice.
* **Orchestration**:  
  + Use of Kubernetes, Operators, service mesh.
  + Represent orchestration platforms (e.g. :Orchestrator :Kubernetes).
* **Edge vs Core**:  
  + Logic to deploy UPF and DU/CU at network edge for latency-sensitive services.
  + Use property :deployedAt :Edge|:Core.
* **Network Ownership**:  
  + Private vs public MNO deployment.
  + Capture vendor, orchestration constraints.

#### Network Planning & Topology

* **Instance Scaling**:  
  + UPF active-standby GR set; NF Set concept with geo-redundancy.
  + Define :hasRedundancyModel :ActiveStandby|:NFSet.
* **IP Scheme Separation**:  
  + CP vs UP traffic separation, addressing schemes for spine-leaf/K8 clusters.
* **Redundancy Mechanisms**:  
  + Packet-level replication via PREOF; dual-UE; disjoint paths.
* **Latency & Jitter Considerations**:  
  + Impact of edge placement, traffic steering, redundant paths.

### Next Steps

You can now:

* **Expand ontology** with deployment properties:  
  + :Node entities with :hasDeploymentType, :usesOrchestrator.
  + :UPF :hasDeploymentLocation :Edge.
* **Define instance planning rules**:  
  + RDF triple like :UPF :redundancyModel :ActiveStandby.
  + Use cardinality constraints (e.g., ≥2 instances).
* **Ingest paths and metrics** into KG and vector store:  
  + Document relevant snippets, config examples, whitepaper mathematical optimization models.

## **7. QoS and Performance Management**

### Optimizing 5G network performance with dynamic resource allocation & QoS modeling

*Literature review focusing on latency/jitter targets, SLA enforcement, URLLC/eMBB throughput* 📄<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11622846/> [pmc.ncbi.nlm.nih.gov+1pmc.ncbi.nlm.nih.gov+1](https://pmc.ncbi.nlm.nih.gov/articles/PMC11622846/?utm_source=chatgpt.com)[arxiv.org+7pmc.ncbi.nlm.nih.gov+7diva-portal.org+7](https://pmc.ncbi.nlm.nih.gov/articles/PMC10707587/?utm_source=chatgpt.com)

#### **Key takeaways**

Dynamic vs Static Resource Allocation:

* Static resource allocation cannot adapt to 5G’s highly variable traffic patterns; dynamic models like the **Maximum Capacity Model (MCM)** react in real time.
* MCM integrates dynamic bandwidth allocation, traffic prioritization, network slicing, and security early in session setup.

Latency & Jitter Targets for URLLC/eMBB:

* QoS metrics include **latency**, **jitter**, **packet loss**, **throughput**, **fairness**, **availability**, and **energy efficiency**
* Resources must meet **sub-ms latency** for URLLC and **high throughput (100sMbps)** for eMBB usage.

Machine Learning for Traffic Prediction & Prioritization:

* MCM applies **ML** to forecast upcoming UE traffic and adapt allocation strategy dynamically, optimizing QoS delivery.
* DRL and MDP-based schedulers in RAN optimize slot-level decisions for URLLC/eMBB coexistence and performance trade-offs

Network Slicing Integration:

* Network slices isolate resource pools (RAN and core) with tailored SLA enforcement across DRB, queue, and slice boundaries.
* Vertical slicing with QoS-aware models optimizes eMBB and URLLC traffic distribution per slice.

Security Trade-offs:

* Encryption introduces processing latency that must be factored into resource planning; MCM includes encryption overhead in initial allocation.

Scheduler Design: Radio vs Core:

* At the RAN level, **EDQAS** and **LDI** schedulers assign RBs to minimize uRLLC latency while maintaining eMBB throughput.
* At the core-plane, packet scheduling and queue prioritization use resource modeling to ensure E2E SLA compliance across domains.

**Summary factoids**

**Factoid**: "Static resource allocation fails in 5G; dynamic models like MCM are needed for real-time adaptability."

**Factoid**: "MCM integrates bandwidth allocation, traffic prioritization, encryption, and network slicing to enforce QoS."

**Factoid**: "Target QoS metrics include sub-ms latency and bounded jitter for URLLC, plus hundreds of Mbps throughput for eMBB."

**Factoid**: "ML-based traffic prediction is used in MCM to dynamically reallocate resources in anticipation of demand peaks."

**Factoid**: "MDP/DRL-based RAN schedulers dynamically balance URLLC and eMBB performance on slot and mini-slot timescales."

**Factoid**: "EDQAS/LDI schedulers at MAC layer schedule resource blocks to minimize uRLLC latency and eMBB rate loss."

**Factoid**: "Network slices partition resources with per-slice SLA-driven enforcement across RAN and core domains."

**Factoid**: "Encryption overhead is modeled in resource allocation decisions to meet QoS while ensuring security."

**Factoid**: "QoS parameters considered include latency, jitter, packet loss, throughput, spectral efficiency, and energy efficiency."

**Factoid**: "Increasing URLLC performance often involves puncturing eMBB traffic in real-time, trading off throughput."

**Factoid**: "QoS-aware slicing leverages vertical slice models to allocate bandwidth and priority between service types."

**Factoid**: "Dynamic resource allocation models consider fairness, availability, and service resilience during congestion."

**Factoid**: "Resource allocation frameworks now combine RAN-level scheduling with core-level queue management for SLA delivery."

### 5G QoS: Impact of Security Functions on Latency

*Examines how security (e.g., IPS) affects URLLC latency/jitter; includes throughput modeling and traffic policing* 📄<https://arxiv.org/abs/1909.08397> [arxiv.org+3arxiv.org+3researchgate.net+3](https://arxiv.org/abs/1909.08397?utm_source=chatgpt.com)

#### **Key takeaways**

URLLC QoS Requirements:

* URLLC services demand ≤1 ms one-way latency and ≥99.999% reliability

Security vs. Performance Trade-off:

* Inline IPS (Snort using DPDK in a VM) can meet median latency targets (<107 µs) but exhibits tail latency spikes (>1 ms) even without rule matching
* Latency spikes persist beyond cache warm-up, causing unpredictable behavior – a significant concern for URLLC

Packet Processing Architecture:

* Leveraging **Linux + DPDK + Snort** provides predictable low-latency packet handling. Hardware timestamping reveals worst-case delays (up to 2.5 ms at 99.999‑th percentile)

Predictive Load Modeling:

* Authors present a mathematical model to estimate **maximum IPS load** sustainable under URLLC latency budgets

Virtualization Overhead & Optimization:

* Virtualization layer contributes unpredictability due to interrupts, CPU frequency scaling, context switches
* Remedies include dedicated cores, isolcpus, interrupt affinity, and run-to-completion packet process design.

**Summary factoids**

**Factoid**: “URLLC requires ≤1 ms one-way latency and ≥99.999% reliability, posing stringent QoS targets.”

**Factoid**: “An inline DPDK-accelerated Snort IPS can meet median latency but shows tail latency spikes above 1 ms—problematic for URLLC.”

**Factoid**: “Worst-case latency with Snort IPS in a VM hit up to 2.5 ms at the 99.999th percentile even with no rule matching.”

**Factoid**: “Packet processing stacks using Linux + DPDK + Snort reveal virtualization unpredictability due to interrupts and CPU scaling.”

**Factoid**: “Mitigation for virtualization-induced latency includes dedicated cores, CPU isolation, interrupt pinning, and run-to-completion processing.”

**Factoid**: “A predictive model estimates maximum sustainable IPS load under URLLC, enabling SLA-driven capacity planning.”

**Factoid**: “Middleware security functions (like IPS) must be carefully optimized to avoid violating URLLC tail latency requirements.”

**Factoid**: “Hardware-assisted timestamping is essential for precise tail-latency measurement when evaluating IPS impact on URLLC.”

**Factoid**: “Even with minimal packet inspection, packet processing latency remains unpredictable due to virtual environment effects.”

**Factoid**: “To support URLLC with inline security, packet processing pipelines must be architected for low jitter and deterministic behavior.”

### Ultra-Reliable Low-Latency Communication – 5G Americas White Paper

*Defines latency/jitter targets, QoS Flow Descriptions (QFI), and resource reservation strategies* 📄<https://www.5gamericas.org/wp-content/uploads/2019/07/5G_Americas_URLLLC_White_Paper_Final__updateJW.pdf> [tec.gov.in+45gamericas.org+4researchgate.net+4](https://www.5gamericas.org/wp-content/uploads/2019/07/5G_Americas_URLLLC_White_Paper_Final__updateJW.pdf?utm_source=chatgpt.com)[devopedia.org](https://devopedia.org/5g-quality-of-service?utm_source=chatgpt.com)

#### **Key takeaways**

URLLC Latency & Reliability Targets:

* Aims for ≤ 1 ms user-plane latency, a significant improvement over LTE’s ~4 ms
* Strives for ultra-high reliability: 99.999% (five 9s) packet delivery and block error rates as low as 10⁻⁹

End-to-End Latency Reduction Strategies:

* Optimization across all components: modem processing, sub‑ms Transmission Time Intervals (TTIs), and minimal HARQ retransmissions
* **Edge computing/MEC** is critical to cut out ~100 ms transport delay.

QoS Flow Descriptions (QFI):

* QFI is used to mark individual QoS flows at the user plane, enabling differentiated forwarding such as URLLC vs eMBB

Resource Reservation & Scheduling:

* URLLC services utilize prioritized scheduling with preemption, mini-slots, and flexible TTIs to ensure timely transmission
* Physical-layer techniques include pulse shaping, ultra-robust coding, and redundant transmissions to minimize errors

Support for Disjoint User-Plane Paths:

* Multiple PDU session anchor points (UPFs) support disjointed paths with local breakout to reduce latency and increase resiliency

**Summary factoids**

**Factoid**: “5G URLLC targets ≤ 1 ms user-plane latency—a ~4× reduction compared to LTE.”

**Factoid**: “URLLC demands ultra-reliability (≥ 99.999%), with block error rates down to 10⁻⁹.”

**Factoid**: “QoS Flow Identifier (QFI) allows differentiated packet handling in the user plane.”

**Factoid**: “URLLC flows use resource reservation with preemption and mini-slots to meet sub‑ms deadlines.”

**Factoid**: “Edge computing eliminates ~100 ms transport delay, enabling end-to-end latency ≤ 1 ms.”

**Factoid**: “Flexible TTIs and robust coding techniques are used to reduce transmission latency and BER.”

**Factoid**: “Multiple UPFs with local breakout support disjoint user-plane paths for low-latency and resilience.”

**Factoid**: “URLLC scheduling uses prioritized access and TTI scaling to minimize delay and jitter.”

**Factoid**: “Reliable URLLC transmission combines mini-slots, redundancy, and HARQ optimization.”

**Factoid**: “Edge-deployed UPFs and MEC collaborate to enforce QoS and satisfy URLLC requirements.”

### Quality of Service (QoS) in 5G networks – 5G Hub Technologies

*Overview of QoS parameters, traffic shaping, marking/policing among URLLC, eMBB, mMTC* 🌐<https://5ghub.us/quality-of-service-qos-in-5g-networks/> [arxiv.org+45ghub.us+4devopedia.org+4](https://5ghub.us/quality-of-service-qos-in-5g-networks/?utm_source=chatgpt.com)[arxiv.org+5pmc.ncbi.nlm.nih.gov+5researchgate.net+5](https://pmc.ncbi.nlm.nih.gov/articles/PMC10707587/?utm_source=chatgpt.com)

#### **Key takeaways**

QoS Service Classes in 5G:

* **URLLC**: Requires ultra-low latency (≤1 ms), minimal jitter and high reliability.
* **eMBB**: High throughput demands (100s of Mbps) with moderate latency tolerance (≲100 ms).
* **mMTC**: Massive device support (up to 1M devices/km²) with low data rates and relaxed latency/jitter.

QoS Parameters & Metrics:

* Key performance indicators include:
  + **Latency**, **jitter**, **packet loss**
  + **Throughput, reliability, energy efficiency, fairness**
* Network slicing uses slice-specific SLAs enforced via bandwidth allocation and priority rules.

Traffic Shaping, Marking & Policing:

* **DiffServ** architecture applied: packets are classified, metered, and marked at ingress edge.
* **Per-hop behavior (PHB)** ensures scheduling and shaping along paths.

Scheduling & Resource Allocation:

* **eMBB scheduling** uses proportional fair or weighted mechanisms.
* **URLLC scheduling** employs preemption/puncturing or mini-slot pre-scheduling to meet hard latency.
* **mMTC** uses dynamic/random access strategies to support many sporadic devices.

QoS Enforcement in Slicing & RAN:

* **Resource blocks** are partitioned per slice to preserve isolation.
* **Priority queuing models** in transport/fronthaul ensure UL/DL traffic for URLLC is served first.
* Slice behavior templates (hard vs. soft slicing) configure dedicated or shared resources.

**ServiceClass**: URLLC, eMBB, mMTC

**QoSMetric**: Latency, Jitter, Throughput, Reliability, PacketLoss

**TrafficControlMechanism**: DiffServ, Preemption, Puncturing, MiniSlot

**SchedulerType**: PF\_RR, Preemptive, EDQAS

**SliceType**: HardSlice, SoftSlice

**ResourceUnit**: ResourceBlock, BandwidthSegment

**TopologyLayer**: RAN, Transport, Core

**Summary factoids**

**Factoid**: “URLLC flows demand ≤1 ms latency, low jitter, and ultra-high reliability; eMBB flows prioritize high throughput with moderate latency; mMTC supports up to 1 M devices/km² with low rate requirements.”

**Factoid**: “QoS metrics include latency, jitter, packet loss, throughput, reliability, fairness, and energy efficiency across 5G slices.”

**Factoid**: “Slice-specific SLAs are enforced through dynamic bandwidth allocation and priority queueing per network slice.”

**Factoid**: “Traffic shaping uses DiffServ: ingress classification, metering, and marking enable per-hop QoS enforcement.”

**Factoid**: “URLLC scheduling uses preemption/puncturing or mini-slots at RAN to meet sub-ms latency requirements.”

**Factoid**: “eMBB scheduling applies proportional fairness or weighted round-robin to balance throughput and fairness.”

**Factoid**: “mMTC devices use dynamic channel access protocols to handle high device density and sporadic traffic.”

**Factoid**: “Transport-priority queuing ensures UL/DL URLLC packets are served ahead of eMBB and mMTC traffic.”

**Factoid**: “Hard slicing uses dedicated resources per slice; soft slicing shares resources with prioritization—supporting mixed traffic isolation.”

**Factoid**: “Reliability and latency goals for URLLC are modeled via resource block reservation and robust coding at physical and transport layers.”

**Factoid**: “QoS enforcement spans RAN, transport, and core layers—ensuring consistent SLA adherence.”

**Factoid**: “Scheduler models use optimization and ML techniques to dynamically allocate resources based on real-time QoS demands.”

### 5G Quality of Service – Devopedia

*Explains QoS flows, QFI, mapping to DRBs, NAS/AS packet filtering* 🌐<https://devopedia.org/5g-quality-of-service> [net.in.tum.de+12devopedia.org+12gsma.com+12](https://devopedia.org/5g-quality-of-service?utm_source=chatgpt.com)

#### **Key takeaways**

QoS Flow Fundamentals & QFI:

* In 5G, **QoS Flows** are identified by a **4-bit QoS Flow Identifier (QFI)**, unique within each PDU session
* Flows are classified as **GBR** (Guaranteed Bit Rate) or **Non-GBR**, each with distinct QoS parameters

NAS-Level Packet Filtering (UE & 5GC):

* Packet Detection Rules (PDRs) in the **UPF** inspect downlink IP packets; UE applies similar **QoS rules** for uplink
* Filters use IP headers (addresses, ports, protocols) and up to Ethernet tags to determine packets belonging to specific QoS flows

AS-Level Mapping to DRBs via SDAP:

* The **SDAP sublayer** at both UE and gNB maps QoS flows to DRBs using QFI and is configured by RRC
* Multiple QoS flows can be multiplexed onto a single DRB, depending on QoS similarity
* Reflective QoS allows the UE to infer uplink mapping based on downlink rules using SDAP flags RQI and RDI

N3 Marking & Tunnel Integration:

* QFI is encoded in the **GTP‑U header** on the N3 interface between UPF and gNB for flow tracking
* SDAP then uses the QFI to select the appropriate DRB in the radio domain

Flow Setup & Signaling:

* **SMF** obtains QoS parameters (5QI, ARP, etc.) from UDM/PCF and configures QoS flows during PDU session establishment.
* It sends QoS rules to UE via AMF (N1), to gNB via AMF (N2), and to UPF via PFCP (N4)

**Summary factoids**

**Factoid**: “Each QoS Flow in 5G is tagged with a unique 4-bit QFI within its PDU session.”

**Factoid**: “QoS Flows are classified as GBR or Non‑GBR, with specific QoS profiles defined.”

**Factoid**: “UPF applies Packet Detection Rules (PDRs) and UE applies QoS rules to map packets to QoS flows at NAS layer.”

**Factoid**: “Packet filters use IP addresses, ports, protocols, and Ethernet tags to classify packets.”

**Factoid**: “SDAP sublayer maps QoS flows with given QFI to DRBs, supported by RRC-configured mapping rules.”

**Factoid**: “Multiple QoS flows can share a DRB if their service requirements align.”

**Factoid**: “Reflective QoS enables UE to derive uplink QoS-to-DRB mapping from downlink rules, using RQI/RDI flags.”

**Factoid**: “N3 GTP-U headers carry QFI for QoS identification between UPF and gNB.”

**Factoid**: “SMF configures QoS flows by distributing QoS rules and PDRs to UE (N1), gNB (N2), and UPF (N4).”

**Factoid**: “QoS control spans NAS classification, SDAP mapping, and GTP-U marking to enforce consistent end-to-end service quality.”

## 8**. Security**

### 5G Security Guide Version 3.0 – GSMA (July 2024)

*Covers 5G AKA, EAP-AKA′, AUSF/UDM roles, SIM provisioning, TLS/IPsec usage* 📄<https://www.gsma.com/solutions-and-impact/technologies/security/wp-content/uploads/2024/07/FS.40-v3.0-002-19-July.pdf> [en.wikipedia.org+6gsma.com+6cablelabs.com+6](https://www.gsma.com/solutions-and-impact/technologies/security/wp-content/uploads/2024/07/FS.40-v3.0-002-19-July.pdf?utm_source=chatgpt.com)

#### **Key takeaways**

Unified Authentication Framework & Protocols:

* Primary 5G authentication uses **5G‑AKA** and **EAP‑AKA′**, mandatory across both cellular and non‑3GPP access
* Secondary methods like **EAP‑TLS** are supported in private or enterprise settings using TLS 1.3

Authentication Functions & Context:

* **AUSF** provides 5G‑AKA authentication vectors (KSEAF anchor key) during initial UE authentication
* **UDM/ARPF** stores subscriber credentials and assists in both 5G‑AKA and EAP‑AKA′ flows

Secure Channel Usage: TLS & IPsec:

* **SBA APIs** (e.g., Namf, Nsmf, Npcf) require **mutual TLS (mTLS)** with client/server certificates and support for OAuth authorization
* **NAS/S1‑mode signaling over non‑3GPP access** is protected using **IPsec tunnels** (e.g., NWu/NWt), with NULL encryption allowed for trusted access

Secure Roaming & Inter‑PLMN Interfaces:

* **SEPP** secures inter‑PLMN signaling (N32) using TLS and optional **PRINS** for end‑to‑end integrity/authenticity
* **IPUPS** secures N9 UP‑plane GTP‑U traffic with IPsec filtering to ensure tunnel integrity between UPFs

Slice‑Specific Security:

* **NSSAAF** enables slice‑specific authentication via EAP flows for each S‑NSSAI, using client‑UICC credentials
* Slice isolation and management use TLS (1.2+/1.3) for slice APIs, OAuth for authorization, and mutual authentication among management entities

Advanced Cryptography & Zero‑Trust Practices:

* TLS profiles mandate AEAD suites (e.g., ECDHE, DHE), OCSP support, and the deprecation of weak ciphers
* GSMA endorses **Zero‑Trust** approach: define protect surfaces (SBA APIs), map flows, implement policies, and maintain ongoing monitoring

**Factoid**: “5G mandates support for 5G‑AKA and EAP‑AKA′ for unified, access‑agnostic authentication.”

**Factoid**: “EAP‑TLS can be used as a secondary authentication method in private or enterprise deployments.”

**Factoid**: “AUSF anchors the security key KSEAF during initial authentication and supplies authentication vectors.”

**Factoid**: “UDM/ARPF stores subscriber credentials and supports authentication via both AKA and EAP frameworks.”

**Factoid**: “All SBA APIs must use mutual TLS with client and server certificates, often coupled with OAuth authorization.”

**Factoid**: “IPsec tunnels (NWu/NWt) protect NAS signaling over non‑3GPP access, with optional NULL encryption for trusted networks.”

**Factoid**: “SEPP secures inter‑PLMN signaling on N32 via TLS or PRINS, ensuring integrity and confidentiality.”

**Factoid**: “IPUPS secures N9 GTP‑U traffic between UPFs with IPsec-based filtering to avoid tunnel spoofing.”

**Factoid**: “NSSAAF enables slice‑level authentication using EAP‑based credentials for each network slice.”

**Factoid**: “TLS 1.2/1.3 configurations require AEAD cipher suites and OCSP, removing legacy weak cipher support.”

**Factoid**: “Zero‑Trust security within 5G SBA demands explicit flow definitions, strict policies, and continuous monitoring.”

**Factoid**: “Slice management interfaces leverage OAuth for access control, combined with mTLS for secure management messaging.”

**Summary factoids**

### A Comparative Introduction to 4G and 5G Authentication – CableLabs

*Describes 5G-AKA, EAP-AKA′, EAP-TLS, compares trust models and flow diagrams* 🌐<https://www.cablelabs.com/insights/a-comparative-introduction-to-4g-and-5g-authentication> [nctatechnicalpapers.com+2cablelabs.com+2sciencedirect.com+2](https://www.cablelabs.com/insights/a-comparative-introduction-to-4g-and-5g-authentication?utm_source=chatgpt.com)

#### **Key takeaways**

Unified 5G Authentication Framework:

* **Three methods supported**: 5G‑AKA, EAP‑AKA′, and EAP‑TLS
* Framework is **access‑agnostic** (works over 3GPP and non‑3GPP networks) via EAP transport

Stronger UE Identity Protection:

* Use of **SUPI encrypted as SUCI** prevents exposure of permanent identifiers over the air

Expanded Trust Anchor Entities:

* New SBA functions: **SEAF**, **AUSF**, **UDM/ARPF**, **SIDF** participate in authentication — redefining trust roles

Advanced Key Hierarchy:

* Additional keys: **KAUSF**, **KSEAF**, **KAMF**, with SUCI-based flows — significantly deeper than 4G’s Ki → KASME

Support for Multiple Security Contexts:

* A single authentication run can establish **multiple security contexts**, enabling seamless mobility and multi-access coordination

Diverse Trust Models:

* **5G‑AKA/EAP‑AKA′**: Based on symmetric keys and subscriber credentials.
* **EAP‑TLS**: Based on PKI and X.509 certificates—ideal for non‑USIM environments

Roaming & SEPP Involvement:

* **SEPP** mediates inter‑PLMN communications, preventing downgrade attacks in roaming scenarios

**Summary factoids**

**Factoid**: “5G supports three authentication methods—5G‑AKA, EAP‑AKA′, and EAP‑TLS—within a unified, access‑agnostic framework.”

