**Factoid**: 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).

* **5G Core Network** *is introduced in* **3GPP Release 15**
* **5G Core Network** *adopts architecture* **SBA**
* **Network Function** *exposes* **Service**
* **Service** *is available via* **API** (HTTP/2 + JSON)
* **API** *uses protocol* **HTTP/2**
* **API** *uses format* **JSON**

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

* **Service-Based Architecture (SBA)** *enables* **dynamic service discovery**
* **Service-Based Architecture (SBA)** *enables* **direct NF-to-NF communication**
* **Direct NF-to-NF communication** *does not require* **intermediaries or brokers**

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

* **SBA model** *decouples* **service consumer** from **service provider**
* **SBA model** *achieves decoupling via* **Network Repository Function (NRF)**
* **Network Repository Function (NRF)** *mediates interaction between* **service consumer** and **service provider**

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

* **Service-Based Architecture (SBA)** *supports* **horizontal scalability**
* **Service-Based Architecture (SBA)** *supports* **modularity**
* **Service-Based Architecture (SBA)** *treats* **AMF** *as* **an independent service**
* **Service-Based Architecture (SBA)** *treats* **SMF** *as* **an independent service**
* **Service-Based Architecture (SBA)** *treats* **PCF** *as* **an independent service**

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

* **5G core functions** *are designed as* **cloud-native components**
* **RAN functions** *are designed as* **cloud-native components**
* **Cloud-native components** *follow the principle of* **containerization**
* **Cloud-native components** *follow the principle of* **statelessness**
* **Cloud-native components** *follow the principle of* **modularity**

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

* **Network Functions (NFs)** *are implemented as* **microservices**
* **Microservices** *enable* **CI/CD deployment**
* **Microservices** *enable* **independent updates**
* **Microservices** *enable* **fine-grained scaling**

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

* **Network Function (NF)** *runs in* **isolated execution environments**
* **Isolated execution environments** *are typically realized as* **containers**
* **Containers** *are orchestrated via* **Kubernetes**
* **Containers** *may be orchestrated via* **NFVO frameworks**

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

* **Microservices** *support* **vendor-agnostic deployments**
* **Microservices** *support* **hardware-agnostic deployments**
* **Container platforms** *support* **vendor-agnostic deployments**
* **Container platforms** *support* **hardware-agnostic deployments**
* **Vendor-agnostic deployments** *occur in* **hybrid cloud environments**
* **Hardware-agnostic deployments** *occur in* **hybrid cloud environments**

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

* **Cloud-native features** *allow* **slicing**
* **Cloud-native features** *allow* **multi-tenancy**
* **Cloud-native features** *allow* **elasticity**
* **Slicing** *is used in managing* **network resources**
* **Multi-tenancy** *is used in managing* **network resources**
* **Elasticity** *is used in managing* **network resources**

**Factoid**: 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.

* **5G** *continues* **Control and User Plane Separation (CUPS) principle**
* **Control and User Plane Separation (CUPS) principle** *was introduced in* **3GPP Release 14**
* **CUPS principle** *allows* **independent placement** of **control-plane functions**
* **CUPS principle** *allows* **independent scaling** of **control-plane functions**
* **CUPS principle** *allows* **independent placement** of **user-plane functions**
* **CUPS principle** *allows* **independent scaling** of **user-plane functions**
* **Control-plane functions** *include* **AMF**
* **Control-plane functions** *include* **SMF**
* **User-plane functions** *include* **UPF**

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

* **CUPS** *enables deploying* **UPF close to the network edge**
* **UPF close to the network edge** *supports* **latency-sensitive services**
* **Control-plane functions** *remain* **centralized**

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

* **Separation of control and user planes (CUPS)** *enhances* **fault isolation**
* **Separation of control and user planes (CUPS)** *enhances* **resource utilization**
* **Resource utilization** *occurs in* **the network**

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

* **CUPS** *is a prerequisite for* **flexible network slicing**
* **CUPS** *is a prerequisite for* **performant network slicing**
* **CUPS** *is a prerequisite for* **multi-access edge computing (MEC)**

**Factoid**: 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).

* **5G SBA** *enables* **stateless operation of AMF**
* **5G SBA** *enables* **stateless operation of SMF**
* **5G SBA** *enables* **stateless operation of AUSF**
* **5G SBA** *enables* **stateless operation of UDM**
* **AMF** *offloads* **UE context to Unstructured Data Storage Function (UDSF)**
* **SMF** *offloads* **UE context to Unstructured Data Storage Function (UDSF)**
* **AUSF** *offloads* **UE context to Unstructured Data Storage Function (UDSF)**
* **UDM** *offloads* **UE context to Unstructured Data Storage Function (UDSF)**
* **Offloading UE context to UDSF** *enables* **stateless operation**

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

* **Stateless NFs** *gain* **elasticity**
* **Stateless NFs** *gain* **failure resilience**
* **Stateless NFs** *enable* **container-based deployments**
* **Stateless NFs** *enable* **fast restarts**
* **Fast restarts** *occur without* **local state loss**

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

* **UDSF** *acts as* **centralized non-relational key-value store**
* **UDSF** *enables* **fast read/write of UE-related state**
* **Fast read/write** *operates on* **UE-related state**

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

* **Stateful NFs** *are tightly coupled with* **internal UE context**
* **Stateful NFs tightly coupled with internal UE context** *incur* **failure risks**
* **Stateful NFs tightly coupled with internal UE context** *incur* **scaling limitations**

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

* **Piggyback-based approach** *retrieves* **UE context once per procedure**
* **Piggyback-based approach** *embeds* **UE context** in **NF-to-NF HTTP calls**
* **Embedding UE context in NF-to-NF HTTP calls** *reduces* **repeated fetches**

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

* **Piggybacking** *reduces* **registration procedure latency by ≈ 44 %**
* **Piggybacking** *reduces* **deregistration procedure latency by up to ≈ 70 %**

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

* **Proactive-push approach** *enables* **AMF to preemptively instruct UDSF**
* **AMF** *preemptively instructs* **UDSF to push context**
* **UDSF** *pushes* **context to downstream NFs**
* **Context** *is pushed via* **new SBI POST endpoints**

ex:AMF a rdfs:Class .

ex:UDSF a rdfs:Class .

ex:NF a rdfs:Class .

ex:ContextPush a rdfs:Class .

ex:SBIPostEndpoint a rdfs:Class .

ex:proactivePushApproach a ex:Approach ;

rdfs:label "Proactive-push approach" .

ex:preemptivelyInstructs a rdf:Property ;

rdfs:domain ex:AMF ;

rdfs:range ex:UDSF .

ex:pushesContextTo a rdf:Property ;

rdfs:domain ex:UDSF ;

rdfs:range ex:NF .

ex:usesSBIendpoint a rdf:Property ;

rdfs:domain ex:ContextPush ;

rdfs:range ex:SBIPostEndpoint .

ex:proactivePushApproach ex:enables ex:AMFpreemptivelyInstructingUDSF .

ex:AMFpreemptivelyInstructingUDSF

a ex:ContextPush ;

ex:performedBy ex:AMF ;

ex:target ex:UDSF ;

rdfs:comment "AMF preemptively instructs UDSF to push context." .

ex:AMF ex:preemptivelyInstructs ex:UDSF .

ex:UDSF ex:pushesContextTo ex:NF .

ex:**ContextPush ex:usesSBIendpoint ex:SBIPostEndpoint .**

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

* **Proactive-push** *improves* **asynchronous procedure completion times by ≈ 13–22 %**
* **Asynchronous procedure completion times** *include* **PDU session setup**

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

* **Piggybacking** *is optimal for* **synchronous flows**
* **Proactive-push** *works best for* **asynchronous flows**

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

* **Hybrid approach** *combines* **piggyback** and **proactive-push**
* **Hybrid approach** *yields* **best end-to-end control procedure performance**

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

* **These approaches** *require no major changes to* **core 5G architecture**
* **These approaches** *depend on* **UDSF**
* **These approaches** *depend on* **SBI endpoint extension**

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

* **Prototype deployments** *were conducted on* **Kubernetes**
* **Prototype deployments** *used* **Free5GC**
* **Prototype deployments** *confirmed* **stateless NFs**
* **Stateless NFs** *incur* **minimal CPU/memory overhead**

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

* **Procedure-aware optimizations** *are backward compatible with* **existing SBI-based service flows**
* **Procedure-aware optimizations** *are non-intrusive to* **existing SBI-based service flows**

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

* **NG-RAN** *is split into* **gNB-DU**
* **NG-RAN** *is split into* **gNB-CU**
* **gNB-DU** *performs* **real-time processing**
* **gNB-CU** *performs* **centralized control**
* **gNB-CU** *performs* **core interfacing**

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

* **gNB-CU** *is functionally divided into* **CU-CP**
* **gNB-CU** *is functionally divided into* **CU-UP**
* **gNB-CU functional division** *mirrors* **CUPS design**

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

* **CU-CP** *connects to* **AMF**
* **CU-CP–AMF connection** *uses* **NG-AP**
* **CU-CP–AMF connection** *uses* **SCTP**
* **CU-CP–AMF connection** *constitutes* **NG-C link**
* **CU-UP** *transfers data to* **UPF**
* **Data transfer between CU-UP and UPF** *occurs via* **NG-U**
* **Data transfer between CU-UP and UPF** *uses* **GTP-U**

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

* **E1 interface** *supports* **CU-CP ↔ CU-UP communication**
* **CU-CP ↔ CU-UP communication** *uses* **E1AP**
* **E1AP** *runs over* **SCTP**
* **CU-CP ↔ CU-UP communication** *is used for* **bearer/context management**

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

* **F1-C split** *handles* **control-plane data** between **CU** and **DU**
* **F1-U split** *handles* **user-plane data** between **CU** and **DU**
* **F1-C split** *uses* **SCTP**
* **F1-U split** *uses* **SCTP**

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

* **CUPS architecture** *enables* **horizontal scaling**
* **CUPS architecture** *enables* **placement flexibility**
* **Horizontal scaling** *applies across* **RAN**
* **Horizontal scaling** *applies across* **core**
* **Placement flexibility** *applies across* **RAN**
* **Placement flexibility** *applies across* **core**

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

* **5G NR** *uses* **scalable OFDM numerology**
* **5G NR** *uses* **ultra-lean design**
* **5G NR** *uses* **beamforming**
* **Scalable OFDM numerology** *supports* **diverse service types**
* **Ultra-lean design** *supports* **diverse service types**
* **Beamforming** *supports* **diverse service types**
* **Diverse service types** *include* **eMBB**
* **Diverse service types** *include* **mMTC**
* **Diverse service types** *include* **URLLC**

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

* **Separation of control and data functions** *allows* **independent edge deployment**
* **Separation of control and data functions** *allows deploying* **CU-UP at the edge**
* **Separation of control and data functions** *allows keeping* **CU-CP centralized**
* **CU-UP at the edge** *reduces* **latency**
* **Independent edge deployment** *enables* **low latency**

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

* **Functional splits** *enhance* **vendor diversity**
* **Functional splits** *enhance* **network slicing**
* **Functional splits** *enhance* **multi-tenant orchestration**
* **Multi-tenant orchestration** *occurs in* **cloud-native environments**

**Factoid**: 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.

* **SBA control plane** *is more vulnerable due to* **REST over HTTP/2 interfaces**
* **REST over HTTP/2 interfaces** *connect* **network functions (NFs)**
* **REST over HTTP/2 interfaces between NFs** *enable* **new attack vectors**
* **New attack vectors** *include* **header manipulation**
* **New attack vectors** *include* **horizontal DDoS**

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

* **NRF** *centralizes* **service registry**
* **NRF** *centralizes* **service discovery**
* **Compromised NRF** *can route* **traffic to malicious NFs**
* **Compromised NRF** *can route* **traffic to rogue NFs**

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

* **End-to-end mutual TLS (mTLS)** *is required to secure* **NF-to-NF communication**
* **NF-to-NF communication** *occurs in* **SBA control plane**

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

* **API gateways** *detect* **anomalous patterns**
* **Service proxies** *detect* **anomalous patterns**
* **API gateways** *rate-limit* **flows**
* **Service proxies** *rate-limit* **flows**
* **API gateways** *enforce* **role-based access to NF services**
* **Service proxies** *enforce* **role-based access to NF services**

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

* **PKI with PLMN-bound CAs** *allows* **strong NF identity verification**
* **PKI with PLMN-bound CAs** *ensures* **integrity within multitenant operator domains**
* **Strong NF identity verification** *applies within* **multitenant operator domains**
* **Integrity** *applies within* **multitenant operator domains**

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

* **NF deployments** *must include* **micro-segmentation**
* **NF deployments** *must include* **zero-trust policies**
* **Micro-segmentation** *mitigates* **lateral threats**
* **Zero-trust policies** *mitigate* **lateral threats**
* **Lateral threats** *occur in* **disaggregated environments**

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

* **Control plane security** *must encompass* **confidentiality**
* **Control plane security** *must encompass* **integrity**
* **Control plane security** *must encompass* **authentication**
* **Control plane security** *must encompass* **segmentation**
* **Control plane security** *must encompass* **runtime monitoring**
* **Confidentiality** *must apply across* **all NF interfaces**
* **Integrity** *must apply across* **all NF interfaces**
* **Authentication** *must apply across* **all NF interfaces**
* **Segmentation** *must apply across* **all NF interfaces**
* **Runtime monitoring** *must apply across* **all NF interfaces**

**Factoid**: AMF manages UE registration, mobility, and NAS signaling.

* **AMF** *manages* **UE registration**
* **AMF** *manages* **mobility**
* **AMF** *manages* **NAS signaling**

**Factoid:** AMF authenticates UE using NAS and passes session control to SMF.

* **AMF** *authenticates* **UE using NAS**
* **AMF** *passes* **session control to SMF**

**Factoid:** AMF is control-plane only; it does not handle user traffic.

* **AMF** *is* **control-plane only**
* **AMF** *does not handle* **user traffic**

**Factoid:** AMF can redirect UEs to appropriate SMF/UPF combinations based on policies.

* **AMF** *redirects* **UEs** to **appropriate SMF/UPF combinations**
* **AMF** *selects* **SMF/UPF combinations** based on **policies**

**Factoid:** AMF is dependent on UDM (for subscription), AUSF (for auth), SMF (for session setup).

* **AMF** *depends on* **UDM** for subscription
* **AMF** *depends on* **AUSF** for authentication
* **AMF** *depends on* **SMF** for session setup

**Factoid:** AMF uses such interfaces as: N1 (UE-AMF) NAS over IP, N2 (AMF-gNB) NGAP over SCTP, N11 (AMF-SMF) SBI (HTTP/2+JSON).