**Factoid**: “UE identity is encrypted as SUCI using public-key encryption, protecting SUPI from over-the-air exposure.”

**Factoid**: “New SBA functions SEAF, AUSF, UDM/ARPF, and SIDF coordinate to authenticate UE and manage key derivation.”

**Factoid**: “5G key hierarchy includes KAUSF, KSEAF, and KAMF, offering deeper security separation than 4G.”

**Factoid**: “A single 5G authentication session can establish multiple security contexts across access types.”

**Factoid**: “EAP‑TLS leverages X.509 certificates for authentication without requiring USIM, suiting BYOD or enterprise devices.”

**Factoid**: “EAP‑AKA′ is a symmetric key-based EAP method offering similar trust as 5G‑AKA but via EAP exchange.”

**Factoid**: “SIDF decrypts SUCI to SUPI, enabling identifier confidentiality with public-key protection.”

**Factoid**: “SEAF uses mutual TLS and PRINS to secure inter-PLMN communication and prevent downgrade attacks.”

**Factoid**: “Mapping of keys: KAUSF → KSEAF → KAMF ensures layered trust and key separation in 5G.”

**Factoid**: “SUPI encryption, SBA authentication functions, and deeper key derivation collectively enhance privacy and home‑network control in 5G.”

**Factoid**: “EAP‑TLS eliminates symmetric key dependency but introduces certificate lifecycle overhead—trading key management for lifecycle complexity.”

### Security Analysis of Critical 5G Interfaces

*Focuses on EAP-AKA′ and 5G-AKA signaling, protection via IPsec/TLS* 📄<https://www.techrxiv.org/users/692862/articles/682789/master/file/data/Security_Analysis_of_Critical_5G_Interfaces/Security_Analysis_of_Critical_5G_Interfaces.pdf> [techrxiv.org](https://www.techrxiv.org/users/692862/articles/682789/master/file/data/Security_Analysis_of_Critical_5G_Interfaces/Security_Analysis_of_Critical_5G_Interfaces.pdf?utm_source=chatgpt.com)

#### **Key takeaways**

Scope and Objective:

* The paper systematically assesses the security of critical 5G interfaces—those used for authentication signaling—mapping them to the STRIDE threat model

Authentication Protocols in Use:

* **5G‑AKA** and **EAP‑AKA′** remain the mandatory baseline for primary authentication over N1, N12, N13, and non-3GPP access
* EAP enables unified authentication across both 3GPP and non‑3GPP RATs.

Threats Identified at Key Interfaces:

* Interface endpoints (e.g., N1, N12) are vulnerable to **replay**, **man-in-the-middle (MitM)**, and **downgrade attacks** if security is improperly enforced

Protection Measures Required:

* **N1 (UE–AMF)** and **N2 (gNB–AMF)**: Must use **IPsec** or **NAS with integrity protection**, depending on access type
* **N12/N13 (AMF↔AUSF/UDM)**: Use **mTLS** with mutual authentication to secure control-plane interactions
* Inter-domain (e.g., N32, N16) mandates **TLS** plus forwarding integrity protocols such as **PRINS**, to prevent MitM

Residual Weaknesses:

* Formal security analyses (e.g., Tamarin) of 5G‑AKA highlight **linkability attacks**, authenticity challenges, and leakage in corner cases unless protocol extensions are applied

**Summary factoids**

**Factoid**: “5G-AKA and EAP-AKA′ are mandatory primary authentication protocols across N1, N12, and N13 interfaces.”

**Factoid**: “EAP enables unified authentication regardless of whether a UE accesses via 3GPP or non-3GPP RAT.”

**Factoid**: “Security threats like replay, MitM, and downgrade attacks target N1 and N12 if proper cryptographic protections are absent.”

**Factoid**: “N1 and N2 interfaces must be protected using IPsec for non-3GPP access, and NAS integrity algorithms over 3GPP access.”

**Factoid**: “Control-plane interfaces to AUSF and UDM (N12/N13) require mutual TLS to enforce authentication and confidentiality.”

**Factoid**: “Inter-domain interfaces such as N32/N16 need TLS with PRINS or equivalent to ensure end-to-end signaling integrity.”

**Factoid**: “Formal verification tools like Tamarin uncovered potential linkability vulnerabilities in 5G-AKA unless mitigations are implemented.”

**Factoid**: “Even when authentication exchanges are secure, implementations must ensure serving network binding via proper key derivation to prevent impersonation.”

**Factoid**: “Non-3GPP access (e.g., Wi‑Fi) requires IPsec tunnels (e.g., NWu) to secure NAS and user-plane traffic.”

**Factoid**: “Upgrade or downgrade prevention is critical—NAS and interface-level protections must enforce version/context awareness to avoid downgrade exploits.”

### 5G and Wi‑Fi RAN Convergence – WBA Whitepaper

*Details EAP‑5G, IKEv2, IPsec for N3 (untrusted access), and TLS/IPsec for backhaul* 📄<https://wballiance.com/wp-content/uploads/2021/04/WBA-5G-and-Wi-Fi-RAN-Convergence-Whitepaper-Online-Version-2021-V1.0.pdf> [sciencedirect.com+7wballiance.com+7nctatechnicalpapers.com+7](https://wballiance.com/wp-content/uploads/2021/04/WBA-5G-and-Wi-Fi-RAN-Convergence-Whitepaper-Online-Version-2021-V1.0.pdf?utm_source=chatgpt.com)

**Key takeaways**

Untrusted and Trusted Wi‑Fi Access:

* **Untrusted WLAN**: UE connects over public Wi‑Fi, then establishes IPsec signaling SA (NWu) with N3IWF using IKEv2 and EAP‑5G. NAS messages and user-plane traffic are encapsulated in child IPsec SAs.
* **Trusted WLAN**: UE first connects to a trusted WLAN AP (TNAP/TNGF). It uses EAP‑5G over IKEv2 to establish a signaling IPsec SA (NWt with NULL encryption) to the TNGF via layer-2 AAA proxy (e.g., RADIUS). Separate child IPsec SAs carry user-plane.

Auth Methods & Protocol Flow:

* **EAP‑5G** is used within IKEv2 to encapsulate NAS authentication messages (EAP‑AKA′ or 5G‑AKA). After successful UE authentication, both UE and (N3IWF/TNGF) generate a shared key and complete with EAP-Success.
* Signaling IPsec SA uses **TCP** transport for reliable NAS, and user-plane SAs use **NULL-encrypted or encrypted IPsec** depending on trust level.

Interfaces Outside 3GPP Scope:

* The **Ta** and **Yw** interfaces connect WLAN AP to TNGF/TWIF for trusted access, operating outside of 3GPP and managed within WLAN-specific domains.

Support for Wi‑Fi‑Only Devices:

* Devices without USIM (Wi‑Fi only) rely on **EAP‑TLS/EAP‑TTLS**, using certificate-based identity, particularly for SNPN (private 5G). This is still evolving in standards.

ATSSS Policy & QoS:

* ATSSS framework handles traffic steering, switching, or splitting across 3GPP and Wi‑Fi using rules from the SMF and policy from PCF via unified control-plane. (Detailed QoS integration is part of the framework.)

**Summary factoids**

**Factoid**: “UE establishes an IPsec signaling SA (NWu) with the N3IWF over IKEv2 and EAP‑5G when connected via untrusted WLAN.”

**Factoid**: “EAP‑5G encapsulates NAS-based authentication (5G‑AKA/EAP‑AKA′) within IKEv2 exchanges over non-3GPP access.”

**Factoid**: “User-plane over Wi‑Fi uses separate IPsec child SA(s) after PDU session establishment, with encryption based on trust.”

**Factoid**: “Trusted WLAN setups use layer‑2 authentication (802.1x via TNAP) followed by IPsec signaling SA with NULL encryption to avoid double encryption.”

**Factoid**: “Ta/Yw interfaces connect WLAN APs to TNGF/TWIF gateways, existing outside 3GPP scope but essential for trusted Wi‑Fi integration.”

**Factoid**: “Wi‑Fi only devices without USIM rely on certificate-based EAP‑TLS/EAP‑TTLS, especially in private SNPN environments.”

**Factoid**: “ATSSS enables steering or splitting of traffic between 3GPP and Wi‑Fi based on central policy from SMF/PCF.”

**Factoid**: “NAS messages over Wi‑Fi rely on TCP/IP transport within IPsec tunnels to ensure message reliability and ordering.”

**Factoid**: “Building trust zones in Wi‑Fi (untrusted vs trusted) determines whether IPsec encryption is NULL or standard to prevent redundancy.”

**Factoid**: “Integration of EAP‑5G, IKEv2, and IPsec ensures secure control and data-plane transport across Wi‑Fi links.”

### Security Guide – Secure Integration of 5G in Industrial Networks

*Explores authentication methods (5G‑AKA, EAP-AKA, EAP-TLS), IPsec tunnels, SIM provisioning via UDM/HSS* 📄<https://www.sciencedirect.com/science/article/pii/S0167739X24006095> [diva-portal.org+8sciencedirect.com+8cablelabs.com+8](https://www.sciencedirect.com/science/article/pii/S0167739X24006095?utm_source=chatgpt.com)

#### **Key takeaways**

Primary Authentication in Industrial 5G:

* Mandatory primary authentication methods include **5G‑AKA**, **EAP‑AKA**, and **EAP‑TLS**
* **Primary authentication ensures only authorized devices (e.g., industrial sensors, actuators) access the 5G network**

Secondary & Slice‑Specific Authentication:

* After primary authentication, networks can optionally invoke **secondary EAP** or **slice-specific EAP authentication** to access external resources or segmented network slices
* This enables fine-grained access control—for example, granting specific operator credentials for slice access.

SIM Provisioning & Credential Management:

* Device credentials (USIM) are securely managed by **UDM/HSS**, with explicit bindings required for industrial-grade security and lifecycle management
* The guide emphasizes secure USIM provisioning and update workflows—critical where device impersonation risks exist.

Use of IPsec Tunnels for Industrial RAN:

* **IPsec tunnels (NWu/NWt)** are essential for protecting both NAS signaling and user-plane traffic over industrial and wireless RAN connections, with proper SPIs and tunneling configurations
* Trusted and untrusted Wi‑Fi variations influence encryption use (e.g., NULL encryption in closed-loop control environments).

Industrial Security Prioritization:

* Industrial 5G deployments prioritize **safety and availability**, rather than confidentiality alone
* Integration of 5G in ICS includes **TSN mapping**, real-time scheduling, and protective measures against attack vectors specific to industrial control systems.

**Summary factoids**

**Factoid**: “Primary authentication in industrial 5G mandates use of 5G‑AKA, EAP‑AKA, or EAP‑TLS to verify device identity.”

**Factoid**: “Secondary EAP authentication enables access to external services or slice-specific networks using alternate credentials.”

**Factoid**: “UDM/HSS securely stores USIM credentials and supports provisioning workflows for industrial devices.”

**Factoid**: “SIM provisioning in industrial environments requires secure lifecycle procedures including revocation and updates.”

**Factoid**: “Industrial RAN requires IPsec tunnels (NWu/NWt) to protect NAS signaling and data-plane, with SPI protections.”

**Factoid**: “Trusted industrial deployments may use NULL-encryption under IPsec tunnels to avoid double-layer encryption overhead.”

**Factoid**: “Industrial 5G prioritizes availability and deterministic latency over confidentiality to maintain real-time control.”

**Factoid**: “TSN traffic must be mapped into 5G slices with real-time scheduling and IPsec security for industrial applications.”

**Factoid**: “Industrial 5G must mitigate wireless-specific threats (e.g., jamming, spoofing) via hardened SIM provisioning and tunnel integrity.”

**Factoid**: “Slice-specific EAP authentication is critical where multiple industrial operations share the same 5G infrastructure.”

### Key Insights for Your Knowledge Model

#### QoS and Performance Management

* **Latency/Jitter**: Defined for URLLC (~1 ms, <1 ms jitter), eMBB, mMTC.
* **Throughput Modeling**: Based on expected service (e.g., video ~10s of Mbps, IoT low kbps).
* **QoS Flows & QFI**: QFI uniquely identifies flows; GBR vs non‑GBR mapping to DRBs.
* **Traffic Shaping/Marking**: Use of filters in NAS/AS; policing at UPF/RAN.
* **Latency Impact by Security**: Measure adds from IPsec/TLS/IPS; trade-offs in URLLC.

#### Security

* **Authentication Flows**:  
  + **5G-AKA**: Challenge-response using AUSF/UDM, SUPI/SUCI.
  + **EAP-AKA′ & EAP-TLS**: Alternative methods with certificate or SIM-based models.
* **IPsec Tunnels**: For N3 (UE–N3IWF), N6 (UPF–DN), backhaul encryption.
* **TLS for SBA**: HTTP/2 + TLS for service-based NF interfaces (e.g., NRF, NSSF).
* **SIM Provisioning**: Handled by UDM/HSS; supports over-the-air and factory provisioning.
* **Security Anchors**: Use of SEAF, key derivation KSEAF, layered trust via AUSF/UDM.

### Mapping to Your Pipeline

* **Ontology Nodes**: :QFI, :LatencyTarget, :AuthenticationMethod, :SecurityProtocol.
* **Relations & Properties**:  
  + :URLLCFlow :hasLatencyTarget "≤1ms".
  + :UPF :performsTrafficPolicing :QoSFlow.
  + :AMF :usesAuthenticationFlow :5G‑AKA.
  + :SBA\_Interface :isProtectedBy :TLS.
* **RDF Examples**:  
  + :UE :authenticatedVia :5G‑AKA . :5G‑AKA :involves :AUSF, :UDM .
  + :QoSFlow :identifiedBy :QFI . :QFI :mappedTo :DRB .

## **9. Intent Interpretation and Mapping**

*(Translating natural-language requirements into technical configurations)*

### IETF Draft – Intent-Based Network Management Automation in 5G Networks

<https://www.ietf.org/archive/id/draft-jeong-nmrg-ibn-network-management-automation-03.html> [gsma.com+15ietf.org+15amdocs.com+15](https://www.ietf.org/archive/id/draft-jeong-nmrg-ibn-network-management-automation-03.html?utm_source=chatgpt.com) → Defines architecture with Network Intent Translator (NIT), combining IBN and NWDAF.

#### **Key takeaways**

Intent-Based Network Management (IBN) in 5G Core:

* The draft defines a 5G-native **Network Management Automation (NMA)** system driven by **Intent-Based Networking (IBN)**, enabling human or system-specified “intents” to translate into enforceable network policies
* It integrates **NWDAF** for analytics and monitoring, creating a **closed-loop system** capable of intent verification and dynamic adjustment

Architectural Components & Flow:

* **IBN User**: Origin of intents, which might include SLA, network-slicing, IoT, or V2X QoS intents, following TS‑28.312 semantics
* **IBN Controller**: Central hub translating intent into low-level network policies (via NIT) and distributing these policies to relevant NFs (VNFs/CNFs/PNFs)
* **Network Intent Translator (NIT)**: Maps high-level intent into specific policies understandable by NFs, using a *data model mapper* and optional NLP for natural language parsing
* **IBN Analyzer (NWDAF)**: Collects real-time telemetry via NF facing APIs, applies ML analytics, and feeds back performance data for audit and corrective action

Interfaces & Closed‑Loop Automation:

* **Consumer-Facing Interface**: Accepts user intent via models defined in 3GPP TS-28.312.
* **Analytics Interface**: Connects NWDAF to Controller for learning and policy refinement.
* **NF-Facing Interface**: Pushes translated policies to NFs.
* **Monitoring Interface**: Retrieves telemetry from each NF for auditing and loop closure

Representative Use Cases:

* **IoT Data Aggregation**: Intent expresses need to collect data from certain device types; NIT configures slice, QoS, and routing accordingly. NWDAF monitors traffic and validates intent success
* **Network Slicing and V2X QoS**: Intent can specify slice parameters and QoS metrics; system enforces slice creation, QoS configuration, then audits slice performance via NWDAF .

**Summary factoids**

**Factoid**: “Intent-Based Networking (IBN) enables high-level operator intent to be translated into network policy via NIT in 5G Pr-purposed NMA systems.”

**Factoid**: “The architectural framework integrates NWDAF as IBN Analyzer to create a closed-loop system—monitoring, validation, and policy refinement.”

**Factoid**: “NIT uses data-model mapping and optionally NLP to convert user intents (e.g., slice QoS, IoT data collection) into NF-specific configurations.”

**Factoid**: “IBN Controller dispatches translated policy rules to VNFs, CNFs, or PNFs using the NF-facing interface.”

**Factoid**: “Network telemetry is collected via a monitoring interface from NFs, analyzed by NWDAF, and used to audit intent success.”

**Factoid**: “Consumer-facing interface accepts user intent defined via 3GPP TS-28.312 intent schemas.”

**Factoid**: “Use cases like IoT aggregation and V2X QoS demonstrate how intent drives slice creation and SLA enforcement end-to-end.”

**Factoid**: “Closed-loop automation ensures intent enforcement accuracy by verifying via telemetry and updating policies as needed.”

### 3GPP – Intent-Driven Network Management (TS 28.312/TR 28.912 overview)

<https://www.3gpp.org/technologies/intent> [itu.int+4ietf.org+4policyreview.info+4](https://www.ietf.org/archive/id/draft-jeong-nmrg-ibn-network-management-automation-03.html?utm_source=chatgpt.com)[policyreview.info+43gpp.org+4ietf.org+4](https://www.3gpp.org/technologies/intent?utm_source=chatgpt.com) → Shows how high-level operator/user intents map to management actions via intent services.

#### **Key takeaways**

Intent‑Driven Management Service Architecture:

* 3GPP defines an **Intent‑Driven Management Service (MnS)** within the Service-Based Management Architecture (SBMA) that enables **intent producers** to accept, fulfill, and provide feedback on operator or user intents
* TS 28.312 specifies a **model-driven approach** separating **what** needs to be done (intent) from **how** it’s implemented (policies/actions)

Intent Lifecycle & Data Model:

* Intent operations include: **create**, **query**, **modify**, **delete**, and **activate/deactivate** modes; an **intent** comprises expectations, targets, and contexts
* Example expectations include **RadioNetworkExpectation** (e.g., target throughput, coverage area, energy savings) with context parameters like PLMN, TAC, RAT type

Enhanced Features in Release 18 (TR 28.912):

* Introduces **intent verification reports**, conflict detection, **feasibility checking**, capability discovery, and optional **AI/ML mapping** for intent processing
* Supports **intent-to-SON orchestration** and **management‑data analytics** for closed-loop, intelligence-driven network operations

Alignment with Other Standards:

* 3GPP intent services are designed to **cooperate with TM‑Forum Intent APIs** and **O‑RAN Intent** frameworks, enabling multi‑SDO ecosystem interoperability

**Summary factoids**

**Factoid**: “TS 28.312 defines an SBMA-based Intent‑Driven Management Service (MnS) that allows intent producers to accept, fulfill, and report on network or service intents.”

**Factoid**: “An intent is decoupled from execution—it specifies **what** (intent expectations), not **how** (policies/actions), aligning with a model-driven SBMA approach.”

**Factoid**: “Intent operations include create, modify, query, delete, activate, and deactivate, each managing intent lifecycle states.”

**Factoid**: “Intent content includes expectations (e.g., throughput targets, area coverage) linked to context objects such as PLMN and TAC.”

**Factoid**: “TR 28.912 enhances TS 28.312 with verification reports, conflict resolution, feasibility checks, AI/ML mapping, and intent‑driven SON orchestration.”

**Factoid**: “Intent producers may implement actions via rule-based, closed-loop, or AI/ML-driven mechanisms.”

**Factoid**: “3GPP intent services are interoperable with TM‑Forum and O‑RAN intent frameworks to enable cross-domain automation.”

**Factoid**: “Intent expectations can include energy savings targets balanced against service performance metrics.”

**Factoid**: “Release 19 (TR 28.914) extends intent models, including RAN-level intent autopolicies for 6G readiness.”

**Factoid**: “RadioNetworkExpectation in TS 28.312 includes attributes like coverageAreaPolygon, RAT types, target throughput, and latency thresholds.”

### ITU Journal – Intent‑Driven Network and Service Management

[https://www.itu.int/dms\_pub/itu‑s/opb/jnl/S‑JNL‑VOL3.ISSUE3‑2022‑A43‑PDF‑E.pdf](https://www.itu.int/dms_pub/itu%E2%80%91s/opb/jnl/S%E2%80%91JNL%E2%80%91VOL3.ISSUE3%E2%80%912022%E2%80%91A43%E2%80%91PDF%E2%80%91E.pdf) [sciencedirect.com+15itu.int+153gpp.org+15](https://www.itu.int/dms_pub/itu-s/opb/jnl/S-JNL-VOL3.ISSUE3-2022-A43-PDF-E.pdf?utm_source=chatgpt.com) → Deep-dive on intent lifecycles, classification, translation, and interfaces across TM Forum, ETSI ZSM.

#### **Key takeaways**

Intent Definition & Classification:

* Intent-driven management was first explored in IRTF‑NMRG drafts, defining **network intents** as high-level declarative objectives initiated by stakeholders across IBN/NF orchestration
* The article introduces a **flexible generic intent model**: each intent comprises a list of **IntentTargets**, **IntentExpectation** (desired outcomes), and **contextual scope** (e.g., PLMN, slice, network segments)

End-to-End IDM Architecture:

* The proposed architecture includes:
  + **Intent Ingestion/Recognition**: Interactive and iterative refinements toward well-formed intents.
  + **Intent Translation/Orchestration**: Maps intents via Intent Logic Units (ILUs) and Intent Logic Library (ILL).
  + **Intent Assurance**: Observes network behavior and detects “intent drift.”
  + **Intent Reporting/Abstraction**: Aggregates status into user-understandable feedback

Intent Lifecycle Management:

* The lifecycle stages include: **Create → Refine → Validate → Fulfill → Monitor → Assure → Report → Drift Detection → Corrective Action**
* Two nested control loops ensure::
  + Inner (auto): intent fulfilment and assurance via IBS and network components.
  + Outer (interactive): user involvement in clarifying/refining intents

Conflict Detection & Semantic Assurance:

* The model addresses **conflict resolution**, where overlapping intents are detected and need reconciliation or prioritization strategies (e.g. latest-first, priority-based) .
* **Assurance components** perform semantic alignment: map real telemetry to intent expectations, derive compliance scores, and trigger drift-aware corrective actions

Multi-SDO Alignment:

* Intent frameworks are mapped to standards from **TM Forum (IG1253)**, **3GPP TS28.312/28.912**, and **ETSI ZSM**, ensuring interoperability across intent definitions, NBI schemas, and orchestration infrastructures
* Supports AI-assisted execution (e.g., learning via LLMs for drift detection and policy suggestions)

**Summary factoids**

**Factoid**: “Network intents are high-level objectives defined as IntentTargets, IntentExpectations, and context constraints.”

**Factoid**: “End-to-end IDM architecture includes ingestion, translation (via ILUs/ILL), orchestration, assurance, and reporting components.”