* **AMF** *uses* **N1 interface**
* **N1 interface** *connects* **UE** and **AMF**
* **N1 interface** *carries* **NAS over IP**
* **AMF** *uses* **N2 interface**
* **N2 interface** *connects* **AMF** and **gNB**
* **N2 interface** *carries* **NGAP over SCTP**
* **AMF** *uses* **N11 interface**
* **N11 interface** *connects* **AMF** and **SMF**
* **N11 interface** *implements* **SBI over HTTP/2 + JSON**

**Factoid:** SMF handles PDU session lifecycle: setup, modification, and release.

* **SMF** *handles* **PDU session lifecycle**
* **SMF** *handles* **PDU session setup**
* **SMF** *handles* **PDU session modification**
* **SMF** *handles* **PDU session release**

**Factoid:** SMF allocates IP addresses and installs traffic steering rules in UPFs.

* **SMF** *allocates* **IP addresses**
* **SMF** *installs* **traffic steering rules**
* **Traffic steering rules** *are installed in* **UPFs**

**Factoid:** SMF enforces QoS via PCF policies.

* **SMF** *enforces* **QoS**
* **SMF** *uses* **PCF policies** to enforce **QoS**

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

* **SMF** *connects to* **UPFs**
* **SMF–UPF connection** *uses* **PFCP**
* **PFCP** *is designed for* **fast user-plane control**
* **PFCP** *is designed for* **stateless user-plane control**

**Factoid:** SMF uses the following interfaces: N11 (SMF-AMF), N4 (SMF-UPF) PFCP over UDP, and N10 (SMF-UDM).

* **SMF** *uses* **N11 interface**
* **N11 interface** *connects* **SMF** and **AMF**
* **SMF** *uses* **N4 interface**
* **N4 interface** *connects* **SMF** and **UPF**
* **N4 interface** *uses* **PFCP**
* **PFCP** *runs over* **UDP**
* **SMF** *uses* **N10 interface**
* **N10 interface** *connects* **SMF** and **UDM**

**Factoid:** SMF is dependent on AMF, UPF, and PCF.

* **SMF** *depends on* **AMF**
* **SMF** *depends on* **UPF**
* **SMF** *depends on* **PCF**

**Factoid:** SMF exchanges data such as session context, QoS rules, and IP address allocation.

* **SMF** *exchanges* **session context**
* **SMF** *exchanges* **QoS rules**
* **SMF** *exchanges* **IP address allocation**

**Factoid:** SMF uses protocols including HTTP/2 and PFCP.

* **SMF** *uses* **HTTP/2**
* **SMF** *uses* **PFCP**

**Factoid:** UPF handles user traffic forwarding and QoS enforcement.

* **UPF** *handles* **user traffic forwarding**
* **UPF** *handles* **QoS enforcement**

**Factoid:** UPF forwards traffic between RAN and external data networks.

* **UPF** *forwards* **traffic between RAN and external data networks**
* **UPF** *connects* **RAN** to **external data networks**

**Factoid:** UPF supports data buffering, downlink packet marking, and traffic shaping.

* **UPF** *supports* **data buffering**
* **UPF** *supports* **downlink packet marking**
* **UPF** *supports* **traffic shaping**

**Factoid:** Multiple UPFs can be deployed regionally to support MEC and network slicing.

* **Multiple UPFs** *can be deployed* **regionally**
* **Multiple UPFs** *support* **multi-access edge computing (MEC)**
* **Multiple UPFs** *support* **network slicing**

**Factoid:** UPF uses the following interfaces: N3 (UPF-gNB) GTP-U, N4 (UPF-SMF) PFCP, N6 (UPF-DN) IP forwarding.

* **UPF** *uses* **N3 interface**
* **N3 interface** *connects* **UPF** and **gNB**
* **N3 interface** *carries* **GTP-U**
* **UPF** *uses* **N4 interface**
* **N4 interface** *connects* **UPF** and **SMF**
* **N4 interface** *uses* **PFCP**
* **UPF** *uses* **N6 interface**
* **N6 interface** *connects* **UPF** and **data network (DN)**
* **N6 interface** *provides* **IP forwarding**

**Factoid:** UPF is dependent on SMF.

* **UPF** *depends on* **SMF**

**Factoid:** UPF manages data such as PDU sessions and QoS enforcement rules.

* **UPF** *manages* **PDU sessions**
* **UPF** *manages* **QoS enforcement rules**

**Factoid:** UPF uses protocols including PFCP and GTP-U.

* **UPF** *uses* **PFCP**
* **UPF** *uses* **GTP-U**

**Factoid:** AUSF handles UE authentication using 5G AKA or EAP-AKA′.

* **AUSF** *handles* **UE authentication**
* **UE authentication** *uses* **5G AKA**
* **UE authentication** *uses* **EAP-AKA′**

**Factoid:** AUSF performs challenge–response procedures with UEs during registration.

* **AUSF** *performs* **challenge–response procedures**
* **Challenge–response procedures** *involve* **UEs**
* **Challenge–response procedures** *occur during* **registration**

**Factoid:** AUSF is stateless by design and uses UDM to obtain authentication data (e.g., K, OPc, SQN).

* **AUSF** *is* **stateless by design**
* **AUSF** *uses* **UDM**
* **AUSF** *obtains* **authentication data** from **UDM**
* **Authentication data** *includes* **K**
* **Authentication data** *includes* **OPc**
* **Authentication data** *includes* **SQN**

**Factoid:** AUSF uses the following interfaces: N12 (AMF-AUSF) and N13 (AUSF-UDM).

* **AUSF** *uses* **N12 interface**
* **N12 interface** *connects* **AMF** and **AUSF**
* **AUSF** *uses* **N13 interface**
* **N13 interface** *connects* **AUSF** and **UDM**

**Factoid:** AUSF is dependent on UDM for subscription credentials.

* **AUSF** *depends on* **UDM**
* **AUSF** *obtains* **subscription credentials** from **UDM**

**Factoid:** AUSF processes authentication vectors as part of its data handling.

* **AUSF** *processes* **authentication vectors**
* **Authentication vectors** *are part of* **AUSF data handling**

**Factoid:** UDM serves as the central database for subscriber profiles and policies.

* **UDM** *serves as* **central database**
* **Central database** *stores* **subscriber profiles**
* **Central database** *stores* **policies**

**Factoid:** UDM provides authentication data to AUSF and access policies to AMF.

* **UDM** *provides* **authentication data** to **AUSF**
* **UDM** *provides* **access policies** to **AMF**

**Factoid:** UDM stores SUPI, subscription profiles, and AM Policy Association.

* **UDM** *stores* **SUPI**
* **UDM** *stores* **subscription profiles**
* **UDM** *stores* **AM Policy Association**

**Factoid:** UDM uses the following interfaces: N8 (UDM-AMF), N10 (UDM-SMF), N13 (UDM-AUSF).

* **UDM** *uses* **N8 interface**
* **N8 interface** *connects* **UDM** and **AMF**
* **UDM** *uses* **N10 interface**
* **N10 interface** *connects* **UDM** and **SMF**
* **UDM** *uses* **N13 interface**
* **N13 interface** *connects* **UDM** and **AUSF**

**Factoid:** UDM manages data including subscription data, access profiles, and authentication keys.

* **UDM** *manages* **subscription data**
* **UDM** *manages* **access profiles**
* **UDM** *manages* **authentication keys**

**Factoid:** UDM may link with the HSS in interworking scenarios for backward compatibility.

* **UDM** *may link with* **HSS**
* **UDM–HSS linking** *occurs in* **interworking scenarios**
* **UDM–HSS linking** *provides* **backward compatibility**

**Factoid:** PCF provides policy decisions such as QoS and charging to other network functions.

* **PCF** *provides* **policy decisions**
* **PCF** *provides* **policy decisions** to **other network functions**
* **Policy decisions** *include* **QoS**
* **Policy decisions** *include* **charging**

**Factoid:** PCF enforces subscriber-specific policies via SMF and AMF.

* **PCF** *enforces* **subscriber-specific policies**
* **PCF** *enforces* \**subscriber-specific policies via* **SMF**
* **PCF** *enforces* \**subscriber-specific policies via* **AMF**

**Factoid:** PCF works with AF (Application Function) to enforce application-level QoS.

* **PCF** *works with* **AF (Application Function)**
* **PCF and AF (Application Function)** *enforce* **application-level QoS**

**Factoid:** PCF uses the following interfaces: N7 (PCF-SMF) and N15 (PCF-AMF).

* **PCF** *uses* **N7 interface**
* **N7 interface** *connects* **PCF** and **SMF**
* **PCF** *uses* **N15 interface**
* **N15 interface** *connects* **PCF** and **AMF**

**Factoid:** PCF manages data including policy rules, network slicing information, and QoS profiles.

* **PCF** *manages* **policy rules**
* **PCF** *manages* **network slicing information**
* **PCF** *manages* **QoS profiles**

**Factoid:** PCF may connect to UDR for policy rule storage.

* **PCF** *may connect to* **UDR**
* **UDR** *stores* **policy rules**

**Factoid:** NRF provides service discovery and registration for the Service-Based Architecture (SBA).

* **NRF** *provides* **service discovery**
* **NRF** *provides* **registration**
* **Service-Based Architecture (SBA)** *relies on* **NRF**

**Factoid:** NRF enables dynamic discovery of NF services using API lookup.

* **NRF** *enables* **dynamic discovery of NF services**
* **Dynamic discovery of NF services** *uses* **API lookup**

**Factoid:** NRF ensures service-level routing and load-aware NF selection.

* **NRF** *ensures* **service-level routing**
* **NRF** *ensures* **load-aware NF selection**

**Factoid:** NRF uses the Nnrf interface (NF-NRF) over HTTP/2.

* **NRF** *uses* **Nnrf interface**
* **Nnrf interface** *connects* **network functions (NFs)** and **NRF**
* **Nnrf interface** *runs over* **HTTP/2**

**Factoid:** NRF manages data such as NF profiles, service status, and availability.

* **NRF** *manages* **NF profiles**
* **NRF** *manages* **service status**
* **NRF** *manages* **availability**

**Factoid:** NRF is a dependency for all SBA-based network functions.

* **NRF** *is a dependency for* **all SBA-based network functions**

**Factoid:** UDR acts as a centralized database backend for UDM and PCF.

* **UDR** *acts as* **centralized database backend**
* **UDR** *serves as backend for* **UDM**
* **UDR** *serves as backend for* **PCF**

**Factoid:** UDR decouples business logic (UDM/PCF) from persistent storage.

* **UDR** *decouples* **business logic** from **persistent storage**
* **Business logic** *includes* **UDM**
* **Business logic** *includes* **PCF**

**Factoid:** UDR improves fault tolerance and scaling of data-centric functions.

* **UDR** *improves* **fault tolerance**
* **UDR** *improves* **scaling**
* **Fault tolerance** *applies to* **data-centric functions**
* **Scaling** *applies to* **data-centric functions**

**Factoid:** UDR uses the N5 interface (UDM/PCF-UDR).

* **UDR** *uses* **N5 interface**
* **N5 interface** *connects* **UDM** and **UDR**
* **N5 interface** *connects* **PCF** and **UDR**

**Factoid:** UDR manages data such as subscriber data, policy data, and configuration values.

* **UDR** *manages* **subscriber data**
* **UDR** *manages* **policy data**
* **UDR** *manages* **configuration values**

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

* **PFCP** *is* **standardized protocol for SMF-to-UPF control**
* **PFCP** *operates over* **UDP**
* **PFCP** *operates at* **N4 reference point**
* **PFCP** *operates at* **Sx reference point**
* **SMF-to-UPF control** *uses* **PFCP**

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

* **SMF** *uses* **PFCP**
* **SMF** *installs* **PFDs** in **UPF**
* **SMF** *installs* **FARs** in **UPF**
* **PFDs** *govern* **user data forwarding rules**
* **FARs** *govern* **user data forwarding rules**

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

* **PFCP** *is fundamental for enabling* **Control-User Plane Separation (CUPS)**
* **Control-User Plane Separation (CUPS)** *is applied in* **5G core**

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

* **PFCP** *includes* **Keepalive mechanism**
* **PFCP** *includes* **Session mechanism**
* **PFCP** *includes* **QER mechanism**
* **PFCP** *includes* **BAR mechanism**
* **Keepalive mechanism** *manages* **forwarding**
* **Keepalive mechanism** *manages* **buffering**
* **Session mechanism** *manages* **forwarding**
* **Session mechanism** *manages* **buffering**
* **QER mechanism** *manages* **forwarding**
* **QER mechanism** *manages* **buffering**
* **BAR mechanism** *manages* **forwarding**
* **BAR mechanism** *manages* **buffering**

**Factoid:** PFCP enables dynamic session lifecycle control by allowing SMF to initiate, modify, or tear down UPF forwarding behavior in real time.

* **PFCP** *enables* **dynamic session lifecycle control**
* **PFCP** *allows* **SMF** to initiate **UPF forwarding behavior** in real time
* **PFCP** *allows* **SMF** to modify **UPF forwarding behavior** in real time
* **PFCP** *allows* **SMF** to tear down **UPF forwarding behavior** in real time

**Factoid:** Through rule-based provisioning, PFCP allows SMF to granularly instruct UPF on how to classify, buffer, or prioritize specific traffic flows.

* **SMF** *uses* **PFCP;**
* **SMF** *instructs* **UPF;**
* **UPF** *classifies* **SpecificTrafficFlows.**
* **UPF** *buffers* **SpecificTrafficFlows.**
* **UPF** *prioritizes* **SpecificTrafficFlows.**

**Factoid:** PFCP sessions maintain stateful associations between SMF and UPF, ensuring coherent control across multiple user sessions.

* **PFCP sessions** *maintain* **stateful associations between SMF and UPF**
* **Stateful associations between SMF and UPF** *ensure* **coherent control across multiple user sessions**

**Factoid:** The PFCP protocol supports heartbeats and recovery procedures to detect failure conditions and maintain high-availability pathways.