**Factoid**: “Intent lifecycle follows Create→Refine→Validate→Fulfill→Monitor→Assure→Report→Drift→Correct.”

**Factoid**: “Inner loop automates fulfillment and assurance; outer loop involves user refinement and intent updates.”

**Factoid**: “Conflict detection mechanisms identify overlapping or contradictory intents, requiring resolution rules.”

**Factoid**: “Intent assurance semantically maps network telemetry to intent expectations to detect drift.”

**Factoid**: “Assurance systems may deploy LLMs to recommend policy changes when intent drift arises.”

**Factoid**: “Intent frameworks align with TM Forum, 3GPP, and ETSI ZSM to enable multi-domain interoperability.”

**Factoid**: “Intent ingestion supports both user-interactive refinement and automated processing toward machine-actionable formats.”

**Factoid**: “Intent translation uses reusable Intent Logic Units from an Intent Logic Library for mapping abstract intent to policies.”

**Factoid**: “Reporting abstracts low-level telemetry into intent-aligned summaries for operator decision-making.”

**Factoid**: “Assurance loop may trigger corrective workflows or escalate to user when semantic compliance falls below thresholds.”

### IEEE Access – NLP-Powered Intent-Based Network Management for Private 5G

[https://www.ihp‑microelectronics.com/.../goodarzi-mcnamara-ieee-access-11-36642-2023-2023.pdf](about:blank) [3gpp.org+2itu.int+2amdocs.com+2](https://www.itu.int/dms_pub/itu-s/opb/jnl/S-JNL-VOL3.ISSUE3-2022-A43-PDF-E.pdf?utm_source=chatgpt.com)[cloud.google.com+7ihp-microelectronics.com+7amdocs.com+7](https://www.ihp-microelectronics.com/php_scripts/publications/full_text_final_files/goodarzi-mcnamara-ieee-access-11-36642-2023-2023.pdf?utm_source=chatgpt.com) → Demonstrates natural-language intent interpretation, translation to NFV/RAN orchestrator tasks.

#### **Key takeaways**

NLP-Driven Intent Interface:

* The platform features a natural-language interface that accepts English intents (e.g., “provision a slice for throughput X in area Y”) and converts them into machine-interpretable policy definitions.

5G-CLARITY Intelligence Stratum:

* Builds on the 5G‑CLARITY architecture with an added **Intent Engine**, **ML Function Orchestrator (MLFO)**, and **Data Lake** for telemetry and model hosting.

Intent Translation via ML:

* An AI engine within the intent platform hosts ML models that map high-level intent to specific API workflows (NFV/RAN orchestrator commands).

Use Case Demonstrations:

* The paper evaluates three key private‑5G use cases with the intent interface:
  + Slice provisioning
  + Indoor positioning setup
  + Service deployment workflows
* Each is benchmarked for intent provisioning time.

Standards Alignment:

* The design aligns with intent frameworks from **3GPP SA5 (TS 28.312/TR 28.912)**, **ETSI ZSM**, and **TM Forum IG1253/IG1253A-D** for interoperability.

Future Enhancements:

* The authors propose integrating **LLMs** to further automate intent-to-API mappings and reduce reliance on manually defined workflows.

**Summary factoids**

**Factoid**: “A natural-language intent interface parses English statements into formal policy definitions for private 5G management.”

**Factoid**: “The 5G‑CLARITY platform includes an Intent Engine plus MLFO and Data Lake for telemetry-driven intent mapping.”

**Factoid**: “ML models convert high-level intents into API workflows targeting NFV or RAN orchestrators.”

**Factoid**: “Use cases—slice provisioning, indoor positioning, and service deployment—were benchmarked for provisioning latency.”

**Factoid**: “Platform aligns with 3GPP TS 28.312, ETSI ZSM, and TM Forum intent frameworks for cross-SDO integration.”

**Factoid**: “Future work includes embedding LLMs to automate intent-to-workflow translation and reduce manual configuration effort.”

### Amdocs + AvidThink – Achieving Success with Intent‑Driven Next‑Gen Networks

<https://www.amdocs.com/.../Achieving-Success-with-Intent-Driven-Next-Generation-Networks-Amdocs-AvidThink-aug22.pdf> [ihp-microelectronics.com+13gpp.org+1](https://www.ihp-microelectronics.com/php_scripts/publications/full_text_final_files/goodarzi-mcnamara-ieee-access-11-36642-2023-2023.pdf?utm_source=chatgpt.com)[itu.int+2amdocs.com+23gpp.org+2](https://www.amdocs.com/sites/default/files/2022-09/Achieving-Success-with-Intent-Driven-Next-Generation-Networks-Amdocs-AvidThink-aug22.pdf?utm_source=chatgpt.com) → Architectural analysis, domain/data-model mapping constraints, reliance on knowledge models.

#### **Key takeaways**

Scope of Intent-Driven Systems:

* Intent systems operate across **all network domains**—wireless, wired, data center, RAN—governing parameters in elements like RUs, DUs, and CUs to meet performance or power goals
* They **cannot invent new capabilities**; intents are constrained to existing features defined in data models and policy languages

Standards Ecosystem & SDO Progress:

* ETSI ENI and ZSM, TM Forum IG1253, and 3GPP R17 (via TS 28.312, TR 28.821) are involved in intent abstractions, SON, and framework definitions
* IETF’s intent lifecycle model featuring inner autonomous and outer human-involved loops is acknowledged

Architectural Components Required:

* Core components include: **Policy & Action Subsystem**, **Monitoring & Assurance Subsystem**, and optional AI/ML for mapping and remediation
* Intent updates trigger regeneration of policy translations and data model revisions

Gen-2 Orchestration System Requirements:

* Systems need vendor-agnostic, multi-domain orchestration spanning PNFs, VNFs, CNFs, bare metal, VMs, and containers
* Must work across multiple cloud environments with workload placement aware of SLA, cost, and resource context
* Closed-loop telemetry integration is critical for dynamic adaptation and SLA enforcement

**Summary factoids**

**Factoid**: “Intent systems govern RAN, core, and data-center domains, adjusting parameters in RUs, DUs, CUs to meet performance or power objectives.”

**Factoid**: “Intents are constrained to pre-defined capabilities; they cannot create new functionality beyond the data model.”

**Factoid**: “ETSI ENI, ZSM, TM Forum IG1253, and 3GPP TS 28.312 are the primary SDOs driving intent-driven network standards.”

**Factoid**: “Intent architectures require a policy/action engine, monitoring/assurance subsystem, and optional AI/ML-enhanced mapping.”

**Factoid**: “Policy translation regenerates and updates when intents change to keep intent-data models and actions aligned.”

**Factoid**: “Gen‑2 orchestration must span PNFs, VNFs, CNFs, VMs, containers, and serve across private/public/edge clouds.”

**Factoid**: “Closed-loop telemetry is mandatory for dynamic reconfiguration and SLA compliance within intent-driven systems.”

**Factoid**: “IETF’s intent model uses nested loops: an inner autonomous control loop and an outer user-in-the-loop refinement loop.”

**Factoid**: “Vendor data models vary, so intent systems need abstraction bridges across polymorphic device APIs and configurations.”

**Factoid**: “AI/ML can enhance intent mapping and assurance but remains bounded by intended expressiveness of policy and data models.”

### Key Ontology/RDF Mapping Insights

* **Intent**: Defines high-level requirements in natural language (e.g., “allocate URLLC slice for X”).
* **Translator/NIT**: Maps intents to policy/data models/NF configuration specs.
* **Model Nodes/Properties**:  
  + :Intent → :hasType (e.g., Throughput, Resilience)
  + :IntentTranslator → :generatesPolicy → :NetworkPolicy
  + :NetworkPolicy → applied to :NF, :Slice, :RAN
* **Vector DB Context**: store sample intents, mappings, policy templates, DSL snippets.

## **10. Constraints and Policies**

*(Regulatory, operator-specific policies, and SLA enforcement)*

### GSMA – Network Slicing Use Case Requirements

[https://www.gsma.com/futurenetworks/.../Network‑Slicing‑Use‑Case‑Requirements‑fixed.pdf](https://www.gsma.com/futurenetworks/.../Network%E2%80%91Slicing%E2%80%91Use%E2%80%91Case%E2%80%91Requirements%E2%80%91fixed.pdf) [en.wikipedia.org](https://en.wikipedia.org/wiki/5G_network_slicing?utm_source=chatgpt.com)[amdocs.com](https://www.amdocs.com/sites/default/files/2022-09/Achieving-Success-with-Intent-Driven-Next-Generation-Networks-Amdocs-AvidThink-aug22.pdf?utm_source=chatgpt.com)[nybsys.com](https://nybsys.com/intent-based-networking-for-telecom/?utm_source=chatgpt.com)[ietf.org](https://www.ietf.org/archive/id/draft-jeong-nmrg-ibn-network-management-automation-03.html?utm_source=chatgpt.com)[arxiv.org](https://arxiv.org/abs/2302.08544?utm_source=chatgpt.com)[gsma.com+13gsma.com+13jis-eurasipjournals.springeropen.com+13](https://www.gsma.com/futurenetworks/wp-content/uploads/2018/07/Network-Slicing-Use-Case-Requirements-fixed.pdf?utm_source=chatgpt.com) → Specifies use-case requirements, slice customization boundaries, regulatory impact frameworks.

#### **Key takeaways**

Network Slice Concept & SLA:

* A **network slice** is an end-to-end logical network operating on shared physical infrastructure that complies with agreed **Service Level Agreements (SLAs)** covering data speed, quality, latency, reliability, and security
* Slices combine **dedicated and shared resources** (e.g., processing, storage, bandwidth) and maintain **isolation** from other slices
* Operators can create **“business bundles”** by packaging multiple slice types (e.g., URLLC + eMBB) under a single SLA perimeter

Generic Slice Template (GST) & Slice Types (NEST):

* A **Generic Slice Template (GST)** defines the standard set of slice attributes (performance, functionality, isolation), applicable industry-wide
* A **Network Slice Type (NEST)** is a GST plus specific numeric values (e.g., latency <10 ms, 99.999% availability), tailored to vertical use cases

Domain Span & Multi-Operator Support:

* Slices can traverse multiple domains: **RAN, Core, Transport**, and may span **multiple operators** for roaming or multi-MNO deployments
* Descriptors enable slice **blueprints** to be shared across visited PLMNs, supporting standardization and service continuity

Customization Boundaries & Constraints:

* Operators must enforce **slice customization boundaries**—customers can select only from attributes predefined in GST/NEST; arbitrary changes aren’t allowed
* Use-case requirements (e.g., V2X or industrial IoT) drive attribute selection and mandatory slice capabilities like **slicing elasticity** and traffic isolation

Regulatory & Interoperability Considerations:

* GSMA highlights regulatory needs like **network neutrality**, **data sovereignty**, and cross-border interoperability for slicing
* Slicing must comply with inter-PLMN roaming frameworks; **slice descriptors and SLAs** play central roles in interoperability

**Summary factoids**

**Factoid**: “A slice is a customer SLA-backed end-to-end logical network running on shared infrastructure with performance bounds.”

**Factoid**: “Slices comprise both dedicated and shared resources and must be logically isolated from one another.”

**Factoid**: “Business bundles allow operators to combine multiple slice types (e.g., URLLC + eMBB) under a unified SLA.”

**Factoid**: “Generic Slice Templates define attribute domains (e.g., latency range, reliability targets) for network slices.”

**Factoid**: “Network Slice Type instantiates a GST with specific values tailored to a vertical use case’s requirements.”

**Factoid**: “Slices can span RAN, Core, and Transport domains and be shared across multiple operators for roaming support.”

**Factoid**: “Operators publish standardized slice blueprints to visited PLMNs to preserve SLAs and enable slice continuity.”

**Factoid**: “Slice customization is capped by GST/NEST—customers cannot exceed operator-defined attribute ranges.”

**Factoid**: “Use cases like V2X and industrial IoT require elasticity and strict traffic isolation from slices.”

**Factoid**: “Regulatory models must handle neutrality, data sovereignty, and cross-border slicing SLAs.”

**Factoid**: “Slice descriptors and SLAs are central to operator interoperability and roaming in sliced 5G environments.”

### PolicyReview.info – Mitigating Information Asymmetry in 5G Networks

<https://policyreview.info/articles/analysis/mitigating-information-asymmetry-5g-networks> [policyreview.info](https://policyreview.info/articles/analysis/mitigating-information-asymmetry-5g-networks?utm_source=chatgpt.com) → Discusses regulatory monitoring of slice performance, net-neutrality, NWDAF visibility.

#### **Key takeaways**

Regulatory Challenge of Information Asymmetry:

* 5G slicing enables dynamic creation of logically segregated networks, allowing customized traffic treatment per slice, which can unintentionally hinder regulatory oversight and conflict with net-neutrality principles.

NWDAF as a Regulatory Instrument:

* The paper proposes leveraging the **3GPP-standardized NWDAF** to provide regulators with automated access to performance metrics (e.g., packet loss, delay) on a per-slice basis.

Consumer–Producer Model & Data Exposure:

* NWDAF uses a **consumer–producer architecture** with standardized interfaces to periodically collect slice-level network KPIs and analytics, which may be exposed to external bodies via **NEF**.

Analytics Capabilities of NWDAF:

* NWDAF supports **historical analysis**, **predictive analytics**, **anomaly detection**, and involvement in **QoS sustainability** measurement, aiding transparency into ISP traffic management.

Proof‑of‑Concept Implementation:

* A prototype demonstrated NWDAF collecting eMBB slice metrics, enabling regulators to assess traffic differentiation practices effectively.

Regulatory Monitoring Use‑Case:

* NWDAF-derived metrics can support regulators in monitoring **net neutrality compliance**, especially where QoS differentiation could become discriminatory.

**Summary factoids**

**Factoid**: “5G slicing enables dynamic per-slice traffic treatment, creating potential information asymmetries vis-à-vis regulators.”

**Factoid**: “NWDAF can expose per-slice KPIs (packet loss, delay, QoS metrics) to external regulators via NEF.”

**Factoid**: “NWDAF employs a consumer–producer model with standardized interfaces for periodic KPI and analytics distribution.”

**Factoid**: “Analytics functions include historical trend analysis, predictive modeling, anomaly detection, and QoS sustainability reports.”

**Factoid**: “A proof-of-concept showed eMBB slice analytics can be used by regulators to detect traffic differentiation.”

**Factoid**: “NWDAF enables real-time regulatory monitoring of net neutrality compliance based on slice-level transparency.”

### 5G-SMART – Second Report on 5G Network Architecture Options

(Download via 5G-SMART project)  
 → Examines constraint models, isolation, operator policies, geo-constraints for slicing.

#### **Key takeaways**

Network Architecture Models for NPN:

* Explores deployment options: **Standalone Non-Public Networks (SNPNs)**, **Public Network Integrated NPNs (PNI-NPNs)**, and hybrids including shared RAN and URLLC-enabled TSN-integration
* Analyzes integration with edge cloud and Time-Sensitive Networking (TSN) tailored for smart manufacturing demands

Constraint Models & Slice Isolation:

* Highlights shortcomings in **3GPP Rel‑16 slicing mechanisms** regarding industrial Ethernet integration; proposes enhanced slice constraint models in Rel‑17
* Introduces dynamic traffic-driven forwarding rules and **security zones** via VLANs and slice-based isolation zones for segregated traffic flows

Operator Policy and Customization Boundaries:

* Defines operator policy boundaries for NPN deployments, including which features can be customized by tenant (e.g., coverage area, TSN integration level)
* Uses qualitative evaluation to align use-case requirements (latency, reliability) with appropriate NPN architecture (e.g., SNPN vs PNI-NPN)

Geo-Constraints & Deployment Recommendations:

* Geographic constraints (e.g., TAC-based slice availability) are enforced to restrict slice access to specific physical zones
* TSN and edge cloud placement decisions incorporate geographic and network proximity constraints to meet latency SLAs

**Summary factoids**

**Factoid**: “Deployment options include SNPN, PNI‑NPN, and hybrid models with shared RAN and TSN integration.”

**Factoid**: “Rel‑16 slicing lacks native support for industrial Ethernet integration; enhancements planned in Rel‑17.”

**Factoid**: “Security zones and VLAN-based segmentation enable dynamic isolation of traffic within slices.”

**Factoid**: “Operator policy defines tenant-customizable parameters like coverage area and TSN support levels.”

**Factoid**: “Qualitative mapping of use-case requirements (latency, reliability) to NPN architecture types informs deployment choice.”

**Factoid**: “Slice access can be geographically constrained using TAC or other location identifiers.”

**Factoid**: “Edge cloud and TSN functions are co-located based on geo-proximity to meet tight latency targets.”

**Factoid**: “Dynamic traffic-driven forwarding rules support adaptive slice isolation and performance optimization.”

### 3GPP TS 23.503 – Slice-Related Policy Control (Max Slice Data Rate)

<https://www.tech-invite.com/3m23/toc/tinv-3gpp-23-503_k.html> [ietf.org+1policyreview.info+1](https://www.ietf.org/archive/id/draft-jeong-nmrg-ibn-network-management-automation-03.html?utm_source=chatgpt.com)[amdocs.com+3tech-invite.com+3en.wikipedia.org+3](https://www.tech-invite.com/3m23/toc/tinv-3gpp-23-503_k.html?utm_source=chatgpt.com) → Technical spec: how S-NSSAI is limited by Max Slice Data Rate policy enforcement.

#### **Key takeaways**

Maximum Slice Data Rate per S‑NSSAI:

* Operators can define a **Maximum Slice Data Rate (MSDR)**—with separate uplink (UL) and downlink (DL) limits—for each S‑NSSAI, covering all GBR and non‑GBR flows
* The **PCF** is responsible for enforcing this policy slice-wide, by rejecting SM Policy or PDU session requests, adjusting AMBR/MBR, or modifying PCC rules

Monitoring Options: NWDAF vs. PCF:

* **With NWDAF**: The PCF subscribes to analytics (e.g., volume/duration) and calculates actual usage; when usage nears or exceeds MSDR, the PCF tightens restrictions, and when it drops, relaxes them
* **Without NWDAF**: The PCF reads the **Remaining MS** from UDR and deducts authorized AMBR/MBR values for sessions and flows. Similar restricting/releasing logic applies

Session-Level and Slice-Level Enforcement:

* For each PDU session or GBR rule creation/modification, the PCF checks if remaining slice RATE suffices; otherwise it rejects with **HTTP 403 EXCEEDED\_SLICE\_DATA\_RATE**
* When sessions or rules end, their budget is returned to the UDR, reopening capacity within slice limits .

Support for Redundancy and Group Rate Control:

* Multiple PCFs can coordinate via UDR with conditional updates and etags to maintain consistency in Remaining MD across distributed instances
* The same mechanism supports **Maximum Group Data Rate** limits applied to VN groups (collection of slice traffic)

Regulatory & Service Exceptions:

* Operators can configure exceptions (e.g., emergency, prioritized services) to exceed slice caps, applying differentiated charging if needed

**Summary factoids**

**Factoid**: “A Maximum Slice Data Rate (UL/DL) can be configured by the operator per S‑NSSAI to cap slice-wide throughput.”

**Factoid**: “PCF enforces Maximum Slice Data Rate by rejecting SM Policy or PDU session establishment if slice budget is exceeded.”

**Factoid**: “With NWDAF integration, PCF uses slice usage data (volume and duration) to apply or relax constraints dynamically.”

**Factoid**: “Without NWDAF, the PCF deducts used capacity from UDR via Session-AMBR and MBR during flow provisioning.”

**Factoid**: “If PCF rejects due to exceeded data rate, SMF returns HTTP 403 'EXCEEDED\_SLICE\_DATA\_RATE' to UE.”

**Factoid**: “Slice capacity is restored to UDR when sessions or GBR rules terminate or are modified downward.”

**Factoid**: “Multiple PCFs can synchronously enforce slice caps using conditional UDR updates with etags.”

**Factoid**: “Maximum Group Data Rate control is extended to VN groups similarly to slice-based rate limiting.”

**Factoid**: “Emergency or prioritized services can bypass slice data rate limits based on operator policy or regulation.”

**Factoid**: “Slice caps apply across both GBR and non-GBR flows—Non‑GBR via Session-AMBR, GBR via MBR in PCC rules.”

### ENISA – Threat Landscape for 5G

[https://www.researchgate.net/.../ENISA‑THREAT‑LANDSCAPE‑FOR‑5G‑NETWORKS.pdf](https://www.researchgate.net/.../ENISA%E2%80%91THREAT%E2%80%91LANDSCAPE%E2%80%91FOR%E2%80%915G%E2%80%91NETWORKS.pdf) [policyreview.info+12researchgate.net+12tech-invite.com+12](https://www.researchgate.net/profile/Marco-Lourenco-4/publication/337495468_ENISA_THREAT_LANDSCAPE_FOR_5G_NETWORKS/links/5ddbe9f0299bf10c5a3233e8/ENISA-THREAT-LANDSCAPE-FOR-5G-NETWORKS.pdf?utm_source=chatgpt.com) → Regulatory and isolation-related constraints; mapping of CP NFs and trust boundaries.

#### **Key takeaways**

CP–UP Trust Zones and NF Exposure:

* The report provides an **asset map** detailing 5G core NFs and CP trust zones, including **AUSF, SEAF, SEPP, SIDP, AMF, UDM, NSSAAF, AAA-server**, and **NFV management components**
* These functions reside within specific **trust zones**, with boundary rules enforced through certificate, API profiles, and isolation layers.

CP Interface Vulnerability Mapping:

* ENISA uses **nine “zoom‑ins”** mapping core functions to threat exposure, such as attacks on MANO, NFV, SDN, MEC, slicing management, and CP-interfaces

Regulatory & Operational Controls:

* Operational controls (identity & access management, configuration management, monitoring) are mandated under EU frameworks (NIS Directive, Telecom Security Act, EU 5G Toolbox)
* Includes **incident reporting obligations**, **asset/inventory audits**, and **service continuity/disaster recovery measures** under Article 13a of EU regulation

CP–UP Isolation in 5G Architecture:

* The architecture defines clear separation: **CP functions** within 5GC are isolated from UP components, enabling mitigation of CP-targeted attacks without UP disruption .
* NFV and SDN orchestrators (e.g., MANO, NFVO, VNF Manager) represent high-value targets due to their role in controlling both CP and UP infrastructure

Threat Actors and Attack Pathways:

* Threats include **DoS, MitM, software supply-chain attacks, SDN compromise**, and **side-channel attacks** on virtualization infrastructure
* Attackers exploit weaknesses in CP APIs, orchestration systems, slice control mechanisms, and cross‑domain trust models.

**Summary factoids**

**Factoid**: “ENISA identifies core control-plane NFs (AUSF, SEAF, SEPP, SIDP, AMF, UDM, NSSAAF) and NFV/MANO components as critical assets requiring trust zone isolation.”

**Factoid**: “ENISA’s threat landscape uses nine architectural ‘zoom‑ins’ to map vulnerabilities across CP–UP, NFV, SDN, MEC, slicing, and orchestration domains.”

**Factoid**: “EU regulations (NISD, Telecom Security Act, Article 13a) require strict operational controls: IAM, configuration management, logging, continuity planning.”

**Factoid**: “Control‑plane functions are architecturally isolated from user‑plane, mitigating CP-targeted attacks without affecting UP traffic.”

**Factoid**: “MANO, NFVO, and VNF/VIM orchestrators are critical CP assets vulnerable to supply chain and virtualization-layer attacks.”

**Factoid**: “Attacks on CP APIs and slice orchestration functions can compromise multiple domains—ENISA recommends hardened authentication and policy enforcement.”

**Factoid**: “Operational resilience requirements include incident reporting, network audits, disaster recovery, and service continuity under EU telecom law.”

**Factoid**: “NFV and SDN systems centralize both CP and UP control, making them high-value attack vectors for DoS or configuration tampering.”

**Factoid**: “Trust boundaries are enforced using certificate-based isolation, role-based access, and dedicated API profiles per NF in control plane.”

**Factoid**: “Threat actors may exploit cross‑domain trust misuse in orchestration and roaming as part of multi-stage CP attacks.”

### **Key Ontology/RDF & Policy Context**

* **Regulatory Constraints**:  
  + Data sovereignty: :Slice :mustResideIn :CountryX.
  + Net-neutrality: :OperatorPolicy prohibitsTrafficDifferentiation.
* **Operator Policies**:  
  + Closed-loop automation: :Policy :enforces :AutoHealing.
  + Slice constraints: :Slice :limitedBy :MaxDataRate.
* **SLAs**:  
  + Mapped as :Slice :hasSLA {latency, throughput, availability}.
  + Enforcement via NWDAF or NEF analytic reporting onto SLA compliance metrics.
* **RDF Triples**:  
  + :Slice1 :hasRegulatoryConstraint :DataSovereignty.
  + :Slice1 :hasSLA :URLLCSLA . :URLLCSLA :latencyTarget "≤1ms" .
  + :OperatorPolicy :conflictsWith :Intent ? to capture constraint/satisfaction relations.

### What’s Next?

Would you like help to:

1. Design **JSON-LD ontologies** aligning with 3GPP/ENI/TMF/ZSM standards?
2. Generate **sample intents and policies with YAML/DSL templates**?
3. Create **knowledge graph triples** representing intents, constraints, policies, SLA relations?

## **11. Inter-NF Relationships**

(Logical/physical connections, interface types, protocols, bandwidth/latency targets)

### Amantya Tech – What is the 5G User Plane Function (UPF)?

<https://www.amantyatech.com/5g-user-plane-function-upf> → Describes N3/N4/N6/N9 interfaces, UPF ↔ DN via N6, and emphasizes latency throughput roles [upcommons.upc.edu](https://upcommons.upc.edu/bitstream/2117/398691/1/TILP1de1.pdf?utm_source=chatgpt.com)[etsi.org+10amantyatech.com+10emblasoft.com+10](https://www.amantyatech.com/5g-user-plane-function-upf?utm_source=chatgpt.com).

#### **Key takeaways**

Functional Responsibilities of UPF:

* The UPF is a **core user-plane NF** in 5GC, responsible for **packet routing/forwarding**, **QoS enforcement**, **traffic buffering**, **usage reporting**, **uplink classification**, and acting as the **PDU‑Session anchor** for both intra- and inter-RAT mobility
* It interconnects the RAN to Data Networks (DN) and supports reflective QoS marking and application-level flow detection via Packet Flow Descriptions (PFDs)

Reference Points and Protocols:

* Four key interfaces: **N3** (gNB ↔ UPF), **N4** (SMF ↔ UPF using PFCP), **N6** (UPF ↔ Data Network), and **N9** (UPF-to-UPF chaining for branching or multi‑hop)
* N4 uses **PFCP** to install PDR, FAR, BAR, QER, and URR rules
* N3/N9 typically carry **GTP‑U** encapsulated user traffic between the RAN and UPF(s), and between UPFs

CUPS & Edge Deployment:

* UPF embodies 5G's **Control‑User Plane Separation (CUPS)**, enabling data-plane functions to be distributed—often near the edge—while controlled via SMF
* Cloud-native, microservices-based architectures enable UPF to scale dynamically via orchestration systems like Kubernetes, and to co-locate with edge computing environments

Performance and Hardware Offload:

* UPF implementations utilize VPP/DPDK and **SmartNIC offload** (e.g., Napatech) to achieve ultra-high throughput (≥ 100 Gbps) and low CPU usage, facilitating multi-tenancy and edge performance

**Summary factoids**

**Factoid**: “UPF serves as the primary User‑Plane NF in 5GC, handling packet forwarding, QoS, buffering, usage reporting, and mobility anchoring.”

**Factoid**: “The UPF interconnects RAN and Data Networks, and performs reflective QoS marking and application-specific flow detection.”

**Factoid**: “UPF uses four reference points: N3 (gNB), N4 (SMF via PFCP), N6 (Data Network), and N9 (UPF‑to‑UPF chaining).”

**Factoid**: “PFCP over N4 installs forwarding (FAR), buffering (BAR), QoS (QER), usage (URR), and detection (PDR) rules in UPF.”

**Factoid**: “GTP‑U encapsulated traffic traverses N3/N9 for user-plane data exchange between RAN and/or UPF instances.”

**Factoid**: “UPF fulfills the CUPS model by decoupling data-plane processing from control-plane SMF logic and enabling edge deployment.”

**Factoid**: “Cloud‑native UPF architectures using microservices and Kubernetes enable elastic scaling for diverse throughput demands.”

**Factoid**: “SmartNIC offload (Napatech) enables UPFs to process dual 100 Gbps data-plane loads with zero CPU usage.”

**Factoid**: “Branching via N9 or uplink classification allows UPF to intelligently split or redirect traffic flows.”

**Factoid**: “UPF supports IPv4/IPv6, NAT at N6, multi-tenancy, session anchoring, and reflective QoS for end-to-end service compliance.”

### Exploring the 3GPP UPF – Emblasoft

<https://emblasoft.com/blog/exploring-the-3gpp-upf-user-plane-function> → Covers N3 (gNB ↔ UPF), N6, N9, and N4 (SMF ↔ UPF via PFCP), includes protocol stacks and performance expectations [emblasoft.com](https://emblasoft.com/blog/exploring-the-3gpp-upf-user-plane-function?utm_source=chatgpt.com).

#### **Key takeaways**

Control-User Plane Separation (CUPS) & UPF Role:

* UPF represents the user-plane evolution of 3GPP's **CUPS** strategy (from Release 14), enabling independent scaling, edge deployment, and cloud-native operation
* CUPS decouples packet processing (UPF) from session control (SMF), enabling isolated feature upgrades and flexible deployment placement

UPF Functional Capabilities, As defined in TS 23.501 Release 15+, UPF includes functions such as:

* **Intra-/Inter-RAT mobility anchoring** and **IP address allocation**
* **PDU Session anchoring** to Data Networks
* **Packet routing/forwarding**, **classification**, **branching**, **buffering**, and **application-layer traffic inspection** (via PFDs)
* **Policy enforcement**: gating, redirection, traffic steering
* **Lawful interception**, **usage reporting**, and **QoS mechanisms** including rate enforcement, reflective marking, SDF→QoS mapping
* **Transport-level marking**, **downlink buffering and data notification**, **end-marker signaling during handover**, and **GTP-U packet duplication/elimination**

Reference Interfaces and Protocol Stacks:

* UPF interacts via four key reference points:
  + **N3**: gNB ↔ UPF – carries GTP-U user-plane traffic
  + **N4**: SMF ↔ UPF – PFCP protocol installs PDR, FAR, BAR, QER, and URR rules
  + **N6**: UPF ↔ Data Network – bridges to internet or other service platforms
  + **N9**: Inter-UPF chaining – for multi-hop, split-anchor, or breakout scenarios

Performance & Deployment Expectations:

* UPF must support cloud-native, microservice-based orchestration (e.g., Kubernetes) for elasticity and automation
* Edge placement optimizes low-latency, high-bandwidth connectivity for user-plane traffic

**Summary factoids**

**Factoid**: “UPF is the CUPS-based user-plane NF in 5GC, enabling separation from SMF and independent scaling/deployment.”

**Factoid**: “CUPS allows UPF to evolve and scale independently, enabling edge deployment for low-latency use cases.”

**Factoid**: “UPF handles mobility anchoring, IP allocation, routing, classification, branching, buffering, and PFD-based application detection.”

**Factoid**: “Packet-level policy enforcement in UPF includes gating, redirection, traffic steering, reflective QoS marking, and rate control.”

**Factoid**: “Scientific interfaces include N3 for GTP-U, N4 for PFCP rule installation, N6 for Data Network connectivity, and N9 for UPF-to-UPF chaining.”

**Factoid**: “PFCP enables UPF to install PDR, FAR, BAR, QER, and URR via N4 under SMF control.”

**Factoid**: “N9 chaining supports branching, multi-anchoring, and breakout of traffic flows in complex deployment scenarios.”

**Factoid**: “Edge-deployed UPFs and cloud-native architecture support elastic scaling managed via orchestration platforms.”

**Factoid**: “UPF supports GTP-U-level packet duplication and elimination for handover and redundancy.”

**Factoid**: “Intent-driven orchestration of UPF includes lifecycle automated deployment, test validation, and performance assurance using DevOps aligned tools.”

### ETSI TS 128 552 (Release 16) – Measurement of N3/N4/N6

<https://www.etsi.org/deliver/etsi_ts/128500_128599/128552> → Provides bandwidth and latency metrics per interface; essential for QoS-based modeling [cisa.gov+12etsi.org+12etsi.org+12](https://www.etsi.org/deliver/etsi_ts/128500_128599/128552/16.10.00_60/ts_128552v161000p.pdf?utm_source=chatgpt.com).

#### **Key takeaways**

N3 Interface (gNB–UPF over GTP-U):

* **Data volume metrics** (GTP.In/OutDataOctetsN3) measure incoming/outgoing GTP packet octets to monitor transport bandwidth usage, guiding capacity planning
* **Packet loss counts** (GTP.InDataPktLossN3) track packet loss on the interface per QoS class, crucial for identifying service degradation
* **Round-trip delay measurements**:
* GTP.RttDelayN3DlPsaUpfMean.DSCP: average downlink RTT per DSCP class.
* ...Dist.Bin.DSCP: distribution of delays in configurable bins.
* Equivalent uplink RTT metrics for I-UPF

N4 Interface (SMF ↔ UPF via PFCP):

* **Session establishment counters** track N4 session setup attempts and failures (SM.N4SessionEstabReq, SM.N4SessionEstabFail)
* **Session report counts** record the number of N4 session report messages sent and acknowledged (SM.N4SessionReport, SM.N4SessionReportSucc)

N6 Interface (UPF ↔ Data Network):

* **Link usage metrics** monitor incoming/outgoing IP octets (IP.N6IncLinkUsage, IP.N6OutLinkUsage) to understand real user-plane load
* Metrics align with IETF RFC 5136 definitions for IP link usage, facilitating interoperability

QoS, Slicing & Performance:

* Measurements are **per-QoS-class (DSCP)** and can also be broken out by **S‑NSSAI**, supporting slice-aware monitoring
* UL/DL metrics contribute to KPIs like throughput, latency, jitter, loss, enabling SLA compliance checks across slices

**Summary factoids**

**Factoid**: “GTP octet counters on N3 (GTP.In/OutDataOctetsN3) monitor transport bandwidth usage per QoS class.”

**Factoid**: “Packet loss on N3 is tracked via GTP.InDataPktLossN3QoS, enabling detection of per-class service degradation.”

**Factoid**: “Average and distribution metrics of N3 round-trip delay per DSCP (e.g., GTP.RttDelayN3DlPsaUpfMean.DSCP) support QoS-level latency modelling.”

**Factoid**: “N4 PFCP session metrics measure N4SessionEstabReq/Fail and N4SessionReport/ReportSucc to assess control-plane signaling performance.”

**Factoid**: “N6 interface metrics (IP.N6IncLinkUsage, IP.N6OutLinkUsage) record IP-layer data volume, consistent with RFC 5136 counters.”

**Factoid**: “Per-QoS-class and per-slice measurements enable slice-specific SLA validation across UPF interfaces.”

**Factoid**: “N3, N4, and N6 measurement data provide key inputs to dynamic QoS and scaling actions in intent-driven orchestration.”

**Factoid**: “Monitoring of GTP RTT and packet loss informs real-time traffic steering or re-orchestration decisions.”

**Factoid**: “Control-plane PFCP session metrics support monitoring of SMF‑UPF coordination health.”

**Factoid**: “UESPLIT per-interface metrics feed operators’ QoS dashboards and NWDAF analytics for performance assurance.”

### Dynamic UPF Placement & Chaining in 5G Networks

<https://www.sciencedirect.com/science/article/pii/S1389128622002900> → Discusses re-chaining and latency constraints when placing UPFs near edge core [etsi.org+11sciencedirect.com+11emblasoft.com+11](https://www.sciencedirect.com/science/article/pii/S1389128622002900?utm_source=chatgpt.com)[emblasoft.com+1iplook.com+1](https://emblasoft.com/blog/exploring-the-3gpp-upf-user-plane-function?utm_source=chatgpt.com).

#### **Key takeaways**

Dynamic UPF Placement with MEC:

* The paper formulates the UPF placement and chaining reconfiguration (UPCR) problem in a Multi-access Edge Computing (MEC) environment to support user mobility while optimizing cost and QoS

Multi-Objective Optimization Model:

* Uses an Integer Linear Programming (ILP) model to minimize costs (e.g., instantiation, migration, resource usage) while satisfying QoS thresholds, particularly latency

Heuristic Algorithm: DPC‑UPCR:

* Proposes a “Dynamic Priority and Cautious UPCR” heuristic that achieves near-optimal solutions (within 15%) significantly faster than the ILP

Optimal Reconfiguration Scheduler:

* Introduces a scheduler based on **optimal stopping theory**, which determines ideal timing for UPF reconfiguration by balancing latency violations and a QoS threshold

Trade-Offs: Reconfiguration vs Performance:

* Results demonstrate the heuristic scheduler reduces both the number of UPF reconfigurations and QoS violations compared to baseline approaches

Scaling & Mobility Handling:

* Dynamic chaining accommodates user mobility by relocating user-plane anchors (UPFs) closer to end-users, minimizing latency due to handovers

**Summary factoids**

**Factoid**: “UPCR (UPF placement and chaining reconfiguration) addresses UPF placement in MEC environments to handle user mobility.”

**Factoid**: “An ILP-based model optimizes UPF placement to minimize deployment and migration costs under QoS constraints.”

**Factoid**: “The DPC‑UPCR heuristic achieves within 15% of optimal placement outcomes with significantly reduced computation time.”

**Factoid**: “An optimal stopping theory–based scheduler triggers UPF reconfiguration when latency thresholds are breached.”

**Factoid**: “DPC‑UPCR outperforms baseline schedulers by reducing reconfig event count and QoS violations.”

**Factoid**: “UPF repositioning supports proximity anchoring to minimize latency during access node handovers.”

**Factoid**: “Cost-QoS trade-offs are balanced via dynamic UPF chaining instead of static placement clusters.”

**Factoid**: “User mobility patterns drive dynamic reconfiguration decisions rather than purely periodic adjustments.”

**Factoid**: “Scheduling parameters—e.g., latency violation rate and QoS thresholds—govern when to deploy UPF migrations.”

**Factoid**: “The heuristic scheduler aims to minimize cost of migration while preserving slice-level performance during reconfigurations.”

### 5G-PPP – View on 5G Architecture (Whitepaper)

<https://5g-ppp.eu/wp-content/uploads/2019/07/5G-Architecture-White-Paper_v3.0_PublicConsultation.pdf> → Contains NF logical/physical topology, inter-NF connections, placement best practices [5g-ppp.eu+1iplook.com+1](https://5g-ppp.eu/wp-content/uploads/2019/07/5G-PPP-5G-Architecture-White-Paper_v3.0_PublicConsultation.pdf?utm_source=chatgpt.com)[researchgate.net+9arxiv.org+9researchgate.net+9](https://arxiv.org/abs/2204.00178?utm_source=chatgpt.com).

#### **Key takeaways**

Multi‑Domain and SDN/NFV‑Based Architecture:

* The architecture envisions a **multi-domain orchestration framework** spanning RAN, transport/backhaul, core/control and service levels, built on **ETSI NFV-MANO** and **software-defined networking (SDN)** integration
* It supports **application-aware orchestration**, integrating NFV-MANO with service-specific managers and SDN agents at edge/cloud/transport nodes

Logical and Functional Topology:

* Defines logical separation of RAN into **softwarized layers**: DU/PHY/MAC at edge and centralized PDCP/RRC as VNFs in data-centers
* Physical topology includes **core NFVI**, **transport**, **edge NFVI**, **fronthaul/backhaul networks**, and **WLAN APs/LTE small-cells**
* Emphasizes modular NF deployment: DU, CU-CP, CU-UP, and gNB split across centralized and distributed layers

Inter‑NF Connections & Placement Practices:

* Advocates **distributed control plane and centralized user-plane splits** (e.g., CU‑CP vs CU‑UP; CUPS architecture with SMF-UPF separation)
* Supports **edge-native functional placements**: DU/CU-UP/UPFs deployed at edge for low latency, RRC at central cloud for macro-control .
* SDN agents are co-located with domain-specific components (RAN, transport), controlled via NFVO/VNFM and SDN controllers

Best Practices & Programmability:

* Emphasises **Northbound Intent/SLA** ingestion, closed-loop telemetry, and AI/ML-driven policy translation via NFV-MANO enhancements
* Urges balanced placement between **granular programmability** and **hardware acceleration**, such as offloading PHY and MAC at DU, more flexibility at PDCP/RRC layers
* Encourages use of **Transport API (T‑API)**, intent-based North-Bound Interfaces, and alignment with SDN frameworks like ONF

**Summary factoids**

**Factoid**: “The 5G‑PPP architecture integrates multi-domain orchestration through ETSI NFV-MANO and SDN agents across RAN, transport, core, and edge.”

**Factoid**: “RAN functions are split: DU/PHY/MAC run at edge for performance, while PDCP/RRC run centrally as VNFs.”

**Factoid**: “Core network topology includes NFVI in core and edge, fronthaul/backhaul, and WLAN/small-cell components.”

**Factoid**: “CUPS architecture enables independent scaling by separating CU-CP from CU-UP and using SMF–UPF control-plane splits.”

**Factoid**: “Edge deployment of DU, CU-UP, and UPF ensures low-latency, bandwidth-intensive data handling close to users.”

**Factoid**: “SDN agents co-locate with domain NFs and are managed via NFVO/VNFM, enabling programmable domain control.”

**Factoid**: “Intent ingestion, telemetry, and AI/ML closed-loop orchestration are best-practice control methods for slice and service SLAs.”

**Factoid**: “PHY and MAC layers benefit from hardware acceleration at the DU, while PDCP/RRC benefit from centralized programmability.”

**Factoid**: “Transport-level programmability leveraged via T‑API and intent-based NBIs integrated with ONF/SDN frameworks.”

**Factoid**: “Application-aware orchestration includes plug-in service and function managers integrated into MANO for vertical-specific services.”

### **🧩 Key Ontology & RDF Considerations**

* **Interfaces/Protocols**:  
  + Define each interface (N2, N3, N4, N6, N9) as an entity with properties: :usesProtocol, :connects, :latencyTarget, :bandwidthTarget.

**Example Triples**:  
  
:UPF :hasInterface :N6 .

:N6 :connectsTo :DataNetwork .

:N6 :usesProtocol :IP .

:N4 :usesProtocol :PFCP (UDP).

**Latency/Bandwidth Constraints**:

:N3 :latencyTarget "≤1 ms" ; :bandwidthTarget "10–100 Gbps".

**Mapping Relationships**:  
  
:AMF :connectsTo :gNB via :N2 .

:gNB :connectsTo :UPF via :N3 .

These structured properties can be leveraged in your vector DB to support context-aware responses or mapping.

## **12. Topological & Deployment Information**

(Node placement: edge vs core; virtualization options; physical layout)

### Sebastian Böhm & Guido Wirtz – Towards Orchestration of Cloud‑Edge with Kubernetes

<https://www.researchgate.net/publication/356641902_Towards_Orchestration_of_Cloud-Edge_Architectures_with_Kubernetes> → Discusses container/node placement strategies, Kubernetes limitations, real-time metrics in edge deployments[researchgate.net](http://researchgate.net)[+4researchgate.net+4mdpi.com+4](https://www.researchgate.net/publication/356641902_Towards_Orchestration_of_Cloud-Edge_Architectures_with_Kubernetes?utm_source=chatgpt.com).

#### **Key takeaways**

Edge-Oriented Kubernetes Use:

* Kubernetes is widely adopted for edge orchestration, leveraging containers to deploy workloads closer to devices for **<20 ms latency**
* Edge nodes are typically resource-constrained IoT environments requiring location-aware scheduling

Missing Scheduling Capabilities:

* Vanilla Kubernetes lacks **real-time network metric** integration and **topology awareness** needed for optimal edge placement

Evaluation of Edge-Oriented Kubernetes Implementations:

* Several custom K8s architectures have been evaluated: use of **virtual kubelets**, **multi-cluster federation**, and custom schedulers addressing edge needs

Identified Architectural Shortcomings:

* Key challenges remain:
  + Real-time processing of network metrics
  + **Fault tolerance** of edge clusters
  + **Registry placement** for container images on edge nodes

Proposed Architectural Enhancements:

* Integration of **custom metrics servers**, use of **custom schedulers**, topology-aware placement logic, and placement of container registries at the edge are recommended

**Summary factoids**

**Factoid**: “Kubernetes is used for edge orchestration to support sub‑20 ms latency use cases in IoT and real-time applications.”

**Factoid**: “Native Kubernetes lacks network‑aware scheduling and topology‑based placement essential for edge scenarios.”

**Factoid**: “Custom edge Kubernetes deployments use virtual kubelets, multi‑cluster federation, and custom schedulers.”

**Factoid**: “Edge K8s implementations struggle with real-time metric processing, fault-tolerance, and container registry placement.”

**Factoid**: “Edge-focused improvements include installing local metrics servers, topology-aware custom schedulers, and edge-located container registries.”

**Factoid**: “Edge orchestration requires orchestration decisions based on realtime network measurements and locality knowledge.”

**Factoid**: “Fault tolerance at the edge demands resilient control planes and backup mechanisms for remote node failures.”

**Factoid**: “Placing container registries on edge nodes reduces network usage and speeds up workload deployment.”

**Factoid**: “Virtual kubelets enable edge nodes to publish as Kubernetes nodes without running full control plane components.”

**Factoid**: “Topology‑aware scheduling uses knowledge of node network connectivity to prioritize workload placement.”

### Container Placement & Migration: Survey (ResearchGate)

<https://www.researchgate.net/publication/362814993_Container_placement_and_migration_strategies_for_cloud_fog_and_edge_data_centers_A_survey> → Reviews container placement across cloud/fog/edge, node placement trade-offs, fault tolerance, load balancing [arxiv.org+2researchgate.net+2arxiv.org+2](https://www.researchgate.net/publication/362814993_Container_placement_and_migration_strategies_for_cloud_fog_and_edge_data_centers_A_survey?utm_source=chatgpt.com).