* **PFCP protocol** *supports* **heartbeats**
* **PFCP protocol** *supports* **recovery procedures**
* **Heartbeats** *detect* **failure conditions**
* **Recovery procedures** *detect* **failure conditions**
* **Heartbeats** *maintain* **high-availability pathways**
* **Recovery procedures** *maintain* **high-availability pathways**

**Factoid:** By separating control and data responsibilities, PFCP allows distributed UPF instances to be managed centrally by fewer SMF nodes.

* **PFCP** *allows* **central management of distributed UPF instances**
* **Central management of distributed UPF instances** *is performed by* **fewer SMF nodes**

**Factoid:** Advanced rule types in PFCP such as QER and BAR enable precise QoS enforcement and packet buffering, tailored to per-session policy.

* **Advanced rule types in PFCP** *include* **QER**
* **Advanced rule types in PFCP** *include* **BAR**
* **Advanced rule types in PFCP** *enable* **precise QoS enforcement**
* **Advanced rule types in PFCP** *enable* **packet buffering**
* **Precise QoS enforcement** *is tailored to* **per-session policy**
* **Packet buffering** *is tailored to* **per-session policy**

**Factoid:** PFCP supports scalable deployment by minimizing control overhead using compact rule structures and stateless UDP transport.

* **PFCP** *supports* **scalable deployment**
* **PFCP** *minimizes* **control overhead**
* **Control overhead** *is minimized using* **compact rule structures**
* **Control overhead** *is minimized using* **stateless UDP transport**

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

* **PFCP** *is specified in* **3GPP TS 29.244**
* **PFCP** *relies on* **UDP transport**
* **UDP transport** *carries* **control messaging**

**Factoid:** SEPP encrypts and proxies inter-PLMN control signaling across roaming borders.

* **SEPP** *encrypts* **inter-PLMN control signaling**
* **SEPP** *proxies* **inter-PLMN control signaling**
* **Inter-PLMN control signaling** *crosses* **roaming borders**

**Factoid:** SEPP ensures end-to-end control-plane security via TLS/IPsec tunnels.

* **SEPP** *ensures* **end-to-end control-plane security**
* **SEPP** *uses* **TLS/IPsec tunnels**
* **TLS/IPsec tunnels** *provide* **end-to-end control-plane security**

**Factoid:** N3IWF acts as a gateway for non-3GPP access, such as Wi-Fi, using IKEv2 tunnels.

* **N3IWF** *acts as* **gateway for non-3GPP access**
* **Non-3GPP access** *includes* **Wi-Fi**
* **N3IWF** *uses* **IKEv2 tunnels**

**Factoid:** N3IWF bridges Wi-Fi clients to the AMF securely, without requiring native RAN.

* **N3IWF** *bridges* **Wi-Fi clients** to **AMF securely**
* **N3IWF-mediated Wi-Fi–AMF connection** *does not require* **native RAN**

**Factoid:** W-AGF is the Wireline Access Gateway Function used for fixed/mobile network convergence.

* **W-AGF** *is* **Wireline Access Gateway Function**
* **Wireline Access Gateway Function** *is used for* **fixed/mobile network convergence**

**Factoid:** W-AGF enables fixed wireless access (FWA) and fixed-line services to connect to the 5G Core (5GC).

* **W-AGF** *enables* **fixed wireless access (FWA)**
* **W-AGF** *enables* **fixed-line services**
* **Fixed wireless access (FWA)** *connects to* **5G Core (5GC)**
* **Fixed-line services** *connect to* **5G Core (5GC)**

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

* **5G core** *employs* **Service-Based Architecture (SBA)**
* **NFs (AMF, SMF, UPF)** *register services via* **NRF**
* **NFs (AMF, SMF, UPF)** *discover services via* **NRF**
* **Service registration/discovery via NRF *uses* HTTP/2 + REST**

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

* **Service-Based Architecture (SBA)** *provides* **modularity**
* **Modularity** *enables* **dynamic NF registration**
* **Modularity** *enables* **vendor interoperability**

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

* **NSA (Non-Standalone architecture)** *combines* **5G NR radio**
* **NSA (Non-Standalone architecture)** *combines* **4G EPC core**
* **SA (Standalone architecture)** *uses* **5GC**
* **5GC** *supports* **network slicing**
* **5GC** *supports* **URLLC**
* **5GC** *supports* **mMTC**

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

* **Cybersecurity in 5G** *requires* **layered defenses**
* **Layered defenses** *span* **virtualization**
* **Layered defenses** *span* **SBA APIs**
* **Layered defenses** *span* **RAN**
* **Layered defenses** *span* **network slices**

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

* **Service-Based Architecture (SBA)** *includes* **AMF**
* **Service-Based Architecture (SBA)** *includes* **SMF**
* **Service-Based Architecture (SBA)** *includes* **UPF**
* **Service-Based Architecture (SBA)** *includes* **UDM**
* **Service-Based Architecture (SBA)** *includes* **PCF**
* **Service-Based Architecture (SBA)** *includes* **SEPP -> roaming entity**
* **Service-Based Architecture (SBA)** *includes* **N3IWF -> roaming entity**
* **Service-Based Architecture (SBA)** *includes* **W-AGF -> roaming entity**

**Factoid:** AMF can operate statelessly by externalizing UE context to a Unified Data Storage Function (UDSF), enhancing horizontal scalability.

* **AMF** *can operate* **statelessly**
* **AMF** *externalizes* **UE context** to **Unified Data Storage Function (UDSF)**
* **Externalizing UE context to UDSF** *enhances* **horizontal scalability**

**Factoid:** SMF uses the N7 interface to retrieve policy decisions from PCF, enabling differentiated handling per session.

* **SMF** *uses* **N7 interface**
* **SMF** *retrieves* **policy decisions** from **PCF**
* **Policy decisions** *enable* **differentiated per-session handling**

**Factoid:** UPF maintains stateful session information to perform precise flow routing and per-user QoS enforcement.

* **UPF** *maintains* **stateful session information**
* **Stateful session information** *enables* **precise flow routing**
* **Stateful session information** *enables* **per-user QoS enforcement**

**Factoid:** NRF supports load-aware NF instance selection by maintaining up-to-date service availability data.

* **NRF** *supports* **load-aware NF instance selection**
* **NRF** *maintains* **up-to-date service availability data**
* **Maintaining up-to-date service availability data** *enables* **load-aware NF instance selection**

**Factoid:** AUSF leverages HTTP/2 interfaces to securely interact with UDM for retrieving authentication keys and vectors.

* **AUSF** *leverages* **HTTP/2 interfaces**
* **AUSF** *securely interacts with* **UDM**
* **Secure interaction with UDM** *retrieves* **authentication keys and vectors**

**Factoid:** UDM is a stateful front-end to subscriber data, often backed by UDR for persistence and redundancy.

* **UDM** *is stateful front-end to* **subscriber data**
* **UDM** *is backed by* **UDR**
* **UDR** *provides* **persistence**
* **UDR** *provides* **redundancy**

**Factoid:** UDR acts as a persistent, central repository supporting both subscriber data (UDM) and policy data (PCF), facilitating decoupling.

* **UDR** *acts as* **persistent central repository**
* **UDR** *supports* **subscriber data for UDM**
* **UDR** *supports* **policy data for PCF**
* **UDR’s support for subscriber and policy data** *facilitates* **decoupling**

**Factoid:** PCF is inherently stateless and offloads data persistence to UDR, allowing lightweight and scalable deployment.

* **PCF** *is* **inherently stateless**
* **PCF** *offloads* **data persistence** to **UDR**
* **Offloading data persistence to UDR** *allows* **lightweight deployment**
* **Offloading data persistence to UDR** *allows* **scalable deployment**

**Factoid:** NSSF enables network slicing by selecting appropriate slice instances based on policies and UE subscription data.

* **NSSF** *enables* **network slicing**
* **NSSF** *selects* **slice instances**
* **Slice instance selection** *is based on* **policies**
* **Slice instance selection** *is based on* **UE subscription data**

**Factoid:** SEPP enforces inter-PLMN security by encrypting control-plane messages using TLS/IPsec tunnels, while hiding internal NF topology.

* **SEPP** *enforces* **inter-PLMN security**
* **SliceSEPP** *encrypts* **control-plane messages**
* **SEPP** *uses* **TLS/IPsec tunnels**
* **SEPP** *hides* **internal NF topology**

**Factoid:** N3IWF provides secure IPsec tunnels over Wi-Fi, allowing non-3GPP devices to attach directly to the 5G core via AMF.

* **N3IWF** *provides* **secure IPsec tunnels**
* **Secure IPsec tunnels** *operate over* **Wi-Fi**
* **N3IWF** *allows* **non-3GPP devices** to attach directly to **5G core**
* **AMF** *facilitates* **non-3GPP device attachment to 5G core**

**Factoid:** SBA enables all control-plane NFs to expose RESTful APIs over HTTP/2, facilitating microservice-based orchestration and interop.

* **Service-Based Architecture (SBA)** *enables* **control-plane NFs** *to expose* **RESTful APIs**
* **Control-plane NFs** *expose* **RESTful APIs** *over* **HTTP/2**
* **Exposing RESTful APIs over HTTP/2** *facilitates* **microservice-based orchestration**
* **Exposing RESTful APIs over HTTP/2** *facilitates* **interoperability**

**Factoid:** The choice between stateful and stateless NF design impacts how context is managed, with stateless NFs enabling container-based scaling.

* **Stateful NF design** *impacts* **context management**
* **Stateless NF design** *impacts* **context management**
* **Stateless NF design** *enables* **container-based scaling**

**Factoid:** Stateless core NFs like AMF and SMF require external state repositories (e.g., UDSF, UDR) to restore context during horizontal scaling or failure recovery.

* **Stateless core NFs** *require* **external state repositories**
* **AMF** *requires* **external state repositories**
* **SMF** *requires* **external state repositories**
* **External state repositories** *include* **UDSF**
* **External state repositories** *include* **UDR**
* **External state repositories** *restore* **context during horizontal scaling**
* **External state repositories** *restore* **context during failure recovery**

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

* **NRF** *is* **stateless**
* **NSSF** *is* **stateless**
* **AMF** *is* **stateful**
* **AMF** *stores* **UE context in UDSF**
* **SMF** *is* **stateful**
* **SMF** *stores* **UE context in UDSF**

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

* **UDSF** *functions as* **shared key-value store**
* **Shared key-value store** *holds* **UE session context**
* **Shared key-value store** *holds* **security credentials**

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

* **Piggyback-based retrieval** *retrieves* **all required UE context at procedure start**
* **Piggyback-based retrieval** *eliminates* **redundant external requests**

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

* **Piggyback** *reduces* **synchronous procedure latency by 44–70 %**

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

* **Proactive-push** *preloads* **downstream NF state** at **procedure initiation**
* **Proactive-push** *cuts* **asynchronous latency by ≈ 13–22 %**

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

* **Hybrid pattern** *combines* **piggyback** and **proactive-push**
* **Hybrid pattern** *optimizes* **latency across synchronous procedures**
* **Hybrid pattern** *optimizes* **latency across asynchronous procedures**
* **Hybrid pattern** *incurs* **no extra overhead**

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

* **Container-based stateless deployment** *enables* **robust horizontal scaling**
* **Container-based stateless deployment** *enables* **failure resilience**

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

* **Stateless architecture** *maintains* **3GPP compliance**
* **Stateless architecture** *preserves* **standard SBI interfaces**
* **Stateless architecture** *requires only* **new UDSF endpoints**

**Factoid:** Stateful NFs like AMF and SMF can offload internal memory to UDSF to become externally state-managed without altering logical behavior.

* **AMF** *offloads* **internal memory to UDSF**
* **SMF** *offloads* **internal memory to UDSF**
* **Offloading internal memory to UDSF** *enables* **externally state-managed operation**
* **Externally state-managed AMF and SMF** *do not alter* **logical behavior**

**Factoid:** UDSF is optimized for UE-specific metadata such as SUPI, session state, and NAS keys, supporting fine-grained key-value access.

* **UDSF** *is optimized for* **UE-specific metadata**
* **UE-specific metadata** *includes* **SUPI**
* **UE-specific metadata** *includes* **session state**
* **UE-specific metadata** *includes* **NAS keys**
* **UDSF** *supports* **fine-grained key-value access**

**Factoid:** Proactive state push allows AMF to transmit UE context to dependent NFs *before* they are involved in a procedure, enabling smoother control plane flow.

* **Proactive state push** *allows* **AMF** to transmit **UE context**
* **AMF** *transmits* **UE context** to **dependent NFs**
* **Transmitting UE context to dependent NFs before procedure involvement** *enables* **smoother control-plane flow**

**Factoid:** Piggyback and proactive strategies are not mutually exclusive—hybrid use improves end-to-end responsiveness under varied traffic patterns.

* **Piggyback strategies** *are not mutually exclusive with* **proactive strategies**
* **Hybrid use** *combines* **piggyback strategies** and **proactive strategies**
* **Hybrid use** *improves* **end-to-end responsiveness**
* **End-to-end responsiveness** *is improved under* **varied traffic patterns**

**Factoid:** Stateless NF deployment reduces the operational burden of state synchronization across instances during autoscaling or rolling updates.

* **Stateless NF deployment** *reduces* **operational burden of state synchronization across instances**
* **Operational burden of state synchronization** *occurs during* **autoscaling**
* **Operational burden of state synchronization** *occurs during* **rolling updates**

**Factoid:** Kubernetes-based control plane orchestration benefits from stateless NF design by enabling auto-replacement of crashed pods without state loss.

* **Kubernetes-based control plane orchestration** *benefits from* **stateless NF design**
* **Stateless NF design** *enables* **auto-replacement of crashed pods**
* **Auto-replacement of crashed pods** *occurs without* **state loss**

**Factoid:** No changes to 3GPP signaling protocols (e.g., NAS, NGAP) are required to adopt stateless control—only the UDSF interface and optional push hooks are added.

* **Stateless control** *requires no changes to* **3GPP signaling protocols (NAS, NGAP)**
* **Stateless control** *adds* **UDSF interface**
* **Stateless control** *adds* **optional push hooks**

**Factoid:** UDSF enables centralized or distributed state storage strategies depending on network topology, aiding deployment in edge or cloud regions.

* **UDSF** *enables* **centralized state storage strategies**
* **UDSF** *enables* **distributed state storage strategies**
* **Centralized state storage strategies** *depend on* **network topology**
* **Distributed state storage strategies** *depend on* **network topology**
* **Centralized state storage strategies** *aid* **deployment in edge regions**
* **Centralized state storage strategies** *aid* **deployment in cloud regions**
* **Distributed state storage strategies** *aid* **deployment in edge regions**
* **Distributed state storage strategies** *aid* **deployment in cloud regions**

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

* **Stateless NF instances** *exhibit* **lower average CPU usage than stateful NF instances**
* **Lower average CPU usage** *is due to* **wait periods during external state retrieval**
* **Wait periods during external state retrieval** *occur in* **stateless NF instances**

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

* **Queue buildup in AMF** *propagates to* **SMF**
* **Queue buildup in AMF** *propagates to* **UPF**
* **SMF** *experiences* **CPU spikes once state responses are processed**
* **UPF** *experiences* **CPU spikes once state responses are processed**

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

* **Transactional stateless NFs** *increase* **latency**
* **Transactional stateless NFs** *increase* **cost**
* **Transactional stateless NFs** *perform* **synchronous state fetches  
  Transactional stateless NFs** *perform* **JSON deserialization**

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

* **Non-blocking stateless strategies** *improve* **latency**
* **Non-blocking stateless strategies** *improve* **throughput**
* **Non-blocking stateless strategies** *use* **asynchronous state access**

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

* **Stateless NF designs** *increase* **billing costs**
* **Stateless NF designs** *operate in* **cloud environments  
  Stateless NF designs** *cause* **longer request times**
* **Longer request times** *increase* **runtime charges**
* **Increased runtime charges** *lead to* **higher billing costs**

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

* **Global UE context** *is shared between* **NFs**
* **Sharing global UE context between NFs** *reduces* **unnecessary database operations**
* **Sharing global UE context between NFs** *improves* **performance by ≈ 33 %**

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

* **Embedding user context into NF-to-NF messages** *cuts* **DB reads from 4 × n to 2**
* **Cutting DB reads from 4 × n to 2** *reduces* **overhead by ≈ 22 %**

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

* **Stateless NFs** *allow* **container-based scaling**
* **Stateless NFs** *require* **optimized caching strategies**
* **Optimized caching strategies** *avoid* **latency penalties**
* **Optimized caching strategies** *avoid* **cost penalties**

**Factoid:** Stateless NF designs can create a misleading CPU usage profile where underutilized compute masks backend bottlenecks in state access layers.

* **Stateless NF designs** *can create* **misleading CPU usage profile**
* **Stateless NF designs** *can lead to* **underutilized compute**
* **Underutilized compute** *masks* **backend bottlenecks  
  Misleading CPU usage profile** *masks* **backend bottlenecks in state access layers**

**Factoid:** Latency spikes in stateless control chains often correlate with batched state-response arrivals, leading to temporary CPU contention across NF tiers.

* **Latency spikes** *occur in* **stateless control chains**
* **Latency spikes** *correlate with* **batched state-response arrivals**
* **Batched state-response arrivals** *lead to* **temporary CPU contention**
* **Temporary CPU contention** *occurs across* **NF tiers**

**Factoid:** Frequent state serialization and deserialization in transactional stateless models leads to inflated processing delay, especially under high concurrency.

* **Transactional stateless models** *perform* **frequent state serialization and deserialization**
* **Frequent state serialization and deserialization** *leads to* **inflated processing delay**
* **Inflated processing delay** *is exacerbated under* **high concurrency**

**Factoid:** Non-blocking stateless strategies leverage concurrency and pipelining to maintain throughput while masking state-fetch latency.

* **Non-blocking stateless strategies** *leverage* **concurrency**
* **Non-blocking stateless strategies** *leverage* **pipelining**
* **Non-blocking stateless strategies** *maintain* **throughput**
* **Non-blocking stateless strategies** *mask* **state-fetch latency**

**Factoid:** Stateless deployment models require broader horizontal scaling to achieve parity with stateful throughput, despite lower CPU use per instance.

* **Stateless deployment models** *require* **broader horizontal scaling**
* **Broader horizontal scaling** *achieves parity with* **stateful throughput**
* **Stateless deployment models** *exhibit* **lower CPU use per instance**

**Factoid:** Cloud-native stateless NFs must be capacity-planned for bursty workloads, as idle periods followed by sudden spikes are common in real-world flows.

* **Cloud-native stateless NFs** *must be capacity-planned for* **bursty workloads**
* **Bursty workloads** *include* **idle periods**
* **Bursty workloads** *include* **sudden spikes**
* **Idle periods** *are followed by* **sudden spikes**
* **Idle periods** *are common in* **real-world flows**
* **Sudden spikes** *are common in* **real-world flows**

**Factoid:** State-sharing optimizations across service chains can significantly cut control-plane I/O without sacrificing NF modularity.

* **State-sharing optimizations** *occur across* **service chains**
* **State-sharing optimizations** *significantly cut* **control-plane I/O**
* **State-sharing optimizations** *maintain* **NF modularity**

**Factoid:** Embedding minimal but sufficient session metadata in upstream NF messages helps reduce cumulative state-store interactions across the chain.

* **Upstream NF messages** *embed* **minimal session metadata**
* **Embedding minimal session metadata in upstream NF messages** *reduces* **cumulative state-store interactions across the chain**
* **Cumulative state-store interactions** *occur across* **the service chain**

**Factoid:** Persistent caching layers or sidecars that prefetch or retain context between NF invocations offer critical improvements in latency and cost-efficiency.

* **Persistent caching layers** *prefetch* **context between NF invocations**
* **Persistent caching layers** *retain* **context between NF invocations**
* **Sidecars** *prefetch* **context between NF invocations**
* **Sidecars** *retain* **context between NF invocations**
* **Prefetching or retaining context between NF invocations** *offers* **critical improvements in latency**
* **Prefetching or retaining context between NF invocations** *offers* **critical improvements in cost-efficiency**

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

* **Quasi-local model** *uses* **fetch-and-cache strategy**
* **Fetch-and-cache strategy** *stores* **UE state locally**
* **UE state** *is stored after* **initial procedures**

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

* **Decoupling control compute from storage** *enables* **fast NF instance recovery  
  Fast NF instance recovery** *occurs* **mid-session**

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

* **Per-procedure caching** *optimizes* **storage accesses**
* **Per-procedure caching** *controls* **DB IO per NF**

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

* **Quasi-local cache** *supports* **latency-critical use cases**
* **Latency-critical use cases** *include* **V2X**
* **Latency-critical use cases** *include* **telesurgery**

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

* **State metrics** *include* **volume**
* **State metrics** *include* **size**
* **State metrics** *include* **frequency of operations**
* **State metrics** *help dimension* **datastore loads**

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

* **Quasi-local caching** *reduces* **network-wide datastore interactions**
* **Network-wide datastore interactions** *occur during* **user-plane processing**

**Factoid:** The quasi-local model maintains a session-scoped cache that reduces repeated external lookups during procedures like registration and handover.

* **Quasi-local model** *maintains* **session-scoped cache**
* **Session-scoped cache** *reduces* **repeated external lookups**
* **Repeated external lookups** *occur during* **registration procedures**
* **Repeated external lookups** *occur during* **handover procedures**

**Factoid:** Caching state at well-defined checkpoints ensures that NFs have immediate access to necessary context during bursty signaling events.

* **Caching state at well-defined checkpoints** *ensures* **immediate access to necessary context**
* **Immediate access to necessary context** *is provided to* **NFs during bursty signaling events**

**Factoid:** Separating compute from persistent state allows newly spawned NF instances to rejoin active sessions without full protocol reinitialization.

* **Separation of compute from persistent state** *allows* **newly spawned NF instances to rejoin active sessions**
* **Separation of compute from persistent state** *eliminates* **full protocol reinitialization**

**Factoid:** Mid-session recovery is enabled by fetching the most recent session snapshot from a distributed state store, avoiding session resets.

* **Mid-session recovery** *is enabled by* **fetching the most recent session snapshot**
* **Most recent session snapshot** *is stored in* **distributed state store**
* **Fetching the most recent session snapshot** *avoids* **session resets**

**Factoid:** Auto-persistence strategies allow caching to be dynamically tailored per procedure type, optimizing both performance and memory footprint.

* **Auto-persistence strategies** *allow* **caching to be dynamically tailored per procedure type**
* **Auto-persistence strategies** *optimize* **performance**
* **Auto-persistence strategies** *optimize* **memory footprint**

**Factoid:** Procedure-aware caching models help NFs selectively retain short-lived versus long-lived state, balancing responsiveness with memory efficiency.

* **Procedure-aware caching models** *help* **NFs selectively retain short-lived state**
* **Procedure-aware caching models** *help* **NFs selectively retain long-lived state**
* **Selective retention of short-lived state** *balances* **responsiveness**
* **Selective retention of long-lived state** *balances* **memory efficiency**

**Factoid:** The fetch‑and‑cache model minimizes control-plane latency variance by reducing dependency on synchronous state-store reads during critical flows.

* **Fetch-and-cache model** *minimizes* **control-plane latency variance**
* **Fetch-and-cache model** *reduces dependency on* **synchronous state-store reads**
* **Dependency on synchronous state-store reads** *occurs during* **critical flows**

**Factoid:** Tailored cache lifetimes and eviction policies allow NFs to meet reliability requirements across diverse traffic types, including eMBB and IoT.

* **Tailored cache lifetimes** *allow* **NFs to meet reliability requirements**
* **Eviction policies** *allow* **NFs to meet reliability requirements**
* **Reliability requirements** *apply across* **diverse traffic types**
* **Diverse traffic types** *include* **eMBB**
* **Diverse traffic types** *include* **IoT**

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

* **N1 interface** *is* **control-plane interface**
* **N1 interface** *carries* **NAS signaling  
  NAS signaling** *occurs between* **UE** and **AMF**
* **N1 interface** *operates across* **gNB**
* **N1 interface** *operates across* **access network**

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

* **gNB** *transparently forwards* **N1 NAS messages**
* **gNB** *does not process* **N1 NAS messages**

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

* **N1 interface** *uses* **RRC**
* **RRC** *runs over* **Uu**
* **N1 interface** *uses* **NGAP**
* **NGAP** *runs over* **N2**
* **N1 interface** *operates between* **UE and AMF**

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

* **N1 interface** *includes* **IPsec tunnel (NWu)**
* **N1 interface** *uses* **IPsec tunnel (NWu)** in **non-3GPP access**
* **IPsec tunnel (NWu)** *links* **UE** and **AMF** securely
* **IPsec tunnel (NWu)** *is used in* **non-3GPP access**

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

* **N1 mode** *denotes* **Standalone (SA) deployment**
* **Standalone (SA) deployment** *connects* **UE** to **5G Core**
* **S1 mode** *uses* **Non-Standalone (NSA) deployment**
* **NSA deployment** *operates via* **4G**

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

* **N1 interface** *carries* **NAS signaling**
* **NAS signaling** *is carried transparently via* **gNB**
* **N1 interface** *uses* **RRC** *for* **core-plane UE–AMF communication**
* **N1 interface** *uses* **NGAP** *for* **core-plane UE–AMF communication**

**Factoid**: 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.

* **Control-plane interfaces** *use* **HTTP/2 REST**
* **N2** *is a* **Control-plane interface**
* **N11** *is a* **Control-plane interface**
* **N12** *is a* **Control-plane interface**
* **N14** *is a* **Control-plane interface**
* **N15** *is a* **Control-plane interface**
* **HTTP/2 REST** *runs over* **Service-Based Architecture (SBA)**
* **Data-plane interfaces** *use* **UDP-based protocols**
* **N3** *is a* **Data-plane interface**
* **N6** *is a* **Data-plane interface**
* **N9** *is a* **Data-plane interface**
* **N15 is a Control-plane interface**

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

* **N4 interface** *connects* **SMF** and **UPF**
* **N4 interface** *employs* **PFCP**
* **PFCP** *manages* **user-plane sessions**
* **PFCP** *manages* **forwarding behavior**

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

* **Inter-NF service calls** *enable* **SMB workload routing**
* **Inter-NF service calls** *allow* **dynamic policy management**
* **Inter-NF service calls** *allow* **dynamic data flow management**
* **N7 interface** *is an* **Inter-NF service call**
* **N8 interface** *is an* **Inter-NF service call**
* **N10 interface** *is an* **Inter-NF service call**

**\_\_\_\_\_\_\_ BATCH 1 TO GEMINI \_\_\_\_\_\_\_**

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

.

* **Roaming-related interfaces** *facilitate* **control-plane continuity across administrative domains**
* **N16 interface** *is a* **roaming-related interface**
* **N27 interface** *is a* **roaming-related interface**
* **N32 interface** *is a* **roaming-related interface**
* **Control-plane continuity** *occurs across* **administrative domains**

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

* **NSSF** *selects* **network slices**
* **NSSF** *uses* **N22 interface**
* **Slice-specific NF association** *occurs after* **UE–AMF registration**

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

* **N9 interface** *supports* **cascading of UPFs**
* **Cascading of UPFs** *enables* **scaling**
* **Cascading of UPFs** *enables* **multi-hop data-plane routing**

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

* **Interface structure** *supports* **cloud-native scaling**
* **Interface structure** *provides* **clear separation**
* **Interface structure** *provides* **SLA isolation**

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

* **Security features across interfaces** *include* **TLS for HTTP/2 SBA**
* **Security features across interfaces** *include* **IPsec tunneling for non-3GPP access paths**
* **Security features across interfaces** *include* **IPsec tunneling for RAN access paths**

**Factoid:** 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.

* **N1** *carries* **NAS signaling**
* **NAS signaling** *flows between* **UE** and **AMF**
* **NAS signaling** *is transparent through* **gNB**
* **N1** *forms* **control link for UE registration and session setup**

**Factoid:** N2 connects gNB to AMF using NGAP over SCTP and enables the transmission of access signaling and mobility control messages.