#### **Key takeaways**

Scope & Purpose:

* Provides a **comprehensive survey** of container placement algorithms across **cloud, fog, and edge** environments, focusing on the trade-offs among **resource utilization**, **load balancing**, **fault tolerance**, and **energy efficiency**
* Introduces a **taxonomy** of placement strategies and migration techniques specific to containerized microservices in distributed infrastructures

Placement Algorithms:

* Categorizes algorithms into:
  + **Optimization-based** (e.g. LP, bin-packing)
  + **Heuristic/meta-heuristic** (e.g. ant‑colony, ACO; greedy; genetic algorithms)
  + **Machine learning / reinforcement learning**, including Markov decision processes

Migration Techniques:

* Discusses migration strategies: **cold**, **pre-copy**, **post-copy**, and **hybrid**, detailing the trade-offs in **downtime**, **transfer size**, and **complexity**
* **Highlights challenges unique to fog/edge, including state synchronization overhead, container context size reduction, and energy-aware migration controls**

Contextual Edge Constraints:

* Fog/edge nodes face:
* **Resource constraints** (CPU, memory)
* **Geographic and latency demands**
* **Energy limitations**
* Edge placement must trade-off **minimizing latency** against **avoiding resource overloading**

Fault Tolerance & Load Balancing:

* Algorithms address **node failures** and **load surges** via proactive container migrations and replication strategies.
* Meta-heuristics like ACO and grey wolf optimization contribute to multi-objective balancing (load vs cost)

Energy & QoS Trade-offs:

* Energy efficiency is a critical design goal, especially in edge; some strategies use **ant-colony** or **bin-packing** to minimize powered-on resource count
* QoS awareness, such as latency SLAs, are integrated into placement models—particularly via **M/D knapsack** or MILP formulations

**Summary factoids**

**Factoid**: “Container placement spans cloud, fog, and edge domains, optimizing resource use, energy consumption, and fault tolerance.”

**Factoid**: “Placement algorithms include optimization-based (e.g. LP, bin-packing), meta-heuristics (ACO, genetic), and reinforcement learning (MDP).”

**Factoid**: “Migration techniques include cold, pre-copy, post-copy, and hybrid methods, each with different downtime/resource overhead profiles.”

**Factoid**: “Edge/fog placements consider latency, energy, and resource constraints in trade-off-aware optimization.”

**Factoid**: “Pre-copy migration reduces downtime via iterative state replication; post-copy focuses on rapid restart with on-demand state fetch.”

**Factoid**: “Meta-heuristic strategies like ACO and Grey Wolf optimize container placement for load balancing and energy efficiency.”

**Factoid**: “Fault tolerance in edge setups is supported via proactive migrations and container replication to avoid single-node failures.”

**Factoid**: “Energy-aware placement utilizes bin-packing to minimize the number of active fog nodes, saving power.”

**Factoid**: “Placement models often include QoS constraints, formulated using MILP or knapsack to enforce latency and reliability SLAs.”

**Factoid**: “Reinforcement learning enables adaptive container placement sensitive to real-time metrics and evolving constraints.”

### ArXiv – VNF & Container Placement: Recent Advances

<https://arxiv.org/abs/2204.00178> → Taxonomy of placement algorithms for VNFs/containers in edge/5G, impact on latency & resource usage [arxiv.org+11arxiv.org+11arxiv.org+11](https://arxiv.org/abs/2204.00178?utm_source=chatgpt.com).

#### **Key takeaways**

Placement Challenge Scope:

* VNF and container placement is NP-hard, spanning **cloud, fog, and edge environments** over 2016–2021 research
* Placement directly influences **performance**, **cost**, **reliability**, **energy usage**, and **scalability** in 5G and edge contexts

Taxonomy of Placement Methods:

* Methods categorized into:
  + Optimization-based (e.g., linear programming, MIQCP, bin-packing)
  + Heuristic/Meta-heuristic (e.g., greedy, ACO, GWO)
  + Machine learning approaches, especially reinforcement learning and deep RL

Domain-Specific Constraints:

* Edge/fog placement emphasizes latency requirements, unpredictability, resource limits, and intermittent connectivity

Multi-Objective Optimization:

* Solutions often balance energy efficiency, latency, resource utilization, fault resilience, cost, and QoS

Migration & Adaptation Strategies:

* Dynamic placement integrates live adaptations to traffic and resource changes through reactive or predictive decision-making, often using RL

**Summary factoids**

**Factoid**: “VNF and container placement across cloud, fog, and edge are NP-hard and directly impact latency, cost, energy, and scalability.”

**Factoid**: “Placement techniques are classified as optimization-based (e.g., LP, MIQCP), meta-heuristics (ACO, GWO), or ML-based (reinforcement learning, deep RL).”

**Factoid**: “Edge placements prioritize low latency, energy efficiency, resource constraints, and intermittent connectivity.”

**Factoid**: “Multi-objective placement models balance latency, energy, cost, resource use, fault tolerance, and QoS among VNFs/containers.”

**Factoid**: “Reinforcement learning enables dynamic adaptation of placement strategies in response to traffic and resource fluctuations.”

**Factoid**: “Meta-heuristic algorithms like ant-colony and grey wolf are widely used for balancing load, energy, and QoS objectives.”

**Factoid**: “Optimization models like MIQCP or bin‑packing address latency and energy constraints, especially in uRLLC scenarios.”

**Factoid**: “Dynamic placement systems migrate VNFs/containers proactively to maintain SLAs in changing network states.”

**Factoid**: “Container placement extends beyond VNFs to include microservices in MEC and private 5G environments.”

**Factoid**: “Taxonomy connects placement techniques to objectives—single vs multi-objective, static vs dynamic, method class used.”

### ArXiv – Virtualized C‑RAN Orchestration with Docker, Kubernetes, OAI

<https://arxiv.org/abs/2001.08992> → Demonstrates Docker/K8s management of RRH and BBU containers, offers practical placement & scaling insight [arxiv.org+1mdpi.com+1](https://arxiv.org/abs/2001.08992?utm_source=chatgpt.com).

#### **Key takeaways**

Architecture & Containerization:

* Implements a **container-based Cloud‑RAN** architecture, splitting eNodeB into **BBU** and **RRH** components using OpenAirInterface on Docker
* Orchestrates BBU and RRH containers via **Kubernetes**, enabling dynamic scaling based on resource usage

Fronthaul Orchestration Pipeline:

* Uses **Kubernetes StatefulSets** to manage BBU/RRH pod replication with ordered, unique identities
* Employs **Calico CNI** for layer-3 networking between pods and nodes
* Stores runtime configuration and IP discovery details (BBU ↔ RRH) in **etcd**, enabling dynamic connection setup

Dynamic Scaling & Metrics:

* Demonstrates that scaling BBU–RRH pairs clusters linearly increases fronthaul throughput—doubling throughput with two pairs
* Resource usage metrics:
  + **CPU usage >60%** for OAISIM RRH simulation vs. real UE ➝ useful for load-aware scaling
  + Memory and CPU trends reveal scaling patterns vital for placement and autoscaling decisions

**Summary factoids**

**Factoid**: “C-RAN base station functions (BBU/RRH) can be containerized using Docker and orchestrated via Kubernetes.”

**Factoid**: “Kubernetes StatefulSets provide pod identity and ordering guarantees for RAN component scaling.”

**Factoid**: “Calico CNI enables layer-3 networking among RAN containers in orchestration deployments.”

**Factoid**: “etcd acts as a distributed key-value store for dynamic runtime configuration between containerized RRH and BBU modules.”

**Factoid**: “Doubling the number of BBU–RRH pods linearly increases fronthaul data throughput in the containerized RAN testbed.”

**Factoid**: “OAISIM-based RRH emulation consumes over 60% CPU, which is useful for creating autoscaling policies.”

**Factoid**: “Container orchestration enables dynamic resource-driven scaling of RAN baseband functions for performance and efficiency.”

**Factoid**: “Resource metrics (CPU, memory) from Kubernetes can trigger scaling based on observed load patterns.”

### Container-based Virtualization for Real‑Time Industrial Systems (ACM)

<https://dl.acm.org/doi/10.1145/3617591> → Considers bare‑metal, micro‑VM, container trade-offs in latency-sensitive systems; relevant for core/edge NF placement [dl.acm.org+1arxiv.org+1](https://dl.acm.org/doi/10.1145/3617591?utm_source=chatgpt.com).

#### **Key takeaways**

Scope & Motivation:

* The survey reviews how **container-based virtualization** (e.g. Docker, LXC) is used in real-time cyber-physical and industrial control systems, assessing whether it can meet strict timing requirements

Real-Time Metrics Observed:

* Experimental studies report that **task latency ranges** for containers vary widely:
  + Virtualization on Vx ("virtual PLCs"): **37–102 ms**, depending on synchronization states
  + When isolated to kernel-level virtualization: **47–54 ms**, relatively stable .

Container Overheads & Behavior:

* Containers add **non-negligible latency jitter** and are not yet matured enough for hard real-time requirements
* However, soft real-time performance (control loops with 50–100 ms deadlines) is still achievable, depending on orchestration and host kernel tuning

Platform & Orchestration Constraints:

* Most container systems lack built-in **real-time scheduling**, relying instead on RT-kernel or preemption-patch enhancements
* Orchestration layers (e.g., Kubernetes) do **not support real-time awareness**—lack of task priorities, timing-based placement decisions

Future Requirements & Research Gaps:

* Industry must integrate **real-time metrics** via RT-metrics servers and scheduling into container platforms.
* Orchestration mechanisms should incorporate **time-sensitive scheduling**, real-time-aware scaling, and autotuning for latency-sensitive applications

**Summary factoids**

**Factoid**: “Container-based virtualization (Docker/LXC) in real-time industrial systems incurs latency jitter (≈37–102 ms), posing challenges for hard real-time constraints.”

**Factoid**: “Soft real-time deadlines (50–100 ms) are feasible with containers when the host is RT-kernel tuned and orchestration supports deterministic latency.”

**Factoid**: “Lack of built‑in real-time scheduling in container runtimes requires use of RT-patched kernels or RT-specific orchestration frameworks.”

**Factoid**: “Experimental task latency varies by virtualization type: 37–102 ms for virtual PLCs; 47–54 ms for kernel‑level containers.”

**Factoid**: “Current orchestration platforms (e.g., Kubernetes) lack real-time awareness—missing scheduling priorities and time-bound placement logic.”

**Factoid**: “Industrial-grade container platforms need integration of real-time metrics in orchestration controllers to meet strict latency SLAs.”

**Factoid**: “Real-time container maturity is still limited—jitter and unpredictable scheduling up to 100 ms impede hard real-time use.”

**Factoid**: “Future solution directions include real-time scheduling extensions, metrics‑driven orchestration, and explicit task-level prioritization.”

**Factoid**: “Time-sensitive container requirements demand enhancements at both runtime (cgroup, RT kernel) and orchestration (placement, scaling) layers.”

**Factoid**: “Systematic surveys identify container overhead as a key bottleneck for industrial control loops, guiding research into latency-aware virtualization.”

### **Ontology & Knowledge Graph Expansion**

* **Deployment Node Types**:  
  + :EdgeNode, :CoreNode, :CloudNode with properties like :supportsContainer, :containerPlatform.
* **Placement Policies & Constraints**:

Node placement decisions captured as relationships:  
  
:UPF\_Instance1 :deployedOn :EdgeNode .

:EdgeNode :runsPlatform :Kubernetes .

**Virtualization Options**:  
  
:EdgeNode :supports :Docker, :Kubernetes, :BareMetal .

* **Network Layout Structure**:

Define :NodeZone (e.g. city region, PoP):  
  
:CityEdgeZone :contains :EdgeNode1, :EdgeNode2 .

**Latency and Resource Constraints**:  
  
:UPF\_Instance1 :hasLatencyBudget "≤1 ms" .

:KubernetesPlatform :overheadPercentage "≤5 %" .

These structured pieces will help make your RAG system not only context-aware but also actionable by mapping user intent like "deploy UPF near users" to edge container creation.

### Next Steps

Would you like me to:

* Generate **JSON-LD or RDF schema** for these concepts?
* Provide **sample YAML manifests** for Kubernetes-based UPF deployment?
* Create sample **graph query patterns** to extract latency paths or placement info?

## **13. IP/Subnet Planning**

*(Subnet ranges per network zone, NF interface IPs, NAT/firewall rules)*

### Microsoft Azure – Private 5G Core Design Requirements

<https://learn.microsoft.com/en-us/azure/private-5g-core/private-mobile-network-design-requirements> → Describes IP subnet requirements per zone (edge vs core), NAT/firewall for inter-NF communication [reddit.com](https://www.reddit.com/r/networking/comments/u1g2zf/ip_address_planning_philosophy_easy_or_efficient/?utm_source=chatgpt.com)[learn.microsoft.com+1learn.microsoft.com+1](https://learn.microsoft.com/en-us/azure/private-5g-core/private-mobile-network-design-requirements?utm_source=chatgpt.com)

#### **Key takeaways**

IP Subnet Planning per Zone:

* **Separate IP subnets** are required for control-plane (N2) and user-plane (N3) traffic; a unified subnet is permitted only if using the same VLAN
* Edge deployments use dedicated subnets for UE IP allocation pools, both dynamic and static, mapped to N6 interfaces

NAT & Firewall Configuration:

* The **N6 interface** requires firewall and routing policies to enable external access to UEs via specified IPs; Network Address and Port Translation (NAPT) must be disabled if servers initiate connections
* HA deployments involve gateway routers with static routes directing control‑ and user‑plane VLANs to Azure Stack Edge (ASE) devices

High-Availability (HA) Infrastructure:

* **Active-standby ASE pairs** integrate with BFD and VRRP to support sub-2.5 second failover for both control and user planes
* HA IP planning includes virtual IP addresses for routers and gateway redundancy across ASE devices

Zone-Specific Networks:

* Distinct IP address spaces must be defined for:
  + **Access network (N2/N3)**: separate VLANs/subnets per plane
  + **Data network (N6)**: one subnet per connected DN, mapped to ASE ports
  + **Additional services**: management, cluster nodes, virtual IPs, ACS/NFS, etc.

**Summary factoids**

**Factoid**: “Control-plane (N2) and user-plane (N3) traffic require separate IP subnets, unless using the same VLAN.”

**Factoid**: “Edge deployments use dedicated subnets for UE IP pools, allocated via N6 interfaces for dynamic and static addresses.”

**Factoid**: “NAPT must be disabled on N6 when external servers need to initiate connections to UEs.”

**Factoid**: “Firewalls must allow routes and port access between corporate networks and UE subnets connected via N6.”

**Factoid**: “HA setups use active‑standby ASE pairs with VRRP and BFD to maintain control- and user-plane continuity within 2.5s failover.”

**Factoid**: “Gateway routers require single static routes per network, with virtual IPs managed across redundant ASE devices.”

**Factoid**: “Separate IP pools are needed for access, data, management, cluster nodes, ACS/NFS, and virtual IP services in ASE deployments.”

**Factoid**: “Azure reserves five private IPs in each subnet, limiting available resources for NF instances.”

**Factoid**: “UE IP address pool is specified in CIDR block and passed to AP5GC during site deployment via ARM template.”

**Factoid**: “Proper segmentation of N2/N3/N6 subnets supports network isolation, QoS, and firewall rule enforcement.”

### InterLIR – IP Address Management in 5G Private Networks

<https://interlir.com/2024/08/29/ip-address-management-in-5g-private-networks/> → Recommends using IPv6, RFC1918 private ranges, DHCP/NAT segmentation across slices [serverfault.com+2interlir.com+2interlir.com+2](https://interlir.com/2024/08/29/ip-address-management-in-5g-private-networks/?utm_source=chatgpt.com)

#### **Key takeaways**

IPv6 vs. IPv4 and Address Space:

* IPv6 offers **vast address space**, built-in IPsec, and simpler management (SLAAC), while IPv4 is constrained and reliant on NAT
* IPv4 limitations exacerbate IP exhaustion concerns, especially in high-device-density 5G private networks

Best Practices: IPv6 Adoption:

* InterLIR advises migrating to **IPv6** for scalability and performance, with fallback via dual-stack during transition
* IPv6 sidesteps NAT's latency overhead and complexity—critical for URLLC and industrial use cases .

Use of Private IPv4 Ranges & DHCP/NAT Segmentation:

* RFC 1918 private ranges remain valid for network slicing and segmenting devices (e.g., IoT vs. mobile UEs)
* DHCP is recommended for dynamic IP assignment, while static pools may suit anchored devices; NAT segmentation isolates traffic between slices .

IPAM Automation, Monitoring & Security:

* Automated IP Address Management (IPAM) systems enable real-time conflict detection, dynamic provisioning, and utilization tracking
* Monitoring and alerting mechanisms support security by detecting rogue devices and preventing address conflicts .

#### **Summary factoids**

**Factoid**: “IPv6 is preferred in private 5G deployments for its address space, built-in IPsec, and autoconfiguration (SLAAC).”

**Factoid**: “IPv4 address pools are limited and require NAT, which introduces latency and complexity, especially in URLLC contexts.”

**Factoid**: “InterLIR recommends a dual-stack approach during migration from IPv4 to IPv6 in private 5G networks.”

**Factoid**: “RFC 1918 private IPv4 ranges are used to isolate network slices and device classes in private 5G.”

**Factoid**: “DHCP is used for dynamic IP assignment; static pools may serve fixed-function devices.”

**Factoid**: “NAT segmentation across slices enhances isolation and security in multi-slice deployments.”

**Factoid**: “Automated IPAM systems with monitoring capabilities are critical for conflict detection and utilization visibility.”

**Factoid**: “Real-time alerts from IPAM aid in rogue device detection and prevent IP conflicts in private 5G environments.”

**Factoid**: “IPv6 eliminates NAT overhead, improving latency performance in low-latency applications.”

**Factoid**: “Dual-stack deployment supports gradual IPv6 adoption while maintaining IPv4 compatibility with legacy devices.”

### Cisco – Dynamic IP Pool Chunk Allocation for 5G Packet Core

<https://www.cisco.com/dam/en/us/solutions/collateral/executive-perspectives/wp-dynamic-ip-pool-chunk-allocation-5g-packet-core_v2.pdf> → Control-plane assigns IP chunks to UPF and session-level IP per user [cisco.com+1huggingface.co+1](https://www.cisco.com/c/dam/en/us/solutions/collateral/executive-perspectives/wp-dynamic-ip-pool-chunk-allocation-5g-packet-core_v2.pdf?utm_source=chatgpt.com)[zte.com.cn+3builders.intel.com+3huggingface.co+3](https://builders.intel.com/docs/networkbuilders/low-latency-5g-upf-using-priority-based-5g-packet-classification.pdf?utm_source=chatgpt.com)

#### **Key takeaways**

CUPS Architecture & IP Chunking:

* Within 5G CUPS, **the control plane (SMF)** divides IP pools into **smaller “chunks”**, then allocates them dynamically to UPF instances during registration
* The **UPF receives chunks of available addresses** and allocates individual IPs per UE session, reporting usage back to the SMF

IPAM Subsystem Components:

* Cisco defines a **centralized IPAM Server** (manages overall pool and chunking) and distributed **IPAM Cache modules** (deployed per CP cluster) to handle real-time assignments
* This architecture supports **multi-cluster deployments** and ensures global consistent IP resource management

Chunk Allocation Strategies:

* Allocation uses dynamic thresholding: when a UPF consumes >70% of its chunk, the SMF pushes a new chunk; underutilized chunks are withdrawn
* Chunk size must be tuned to balance load distribution (avoid imbalance or exhaustion) and provisioning overhead

Chunk Limits & Scale Implications:

* Maximum chunk sizes (e.g. 65,536) and address pool sizes determine the **maximum number of UPFs per group** (e.g., limit of 16 UPFs with 1M addresses / 65k chunks)
* Static IP pools, IPv4v6 dual-stack requirements, and overlapping pools must be carefully designed to avoid misallocation

UPF Grouping & DNS-Based Selection:

* **UP Group** (tied to APN) defines which UPFs receive chunks. Dynamic IP pool updates are feasible without needing Sx reassociation
* Cisco uses **DNS-based UPF selection** with TAC/RAC for geo-aware loadbalancing and ensures chunk allocation aligns with selected UPF instance

Throttling & Capacity Awareness:

* SMF enforces **IP chunk throttling** based on UPF capacity (max sessions advertised via PFCP), allocating/chunking only when usage falls below thresholds (e.g., 80%)
* This prevents excessive IP pooling and ensures fair distribution among UPFs

Static IP & DHCP Integration:

* Static IP pools are split across UPFs for deterministic address assignment; SMF rejects requests outside static pools
* DHCP-based IPOA is also supported: UPF acts as DHCP client, SMF triggers N4 session with VLAN ID, and VM-based DHCP obtains UE IP which is relayed via PFCP

#### **Summary factoids**

**Factoid**: “In CUPS, SMF dynamically assigns IP chunks to UPFs; UPFs allocate individual IPs per UE and report usage back to SMF.”

**Factoid**: “Cisco’s Cloud‑Native IPAM includes a central IPAM Server and distributed IPAM Caches for multi-cluster consistency.”

**Factoid**: “SMF triggers new IP chunk allocations when UPF usage exceeds 70%, and reclaims underutilized chunks.”

**Factoid**: “Optimal chunk size balances even load distribution against IP exhaustion and provisioning overhead.”

**Factoid**: “Maximum UPFs per chunk-limited pool are determined by pool size divided by chunk size (e.g., 1M/65k → ~16 UPFs).”

**Factoid**: “UP Groups tied to APN define chunk associations; dynamic pool updates work without Sx reassociation.”

**Factoid**: “DNS-based UPF selection uses TAC/RAC to bind UPF and chunk assignment in geo-aware deployments.”

**Factoid**: “SMF enforces chunk throttling based on UPF’s advertised max session capacity to prevent over-allocation.”

**Factoid**: “Static IP pools are split and distributed to multiple UPFs; SMF rejects requests outside defined static blocks.”

**Factoid**: “DHCP-based IP allocation via UPF requires PFCP N4 NF integration with VLAN ID tagging and DHCP DORA exchange.”

### Cisco – IP Addressing Guide (General best practices)

<https://www.cisco.com/c/en/us/td/docs/solutions/strategy/ipv4_addressing_guide.pdf> → Standards for hierarchical subnetting suitable for telecom deployments [cisco.com+1reddit.com+1](https://www.cisco.com/c/dam/global/en_ca/solutions/strategy/docs/sbaBN_IPv4addrG.pdf?utm_source=chatgpt.com)[cisco.com](https://www.cisco.com/c/en/us/td/docs/wireless/ucc/amf/2024-02/config-and-admin/b_ucc-5g-amf-config-and-admin-guide_2024-02/m_amf-overview.html?utm_source=chatgpt.com)

#### **Key takeaways**

Hierarchical IP Addressing Structure:

* Cisco advocates a **hierarchical subnet design** with a modular, three-tier structure: **Access**, **Distribution**, and **Core** layers. This segmentation supports scalability, resiliency, and simplified management
* Subnetting breaks the network into smaller, manageable blocks, each with unique network and host portions, preventing address overlap and supporting efficient routing

Variable Length Subnet Masks (VLSM):

* **VLSM** enables flexible subnet sizing within the same IP block, allowing varying host counts per subnet—optimal for telecom cell sites or NF clusters
* This technique improves address utilization and supports future expansion without restructuring existing allocations

IPv6 Hierarchical Addressing:

* IPv6 uses a **16-bit Subnet ID**, supporting up to **65,535 hierarchically structured subnets** within a /64 global routing prefix—ideal for zonal planning
* Interface IDs follow **modified EUI-64**, ensuring unique host addresses within each subnet

Subnet Planning for Telecom:

* Best practices include **designating separate subnets** for different NF roles or zones (e.g., control-plane, user-plane, management), typically sized for specific host counts (e.g., /28 for NFs, /24 for core-switch uplinks).
* Hierarchical design simplifies routing tables, isolates failures, and localizes changes to subnets rather than the entire network

#### **Summary factoids**

**Factoid**: “Cisco best practice suggests hierarchical subnet design with Access, Distribution, and Core layers to improve scalability and resilience.”