* **N2** *connects* **gNB** to **AMF**
* **N2** *uses* **NGAP over SCTP**
* **N2** *enables* **transmission of access signaling**
* **N2** *enables* **transmission of mobility control messages**

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

* **N3** *transports* **user-plane traffic** between **gNB** and **UPF**
* **N3** *uses* **GTP-U** over **UDP**
* **Transporting user-plane traffic via N3 using GTP-U over UDP** *supports* **high-throughput PDU sessions**

**Factoid:** N4 allows the SMF to control UPF behavior using PFCP, setting up and modifying forwarding rules and QoS enforcement.

* **N4** *allows* **SMF** to **control UPF behavior**
* **SMF** *uses* **PFCP** to **set up forwarding rules**
* **SMF** *uses* **PFCP** to **modify forwarding rules**
* **PFCP** *enables* **QoS enforcement**

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

* **N5 interface** *enables* **application functions (AF)**
* **Application functions (AF)** *communicate* **policy triggers** to **PCF**
* **Communication of policy triggers to PCF** *occurs via* **HTTP/2**
* **Policy triggers** *support* **content-aware optimization**

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

* **N6** *links* **UPF** to **external data networks (DN)** *over* **IP**
* **N6** *handles* **routing** for **user-plane packets**
* **N6** *handles* **NAT** for **user-plane packets**
* **N6** *handles* **service exposure** for **user-plane packets**

**Factoid:** N7 interface is used by the SMF to retrieve policy rules and QoS profiles from the PCF using RESTful HTTP APIs.

* **N7 interface** *is used by* **SMF**
* **SMF** *retrieves* **policy rules**
* **SMF** *retrieves* **QoS profiles**
* **Retrieval of policy rules and QoS profiles** *occurs via* **N7 interface**
* **N7 interface** *uses* **RESTful HTTP APIs**

**Factoid:** N8 provides the AMF access to subscriber and authentication data by querying the UDM over HTTP/2.

* **N8 interface** *provides* **AMF** access to **subscriber data**
* **N8 interface** *provides* **AMF** access to **authentication data**
* **AMF access to subscriber and authentication data** *occurs by querying* **UDM**
* **Querying UDM** *uses* **HTTP/2**

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

* **N9 interface** *enables* **inter-UPF communication**
* **N9 interface** *uses* **GTP-U**
* **Inter-UPF communication** *supports* **distributed user-plane routing**
* **Inter-UPF communication** *supports* **load balancing**
* **Inter-UPF communication** *supports* **service chaining**

**Factoid:** N10 allows SMF to obtain subscriber policy data from the UDM, including PDU session authorization and QoS configuration.

* **N10 interface** *allows* **SMF** to **obtain subscriber policy data**
* **SMF** *obtains* **subscriber policy data** from **UDM**
* **Subscriber policy data** *includes* **PDU session authorization**
* **Subscriber policy data** *includes* **QoS configuration**

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

* **N11 interface** *connects* **AMF** and **SMF**
* **N11 interface** *enables* **control-plane interactions**
* **Control-plane interactions** *include* **PDU session setup**
* **Control-plane interactions** *include* **mobility handling**

**Factoid:** N12 allows the AMF to forward authentication requests to the AUSF during UE registration or security mode command procedures.

* **N12 interface** *allows* **AMF** to forward **authentication requests** to **AUSF**
* **AMF** *forwards* **authentication requests** to **AUSF**
* **Forwarding authentication requests to AUSF** *occurs during* **UE registration**
* **Forwarding authentication requests to AUSF** *occurs during* **security mode command procedures**

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

* **N13 interface** *connects* **AUSF** and **UDM**
* **AUSF** *obtains* **authentication vectors** from **UDM**
* **Authentication vectors** *include* **AV**
* **Authentication vectors** *include* **EAP**
* **Authentication vectors** *are used for* **5G-AKA procedures**

**Factoid:** N14 supports AMF-to-AMF communication for inter-region handover, UE context transfer, and mobility continuity.

* **N14** *supports* **AMF-to-AMF communication**
* **AMF-to-AMF communication** *enables* **inter-region handover**
* **AMF-to-AMF communication** *enables* **UE context transfer**
* **AMF-to-AMF communication** *enables* **mobility continuity**

**Factoid:** N15 is used by the PCF to supply policy rules directly to the AMF, influencing access and mobility behaviors.

* **N15 interface** *is used by* **PCF**
* **PCF** *supplies* **policy rules** to **AMF**
* **Supplying policy rules to AMF** *influences* **access behaviors**
* **Supplying policy rules to AMF** *influences* **mobility behaviors**

**Factoid:** N16 links two SMFs (typically in roaming scenarios) to coordinate session continuity and policy across PLMN boundaries.

* **N16** *links* **two SMFs**
* **Linking two SMFs** *occurs in* **roaming scenarios**
* **N16** *coordinates* **session continuity across PLMN boundaries**
* **N16** *coordinates* **policy across PLMN boundaries**

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

* **N1 interface** *carries* **NAS signaling** between **UE** and **AMF**
* **N1 interface** *is used for* **registration procedures**
* **N1 interface** *is used for* **authentication procedures**
* **N1 interface** *is used for* **mobility procedures**

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

* **N1 signaling** *is* **bidirectional**
* **N1 signaling** *is encapsulated in* **RRC at the UE side**
* **RRC encapsulation of N1 signaling** *occurs at* **UE side**
* **N1 signaling** *is transported via* **NGAP through gNB to AMF**
* **NGAP through gNB** *transports* **N1 signaling** to **AMF**

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

* **N2** *connects* **gNB** and **AMF**
* **N2** *uses* **NGAP over SCTP**
* **NGAP over SCTP** *handles* **UE context**
* **NGAP over SCTP** *handles* **handover signaling**

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

* **N2** *transmits messages such as* **Initial UE Message**
* **N2** *transmits messages such as* **UE Context Release**
* **Initial UE Message** *is transmitted between* **RAN** and **Core**
* **UE Context Release** *is transmitted between* **RAN** and **Core**

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

* **N3** *uses* **GTP-U over UDP**
* **N3** *forwards* **user-plane traffic between gNB and UPF**
* **Forwarding user-plane traffic between gNB and UPF** *occurs after* **PDU session establishment**

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

* **N3** *is unidirectional for* **uplink data flow**
* **N3** *is unidirectional for* **downlink data flow**
* **N3** *is not involved in* **control signaling**

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

* **N4** *is* **control interface between SMF and UPF**
* **N4** *enables* **session creation**
* **N4** *enables* **QoS enforcement**
* **N4** *enables* **traffic routing via PFCP**

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

* **N4** *manages* **forwarding rules**
* **N4** *uses* **PFCP message FAR**
* **N4** *uses* **PFCP message QER**
* **N4** *uses* **PFCP message BAR**
* **Managing forwarding rules through FAR, QER, and BAR** *enables* **dynamic UPF behavior control**

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

* **5G** *supports* **simultaneous NAS sessions over cellular and Wi-Fi**
* **Simultaneous NAS sessions over cellular and Wi-Fi** *are established by* **multiple N1 control-plane connections**

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

* **UEs** *authenticate to* **5G core** *over* **Wi-Fi**
* **Authentication to 5G core over Wi-Fi** *uses* **EAP-AKA′**
* **Authentication to 5G core over Wi-Fi** *uses* **5G-AKA**
* **Authentication using EAP-AKA′ or 5G-AKA** *occurs prior to* **NAS signaling**

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

* **IPsec tunnels (NWu/NWt)** *use* **IKEv2**
* **IPsec tunnels (NWu/NWt)** *use* **EAP-5G**
* **IPsec tunnels (NWu/NWt)** *secure* **NAS traffic to N3IWF gateways**
* **IPsec tunnels (NWu/NWt)** *secure* **NAS traffic to TNGF gateways**

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

* **N1 over trusted Wi-Fi** *uses* **IPsec with NULL encryption**
* **IPsec with NULL encryption** *avoids* **double encryption**
* **IPsec with NULL encryption** *preserves* **link-level security**

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

* **N2 control-plane messages** *occur over* **non-3GPP access**
* **N2 control-plane messages** *use* **NGAP/SCTP**
* **NGAP/SCTP** *connects* **gateway functions** and **AMF**

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

* **TWIF** *enables* **NAS signaling for legacy Wi-Fi devices lacking EAP-5G support**
* **TWIF** *acts as* **NAS proxy**

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

* **N3 user-plane traffic between Wi-Fi gateways and UPF** *uses* **GTP-U over UDP**
* **N3 user-plane traffic between Wi-Fi gateways and UPF** *mirrors* **cellular data flow**

**Factoid**: Service-Based Architecture (SBA) exists only between 5GC control-plane NFs via HTTP/2 REST APIs.

* **Service-Based Architecture (SBA)** *exists only between* **5GC control-plane NFs**
* **Service-Based Architecture (SBA)** *operates via* **HTTP/2 REST APIs**

**Factoid**: Interfaces N1, N2, N3, N4, N6, and N9 are executed outside the SBA domain using traditional control/user-plane protocols.

* **Interfaces N1, N2, N3, N4, N6, and N9** *are executed outside* **SBA domain**
* **Interfaces N1, N2, N3, N4, N6, and N9** *use* **traditional control/user-plane protocols**

**Factoid**: AMF and SMF represent separate functional domains in control-plane, enabling independent scaling and specialization.

* **AMF** *represents* **a separate functional domain in the control-plane**
* **SMF** *represents* **a separate functional domain in the control-plane**
* **Separate functional domains in the control-plane** *enable* **independent scaling**
* **Separate functional domains in the control-plane** *enable* **specialization**

**Factoid**: CUPS design separates control and user gateways for flexible placement and scalability in the 5G core.

* **CUPS design** *separates* **control gateways** and **user gateways**
* **CUPS design** *enables* **flexible placement in the 5G core**
* **CUPS design** *enables* **scalability in the 5G core**

**Factoid**: SBA-capable NFs are cloud-native microservices, orchestrated via container platforms with CI/CD pipelines.

* **SBA-capable NFs** *are* **cloud-native microservices**
* **SBA-capable NFs** *are orchestrated via* **container platforms**
* **Container platforms** *use* **CI/CD pipelines**

**Factoid**: Control-plane APIs (e.g., Nsmf, Namf) are secured via TLS, whereas user-plane protocols (e.g., GTP‑U, PFCP) use UDP/SCTP.

* **Control-plane APIs** *include* **Nsmf**
* **Control-plane APIs** *include* **Namf**
* **Control-plane APIs** *are secured via* **TLS  
  User-plane protocols** *include* **GTP-U**
* **User-plane protocols** *include* **PFCP**
* **User-plane protocols** *use* **UDP/SCTP**

**Factoid**: SBA’s RESTful interfaces cannot directly carry UE or user-data-bound control messages (e.g., RRC or NAS).

* **SBA’s RESTful interfaces** *cannot directly carry* **UE control messages**
* **SBA’s RESTful interfaces** *cannot directly carry* **user-data-bound control messages**
* **UE control messages** *include* **RRC**
* **User-data-bound control messages** *include* **NAS**

**Factoid**: In CUPS, user-plane functions like UPF scale and deploy separately from their control-plane counterparts (SMF, PCF).

* **UPF** *scales separately from* **SMF**
* **UPF** *deploys separately from* **SMF**
* **UPF** *scales separately from* **PCF**
* **UPF** *deploys separately from* **PCF**

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

* **Open5GS NF** *is configured via* **its own YAML file in /etc/open5gs/**
* **YAML file in /etc/open5gs/** *specifies* **protocol ports**
* **YAML file in /etc/open5gs/** *specifies* **PLMN/TAC**
* **YAML file in /etc/open5gs/** *specifies* **features**

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

* **Changing bind addresses in NF YAML** *requires* **consistent updates across RAN configurations**
* **Consistent updates across RAN configurations** *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.

* **AMF** *listens on* **NGAP SCTP port 38412**
* **UPF** *listens for* **GTP-U on PFCP-assigned port**
* **Port listening events** *are reflected in* **NF logs**

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

* **ngap\_server() log entry** *confirms* **NF’s protocol stack initialization**
* **ngap\_server() log entry** *confirms* **port bindings**
* **gtp\_server() log entry** *confirms* **NF’s protocol stack initialization**
* **gtp\_server() log entry** *confirms* **port bindings**

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

* **SCTP INIT/ABORT messages** *occur in* **logs**
* **SCTP INIT/ABORT messages** *reveal* **SCTP handshake status**
* **SCTP handshake status** *is between* **gNB and AMF**

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

* **Core setup** *creates* **ogstun TUN interface**
* **ogstun TUN interface** *provides* **IPv4 subnets**
* **ogstun TUN interface** *provides* **IPv6 subnets**
* **ogstun TUN interface** *supports* **UPF operations**

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

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

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

* **Firewall rules** *enforce* **subnet isolation**
* **Subnet isolation** *applies to* **10.45.0.0/16**
* **Subnet isolation** *blocks* **unauthorized access to NF services**

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

* **Open5GS NFs** *run as* **systemd services**
* **systemd services** *can be stopped or disabled* **individually**
* **open5gs-amfd** *is an example of* **a systemd service for an Open5GS NF**

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

* **Running only subsets of NFs** *is supported by* **stopping irrelevant services**
* **Running only subsets of NFs** *is supported by* **editing YAML configs accordingly**

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

* **NF bind addresses** *must align with* **core network settings**
* **NF bind addresses** *must align with* **RAN network settings**
* **Kubernetes service IPs** *must align with* **core network settings**
* **Kubernetes service IPs** *must align with* **RAN network settings**
* **Port mappings** *must align with* **core network settings**
* **Port mappings** *must align with* **RAN network settings**

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

* **Open5GS** *uses* **Docker manifests**
* **Docker manifests** *configure* **upfPublicIP**
* **Docker manifests** *configure* **amfif.ip/port**
* **Configuring upfPublicIP** *binds* **container network to host services**
* **Configuring amfif.ip/port** *binds* **container network to host services**

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

* **Open5GS** *uses* **MongoDB**
* **MongoDB** *stores* **stateful NF data**
* **Stateful NF data** *includes* **NRF**
* **Stateful NF data** *includes* **PCF**
* **Stateful NF data** *includes* **UDR**
* **MongoDB** *connects using* **mongodb://localhost/open5gs**

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

* **Open5GS PCF** *includes* **dbi configuration**
* **dbi configuration** *points to* **MongoDB URI**
* **dbi configuration** *supports* **service discovery**
* **dbi configuration** *supports* **policy control**