**Factoid**: “Subnetting divides an IP space into network and host portions, ensuring uniqueness and route summarization.”

**Factoid**: “VLSM allows subnets of varying sizes within a block—key for allocating optimal address ranges per NF/service.”

**Factoid**: “IPv6’s 16-bit Subnet ID supports up to 65,535 subnets under a /64 prefix, enabling zonal network segmentation.”

**Factoid**: “Cisco recommends using modified EUI-64 interface IDs for stable, unique host addressing in IPv6 subnets.”

**Factoid**: “Hierarchical design localizes network changes, improves fault isolation, and reduces routing table size.”

**Factoid**: “Typical subnet sizes include /28 for NF instances and /24 for core uplinks, providing headroom for growth.”

**Factoid**: “Hierarchical and CIDR-based subnetting enhances scalability and ease of management in telecom infrastructures.”

**Factoid**: “Separated subnets per network plane (control, user, management) aid in traffic isolation and QoS policy enforcement.”

**Factoid**: “Efficient address planning prevents waste from fixed-length subnetting and traps address exhaustion before it occurs.”

### ZTE – Full‑Scenario UPF Deployment White Paper

<https://www.zte.com.cn/content/dam/zte-site/res-www-zte-com-cn/mediares/zte/files/newsolution/wireless/ccn/hexinwang/whitepaper/ZTE_Full-Scenario_UPF_Deployment_White_Paper.pdf> → Discusses interface IP planning for UPF, user/data/control traffic, and firewall/NAT rules [gsma.com+15zte.com.cn+15builders.intel.com+15](https://www.zte.com.cn/content/dam/zte-site/res-www-zte-com-cn/mediares/zte/files/newsolution/wireless/ccn/hexinwang/whitepaper/ZTE_Full-Scenario_UPF_Deployment_White_Paper.pdf?utm_source=chatgpt.com)

#### **Key takeaways**

Full-Scenario UPF Deployment Architecture:

* ZTE categorizes UPF deployments into **four scenarios**: **Central**, **Regional**, **Edge**, and **Campus**, each tailored to specific SLA needs and performance profiles
  + **Central UPF** (DC): >200 Gbps throughput, >50 ms latency tolerance, supports full functionality and converged multi-generation networks.
  + **Regional UPF** (city DC): 100–200 Gbps, ~30 ms latency.
  + **Edge UPF** (county): <100 Gbps, 10–30 ms latency, cloudified/offloading for local enterprise scenarios.
  + **Campus UPF** (enterprise): ~50 Gbps, <15 ms latency, customized for 5G‑LAN, TSN, URLLC, local security, and simplified O&M

Interface IP and Zone-Specific Subnets:

* Each scenario has tailored **IP addressing and interface plans**, separating **user/data/control traffic**:
  + **N3 (gNB↔UPF)**, **N4 (SMF↔UPF)**, and **N6 (UPF↔DN)** interfaces.
  + Subnet assignments ensure traffic isolation and QoS enforcement per plane/role:
    - Campus/Edge UPFs require VLAN- or subnet‑based segmentation for local data handling and security

Firewall, NAT, and Traffic Control Requirements:

* **NAT/firewall rules** are applied at campus/edge sites to isolate local traffic, enforce access policies and meet enterprise security standards.
* UPF supports **per-DNN or per-slice policy-based steering**, executing firewall/NAT operations locally or upstream as required

Custom Functionality and Local Processing:

* **Campus UPFs** perform on‑demand customization: 5G‑LAN, TSN support, URLLC, data buffering, and local processing to meet industrial-grade security and scenario-specific latency (≤15 ms)

#### **Summary factoids**

**Factoid**: “ZTE classifies UPFs into Central, Regional, Edge, and Campus deployments, each with defined throughput and latency metrics.”

**Factoid**: “Campus UPFs offer ~50 Gbps throughput with sub‑15 ms latency and include TSN, URLLC, and 5G‑LAN enhancements.”

**Factoid**: “Subnet planning for UPFs defines separate IP ranges per interface: N3, N4, N6, ensuring flow isolation and QoS.”

**Factoid**: “VLAN or subnet segmentation at edge/campus UPFs enables localized firewall and NAT policies.”

**Factoid**: “Local firewall/NAT in campus UPFs ensures enterprise-grade security and traffic control at the site level.”

**Factoid**: “UPFs apply DNN or slice‑based policy steering, allowing flexible access control across user data flows.”

**Factoid**: “Separate subnets for user, control, and data traffic per plane support QoS isolation and policy enforcement.”

**Factoid**: “Edge and campus UPFs are positioned at county/data-center proximity to offload traffic and reduce backhaul latency (~10–30 ms).”

**Factoid**: “Campus UPFs integrate simplified O&M and local data storage to satisfy local processing and security compliance.”

**Factoid**: “Central and regional UPFs focus on high throughput and broader connectivity; edge/campus UPFs target performance-sensitive enterprise usage.”

## **14. Performance & QoS Constraints**

*(Latency/jitter targets, throughput, HA sizing, CPU/memory/IOPS)*

### ETSI TS 128 552 Rel 16 – Performance Measurement of Interfaces

<https://www.etsi.org/deliver/etsi_ts/128500_128599/128552> → Defines latency/bandwidth targets per interface (e.g., N3, N4, N6) [etsi.org+1etsi.org+1](https://www.etsi.org/deliver/etsi_ts/128500_128599/128552/16.09.00_60/ts_128552v160900p.pdf?utm_source=chatgpt.com)

#### **Key takeaways**

N3 Interface (gNB ↔ UPF via GTP‑U):

* **Data Volume Metrics**: Octet counters for incoming/outgoing GTP packets, optionally per QoS/5QI/S‑NSSAI (e.g., [GTP.In/OutDataOctetsN3UPF](http://gtp.in/OutDataOctetsN3UPF)).
* **Packet Loss Counters**: Incoming/outgoing packet-loss metrics tracked per N3 and QoS level ([GTP.In/OutDataPktLossN3UPF](http://gtp.in/OutDataPktLossN3UPF))
* **Round-Trip Delay (RTT)**: Average RTT measured per DSCP class (Ultra-low microsecond granularity), plus delay distribution histograms (GTP.RttDelayN3DlPsaUpfMean.DSCP, ...Dist.Bin.DSCP).
* **Out-of-order Packets**: Tracking counts for out-of-sequence GTP packets per QoS (GTP.InDataPktOutOfOrderN3UPF)

N4 Interface (SMF ↔ UPF via PFCP):

* **Session Statistics**: Count of N4 session establishment attempts and failures (SM.N4SessionEstabReq, SM.N4SessionEstabFail).
* **Session Report Metrics**: Number of session reports and acknowledgments (SM.N4SessionReport, SM.N4SessionReportSucc).

N6 Interface (UPF ↔ Data Network):

* **Link Usage Counters**: Incoming and outgoing IP flow volumes (IP.N6IncLinkUsage.N6RP, IP.N6OutLinkUsage.N6RP), based on RFC 5136 definitions.

N9 Interface (UPF ↔ UPF Chaining):

* **RTT Measurements**: Average and distribution for inter-UPF RTT per DSCP class (GTP.RttDelayN9PsaUpfMean.DSCP, ...Dist.Bin.DSCP).
* **GTP Packet Metrics**: Counters for incoming/outgoing GTP packets and bytes, optionally per QoS/S‑NSSAI.

#### **Summary factoids**

**Factoid**: “N3 octet counters (GTP.In/OutDataOctetsN3UPF) measure data volume between gNB and UPF, optionally per QoS or slice.”

**Factoid**: “Packet loss on N3 is captured as GTP.In/OutDataPktLossN3UPF, enabling per-QoS reliability tracking.”

**Factoid**: “Average RTT per DSCP on N3 (GTP.RttDelayN3DlPsaUpfMean.DSCP) provides microsecond-level latency metrics for QoS classes.”

**Factoid**: “Out-of-order GTP packet counts (GTP.InDataPktOutOfOrderN3UPF) help detect sequence and jitter issues on N3.”

**Factoid**: “N4 interface PFCP session metrics (SessionEstab, ReportSucc) reflect SMF‑UPF control-plane signaling health.”

**Factoid**: “N6 link usage counters (IP.N6IncLinkUsage, IP.N6OutLinkUsage) aggregate user-plane bandwidth data toward external networks.”

**Factoid**: “N9 RTT metrics (GTP.RttDelayN9\*) quantify latency across chained UPFs, supporting multi-hop performance assessment.”

**Factoid**: “N9 GTP packet/byte counters support slice-level throughput and control-plane scaling insights.”

**Factoid**: “Measurement split by DSCP/QoS/S‑NSSAI enables slice-specific SLA monitoring and resource orchestration.”

**Factoid**: “These interface-level metrics feed QoS enforcement, auto-scaling, and anomaly detection modules in intent-driven orchestration.”

### Intel/SK Telecom – Low‑Latency UPF via Priority Packet Classification

<https://builders.intel.com/docs/networkbuilders/low-latency-5g-upf-using-priority-based-5g-packet-classification.pdf> → Achieves ~32–45 µs latency and ~12–14 µs jitter for high-priority flows [huggingface.co+5builders.intel.com+5opennetworking.org+5](https://builders.intel.com/docs/networkbuilders/low-latency-5g-upf-using-priority-based-5g-packet-classification.pdf?utm_source=chatgpt.com)

#### **Key takeaways**

Priority Packet Classification for URLLC:

* Uses **hardware-based packet classification and steering** (via Intel® DDP-enabled NICs) along with **software-tier enhancements** to detect and expedite high-priority UPF flows

Deterministic Low-Latency at Scale:

* High-priority packets achieve **~0.07–0.09 ms RTT** for small and large packets, even under **~87% CPU utilization**

Jitter Reduction for Priority Traffic:

* Jitter falls to **~±0.014 ms** for high-priority packets (both 175 B and 550 B), a reduction of **88% vs. normal traffic (±0.1 ms)**

Throughput Neutrality for Lower-Priority Traffic:

* Best-effort traffic maintains ~0.3–0.34 ms latency at high CPU loads; uplifts of ~78% latency and ~88% jitter reductions are achieved for URLLC-class flows

Multi-Traffic Profile Resilience:

* Tested across diverse profiles (2.5%–44% high-priority), high-priority flows consistently show **~32–45 µs latency** and **~12–14 µs jitter**, unaffected by CPU load variability

COTS Hardware Efficacy:

* Achieves deterministic performance using **standard Xeon Gold CPUs** and **Intel Ethernet 800 DDP adapters**, with **no acceleration ASICs**

#### **Summary factoids**

**Factoid**: “Intel + SK Telecom UPF uses hardware NIC classification and software steering to deliver URLLC-grade priority flows.”

**Factoid**: “High-priority UPF traffic achieved consistent RTT of ~0.07–0.09 ms under ~87% CPU load.”

**Factoid**: “Priority jitter was reduced to ~±0.014 ms, an ~88% improvement over best-effort traffic.”

**Factoid**: “Best-effort traffic latency remained at ~0.3 ms, enabling URLLC without sacrificing throughput.”

**Factoid**: “Across varied traffic mixes, high-priority batches consistently saw 32–45 µs latency and 12–14 µs jitter.”

**Factoid**: “Deterministic low-latency performance was achieved on COTS Xeon + Intel Ethernet 800 hardware.”

**Factoid**: “Priority-based packet handling yields ~78% latency and ~88% jitter improvements relative to normal traffic.”

**Factoid**: “Performance remains stable across CPU load variations, ensuring URLLC resilience in production environments.”

**Factoid**: “Hardware-enabled packet steering ensures flow steering to designated cores, optimizing cache and order.”

**Factoid**: “Throughput and latency separation in UPF enable multi-tier traffic handling (eMBB and URLLC) on same infrastructure.”

### IPLOOK – Impact of QoS Parameters on 5G Performance

<https://www.iplook.com/info/the-impact-of-qos-parameters-on-5g-performance-i00471i1.html> → Covers throughput, packet loss, jitter, and latency across service types [etsi.org+4iplook.com+4etsi.org+4](https://www.iplook.com/info/the-impact-of-qos-parameters-on-5g-performance-i00471i1.html?utm_source=chatgpt.com)

#### **Key takeaways**

Core QoS Metrics:

* 5G performance critically depends on managing **throughput**, **latency**, **packet loss**, and **jitter**—which directly impact service quality for voice, autonomous vehicles, and remote healthcare applications
* Meeting strict SLAs for **real-time and mission-critical communications** requires careful tuning of these parameters within the 5GC solution

IPLOOK’s 5GC Feature Set:

* IPLOOK integrates **UPF, AMF, SMF, PCF**, and **AUSF** into a unified, scalable core offering:
  + **Scalability** for high traffic volume
  + **Reliability** for high-availability mission-critical services
  + **Flexibility** supporting smooth migration from 4G/3G to 5G

Future-Proofing and SLA Management:

* Their platform supports multi-generational interoperability while preparing operators for next-gen 5G, optimizing both **performance and cost** under evolving service demands

#### **Summary factoids**

**Factoid**: “Throughput, latency, packet loss, and jitter are the four key QoS metrics defining 5G service quality.”

**Factoid**: “Voice, autonomous vehicles, and remote medical services rely critically on low-latency and minimal packet loss.”

**Factoid**: “UPF, AMF, SMF, PCF, and AUSF form a cohesive 5GC suite enabling scalable, reliable, and flexible network operation.”

**Factoid**: “Scalability supports high traffic load, reliability ensures mission-critical service continuity, and flexibility enables smooth 3G/4G to 5G transition.”

**Factoid**: “Future-proof architecture ensures readiness for emerging 5G services while balancing performance objectives and cost.”

### ITU/ETSI & 5G-Ideal – Network Slicing Based 5G and Future Mobile Networks

<https://arxiv.org/abs/1704.07038> → Includes slice-instance sizing (e.g., 2 AMFs for HA), throughput-per-slice, resource scaling schemes [arxiv.org+1gsma.com+1](https://arxiv.org/abs/1704.07038?utm_source=chatgpt.com)

#### **Key takeaways**

Logical Architecture for Slicing:

* Defines a 5G slicing **logical architecture** that segments physical infrastructure into service-specific logical networks, each with tailored QoS, mobility, and reliability criteria

Mobility Management Across Slices:

* Highlights seamless handover support in slices, particularly under **high mobility (e.g., 500 km/h)**, requiring slice-aware mobility protocols and gateway selection logic

Joint Power & Subchannel Resource Allocation:

* Presents an ILP-based algorithm optimizing **power and subchannel allocation** in spectrum-sharing two-tier networks, balancing co-tier and cross-tier interference while meeting slice demands

Resource Allocation Flexibility & Interference Management:

* Simulation results confirm dynamic resource allocation enables *on-demand flexible resource distribution* among slices under varying traffic and interference conditions

Open Challenges in Slicing:

* Identifies key open issues in slice management: network reconstruction, multi-slice mobility, cross-domain slice coordination, and integration with SDN/NFV paradigms

#### **Summary factoids**

**Factoid**: “A logical slicing architecture partitions 5G infrastructure into QoS-aligned logical networks for tailored service delivery.”

**Factoid**: “Slice-aware mobility mechanisms are required to support seamless handover at high speeds (up to 500 km/h).”

**Factoid**: “Joint optimization of power and subchannel allocation using ILP mitigates co-tier and cross-tier interference in shared spectrum.”

**Factoid**: “Dynamic resource allocation among slices supports flexible, demand-driven QoS fulfillment under interference variability.”

**Factoid**: “Network slicing introduces open challenges: slice orchestration, mobility coordination, SDN/NFV integration, and infrastructure reconfiguration.”

### ArXiv – Survey on Low Latency in RAN & Core/Caching

<https://arxiv.org/abs/1708.02562> → Defines general network latency/jitter budgets and NF resource (CPU/I/O) requirements [arxiv.org+2arxiv.org+2opennetworking.org+2](https://arxiv.org/abs/1708.02562?utm_source=chatgpt.com)[huggingface.co](https://huggingface.co/datasets/greenwich157/telco-5G-core-faults/viewer?utm_source=chatgpt.com)

#### **Key takeaways**

End-to-End Latency Target:

* 5G aims for **≤1 ms E2E latency with ≥99.99% reliability**, crucial for tactile internet, robotics, haptics, and remote surgery
* Requires sweeping architectural optimizations across RAN, core, transport/backhaul, and caching layers

Latency Sources and Constraints:

* **RAN** contributors: transmission processing (ttx), BS processing delay (tbsp), UE processing delay (tmpt), current LTE TTIs (~1 ms) must be shrunk to sub-ms
* **Core/backhaul** delays minimized via SDN/NFV, MEC, and cloud RCA to bypass layers

RAN-layer Enablers:

* Shortened TTIs and flexible frame structures (e.g., sub-ms 0.25 ms TTI) reduce latency but increase control overhead
* Advanced waveforms (e.g., GFDM, SC-FDM), symbol-detection enhancements (MMSE, ZF), and mmWave aggregation are key innovations

Core & Transport Optimizations:

* NFV/SDN splits (e.g., data/control plane separation) and MEC bring functionality closer to the edge
* Techniques like dynamic GTP tunnel termination and ultra-dense WDM backhaul deliver sub-0.1 ms gains

Caching Strategies:

* Edge caching reduces backhaul latency, lowering content delivery delays by storing popular data locally

Resource & NF Requirements:

* To support low latency, NF placement and resource allocation must address CPU, memory, I/O delays. While explicit numbers aren’t specified here, CPU forwarding routines should be offloaded, MEC nodes must offer fast processing, and real-time orchestration is essential.

#### **Summary factoids**

**Factoid**: “5G target is ≤ 1 ms end-to-end latency with 99.99% reliability for use cases like tactile internet.”

**Factoid**: “RAN delays comprise ttx, tbsp, tmpt; LTE’s 1 ms TTI must be reduced to ≤0.25 ms.”

**Factoid**: “Sub‑ms TTI (0.25 ms) lowers latency at the expense of higher control overhead.”

**Factoid**: “Advanced waveforms (GFDM, SC‑FDM), and symbol‑detection (MMSE, ZF) methods reduce RAN processing delays.”

**Factoid**: “SDN/NFV and MEC in core/backhaul reduce per-packet latency by eliminating centralized processing hops.”

**Factoid**: “Dynamic GTP termination and ultra-dense WDM backhaul can contribute < 0.1 ms transport latency improvements.”

**Factoid**: “Edge caching of popular content shortens backhaul dependencies and content retrieval latency.”

**Factoid**: “NF resource requirements for low latency include fast CPU, low-latency I/O, and real-time orchestration responsiveness.”

**Factoid**: “RAN optimization combined with transport and caching enables sub-ms latency but demands trade-off in overhead and complexity.”

**Factoid**: “Reliable ultra-low latency requires harmonised RAN, core, transport, and caching co-design with real-time resource adaptation.”

### How to Integrate into Your Pipeline

#### Ontology & Knowledge Graph Ideas

**IP/Subnet Planning**

:ZoneEdge :usesSubnet "10.1.0.0/24" .

:UPFInstance\_A :hasInterface :N3 ; :ipAddress "10.1.0.5" .

:FirewallN6 :natFrom :UPFInstance\_A ; :toZone :Internet ; :rule "allow tcp/udp outbound".

**Performance & QoS**

:N3 :latencyTarget "≤1 ms"; :bandwidthTarget ">=10 Gbps".

:UPF\_HPFlow :latencyMeasured "32 µs"; :jitterMeasured "12 µs".

:AMF\_Cluster :instanceCount 2 ; :redundancyMode "active‑standby".

:NF\_CPU\_Cores :minCores 4 ; :memory "8 GB"; :iops ">=1000".

#### Vector DB Context Extraction

* Link IP planning to subnet naming, DHCP/NAT rules, zone-to-interface assignment.
* Include performance constraints such as CPU, memory, IOPS for each NF and class of service.
* Create structured JSON-LD fragments combining zone, interface, NF resource limits, and SLA targets.

### Next Steps

Would you like me to:

* **Generate full JSON-LD** schemas incorporating IP planning rules and NF sizing?
* **Produce example YAML templates** for firewall/NAT and NF resource requests?
* **Craft SPARQL queries or graph patterns** to extract SLA compliance or subnet overlaps?

## **15. Intent → Configuration Mapping Rules**

*(Mapping human-readable intent to technical deployment decisions)*

### Ericsson White Paper – Intent-driven → Autonomous Networks

Defines intent patterns like "7 Mbps uplink / ≤50 ms latency" for AR/VR and shows how it's decomposed into RAN/core and slice configurations [mef.net+5rfc-editor.org+5oge.gov+5](https://www.rfc-editor.org/rfc/rfc6312.html?utm_source=chatgpt.com)[researchspace.csir.co.za+8ericsson.com+8sciencedirect.com+8](https://www.ericsson.com/en/reports-and-papers/white-papers/intent-driven-leads-to-autonomous-networks?utm_source=chatgpt.com)<https://www.ericsson.com/en/reports-and-papers/white-papers/intent-driven-leads-to-autonomous-networks>

#### **Key takeaways**

Intent Definition & Role:

* **Intent** represents declarative business-level objectives (e.g., "7 Mbps uplink / ≤ 50 ms latency" for AR/VR) without prescribing how to achieve them .
* Intents are expressed across layers—from business systems to RAN NFs—using identifiers like **5QI** and **S‑NSSAI**, enabling context-aware translation and enforcement

Intent Management Function (IMF):

* The **IMF** acts as an intent handler/owner within autonomous domains, responsible for checking feasibility, decomposing intent, orchestrating resources, and reporting compliance via defined APIs (e.g., TMF921)
* Autonomous network architecture is layered; each layer comprises domains that manage intents and resolve conflicts without human intervention

Intent Lifecycles & Intent APIs:

* Lifecycle stages include: **onboarding**, **handling**, **delivery/execution**, **assurance/monitoring**, and **reporting** on compliance
* APIs, such as **TMF921** and 3GPP intent interfaces, support feasibility checks, negotiation, compliance reporting, and intent modification throughout intent lifecycles

Utility Functions & Service-to-Resource Decomposition:

* Intents carry **utility functions** (e.g., optimize latency vs. energy) to guide autonomous decision-making
* Intent handling involves decomposition: e.g., a 100 ms E2E latency intent may be translated to 60 ms in RAN and residual resources allocated via slice/core configuration

Closed‑Loop Automation & Conflict Resolution:

* Intent-based systems implement **cognitive assurance loops**—monitor, reason, and act cycles—across layers, enabling detection and resolution of operational conflicts (e.g., energy vs. performance)
* Multi‑domain coordination enables autonomous conflict resolution, reducing human dependency and improving operational agility

Evolutionary Adoption Roadmap:

* Deployment progresses in stages—from SLO-based orchestration (Stage 0), to intent-based assurance (Stage 1), to fully autonomous intent-driven operation—integrating with service/product ordering workflows

#### **Summary factoids**

**Factoid**: “An intent like ‘7 Mbps uplink / ≤ 50 ms latency’ can specify AR/VR service requirements without detailing implementation.”

**Factoid**: “Impacted layers parse intents using RAN identifiers like 5QI and S‑NSSAI to scope service intents.”

**Factoid**: “The Intent Management Function (IMF) checks feasibility, decomposes intent, and orchestrates domain resources.”