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

* **AMF** *can expose* **Prometheus metrics via an HTTP server**
* **PCF** *can expose* **Prometheus metrics via an HTTP server**
* **SMF** *can expose* **Prometheus metrics via an HTTP server**
* **UPF** *can expose* **Prometheus metrics via an HTTP server**
* **Prometheus metrics** *are exposed via* **an HTTP server**
* **HTTP server** *is configured under* **metrics:**
* **metrics:** *is defined in* **the NF’s YAML file**

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

* **Metrics endpoints** *are individually defined per* **NF instance**
* **NF instances** *include* **open5gs-amfd**, **open5gs-smfd1**, **open5gs-upfd2**
* **Metrics endpoints** *have* **distinct IP and port**

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

* **Prometheus** *scrapes* **/metrics endpoints** *every 10 seconds*
* **Prometheus** *uses* **job definitions** *in* **prometheus.yml**
* **Job definitions** *match* **NF names**

**\_\_\_\_\_\_ BATCH 2 GEMINI \_\_\_\_\_\_**

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

* **Grafana** *connects to* **Prometheus as a data source**
* **Grafana** *visualizes* **NF-specific metrics**
* **NF-specific metrics** *include* **ues\_active**
* **NF-specific metrics** *include* **amf\_session**

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

* **Multiple SMF and UPF instances** *can be monitored* **in parallel**
* **Parallel monitoring** *supports* **slice-specific deployments**
* **Parallel monitoring** *supports* **region-specific deployments**

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

* **Open5GS’s build process** *includes* **libogsmetrics**
* **libogsmetrics** *compiles* **libprom support for metrics output**
* **libogsmetrics** *compiles* **libmicrohttpd support for metrics output**

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

* **Example metrics** *include* **counters for NAS registration requests (rm\_reginitreq)**
* **Example metrics** *include* **memory/resource usage (process\_resident\_memory\_bytes)**

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

* **Distinct job\_name entries in Prometheus** *enable* **selective scraping**
* **Distinct job\_name entries in Prometheus** *enable* **dashboarding per NF type**

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

* **Sample config** *binds* **metrics HTTP servers** to **host IPs**
* **Binding metrics HTTP servers to host IPs** *facilitates* **external observability**
* **External observability** *applies to* **containerized NFs**

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

* **Metrics integration** *demonstrates* **Open5GS operating as 'Prometheus-enabled' microservices**
* **Open5GS** *operates as* **'Prometheus-enabled' microservices**
* **'Prometheus-enabled' microservices** *run in* **Kubernetes environments**
* **'Prometheus-enabled' microservices** *run in* **Docker environments**

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

* **Open5GS** *is installed via* **Ubuntu PPA**
* **Open5GS** *requires* **MongoDB**
* **MongoDB** *stores* **subscriber context**

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

* **Hardware with Intel i5** *is preferred for* **MongoDB compatibility**
* **Hardware with Intel i5** *is preferred 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.

* **Default Open5GS configs** *use* **loopback IPs**
* **Integration with external RAN simulators** *requires* **replacing loopback addresses in NF YAML**
* **Replacing loopback addresses in NF YAML** *uses* **host LAN IPs**

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

* **PLMN ID** *must match* **Open5GS core configuration**
* **PLMN ID** *must match* **UERANSIM/gNB RAN configuration**
* **TAC** *must match* **Open5GS core configuration**
* **TAC** *must match* **UERANSIM/gNB RAN configuration**
* **Matching PLMN ID and TAC across configurations** *ensures* **successful connectivity**

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

* **UERANSIM open5gs-gnb.yaml** *must specify* **linkIp**
* **UERANSIM open5gs-gnb.yaml** *must specify* **ngapIp**
* **UERANSIM open5gs-gnb.yaml** *must specify* **gtpIp**
* **UERANSIM open5gs-gnb.yaml** *must specify* **core AMF address**
* **Specifying linkIp, ngapIp, gtpIp, and core AMF address** *enables* **N2 connectivity**
* **Specifying linkIp, ngapIp, gtpIp, and core AMF address** *enables* **N3 connectivity**

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

* **UE** *uses* **gnbSearchList**
* **gnbSearchList** *is defined in* **UERANSIM config**
* **gnbSearchList** *locates* **gNB IP**
* **gNB IP** *is used for* **network attachment**

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

* **UE** *forms* **TUN interface (e.g., uesimtun0)**
* **TUN interface (uesimtun0)** *receives* **IP routes**
* **IP routes** *are delivered via* **UPF**
* **PDU session setup** *triggers* **UE to form TUN interface**

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

* **UE external Internet access** *can be validated using* **curl**
* **UE external Internet access** *can be validated using* **ping**
* **curl** *operates over* **TUN interface**
* **ping** *operates over* **TUN interface**

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

* **Test environments** *employ* **TCP proxies**
* **TCP proxies** *operate on* **public IPs**
* **TCP proxies on public IPs** *support* **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.

* **Open5GS core services** *integrate with* **RAN simulators**
* **Integration** *occurs through* **manual IP alignment**
* **Manual IP alignment** *is done in* **core YAML configurations**
* **Manual IP alignment** *is done in* **RAN YAML configurations**

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

* **Open5GS AMF** *binds* **NGAP/SCTP to loopback by default**
* **Open5GS AMF** *must set* **ngap.addr in amf.yaml to LAN IP  
  Setting ngap.addr in amf.yaml to LAN IP** *enables* **RAN connectivity**

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

* **N2 interface** *uses* **NGAP over SCTP**
* **N2 interface** *delivers* **UE NAS signaling from gNB to AMF**
* **N2 interface** *delivers* **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.

* **Multi-host setups** *involve* **Core on separate servers**
* **Multi-host setups** *involve* **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.

* **sudo systemctl restart open5gs-amfd** *applies* **ngap.addr binding change**
* **ngap.addr binding change** *affects* **AMF service**

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

* **UE RAN simulation with UERANSIM** *requires* **libsctp-dev**
* **UE RAN simulation with UERANSIM** *requires* **CMake**
* **libsctp-dev** *provides* **SCTP libraries**
* **CMake** *is used for* **building the gNB module**
* **gNB module** *is part of* **UERANSIM**

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

* **Proper AMF binding** *allows* **remote UERANSIM-created gNB** *to successfully attach*
* **Remote UERANSIM-created gNB** *exchanges* **NGAP over SCTP**

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

* **N2 SCTP handshake logs** *confirm* **successful gNB-AMF connectivity**
* **Successful gNB-AMF connectivity** *validates* **multi-host network setup**

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

* **Open5GS metrics system** *exports* **performance counters**
* **Open5GS metrics system** *exports* **gauges**
* **Performance counters** *monitor* **active PDP contexts**
* **Gauges** *monitor* **NF activity**

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

* **Metrics in Open5GS** *are implemented via* **libprom**
* **Metrics in Open5GS** *are implemented via* **libpromhttp**
* **libprom** *is built on top of* **libmicrohttpd**
* **libpromhttp** *is built on top of* **libmicrohttpd**
* **Building on top of libmicrohttpd** *enables* **embedded HTTP server**

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

* **Metrics subsystem** *uses* **generic API in lib/metrics/  
  Generic API in lib/metrics/** *supports* **conditional Prometheus backend**
* **Generic API in lib/metrics/** *supports* **no-op fallback**

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

* **Each NF (SMF, AMF, UPF, etc.)** *hosts* **its own /metrics endpoint**
* **SMF** *hosts* **its own /metrics endpoint**
* **AMF** *hosts* **its own /metrics endpoint**
* **UPF** *hosts* **its own /metrics endpoint**
* **Hosting its own /metrics endpoint** *allows* **individual scraping by Prometheus**

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

* **Metrics definitions** *include* **counters**
* **Metrics definitions** *include* **gauges**
* **Metrics definitions** *include* **histograms**
* **Counters, gauges, and histograms** *capture* **NF performance and load**

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

* **HTTP metrics server** *is embedded in* **NF process**
* **Exposing metrics** *does not require* **external exporters**

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

* **Prometheus scraping** *is enabled by building* **Open5GS with the Prometheus backend**
* **Metrics API** *is a stub without* **the Prometheus backend**

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

* **Example metrics** *include* **active session counts**
* **Example metrics** *include* **internal NF resource telemetry**

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

* **Conditional build approach** *allows* **operators** to **disable metrics support**
* **Disabling metrics support** *occurs by* **omitting libprom-related dependencies**

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

* **The architecture** *supports* **containerized deployments**
* **Containerized deployments** *allow* **each NF to be independently monitored**

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

* **CSPs** *transition from* **PNFs to CNFs**
* **Transition to CNFs** *uses* **Kubernetes**
* **Transition using Kubernetes** *enables* **vendor-agnostic infrastructure**
* **Transition using Kubernetes** *enables* **scalable infrastructure**

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

* **Disaggregated network functions (RU, DU, CU-UP, UPF)** *run at* **the edge**
* **Disaggregated network functions (RU, DU, CU-UP, UPF)** *run under* **5 ms RTT**
* **Running under 5 ms RTT** *provides* **optimal performance**

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

* **AMF** *is deployed in* **centralized cloud regions**
* **SMF** *is deployed in* **centralized cloud regions**
* **Control-plane NFs deployed in centralized cloud regions** *are* **not latency-critical**

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

* **Anthos/GDC** *provides* **unified orchestration and policy**
* **Unified orchestration and policy** *applies across* **edge deployments**
* **Unified orchestration and policy** *applies across* **private deployments**
* **Unified orchestration and policy** *applies across* **public 5G network deployments**

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

* **CNFs** *adhere to* **microservice design**
* **CNFs** *adhere to* **CI/CD lifecycle models**
* **Microservice design and CI/CD lifecycle models** *enable* **fast feature deployment**
* **Microservice design and CI/CD lifecycle models** *enable* **upgrades**

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

* **Edge deployment** *enables* **CSPs** to host **5G-CNFs** on **shared infrastructure**
* **Edge deployment** *enables* **CSPs** to host **third-party edge applications** on **shared infrastructure**
* **Third-party edge applications** *include* **AR/VR**

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

* **Hybrid clouds** *allow* **dynamic workload placement**
* **Dynamic workload placement** *assigns* **edge** to **real-time tasks**
* **Dynamic workload placement** *assigns* **cloud** to **batch tasks**
* **Dynamic workload placement** *assigns* **cloud** to **training tasks**

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

* **Telco workloads** *use* **Infrastructure-as-Code pipelines**
* **Infrastructure-as-Code pipelines** *enable* **security** across **distributed sites**
* **Infrastructure-as-Code pipelines** *enable* **scaling** across **distributed sites**
* **Infrastructure-as-Code pipelines** *enable* **orchestration** across **distributed sites**

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

* **Latency-aware deployment** *ensures* **UPF/CU-UP services are co-located near users**
* **Co-location near users** *supports* **ultra-low latency applications**

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

* **Container-based CNF approach** *enables* **feature-rich experiences**
* **Container-based CNF approach** *keeps* **costs under control**
* **Container-based CNF approach** *keeps* **TCO under control**

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

* **Telco workloads** *are migrating from* **VNFs in VMs**
* **Telco workloads** *are migrating to* **Kubernetes-managed CNFs deployed as Pods**

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

* **Kubernetes Operators** *use* **Custom Resource Definitions (CRDs)**
* **Kubernetes Operators** *support* **complex lifecycle tasks** for **CNFs**
* **Complex lifecycle tasks** *include* **upgrades**
* **Complex lifecycle tasks** *include* **scaling**
* **Custom Resource Definitions (CRDs)** *enable* **upgrades** of **CNFs**
* **Custom Resource Definitions (CRDs)** *enable* **scaling** of **CNFs**

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

* **Microservice-based CNFs** *run* **each function in separate Pods**
* **Separate Pods** *enable* **independent scaling**
* **Separate Pods** *enable* **failure isolation**

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

* **Monolithic CNFs** *simplify* **deployment**
* **Monolithic CNFs** *lack* **scalability granularity**
* **Scalability granularity** *is offered by* **microservice designs**

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

* **KNI** *combines* **Kubernetes**
* **KNI** *combines* **DPDK**
* **KNI** *combines* **SR-IOV**
* **KNI** *combines* **GPU networking**
* **Combining Kubernetes, DPDK, SR-IOV, and GPU networking** *supports* **telco-grade performance**

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

* **Istio** *secures* **APIs between control-plane CNFs in SBA**
* **Envoy** *secures* **APIs between control-plane CNFs in SBA**
* **Service mesh tools (Istio/Envoy)** *manage* **APIs between control-plane CNFs in SBA**
* **APIs between control-plane CNFs in SBA** *operate under* **Service-Based Architecture (SBA)**

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

* **Operators** *enable* **Site Reliability Engineering for CNFs**
* **Site Reliability Engineering for CNFs** *involves* **automating observability**
* **Site Reliability Engineering for CNFs** *involves* **self-healing**
* **Site Reliability Engineering for CNFs** *involves* **dynamic scaling**

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

* **KNI** *supports* **edge deployments**
* **KNI** *supports* **core deployments**
* **Edge and core deployments** *use* **unified Kubernetes control plane**
* **Unified Kubernetes control plane** *facilitates* **CNF orchestration**

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

* **5G core CNFs** *are deployed in* **multi-tier hub-and-spoke architecture**
* **Multi-tier hub-and-spoke architecture** *features* **centralized control-plane**
* **Multi-tier hub-and-spoke architecture** *features* **user-plane at edge**
* **Control-plane** *is* **centralized**
* **User-plane** *is deployed at* **edge**

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

* **UPF** *is co-located with* **RAN**
* **CU-UP** *is co-located with* **RAN**
* **AMF** *remains in* **central cloud**
* **SMF** *remains in* **central cloud**

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

* **Istio-based service mesh** *provides* **mTLS**
* **Istio-based service mesh** *provides* **secure discovery**
* **Istio-based service mesh** *provides* **tracing**
* **Istio-based service mesh** *provides* **policy enforcement**
* **mTLS**, **secure discovery**, **tracing**, and **policy enforcement** *operate across* **distributed CNFs**

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

* **Logs** *are aggregated centrally from* **edge sites**
* **Metrics** *are aggregated centrally from* **edge sites**
* **Traces** *are aggregated centrally from* **edge sites**
* **Central aggregation** *uses* **Loki**
* **Central aggregation** *provides* **full-stack observability**

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

* **Zero-touch provisioning** *enables* **on-demand scaling of 5G CNF clusters**
* **Zero-touch provisioning** *enables* **burst deployments of 5G CNF clusters**
* **GitOps** *enables* **on-demand scaling of 5G CNF clusters**
* **GitOps** *enables* **burst deployments of 5G CNF clusters**

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

* **ACM placement rules** *ensure* **CNFs are deployed to appropriate clusters**
* **CNFs deployed to appropriate clusters** *are based on* **location policies**
* **CNFs deployed to appropriate clusters** *are based on* **capacity policies**

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

* **Network fabric layers** *include* **hub management**
* **Network fabric layers** *include* **inter-cluster connectivity**
* **Network fabric layers** *include* **RAN access paths**

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

* **UPF selection** *is determined by* **SMF**
* **UPF selection** *is done via* **NRF lookup**
* **NRF lookup** *uses* **DNN metadata**
* **NRF lookup** *uses* **TAC metadata**
* **NRF lookup** *uses* **cell\_id metadata**

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

* **Partial edge deployment** *supports* **only UPF instances**
* **Fully distributed bundles** *include* **SMF CNFs in remote sites**
* **Fully distributed bundles** *include* **UPF CNFs in remote sites**

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

* **Service mesh federation** *allows* **secure cross-cluster communication**
* **Service mesh federation** *allows* **traffic splitting across hub and edge**

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

* **Observability fabric** *leverages* **centralized policy enforcement**
* **Centralized policy enforcement** *applies across* **distributed edge clusters**

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

* **Initial 5G deployments** *use* **NSA**
* **NSA** *combines* **RAN on 5G** and **control on 4G EPC**
* **Evolution to fully standalone (SA) core** *follows in* **later phases**

\_\_\_\_\_\_\_ **BATCH 3 GEMINI** \_\_\_\_\_\_\_

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

* **AWS** *supports* **stateless microservices for 5G NFs**
* **Stateless microservices for 5G NFs** *are deployed via* **containers**
* **Stateless microservices for 5G NFs** *use* **managed databases like DynamoDB, Aurora, and ElastiCache**
* **Managed databases** *include* **DynamoDB**
* **Managed databases** *include* **Aurora**
* **Managed databases** *include* **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.

* **CUPS** *is implemented using* **AWS Outposts  
  CUPS** *is implemented using* **Local Zones**
* **AWS Outposts** *deploy* **UPF at the edge**
* **Local Zones** *deploy* **UPF at the edge**
* **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.

* **Kubernetes (EKS)** *orchestrates* **CNFs on AWS**
* **CI/CD pipelines** *enable* **rapid lifecycle management**
* **CI/CD pipelines** *enable* **blue/green deployments**

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

* **Stateless design** *externalizes* **NF state** to **UDSF-like stores**
* **Externalizing NF state to UDSF-like stores** *enables* **resilience**
* **Externalizing NF state to UDSF-like stores** *enables* **container-based scalability**

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

* **Network slicing** *is orchestrated through* **AWS service mesh (App Mesh)**
* **Network slicing** *is orchestrated through* **programmable infrastructure**
* **Programmable infrastructure** *uses* **CDK**
* **Programmable infrastructure** *uses* **Step Functions**

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

* **AWS** *offers* **enhanced instances**
* **Enhanced instances** *support* **SR-IOV**
* **Enhanced instances** *support* **DPDK  
  Enhanced instances** *support* **huge pages**
* **Enhanced instances** *support* **bare-metal**
* **Enhanced instances** *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.

* **Container-based NF design on AWS** *aligns with* **‘12-factor app’ model**
* **‘12-factor app’ model** *ensures* **process isolation**
* **‘12-factor app’ model** *ensures* **observability**
* **‘12-factor app’ model** *ensures* **lifecycle automation**

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

* **AWS Direct Connect** *delivers* **dedicated, high-performance network paths for 5G workload placement**
* **Global Accelerator** *delivers* **dedicated, high-performance network paths for 5G workload placement**
* **Dedicated, high-performance network paths** *support* **5G workload placement**

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

* **DevOps automation (CodePipeline, CloudFormation)** *allows* **NF operators to treat infrastructure as code**
* **Treating infrastructure as code** *enables* **reproducible deployments at scale**
* **Treating infrastructure as code** *enables* **version-controlled deployments**

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

* **UPF deployment** *may be* **hardware-based**
* **UPF deployment** *may be* **virtualized**
* **Choice of deployment** *depends on* **SLA**
* **Choice of deployment** *depends on* **performance needs**
* **Choice of deployment** *depends on* **vendor maturity**

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

* **UPF deployment** *may be* **hardware-based**
* **UPF deployment** *may be* **virtualized**
* **Choice of deployment** *depends on* **SLA**
* **Choice of deployment** *depends on* **performance needs**
* **Choice of deployment** *depends on* **vendor maturity**
* **Hardware UPF** *suits* **high-performance needs**
* **Virtualized UPF** *aids* **flexibility**
* **Virtualized UPF** *aids* **geo-distribution**

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

* **UPF** *steers* **traffic at edge**
* **UPF** *uses* **standardized APIs**
* **Standardized APIs** *route* **flows toward local applications**
* **Standardized APIs** *route* **flows toward cloud-hosted applications**

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

* **Edge IaaS platforms** *host* **VMs**
* **Edge IaaS platforms** *host* **containers**
* **VMs** *are managed by* **OpenStack**
* **Containers** *are managed by* **OpenStack**
* **VMs** *are managed by* **other platforms**
* **Containers** *are managed by* **other platforms**

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

* **Edge PaaS layers** *provide* **networking services**
* **Edge PaaS layers** *provide* **RNIS services**
* **Edge PaaS layers** *provide* **location services**
* **Edge PaaS layers** *provide* **user identity services**
* **Edge PaaS layers** *provide* **firewall services**
* **Edge PaaS layers** *provide* **DNS services**
* **Edge PaaS layers** *provide* **load balancing services**

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

* **Orchestrator** *deploys* **NFs and applications**
* **Deployment** *is based on* **service requirements**
* **Deployment** *is based on* **policies**
* **Deployment** *is based on* **resource templates mapped to location**

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

* **Traffic steering APIs** *allow* **dynamic flow decisions per application context**
* **Dynamic flow decisions** *are made per* **application context**
* **Application contexts** *include* **video**
* **Application contexts** *include* **IoT**

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

* **Cisco UPF** *implements* **1:1 Active/Standby redundancy**
* **1:1 Active/Standby redundancy** *uses* **SRP**
* **SRP** *performs* **ICSR-based state synchronization**

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

* **Standby UPF** *assumes* **the same Sx/N4 address** during **switchover**
* **Assuming the same Sx/N4 address during switchover** *makes* **the transition transparent to the SMF**

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

* **Sx/N4 control-plane heartbeat monitoring with BFD** *triggers* **fast UPF switchover**
* **Fast UPF switchover** *occurs in* **failure events**

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

* **Active UPF** *replicates* **IP-pool** to **Standby UPF**
* **Active UPF** *replicates* **session context** to **Standby UPF**
* **Replication** *occurs during* **Sx/N4 association**
* **Replication** *occurs during* **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.

* **Standby UPF** *starts in* **‘Pending-Active’**
* **Standby UPF** *transitions out of* **‘Pending-Active’** *after* **SRP elections**
* **Standby UPF** *transitions out of* **‘Pending-Active’** *after* **manual timeout configurations**
* **SRP elections** and **manual timeout configurations** *finalize* **switchover**

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

* **VPP health** *is integrated into* **SRP**
* **BGP monitoring** *is integrated into* **SRP**
* **Integration into SRP** *enables* **multi-layered UPF redundancy triggering**
* **Multi-layered UPF redundancy triggering** *relies on* **SRP**

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

* **SMF** *is unaware of* **standby UPF**
* **SMF** *interacts with* **active UPF endpoint**
* **Interaction** *occurs via* **stable Sx/N4 address**

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

* **SRP Active/Standby redundancy** *is supported without* **dual-active scenarios**
* **SRP Active/Standby redundancy** *relies on* **address takeover**
* **SRP Active/Standby redundancy** *relies on* **heartbeat control**

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

* **SMF heartbeat** *must be timed appropriately relative to* **UPF SRP timeout**
* **Proper timing between SMF heartbeat and UPF SRP timeout** *is crucial to avoid* **false session drop detection**

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

* **NF Set** *groups* **interchangeable NF instances of the same type**
* **Interchangeable NF instances of the same type** *include* **AMF**
* **Interchangeable NF instances of the same type** *include* **SMF**
* **Grouping by NF Set** *enables* **shared context**
* **Grouping by NF Set** *enables* **failover capability**

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

* **NF Sets** *support* **geo-redundancy**
* **Geo-redundancy** *is achieved via* **distributed instances**
* **Geo-redundancy** *is achieved via* **externalized session state**

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

* **Stateless NFs in an NF Set** *externalize* **state to UDSF**
* **Externalizing state to UDSF** *enables* **context retrieval by any instance**

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

* **NF instances in an NF Set** *can have* **different software versions**
* **Different software versions** *enable* **seamless rolling upgrades**

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

* **NF Set** *uses* **N+M redundancy**
* **N+M redundancy** *reduces* **overprovisioning**
* **N+M redundancy** *enables* **fast failover**

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

* **Context-sharing in NF Sets** *prevents* **signaling storms**
* **Context-sharing in NF Sets** *avoids* **re-registration during failover**
* **Avoiding re-registration during failover** *prevents* **signaling storms**

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

* **NF Set** *presents* **single virtual interface (e.g., anycast IP)**
* **Single virtual interface (e.g., anycast IP)** *is presented to* **clients**

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

* **Geographically distributed NF Set instances** *enable* **localized scaling**
* **Geographically distributed NF Set instances** *enable* **resilience per region**

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

* **Disjoint UPF deployment with dual PDU sessions** *increases* **end-to-end availability**
* **End-to-end availability** *is increased 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.

* **Dual-connectivity** *occurs through* **two separate gNBs**
* **Dual-connectivity** *allows* **simultaneous data flows**
* **Simultaneous data flows** *go to* **distinct UPFs**
* **Simultaneous data flows** *form* **a redundant user-plane chain**

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

* **User-plane redundancy** *is achieved via* **dual UPFs**
* **Control-plane** *remains* **single-path**
* **Single-path control-plane** *maintains* **management simplicity**

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

* **Industrial/private 5G deployments** *can reach* **availability levels above 99.9999%**
* **Availability levels above 99.9999%** *are achieved using* **dual-path UPF redundancy**
* **Dual-path UPF redundancy** *provides* **user-plane redundancy**

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

* **E2E availability gains** *are calculated using* **parallel reliability models**
* **Parallel reliability models** *include* **Reliability Block Diagrams**

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

* **Mean Time To Repair (MTTR)** *is a primary enabler of* **achieving telecom-grade availability targets**
* **Path redundancy** *is a primary enabler 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.

* **Redundancy strategy** *leverages* **dual PDU sessions**
* **Dual PDU sessions** *are mapped across* **disjoint UPFs and RAN connections**
* **Mapping across disjoint UPFs and RAN connections** *supports* **ultra-reliable industrial use cases**

**Factoid**: URLLC applications demand sub-millisecond latency and negligible packet loss, requiring stronger resilience than standard Fast ReRoute mechanisms.

* **URLLC applications** *demand* **sub-millisecond latency**
* **URLLC applications** *demand* **negligible packet loss**
* **URLLC applications** *require* **stronger resilience**
* **Stronger resilience** *exceeds* **standard Fast ReRoute mechanisms**

**Factoid**: PREOF enables 1+1 path protection by duplicating packets over two disjoint UPF paths and eliminating redundant packets at the receiver.

* **PREOF** *enables* **1+1 path protection**
* **PREOF** *duplicates* **packets over two disjoint UPF paths**
* **Redundant packets** *are eliminated at* **the receiver**

**Factoid**: The PREOF mechanism can be deployed at the gNB or UE as a Protection Tunnel Ingress (PTI), replicating packets for redundancy.

* **PREOF mechanism** *can be deployed at* **gNB** as **Protection Tunnel Ingress (PTI)**
* **PREOF mechanism** *can be deployed at* **UE** as **Protection Tunnel Ingress (PTI)**
* **Protection Tunnel Ingress (PTI)** *replicates* **packets** for **redundancy**

**Factoid**: Replicated packets traverse two synchronized UPFs before reaching a Protection Tunnel Egress (PTE) node for elimination.

* **Replicated packets** *traverse* **two synchronized UPFs**
* **Replicated packets** *reach* **Protection Tunnel Egress (PTE) node**
* **Protection Tunnel Egress (PTE) node** *eliminates* **replicated packet**

**Factoid**: Ordering of packets is offloaded to an external PTE-O server or implemented via eBPF/DPDK, reducing complexity in programmable hardware switches.

* **Ordering of packets** *is offloaded to* **external PTE-O server**
* **Ordering of packets** *is implemented via* **eBPF/DPDK**
* **Offloading ordering of packets** *reduces* **complexity in programmable hardware switches**

**Factoid**: PREOF is fully compatible with existing GTP-U tunnels, encapsulating replicated packets transparently to UPFs.

* **PREOF** *is fully compatible with* **existing GTP-U tunnels**
* **PREOF** *encapsulates* **replicated packets**
* **Encapsulation of replicated packets** *is transparent to* **UPFs**

**Factoid**: 1+1 path protection increases system resilience at the cost of slightly higher latency and resource usage.

* **1+1 path protection** *increases* **system resilience  
  1+1 path protection** *incurs* **slightly higher latency**
* **1+1 path protection** *incurs* **higher resource usage**

**Factoid**: PREOF aligns with 3GPP Release 18’s support for packet duplication mechanisms to enhance URLLC resilience.

* **PREOF** *aligns with* **3GPP Release 18’s support for packet duplication mechanisms**
* **3GPP Release 18’s support for packet duplication mechanisms** *enhances* **URLLC resilience**

**Factoid**: Disjoint UP paths with replicated UPFs ensure session survival even if one UPF or path experiences a failure.

* **Disjoint UP paths** *use* **replicated UPFs**
* **Disjoint UP paths with replicated UPFs** *ensure* **session survival**
* **Session survival** *occurs even if* **one UPF or path experiences a failure**

**Factoid**: Choosing between PTI placement at gNB versus UE reflects trade-offs in performance, complexity, and protection scope.