**Factoid**: “IMF uses TMF921 and 3GPP APIs for intent lifecycle management, negotiation, and compliance reporting.”

**Factoid**: “Utility functions embedded in intents guide decision-making trade-offs, e.g., latency vs. energy.”

**Factoid**: “E2E intent latency targets (e.g., 100 ms) are decomposed to domain-specific contributions (e.g., 60 ms in RAN).”

**Factoid**: “Closed-loop cognitive operations continuously monitor intent compliance, evaluate metrics, and trigger corrective actions.”

**Factoid**: “Multi-layer autonomous domains resolve policy conflicts (e.g. energy-saving vs. performance) through intent exchange.”

**Factoid**: “Intent adoption is phased—from static SLOs to dynamic assurance, to full autonomy integrated into business-service workflows.”

**Factoid**: “Intent-driven autonomy is expected to materialize in commercial networks between 2025 and 2027.”

### MEF – Automation of LSO APIs using IBN

Provides classification of intent (e.g., Skype for Business → mission-critical SLA) and translation into lexicon and policy enforcement rules [ericsson.com](https://www.ericsson.com/en/reports-and-papers/white-papers/intent-driven-leads-to-autonomous-networks?utm_source=chatgpt.com)[6g-intense.eu+6mef.net+6arxiv.org+6](https://www.mef.net/wp-content/uploads/2019/11/MEF-Presentation-WS-Vid-Automation-of-LSO-APIs-Using-Intent-Based-Networking.pdf?utm_source=chatgpt.com)<https://www.mef.net/wp-content/uploads/2019/11/MEF-Presentation-WS-Vid-Automation-of-LSO-APIs-Using-Intent-Based-Networking.pdf>

#### **Key takeaways**

Natural-Language Intent Expression:

* MEF defines intent using **controlled natural-language DSLs** (e.g., Allegro, Cantata), enabling business users to express service goals (e.g., “Skype for Business → mission-critical SLA”)
* The system harmonizes varied stakeholder expressions into coherent intent via standardized **MEF models**, ensuring consistency across constituencies

​Intent Processing & Lexicon Mapping:

* Intent classification involves mapping natural-language intents into **formal lexicons and policy rules**, then translating them into **LSO API calls** such as Legato, Presto, and Adagio for actual enforcement
* Stakeholders’ intent abstractions span four layers of the policy continuum: Business, System, Admin, and Device, ensuring multi-tier alignment

Continuous Enforcement and Validation:

* Declared intents include **performance and security objectives** that are **continuously validated and enforced** until explicitly removed

AI/ML-Driven Intent Handling:

* The ecosystem leverages **AI/ML engines** to interpret intents, adjust policies, and manage network resources dynamically, enabling automated orchestration wireline networks

#### **Summary factoids**

**Factoid**: “MEF uses controlled natural-language DSLs (Allegro, Cantata) to express business-level intents like ‘Skype for Business → mission-critical SLA.’”

**Factoid**: “Intent expressions from diverse stakeholders are harmonized using MEF models before mapping into enforceable policy rules.”

**Factoid**: “Intent-to-policy mapping translates DSL intent into LSO API calls (Legato, Presto, Adagio) for network enforcement.”

**Factoid**: “MEF enforces declared intents—including performance/security objectives—continuously until manually removed.”

**Factoid**: “The intent processing pipeline roles span Business, System, Admin, and Device layers to ensure end-to-end policy coherence.”

**Factoid**: “AI/ML modules assist in interpreting intent, synthesizing policies, and orchestrating LSO-controlled network resources.”

**Factoid**: “LSO APIs serve as enforcement endpoints for intent-derived policy rules within automated service lifecycles.”

**Factoid**: “Intent-based automation addresses scaling limitations and policy complexity by abstracting intent from implementation details.”

**Factoid**: “Continuous intent validation ensures that performance/security objectives remain enforced over time.”

**Factoid**: “MEF’s IBN framework forms a foundation for autonomous networking by linking declarative intent to operational APIs.”

### arXiv – Intent-Based Meta‑Scheduling in Programmable Networks

Defines architectural mapping between high-level intents and resource allocation (e.g., across edge/cloud) [ericsson.com+6arxiv.org+6mef.net+6](https://arxiv.org/html/2412.04232v1?utm_source=chatgpt.com)[nybsys.com](https://nybsys.com/intent-based-networking-for-telecom/?utm_source=chatgpt.com)<https://arxiv.org/html/2412.04232v1>

#### **Key takeaways**

Need for Meta-Scheduling in Large-Scale Programmable Networks:

* Programmable 5G/B5G networks require rapid, automated resource scheduling to fulfill high-level intents—especially across geographically distributed schedulers handling sub-millisecond latency requirements
* The complexity of coordinating multiple domain-specific schedulers (RAN, edge, core) motivates the design of a **meta-scheduler** to ensure global intent fulfillment

Architecture: Meta-Scheduler & Local Schedulers:

* The proposed **meta-scheduler** acts at a higher level, coordinating independent local schedulers via intent directives, while **local scheduling agents** autonomously manage resource allocations in their domains (e.g., RAN, MEC, transport)

Active Inference with Causal Models:

* They propose leveraging **active inference** (a causal inference framework) to model and predict each scheduler's behavior, enabling the meta-scheduler to make resource allocation decisions that align with intents

Hierarchical & Federated Learning Approaches:

* Meta-scheduling supports **hierarchical learning**, where local schedulers refine resources within domains, and **federated coordination** ensures global optimization while respecting domain autonomy

Sub-millisecond Scheduling Requirements:

* For 6G and advanced 5G, resource scheduling—including scheduling selection, chaining, and deployment—must occur at **sub-ms timescales**, necessitating fully automated and high-speed orchestration

Open Research Agenda:

* The paper outlines research needs:
  + Formalizing meta-scheduler models
  + Designing coordination protocols between meta- and local schedulers
  + Applying active inference/inference theory
  + Real-time experimentation in programmable 5G testbeds

#### **Summary factoids**

**Factoid**: “Large-scale programmable 5G/B5G networks require meta-schedulers to coordinate domain-specific schedulers and fulfill user intents.”

**Factoid**: “Meta-schedulers issue high-level intent directives, while local schedulers handle domain-specific resource allocations autonomously.”

**Factoid**: “Active inference—using causal models—is proposed to predict local scheduler behavior for better meta-scheduling decision-making.”

**Factoid**: “Hierarchical and federated learning allow shared intent-driven optimization while preserving domain autonomy.”

**Factoid**: “Sub-millisecond scheduler coordination is essential for future 5G/6G latency targets.”

**Factoid**: “Meta-scheduler architecture integrates multiple local schedulers via intent-based control loops.”

**Factoid**: “Open research areas include formal meta-scheduler modeling, coordination protocols, active inference application, and real-world validation.”

**Factoid**: “Intent representation at the meta-scheduler level must be both human-readable and machine-operational across network domains.”

**Factoid**: “Programmable network orchestration complexity increases with scale and necessitates intent + meta-scheduling layers.”

**Factoid**: “Federated meta-scheduling enables scalability and conflict resolution among hierarchical domain schedulers.”

### SPIRIT Project – Enabling User Intent‑based Network Path Adaptation

Shows how SDN control uses stored intent → mapping rules to configure path routing and resource allocations [arxiv.org+3arxiv.org+3arxiv.org+3](https://arxiv.org/html/2412.04232v1?utm_source=chatgpt.com)[spirit-project.eu](https://www.spirit-project.eu/wp-content/uploads/sites/85/2024/07/Peng-IFIP.pdf?utm_source=chatgpt.com)<https://www.spirit-project.eu/wp-content/uploads/2024/07/Peng-IFIP.pdf>

#### **Key takeaways**

User-Intent Captured & Registered:

* Supports **offline registration**, followed by **online capture** of user intents (e.g., “move left/right/run”) in immersive 3D volumetric streaming
* Intents are mapped to **QoE requirements** that trigger network-level path adaptations

MAB-Based Path Adaptation:

* Employs a **Multi‑Armed Bandit (MAB)** algorithm to select optimal network paths in real time, guided by **probed application delay** and **network congestion**
* Adaptation balances between **exploitation** (best-known path) and **exploration** (testing alternate routes).

SDN-Controlled Network Reconfiguration:

* Path adaptation is realized via **SDN mechanisms**—installing/updating flow rules across programmable switches dynamically
* This tight coupling aligns user-level intent with packet-level path steering and QoS enforcement.

QoE Assurance Under Variability:

* The framework adapts paths to maintain user-perceived QoE, even as **user intent or network conditions shift**
* Validated in real-world tests with volumetric video workloads, showing consistent QoE under dynamic conditions.

Intents → Path / Resource Mapping Rules:

* Defines an **intent-to-policy translation** layer that binds user intent types (e.g., fast movement) to network actions (e.g., low-latency path selection)
* Policies include path selection thresholds, probing intervals, and flow-rule priorities.

#### **Summary factoids**

**Factoid**: “User intents (e.g., motion commands in volumetric streaming) are captured online and linked to QoE goals in the SPIRIT framework.”

**Factoid**: “A Multi‑Armed Bandit path-selection algorithm dynamically chooses SDN paths using application delay and congestion feedback.”

**Factoid**: “The system issues SDN flow-rule updates to steer traffic along the best path matching user intent and current metrics.”

**Factoid**: “Intent mapping rules define how each recognized user intent maps to network adaptation policies (delay thresholds, path priority).”

**Factoid**: “QoE is maintained through network reconfiguration even as user motion or network load changes.”

**Factoid**: “SPIRIT integrates offline intent registration with online capture and dynamic rule enforcement.”

**Factoid**: “SDN-controlled path adaptation acts as the execution layer for intent translation in real-time streaming scenarios.”

**Factoid**: “MAB-based adaptation balances exploitation vs exploration to handle environmental variability.”

**Factoid**: “Intent-driven adaptation closes the loop: intent capture → path selection → flow steering → QoE validation.”

**Factoid**: “Volumetric streaming under dynamic intent benefits from adaptive pathing to satisfy stringent latency/QoE constraints.”

### arXiv – Intent-Based Management of Next‑Gen Networks (LLM‑centric)

Demonstrates how intents with constraints (e.g., latency, bandwidth) are programmatically transformed into configuration commands [spirit-project.eu](https://www.spirit-project.eu/wp-content/uploads/sites/85/2024/07/Peng-IFIP.pdf?utm_source=chatgpt.com)[mef.net+86g-intense.eu+8arxiv.org+8](https://6g-intense.eu/wp-content/uploads/2025/01/Intent-Based-Management-of-Next-Generation-Networks-an-LLM-centric-Approach.pdf?utm_source=chatgpt.com)[ericsson.com+3nybsys.com+3network-insight.net+3](https://nybsys.com/intent-based-networking-for-telecom/?utm_source=chatgpt.com)<https://6g-intense.eu/wp-content/uploads/2025/01/Intent-Based-Management-of-Next-Generation-Networks-an-LLM-centric-Approach.pdf>

#### **Key takeaways**

LLM‑Centric Intent Life-Cycle (LC):

* Introduces a **novel architecture** where large language models (LLMs) manage the full intent life-cycle: **decomposition**, **translation**, **negotiation**, **activation**, and **assurance**

Natural-Language Input and Expressivity:

* Users specify intents in **plain language** (e.g., "deploy 3 XR apps + 5 GB/s links + ≤5 ms latency"), eliminating the need to handcraft JSON/YAML NBI structures

System Implementation at EURECOM 5G Testbed:

* A prototype was built at **EURECOM**, using **Code Llama** on NVIDIA A100 GPU, to decompose and translate NLP intents into Infrastructure-Level Intents (ILIs), which existing NMS modules then enforce

Few-shot + Human Feedback (HF) Loop:

* The architecture uses **few-shot prompting** and **human-in-the-loop feedback** to improve translation accuracy and intent fidelity over time

End-to-End Deployment Realization:

* Demonstrated capability to **decompose natural intents**, translate them into domain-specific ILIs (e.g., for cloud/edge/RAN), activate services via network controllers, and assure intent compliance

LLM‑centric architecture identified open challenges:

* Highlights key research challenges: **multi-domain orchestration**, **security/privacy**, **NL ambiguity**, **scalability**, **LLM interpretability**, **cost-effectiveness**, and **real-time responsiveness**

#### **Summary factoids**

**Factoid**: “LLM-centric intent life-cycle architecture spans decomposition, translation, negotiation, activation, and assurance.”

**Factoid**: “Plain-language intents obviate the need for manual JSON/YAML structuring by experts.”

**Factoid**: “EURECOM implementation uses Code Llama on A100 GPU to map NL intents into Infrastructure-Level Intents.”

**Factoid**: “Few-shot learning with human-in-loop feedback supports continuous improvement in intent translation.”

**Factoid**: “System demonstrates lifecycle support: NL intent → ILI → activation via NMS → compliance assurance.”

**Factoid**: “Key challenges include multi-domain orchestration, LLM interpretability, scalability, and real-time performance.”

**Factoid**: “NL intent examples include deploying XR apps requiring vCPUs, memory, throughput, and latency constraints.”

**Factoid**: “LLM handles cross-domain decomposition (Cloud, Edge, RAN) in a single unified lifecycle pipeline.”

**Factoid**: “Human feedback loop enables refinement of intent translation accuracy over system lifetime.”

**Factoid**: “Effectiveness demonstrated in real-world deployment within the EURECOM 5G facility.”

## **16. Domain Constraints and Policies**

*(Regulatory, operator, and security-driven policies)*

### IETF draft – IPv6-Only 5G deployments considerations

Details policy-level constraints (e.g., 464XLAT support, no public IPv4 on user plane) [6g-intense.eu](https://6g-intense.eu/wp-content/uploads/2025/01/Intent-Based-Management-of-Next-Generation-Networks-an-LLM-centric-Approach.pdf?utm_source=chatgpt.com)[rfc-editor.org+4datatracker.ietf.org+4ietf.org+4](https://datatracker.ietf.org/doc/draft-ma-v6ops-5g-ipv6only/?utm_source=chatgpt.com)<https://datatracker.ietf.org/doc/draft-ma-v6ops-5g-ipv6only-00/>

#### **Key takeaways**

Pv6‑Only on User Plane with Transition Mechanism:

* The draft outlines **gradual deployment** of **IPv6-only user-plane** leveraging **464XLAT** (CLAT + PLAT) in 5G networks; the network provides IPv6 protocol stack; IPv4-only UEs receive IPv4 via CLAT/PLAT translation

Non-Roaming vs Roaming Scenarios:

* In non-roaming scenarios, IPv6 PDUs are anchored in the home 5GC.
* In roaming, PDU anchoring location (home/visited network) determines where translation (464XLAT) occurs

Technology Applicability & Limitations:

* 464XLAT is preferred in mobile networks; alternatives like DS-Lite (for wireline) exist but IPv6-only ecosystem is **incomplete**, especially in backbone segments

Policy-Level Constraints (RFC 2119):

* Requirements use RFC2119 keywords (“MUST”, “SHOULD”) for key guidelines, ensuring compliance with IETF-defined practice

#### **Summary factoids**

**Factoid**: “5G networks can support IPv6-only user-plane using 464XLAT with CLAT at UE and PLAT in the network.”

**Factoid**: “IPv4-only UEs receive their address via CLAT while IPv6-only UEs are fully native on the user-plane.”

**Factoid**: “In roaming scenarios, the location of PDU anchor (home vs visited 5GC) dictates CLAT/PLAT placement.”

**Factoid**: “464XLAT is deployed in mobile networks; DS-Lite is used in wireline, but end-to-end IPv6 backbone deployment is not yet complete.”

**Factoid**: “The draft uses ‘MUST’, ‘SHOULD’ per RFC2119 to define policy-level IPv6-only deployment rules.”

**Factoid**: “Gradual IPv6-only migration must accommodate mixed UE support: IPv4-only, IPv6-only, and dual-stack.”

**Factoid**: “Policy configurations include static IP assignment and network translation behavior per UE capability.”

**Factoid**: “Backbone IPv6-only deployment lags behind access deployments; draft calls for multi-domain planning.”

**Factoid**: “Network translation points should be stateless and per RFC6877-compliant (464XLAT).”

**Factoid**: “IPv6-only user-plane offers scalability and IPv4 address exhaustion mitigation in 5G networks.”

### Cisco IPv6 Services Whitepaper – Best Practices

Covers IPv6 deployment rules, NAT/firewall constraints, auto-configuration policies [datatracker.ietf.org+1ietf.org+1](https://datatracker.ietf.org/doc/draft-ma-v6ops-5g-ipv6only/?utm_source=chatgpt.com)[cisco.com+1etsi.org+1](https://www.cisco.com/en/US/services/ps6887/ps10716/docs/how_to_get_started_ipv6_services_whitepaper.pdf?utm_source=chatgpt.com)<https://www.cisco.com/en/US/services/ps6887/ps10716/docs/how_to_get_started_ipv6_services_whitepaper.pdf>

#### **Key takeaways**

Phased Dual‑Stack and IPv6‑Only Deployment:

* Cisco advises a **phased approach**: start with dual-stack support, pilot IPv6 in isolated environments, and gradually transition to IPv6-only for capable applications
* IPv6-only sites can reduce IPv4 usage by ~99% while maintaining compatibility through selective dual-stack servers

Auto‑Configuration & SLAAC/DHCPv6:

* **Stateless Address Auto-Configuration (SLAAC)** is recommended for host devices.
* **Stateful DHCPv6** is used for servers or hosts needing controlled address assignment. Dual approach supports segment-level deployment flexibility

NAT and Firewall Considerations:

* In IPv6, **NAT is discouraged**; IPv6 firewalls should use **stateful, implicit-deny policies** for security instead
* Where IPv4-to-IPv6 translation is necessary, **static NAT-PT** is deprecated in favor of **NAT64/DNS64** solutions

DNS64/NAT64 for IPv6‑Only Support:

* Enable **DNS64 + NAT64** at network edges to support IPv6 hosts accessing IPv4-only services. Enables translation of DNS responses and packet headers seamlessly
* SLAAC-only IPv6 mode can co-exist with a fallback to IPv4 via DNS64/NAT64 for legacy compatibility

Security Best Practices:

* **Stateful firewall rules** are key; NAT should not be considered a security mechanism .
* **IPSec** should be used where confidentiality and authentication are required
* Firewalls must enforce **multicast scope boundaries** and monitor transition tunnels (e.g., Teredo, 6to4)

Application & DNS Compatibility:

* IPv6 deployments should **test end-to-end application compatibility**, including support for IPv6 literals, DNS64, and dual-stack fallback
* Ensure that middleboxes, logging systems, and security appliances fully support IPv6 protocols

#### **Summary factoids**

**Factoid**: “A phased deployment begins with dual-stack, pilots IPv6, and transitions to IPv6-only where feasible.”

**Factoid**: “IPv6-only branch office deployment can reduce IPv4 addresses by ~99% using selective dual-stack endpoints.”

**Factoid**: “SLAAC is recommended for host auto-configuration; DHCPv6 is used for server/static assignment.”

**Factoid**: “IPv6 firewalls should use stateful implicit-deny policies rather than NAT for security.”

**Factoid**: “Static NAT-PT is deprecated; NAT64/DNS64 is the preferred IPv4-compatibility mechanism.”

**Factoid**: “DNS64 synthesizes AAAA records when only A records exist, enabling IPv6-only host access via NAT64.”

**Factoid**: “SLAAC-only clients should use DNS64/NAT64 to interact with IPv4-only services.”

**Factoid**: “IPSec is recommended to secure IPv6 communication channels requiring confidentiality.”

**Factoid**: “Firewalls must limit IPv6 multicast scope and monitor IPv6 transition tunnels (e.g., Teredo).”

**Factoid**: “IPv6 compatibility requires testing for IPv6 literals, DNS64 behavior, and supporting dual-stack protocols.”

### LACNIC – IPv6 as a Strategic Decision

Highlights traceability and policy constraints tied to regulatory subscriber mapping [cisco.com](https://www.cisco.com/en/US/services/ps6887/ps10716/docs/how_to_get_started_ipv6_services_whitepaper.pdf?utm_source=chatgpt.com)[lacnic.net](https://www.lacnic.net/innovaportal/file/4943/1/lacnic-ipv6-strategicdecision.pdf?utm_source=chatgpt.com)<https://www.lacnic.net/innovaportal/file/4943/1/lacnic-ipv6-strategicdecision.pdf>

#### **Key takeaways**

Traceability & Subscriber Mapping:

* **IPv6 enables operator-level mapping** of each IP address to an individual subscriber—something not feasible at scale with IPv4 shared addressing
* This mapping enhances **accountability, security**, and **trust**, while potentially reducing demands for mass surveillance by offering transparency

IPv4 Exhaustion & IPv6 Necessity:

* In Latin America and Caribbean, ~400 million internet users are served by only ~190 million IPv4 addresses—resulting in ~2 users per IP and a shortfall for ~300 million unserved users
* IPv6 adoption is essential to connect the unserved population and support exponential device growth in IoT, Industry 4.0, and smart-city solutions

Regulatory & Policy Considerations:

* **Governments and regulators** consider IPv6 deployment strategically critical, not only to avoid address exhaustion but also to improve subscriber-level traceability and compliance

#### **Summary factoids**

**Factoid**: “IPv6 allows mapping of each IP address to a single subscriber, improving accountability and reducing mass surveillance needs.”

**Factoid**: “IPv4 shortage in LAC region (~400M users, ~190M addresses) results in ~2 users sharing each IPv4—IPv6 is required to serve the unconnected ~300M people.”

**Factoid**: “IPv6 deployment supports exponential growth of connected devices, including IoT, smart cities, and Industry 4.0.”

**Factoid**: “Regulators view IPv6 as strategic infrastructure, supporting traceability, security, and resource sustainability.”

**Factoid**: “IPv6 enables transparent subscriber tracking, replacing opaque NAT-based sharing with explicit assignment.”

### NCCoE – Secure IPv6-Only Implementation

Describes security and network policy enforcement (micro-segmentation, zero-trust) in dual-stack/future policy frameworks [lacnic.net](https://www.lacnic.net/innovaportal/file/4943/1/lacnic-ipv6-strategicdecision.pdf?utm_source=chatgpt.com)[arxiv.org+11nccoe.nist.gov+11datatracker.ietf.org+11](https://www.nccoe.nist.gov/sites/default/files/2021-12/ipv6-project-description-draft.pdf?utm_source=chatgpt.com)<https://www.nccoe.nist.gov/sites/default/files/2021-12/ipv6-project-description-draft.pdf>

#### **Key takeaways**

Project Scope & Objectives:

* The NCCoE project aims to **demonstrate secure migration** from IPv4 to **IPv6-only networks** while preserving interoperability via dual-stack transition mechanisms
* It provides practical guidance on implementing **micro-segmentation**, **software-defined perimeters**, and **zero-trust security** within IPv6 environments
* Scenarios include IPv6-only clients, services, and infrastructure, targeting secure deployment across enterprise, public-facing, and hybrid environments

Micro-Segmentation & Zero-Trust Networks:

* Uses **software-defined perimeters** and **micro-segmentation** to enforce **least-privilege access control** and restrict lateral movement
* Emphasizes integrating **zero-trust principles**, including identity-based access, segment-specific enforcement, and continuous risk analysis

Policy Enforcement & Security Controls:

* Includes control over **multicast scope**, **DHCPv6 shielding**, **RADIUS/AAA placement**, **firewall rules**, and **transition mechanism monitoring** (e.g., mix of IPv4/IPv6 tunnels)
* Policies are applied to each scenario element—clients, switches, servers—ensuring defense mechanisms remain robust during transitions

Dual-Stack Coexistence & Phased Deployment:

* Migration stages involve dual-stack coexistence before removal of IPv4, with NIC, DHCP, routing, security, and visibility systems tested at each phase
* Use cases span from endpoint-only clients to full IPv6-only enterprise infrastructure and public services

Standards & Best-Current Practices:

* Draws upon key RFCs like 7610 (DHCPv6 Shield), 7404 (Link-local only), 7381 (Enterprise IPv6), and NIST SP 800-207 (Zero Trust), ensuring policy compliance
* Ensures enterprise-grade support for security monitoring, logging, AAA, MDM, intrusion prevention, and threat intelligence in IPv6 contexts

#### **Summary factoids**

**Factoid**: “NCCoE’s project demonstrates secure IPv6-only deployment using dual-stack transition in enterprise environments.”

**Factoid**: “Micro-segmentation and software-defined perimeters implement zero-trust control by limiting lateral traffic flow.”

**Factoid**: “Zero-trust architecture on IPv6 emphasizes identity-based access, continuous monitoring, and strict firewall enforcement.”

**Factoid**: “IPv6 migration is staged—starting with dual-stack and ending with IPv6-only clients or services.”

**Factoid**: “Policies include DHCPv6 shielding, multicast scope enforcement, RADIUS/AAA, and transition protocol monitoring.”

**Factoid**: “Use cases range from management of IPv6-only clients to fully IPv6-only enterprise infrastructure.”

**Factoid**: “The implementation integrates RFC 7610, 7404, 7381, and NIST SP 800-207 for standards-based IPv6 policy enforcement.”

**Factoid**: “Enterprise components like firewalls, MDM, SIEM, and IPS support dual-stack to IPv6-only migration securely.”

**Factoid**: “The project includes representative lab environments showing the secure deployment of IPv6 across enterprise scenarios.”

**Factoid**: “Zero-trust enforcement in IPv6-only networks hinges on micro-segmentation and strict identity/policy frameworks.”

### Cisco InterLIR – IP Address Management in 5G Private Networks

Covers operator constraints such as no public IPs on control plane, IPv6-only operator policy [nccoe.nist.gov](https://www.nccoe.nist.gov/sites/default/files/2021-12/ipv6-project-description-draft.pdf?utm_source=chatgpt.com)[open5gs.org](https://open5gs.org/open5gs/docs/guide/01-quickstart/?utm_source=chatgpt.com)<https://interlir.com/2024/08/29/ip-address-management-in-5g-private-networks/>

#### **Key takeaways**

IPAM Scalability & Automation:

* IPAM in 5G private networks must support **dynamic, large-scale device pools**, including IoT and UE, via **automated pool provisioning and conflict detection**
* Real-time monitoring and alerting are essential for detecting assignment conflicts or rogue usage in high-density environments .

IPv6 Adoption & Control-Plane Policies:

* **IPv6 is strongly recommended**, especially in control-plane networks where **public IPs must not be assigned**. IPv6-only enhances isolation and policy compliance
* **IPv4 use must be limited** to external-facing slices or appliances, with strict NAT applied at the UPF and **no leakage on control-plane interfaces**

IP Segmentation Across Slices & Zones:

* IPv6/IPv4 ranges are **segmented per slice or device class** (e.g., URLLC vs mMTC), improving **traffic isolation, QoS and network policy enforcement** .
* Subnets are scoped by zone (AMF/SMF vs UPF), enabling tailored firewall/NAT rules and preventing unauthorized cross-zone traffic

IPAM–Orchestrator Integration:

* IPAM must integrate with network orchestration platforms (SMF/PCF/NF controllers), enabling **automated subnets/IPs at NF deployment time**
* The IPAM system should support **IPv6-only, IPv4 dual-stack, and IPv4 translation modes** to accommodate varying slice needs

#### **Summary factoids**

**Factoid**: “5G private network IPAM systems require dynamic automation and conflict resolution to support millions of connected UE and IoT devices.”

**Factoid**: “Operators enforce IPv6-only control-plane policies—no public IPv4 allowed on CP links.”

**Factoid**: “IPv4 addressing is restricted to external-facing slices or middleboxes, with strict NAT at the UPF.”

**Factoid**: “IPAM subnets are isolated per slice (e.g., URLLC, mMTC) and NF zone (e.g., AMF/SMF vs UPF) to enforce QoS and isolation.”

**Factoid**: “Real-time IPAM monitoring should alert network managers to conflicts or rogue addresses in dense deployments.”

**Factoid**: “IPAM–orchestrator integration enables automated allocation of subnets/IPs during on-the-fly NF instantiation.”

**Factoid**: “IPAM must support IPv6-only, dual-stack, and IPv4-translation modes for flexible slice support.”

**Factoid**: “IPv6 control-plane networks simplify policy and isolation strategies, improving compliance with operator frameworks.”

**Factoid**: “Operator policies mandate no public IPv4 on CP while allowing IPv4 for UPF-based NAT translation.”

**Factoid**: “Slice-specific addressing enhances enforcement of slice-level policies and traffic segregation.”

## **17. Deployment Environments**

*(From lab testbeds to production edge, integration specifics)*

### NIST – Blueprint for Deploying 5G O-RAN Testbeds

Offers detailed guidance for testbed configurations, including time sync, orchestration, external services ($DNS$, clock) [nvlpubs.nist.gov+1arxiv.org+1](https://nvlpubs.nist.gov/nistpubs/TechnicalNotes/NIST.TN.2311.pdf?utm_source=chatgpt.com)<https://nvlpubs.nist.gov/nistpubs/TechnicalNotes/NIST.TN.2311.pdf>

#### **Key takeaways**

O‑RAN Testbed Architecture Components:

* Defines **aggregated** (single-server) and **disaggregated** (multi-node) O‑RAN deployments integrating **O‑RU, O‑DU, O‑CU, Near‑RT RIC, Non‑RT RIC, 5G Core (AMF/UPF)** using open-source stacks like **srsRAN**, **OSC FlexRIC**, and **Open5GS**
* Offers end-to-end deployment blueprints including hardware, OS requirements (Ubuntu 20.04/22.04, low‑latency kernel), BIOS tuning, and container/orchestration options for both virtualized and bare-metal environments

Time Synchronization & Hardware Coordination:

* All USRP radios (B210/X310) are synchronized using **shared Pulse-Per-Second (PPS)** and **10 MHz frequency sources** from an OctoClock to maintain accurate time and frequency alignment
* Low-latency kernels and CPU/bios tuning (disable C-states, hyperthreading, secure boot) are recommended to minimize jitter and maximize real-time performance

Automation & Orchestration Tooling:

* Provides a **modular Testbed Automation Tool** (Linux shell, Python, C/C++) for 1-click provisioning of gNB, UE, RIC, xApps, and core components on both VMs and bare-metal, supporting repeatable setup and configuration
* The tool supports multiple RIC and RAN stacks (OSC, FlexRIC, srsRAN, OAI) and configures network addresses (e.g., default Docker IP 10.53.1.2), ZMQ/E2 paths, and tuning parameters for RF chains

External Services: DNS, Clock, and Controller Integration:

* Recommends integrating **external services** such as DNS resolution, NTP/PPS clock synchronization, and SMO/SMF controllers for real network usage scenarios
* testbed blueprints encourage automated settings for network functions to require external service reachability and time synchronization

Monitoring & xApp Support:

* Includes deployment of **xApps** (e.g., KPI‑Monitoring via E2/KPM) running on FlexRIC or OSC near‑RT RIC. The blueprint captures telemetry like **RSRP via E2 indications**
* **Container-based deployments (Docker/K8s) enable visibility into logs, pod statuses, and configurable xApp telemetry endpoints.**

#### **Summary factoids**

**Factoid**: “NIST blueprint supports both aggregated and disaggregated O‑RAN testbeds using srsRAN, FlexRIC, OSC, and Open5GS stacks.”

**Factoid**: “Time synchronization in O‑RAN requires PPS and 10 MHz frequency distribution to USRPs.”

**Factoid**: “Testbed servers use Ubuntu (20.04/22.04), low-latency kernels, and BIOS tuning (disable C‑states, HT, secure boot).”

**Factoid**: “The NIST Testbed Automation Tool automates deployment of gNB, UE, RIC, xApps, and 5G Core on bare-metal or virtual hosts.”

**Factoid**: “Automation tool supports configuration of ZMQ/E2 messaging and default Docker IP like 10.53.1.2.”

**Factoid**: “Modular orchestration with support for multiple RIC and RAN stacks reduces setup complexity and ensures repeatability.”

**Factoid**: “Testbed requires integration of DNS and clock services to support real-time control and function discovery.”

**Factoid**: “xApps (e.g., KPI monitor) are deployed on near‑RT RIC, subscribing to E2KP metrics like RSRP via ZMQ or container orchestration.”

**Factoid**: “Hardware tuning ensures low-latency and deterministic scheduling suitable for RF-based O‑RAN operations.”

**Factoid**: “Disaggregated scenarios use separate servers/nodes for O‑RU/DU/CU, RIC, Core, and UE emulator, enabling network-level isolation.”

### CSIR – Deploying a Stable 5G SA Testbed (srsRAN + Open5GS)

Provides configurations for NTP/DNS integration, monitoring agents for lab environments [nvlpubs.nist.gov](https://nvlpubs.nist.gov/nistpubs/TechnicalNotes/NIST.TN.2311.pdf?utm_source=chatgpt.com)[open5gs.org+3researchspace.csir.co.za+3nvlpubs.nist.gov+3](https://researchspace.csir.co.za/server/api/core/bitstreams/09c11bd7-3e9e-45f8-bbf3-07b181bfd0e0/content?utm_source=chatgpt.com)<https://researchspace.csir.co.za/server/api/core/bitstreams/09c11bd7-3e9e-45f8-bbf3-07b181bfd0e0/content>

#### **Key takeaways**

Testbed Architecture & Components:

* Deploys a **standalone 5G SA testbed** using **srsRAN** (O‑RU/O‑DU/CU/gNB), **Open5GS** (AMF/SMF/UPF core), USRP SDR (NI‑2944R via PCIe), and consumer-grade UEs
* Backhaul is gigabit Ethernet (recommend 10 Gbps); core runs on OpenStack/Docker containers over private VLANs

Time Sync, DNS & Configuration Automation:

* Integrates **NTP** and **DNS services** to support PFCP, NGAP, and HTTP2 control-plane protocols.
* System tuning includes disabling hyperthreading/VT, enabling real-time scheduling, and optimizing socket buffer sizes and MTU

RF Planning & UE Integration:

* Emphasizes managing **RF interference**, USRP gain setup (0–31.5 dB), duplex/SCS mode matching, and UE band/APN compatibility
* Highlights consumer UE variability: some require rooting, APN tweaking, or custom slices to attach successfully

Monitoring and Troubleshooting:

* Logs Open5GS services at /var/log/open5gs/\*; UEs require ZMQ logging; srsRAN console traces help diagnose NGAP, GTP‑U sessions, and slicing messages

Network Slicing Support:

* Implements **basic slicing** using NSSAI (SST and optional SD), enforcing slice logic in both RAN (gNB) and Core (AMF, SMF) domains
* Packet-level trace analysis confirms slice enforcement across PFCP and NGAP control/data sessions.

#### **Summary factoids**

**Factoid**: “Testbed uses srsRAN and Open5GS on commodity hardware and USRP NI‑2944R via PCIe for 5G SA operation.”

**Factoid**: “Backhaul uses gigabit Ethernet; upgrade to 10 Gbps is recommended for high-throughput experiments.”

**Factoid**: “Time sync via NTP and DNS are essential for control-plane protocols like PFCP, NGAP, HTTP2.”

**Factoid**: “Tuning involves disabling HyperThreading/VT, enabling real-time scheduling, and optimizing buffer sizes and MTU.”

**Factoid**: “RF planning requires configuring USRP gains, duplex mode, SCS, and matching UE APN and compatible bands.”

**Factoid**: “Some consumer UEs must be rooted or APN-modified to function reliably in 5G SA testbeds.”

**Factoid**: “Open5GS logs (/var/log/open5gs) and srsRAN/ZMQ traces are critical for NGAP, GTP-U, and slice debugging.”

**Factoid**: “Basic network slicing using NSSAI (SST/SD) is validated in both RAN and Core domains with packet trace inspection.”

**Factoid**: “Consistent slice enforcement is observable in PFCP session setups and NGAP signaling for UE-slice mapping.”

**Factoid**: “Consumer-grade UE variability necessitates RF tuning, root access, and APN configurations to ensure SA connectivity.”

### GitHub – Open5GS Kubernetes Deployment (niloysh/open5gs-k8s)

Supplies manifests for lab vs production environments, including network attachments, storage, external DB and monitoring service integration [researchspace.csir.co.za](https://researchspace.csir.co.za/server/api/core/bitstreams/09c11bd7-3e9e-45f8-bbf3-07b181bfd0e0/content?utm_source=chatgpt.com)[github.com+1github.com+1](https://github.com/niloysh/open5gs-k8s?utm_source=chatgpt.com)<https://github.com/niloysh/open5gs-k8s>

#### **Key takeaways**

Microservice vs All‑in‑One Deployment:

* Offers **Kubernetes manifests** for both **microservice‑oriented** and **all‑in‑one** deployments of Open5GS network functions (NFs), suited to lab or production scale

Network Attachments via Multus/OVS-CNI:

* Uses **Multus** and **OVS‑CNI** to create **separate network attachments** for N2, N3, and N4 interfaces, ensuring proper isolation of control and user plane traffic

Persistent Storage & External MongoDB:

* Deploys **MongoDB** via StatefulSet with **persistent volume claims**, used for subscriber data and NF configuration. External DB support enables scalability

Slice & Subscriber Management Scripts:

* Includes python/bash scripts for adding multiple slices and subscribers via CLI to MongoDB, enabling scripted configuration

UE and RAN Test Setup with UERANSIM:

* Provides manifests and scripts to deploy **UERANSIM gNB and UEs**, with automated verification (ping test, logs) to validate gNB–AMF connectivity

Monitoring with Monarch Integration:

* Offers optional **Monarch monitoring** (5G slice KPI metrics) alongside the deployment, configurable through switches in manifests

Release-Tagged Kubernetes Compatibility:

* Supports **Kubernetes v1.28**, Ubuntu 22.04, and containerd 1.6, with release notes documenting multi-slice and monitoring enhancements

Init Containers Enforce Startup Order:

* Uses **init containers** in each NF pod to handle startup sequencing (waiting for MongoDB, network attachments)

#### **Summary factoids**

**Factoid**: “open5gs‑k8s provides both microservice and all‑in‑one Kubernetes manifests for Open5GS.”

**Factoid**: “Multus and OVS‑CNI enable distinct network attachments for N2, N3, N4, preserving plane separation.”

**Factoid**: “MongoDB statefulsets with PVCs store subscriber and NF profile data externally.”

**Factoid**: “CLI and Python scripts automate slice and subscriber provisioning via MongoDB for multi-slice deployments.”

**Factoid**: “UERANSIM manifests facilitate gNB and UE emulation, including automated ping tests for AMF connectivity.”

**Factoid**: “Monarch integration supports real-time slice KPI monitoring when enabled in manifest.”

**Factoid**: “Supported configurations include Kubernetes v1.28, Ubuntu 22.04, containerd 1.6 – documented via release tags.”

**Factoid**: “Init containers orchestrate proper startup ordering, ensuring dependency readiness before NF launch.”

### Medium – Episode‑V Autoscaled 5G Core

Shows auto-scaling, config drift prevention, external monitoring and sync in dynamic environments [github.com](https://github.com/niloysh/open5gs-k8s?utm_source=chatgpt.com)[medium.com](https://medium.com/open-5g-hypercore/episode-v-autoscaled-5g-core-ab86b4803196?utm_source=chatgpt.com)<https://medium.com/open-5g-hypercore/episode-v-autoscaled-5g-core-ab86b4803196>

#### **Key takeaways**

Autoscaling Strategy for 5G Core CNFs:

* Uses **Horizontal Pod Autoscaler (HPA)** for scaling CNF pods (e.g., AMF, UPF) based on CPU load, supported by **Cluster Autoscaler (CA)** which adds/removes nodes as needed
* **Vertical Pod Autoscaler (VPA)** was avoided due to disruption risk—VPA-driven pod restarts and relocations can cause service outages, so HPA is preferred

Preventing Configuration Drift with GitOps:

* Relies on **GitOps** via an ArgoCD-like management hub to enforce consistent Day‑0/Day‑2 configurations, preventing “snowflake” drift across clusters

Cluster Node Homogeneity & Best Practices:

* All nodes in a group must run identical configurations—same capacity, labels, and system pods—with **PDBs and resource requests** for stability during scale-down/up cycles
* Avoid using multiple autoscalers per group and verify cloud quota before setting min/max settings

Smart Workload Scheduling & 5G Service Continuity:

* Config includes **Scale-Down Fuse**, which blocks scale-down of critical CNFs to maintain service integrity during resource fluctuation

Networking Configuration & SCTP Support:

* SCTP protocol support (Port 132) must be explicitly configured in both firewall/security‑group rules and node OS to enable correct CNF communication
* Passed through examples of SCTP socket errors when host configuration is incomplete

External Monitoring & Observability:

* Leverages external metrics (via Kubernetes metrics API and service mesh like Istio) for autoscaling, health insights, and performance monitoring

#### **Summary factoids**

**Factoid**: “HPA-based autoscaling is used for 5G CNFs, while VPA is avoided due to potential service disruptions.”

**Factoid**: “Cluster Autoscaler dynamically adds/removes nodes based on HPA pod requirements.”

**Factoid**: “GitOps (via ArgoCD) ensures consistent CNF configuration and prevents drift across clusters.”

**Factoid**: “Node pools must remain homogeneous in capacity and configuration, enforced by PodDisruptionBudgets and resource requests.”

**Factoid**: “A scale-down fuse protects critical CNFs (e.g., AMF, SMF) from being evicted during resource scaling events.”

**Factoid**: “SCTP protocol (132) must be permitted at network and OS level to support 5G CNF control-plane communication.”

**Factoid**: “Use of external monitoring (metrics API, Istio) provides data-driven trigger inputs for autoscaling decisions.”

**Factoid**: “Avoid multiple autoscalers in the same node group and check cloud quotas when configuring scaling policies.”

**Factoid**: “GitOps-based management hub enables central control of Day‑0/Day‑2 lifecycle operations for distributed 5G environments.”

**Factoid**: “Cluster and workload configuration consistency is essential for reliable auto-scaling in telecommunication-grade CNFs.”

### MDPI – Experimentation in 5G and Beyond Networks

Covers external connectivity, DNS, time sync, monitoring integration in lab/prod setups [medium.com](https://medium.com/open-5g-hypercore/episode-v-autoscaled-5g-core-ab86b4803196?utm_source=chatgpt.com)[mdpi-res.com](https://mdpi-res.com/bookfiles/book/8662/Experimentation_in_5G_and_beyond_Networks_State_of_the_Art_and_the_Way_Forward.pdf?v=1743296867&utm_source=chatgpt.com) https://mdpi-res.com/bookfiles/book/8662/Experimentation\_in\_5G\_and\_beyond\_Networks\_State\_of\_the\_Art\_and\_the\_Way\_Forward.pdf?v=1743296867

#### **Key takeaways**

Real-World 5G Testbeds Integration:

* Test platforms range from **agricultural remote-area setups (5G-RANGE)** to **city-scale neutral-host networks (5GCity)** and **UAV-aided 5G extensions**, emphasizing programmable NFV stacks and real cellular RAN deployments

End-to-End Testbed Requirements:

* Essential components include:
  + **External network connectivity** (internet access, external DNS)
  + **Time synchronization** (PPS/NTP/GNSS)
  + **External services** (DNS, clock servers)
  + **Monitoring telemetry pipelines** for real-time KPI collection in both lab and field environments

Automated Resource Provisioning and Management:

* Platforms employ **open-source tools** for automating resource instantiation, NF lifecycle management, and data collection—managed through Docker/Kubernetes deployments

Measurement & Validation of 5G KPIs:

* Emphasizes **E2E latency, throughput, reliability**, and **cross-domain synchronization**, along with KPI collection via telemetry and ML-based analysis for NB-IoT, URLLC, slicing, and edge-core performance

Lab ↔ Field Testbed Bridging:

* Recommendations include deploying **both lab and production setups** using uniform toolchains (SDR, open-core, virtualization). This ensures seamless transition of resource pipelines from controlled lab tests to real-world environments

#### **Summary factoids**

**Factoid**: “5G testbeds include remote-area (5G‑RANGE), neutral‑host city deployment (5GCity), and UAV‑assisted RAN with NFV flexibility.”

**Factoid**: “External DNS, internet connectivity, and time sync (PPS/NTP) are mandatory for realistic end-to-end testbed operation.”

**Factoid**: “Automated deployment uses open-source toolchains with Docker/K8s for NF instantiation, telemetry, and orchestration.”

**Factoid**: “KPI measurement includes latency, throughput, reliability, cross‑domain sync, collected via telemetry and ML analytics.”

**Factoid**: “Testbeds bridge lab and field environments through consistent resource pipelines, enabling repeatable validation cycles.”

**Factoid**: “Neutral-host frameworks (5GCity) allow operators to share infrastructure dynamically using slicing and multitenancy.”

**Factoid**: “5G-RANGE combines fixed RAN with UAVs supported by NFV for sporadic deployments in remote regions.”

**Factoid**: “End-to-end KPI validation must consider NFV stack maturity, telemetry quality, and orchestration tool chain alignment.”

**Factoid**: “Edge and core synchronization in testbeds hinges on coordinated time sync, network path management, and telemetry integration.”

**Factoid**: “Lab-to-production deployment is facilitated by standardized virtualization patterns across SDR, core, and orchestration environments.”

### How to Integrate into Your Pipeline

#### Intent → Config Rules

* Recognize intent patterns: e.g. "high throughput low latency for AR/VR" → SLAs (e.g., ≥7 Mbps / ≤50 ms).
* Map to architectural choices: deploy UPF at edge (:deployedOn :EdgeNode), assign slice :S-NSSAI.
* Constraint logic: if throughput >10 Gbps → instantiate multiple UPF instances with geo-redundancy.

#### Domain Constraints & Policies

* Encode regulatory zones and operator policies: IPv6-only, no public IP, micro-seg firewall rules.
* Security guidelines: enforce TLS on APIs and IPsec for N3/N6 tunnels.

#### Deployment Environments

Define testbed vs production environment types with properties:  
  
:Testbed1 :hasComponent :DNS, :NTP, :MonitoringAgent .

:Production1 :usesAutoScaling true .

* Link to manifests or template:  
  + Kubernetes manifest references, external services (e.g., Prometheus, MongoDB, etc.)

### Proposed Next Steps

Would you like me to:

* Provide **JSON-LD templates** capturing these mapping rules, policies, and deployment artifacts?
* Generate **YAML or Kubernetes manifest snippets** illustrating intent-based deployments?
* Create **RDF triples** example for intent, constraint, environment, with compliance rules?