* **Choosing PTI placement at gNB** *reflects trade-offs in* **performance**
* **Choosing PTI placement at gNB** *reflects trade-offs in* **complexity**
* **Choosing PTI placement at gNB** *reflects trade-offs in* **protection scope**
* **Choosing PTI placement at UE** *reflects trade-offs in* **performance**
* **Choosing PTI placement at UE** *reflects trade-offs in* **complexity**
* **Choosing PTI placement at UE** *reflects trade-offs in* **protection scope**

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

* **Static resource allocation** *fails in* **5G**
* **Dynamic models** *include* **MCM**
* **Dynamic models** *are needed for* **real-time adaptability**

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

* **MCM** *integrates* **bandwidth allocation**
* **MCM** *integrates* **traffic prioritization**
* **MCM** *integrates* **encryption**
* **MCM** *integrates* **network slicing**
* **Integration of bandwidth allocation, traffic prioritization, encryption, and network slicing** *enforces* **QoS**

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

* **Target QoS metrics** *include* **sub-ms latency for URLLC**
* **Target QoS metrics** *include* **bounded jitter for URLLC**
* **Target QoS metrics** *include* **hundreds of Mbps throughput for eMBB**
* **Sub-ms latency** *is for* **URLLC**
* **Bounded jitter** *is for* **URLLC**
* **Hundreds of Mbps throughput** *is for* **eMBB**

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

* **ML-based traffic prediction** *is used in* **MCM**
* **ML-based traffic prediction** *dynamically reallocates* **resources**
* **Dynamically reallocating resources** *anticipates* **demand peaks**

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

* **MDP/DRL-based RAN schedulers** *dynamically balance* **URLLC performance  
  MDP/DRL-based RAN schedulers** *dynamically balance* **eMBB performance  
  Dynamic balancing of URLLC and eMBB performance** *occurs on* **slot timescales**
* **Dynamic balancing of URLLC and eMBB performance** *occurs on* **mini-slot timescales**

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

* **EDQAS/LDI schedulers** *schedule* **resource blocks**
* **EDQAS/LDI schedulers** *minimize* **uRLLC latency**
* **EDQAS/LDI schedulers** *minimize* **eMBB rate loss**
* **Scheduling resource blocks** *occurs at* **MAC layer**

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

* **Network slices** *partition* **resources**
* **Network slices** *provide* **per-slice SLA-driven enforcement**
* **Per-slice SLA-driven enforcement** *applies across* **RAN domains**
* **Per-slice SLA-driven enforcement** *applies across* **core domains**

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

* **Encryption overhead** *is modeled in* **resource allocation decisions**
* **Resource allocation decisions** *aim to meet* **QoS**
* **Resource allocation decisions** *ensure* **security**

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

* **QoS parameters** *include* **latency**
* **QoS parameters** *include* **jitter**
* **QoS parameters** *include* **packet loss**
* **QoS parameters** *include* **throughput**
* **QoS parameters** *include* **spectral efficiency**
* **QoS parameters** *include* **energy efficiency**

\_\_\_\_\_\_\_\_\_ BATCH 4 GEMINI \_\_\_\_\_\_\_\_\_

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

* **URLLC performance improvements** *involve* **puncturing eMBB traffic in real-time**
* **Puncturing eMBB traffic in real-time** *trades off* **throughput**

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

**QoS-aware slicing** *leverages* **vertical slice models**

**Vertical slice models** *allocate* **bandwidth between service types**

**Vertical slice models** *allocate* **priority between service types**

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

* **Dynamic resource allocation models** *consider* **fairness**
* **Dynamic resource allocation models** *consider* **availability**
* **Dynamic resource allocation models** *consider* **service resilience**
* **Consideration of fairness, availability, and service resilience** *occurs during* **congestion**

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

* **Resource allocation frameworks** *combine* **RAN-level scheduling** and **core-level queue management**
* **RAN-level scheduling** and **core-level queue management** *enable* **SLA delivery**

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

* **URLLC** *requires* **≤ 1 ms one-way latency**
* **URLLC** *requires* **≥ 99.999 % reliability**
* **Stringent QoS targets** *are posed by* **URLLC**

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

* **Inline DPDK-accelerated Snort IPS** *can meet* **median latency**
* **Inline DPDK-accelerated Snort IPS** *shows* **tail latency spikes above 1 ms**
* **Tail latency spikes above 1 ms** *are* **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.

* **Snort IPS in a VM** *hit* **2.5 ms worst-case latency**
* **2.5 ms worst-case latency** *occurred at* **99.999th percentile**
* **2.5 ms worst-case latency at the 99.999th percentile** *occurred* **even with no rule matching**

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

* **Packet processing stacks** *use* **Linux**
* **Packet processing stacks** *use* **DPDK**
* **Packet processing stacks** *use* **Snort**
* **Virtualization unpredictability** *is revealed due to* **interrupts**
* **Virtualization unpredictability** *is revealed due to* **CPU scaling**

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

* **Packet processing stacks** *use* **Linux**
* **Packet processing stacks** *use* **DPDK**
* **Packet processing stacks** *use* **Snort**
* **Virtualization unpredictability** *is revealed due to* **interrupts**
* **Virtualization unpredictability** *is revealed due to* **CPU scaling**

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

* **Predictive model** *estimates* **maximum sustainable IPS load under URLLC**
* **Estimating maximum sustainable IPS load under URLLC** *enables* **SLA-driven capacity planning**

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

* **Middleware security functions (like IPS)** *must be carefully optimized*
* **Careful optimization of middleware security functions (like IPS)** *avoids* **violating URLLC tail latency requirements**

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

* **Hardware-assisted timestamping** *is essential for* **precise tail-latency measurement**
* **Precise tail-latency measurement** *enables* **evaluating IPS impact on URLLC**

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

* **Packet processing latency** *remains unpredictable due to* **virtual environment effects**
* **Minimal packet inspection** *does not prevent* **unpredictable packet processing latency**

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

* **Packet processing pipelines** *must be architected for* **low jitter**
* **Packet processing pipelines** *must be architected for* **deterministic behavior**
* **Architecting packet processing pipelines for low jitter and deterministic behavior** *supports* **URLLC with inline security**

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

* **5G URLLC targets** *mandate* **≤ 1 ms user-plane latency**
* **≤ 1 ms user-plane latency** *represents* **≈ 4× reduction compared to LTE**

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

* **URLLC** *demands* **ultra-reliability (≥ 99.999 %)**
* **URLLC** *demands* **block error rates down to 10⁻⁹**
* **Ultra-reliability (≥ 99.999 %)** *corresponds to* **block error rates down to 10⁻⁹**

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

* **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.

* **URLLC flows** *use* **resource reservation**
* **URLLC flows** *use* **preemption**
* **URLLC flows** *use* **mini-slots**
* **Use of resource reservation with preemption and mini-slots** *enables* **meeting sub-ms deadlines**

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

* **Edge computing** *eliminates* **∼100 ms transport delay**
* **Eliminating ∼100 ms transport delay** *enables* **end-to-end latency ≤ 1 ms**

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

* **Flexible TTIs** *are used to* **reduce transmission latency**
* **Robust coding techniques** *are used to* **reduce transmission latency**
* **Flexible TTIs** *are used to* **reduce BER**
* **Robust coding techniques** *are used to* **reduce BER**

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

* **Multiple UPFs with local breakout** *support* **disjoint user-plane paths**
* **Disjoint user-plane paths** *provide* **low-latency**
* **Disjoint user-plane paths** *provide* **resilience**

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

* **URLLC scheduling** *uses* **prioritized access**
* **URLLC scheduling** *uses* **TTI scaling**
* **Prioritized access** *minimizes* **delay and jitter**
* **TTI scaling** *minimizes* **delay and jitter**

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

* **Reliable URLLC transmission** *combines* **mini-slots**
* **Reliable URLLC transmission** *combines* **redundancy**
* **Reliable URLLC transmission** *combines* **HARQ optimization**

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

* **Edge-deployed UPFs** *collaborate with* **MEC**
* **Collaboration between Edge-deployed UPFs and MEC** *enforces* **QoS**
* **Collaboration between Edge-deployed UPFs and MEC** *satisfies* **URLLC requirements**

**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.

* **URLLC flows** *demand* **≤ 1 ms latency**
* **URLLC flows** *demand* **low jitter**
* **URLLC flows** *demand* **ultra-high reliability  
  eMBB flows** *prioritize* **high throughput**
* **eMBB flows** *prioritize* **moderate latency**
* **mMTC flows** *support* **up to 1 M devices/km²**
* **mMTC flows** *support* **low rate requirements**

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

* **QoS metrics** *include* **latency**
* **QoS metrics** *include* **jitter**
* **QoS metrics** *include* **packet loss**
* **QoS metrics** *include* **throughput**
* **QoS metrics** *include* **reliability**
* **QoS metrics** *include* **fairness**
* **QoS metrics** *include* **energy efficiency**
* **QoS metrics** *apply across* **5G slices**

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

* **Slice-specific SLAs** *are enforced through* **dynamic bandwidth allocation**
* **Slice-specific SLAs** *are enforced through* **priority queueing per network slice**

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

* **Traffic shaping** *uses* **DiffServ**
* **Ingress classification** *enables* **per-hop QoS enforcement**
* **Metering** *enables* **per-hop QoS enforcement**
* **Marking** *enables* **per-hop QoS enforcement**

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

* **URLLC scheduling** *uses* **preemption/puncturing**
* **URLLC scheduling** *uses* **mini-slots at RAN**
* **Preemption/puncturing and mini-slots at RAN** *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.

* **eMBB scheduling** *applies* **proportional fairness  
  eMBB scheduling** *applies* **weighted round-robin**
* **Proportional fairness** *balances* **throughput and fairness**
* **Weighted round-robin** *balances* **throughput and fairness**

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

* **Transport-priority queuing** *ensures* **UL/DL URLLC packets** *are served ahead of* **eMBB traffic**
* **Transport-priority queuing** *ensures* **UL/DL URLLC packets** *are served ahead of* **mMTC traffic**

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

* **Hard slicing** *uses* **dedicated resources per slice**
* **Soft slicing** *shares* **resources with prioritization**
* **Hard slicing and soft slicing** *support* **mixed traffic isolation**

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

* **Reliability and latency goals for URLLC** *are modeled via* **resource block reservation  
  Reliability and latency goals for URLLC** *are modeled via* **robust coding**
* **Resource block reservation** *occurs at* **physical layer**
* **Resource block reservation** *occurs at* **transport layer**
* **Robust coding** *occurs at* **physical layer**
* **Robust coding** *occurs at* **transport layer**

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

* **QoS enforcement** *spans* **RAN**
* **QoS enforcement** *spans* **transport**
* **QoS enforcement** *spans* **core layers**
* **Spanning RAN, transport, and core layers** *ensures* **consistent SLA adherence**

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

* **Scheduler models** *use* **optimization techniques**
* **Scheduler models** *use* **ML techniques**
* **Scheduler models** *dynamically allocate* **resources**
* **Dynamic allocation of resources** *is based on* **real-time QoS demands**

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

* **QoS Flow** *is tagged with* **unique 4-bit QFI**
* **Tagging with unique 4-bit QFI** *occurs within* **PDU session**

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

* **QoS Flows** *are classified as* **GBR**
* **QoS Flows** *are classified as* **Non-GBR**
* **Specific QoS profiles** *are defined for* **GBR QoS Flows**
* **Specific QoS profiles** *are defined for* **Non-GBR QoS Flows**

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

* **UPF** *applies* **Packet Detection Rules (PDRs)**
* **UE** *applies* **QoS rules**
* **Applying PDRs and QoS rules** *maps* **packets to QoS flows at the NAS layer**

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

* **Packet filters** *use* **IP addresses**
* **Packet filters** *use* **ports**
* **Packet filters** *use* **protocols**
* **Packet filters** *use* **Ethernet tags**
* **Using IP addresses, ports, protocols, and Ethernet tags** *enables* **packet classification**

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

* **Kubernetes-based control plane orchestration** *benefits from* **stateless NF design**
* **Stateless NF design** *enables* **auto-replacement of crashed pods**
* **Auto-replacement of crashed pods** *occurs without* **state loss**

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

* **Multiple QoS flows** *can share* **a DRB**
* **Sharing a DRB** *requires* **aligned service requirements**

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

* **Reflective QoS** *enables* **UE** to derive **uplink QoS-to-DRB mapping** from **downlink rules**
* **Deriving uplink QoS-to-DRB mapping from downlink rules** *uses* **RQI/RDI flags**

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

* **N3 GTP-U headers** *carry* **QFI**
* **QFI** *is used for* **QoS identification**
* **QoS identification** *occurs between* **UPF** and **gNB**

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

* **SMF** *configures* **QoS flows**
* **SMF** *distributes* **QoS rules** to **UE** via **N1**
* **SMF** *distributes* **PDRs** to **gNB** via **N2**
* **SMF** *distributes* **PDRs** to **UPF** via **N4**

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

* **QoS control** *spans* **NAS classification**
* **QoS control** *spans* **SDAP mapping**
* **QoS control** *spans* **GTP-U marking**
* **NAS classification** *enforces* **consistent end-to-end service quality**
* **SDAP mapping** *enforces* **consistent end-to-end service quality**
* **GTP-U marking** *enforces* **consistent end-to-end service quality**

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

* **5G** *mandates support for* **5G-AKA**
* **5G** *mandates support for* **EAP-AKA′**
* **5G-AKA** *provides* **unified, access-agnostic authentication**
* **EAP-AKA′** *provides* **unified, access-agnostic authentication**

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

* **EAP-TLS** *can be used as* **secondary authentication method**
* **Secondary authentication method** *is used in* **private deployments**
* **Secondary authentication method** *is used in* **enterprise deployments**

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

* **AUSF** *anchors* **security key KSEAF**
* **Anchoring security key KSEAF** *occurs during* **initial authentication**
* **AUSF** *supplies* **authentication vectors**

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

* **UDM/ARPF** *stores* **subscriber credentials**
* **UDM/ARPF** *supports* **authentication via AKA framework**
* **UDM/ARPF** *supports* **authentication via EAP framework**

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

* **All SBA APIs** *must use* **mutual TLS**
* **Mutual TLS** *requires* **client certificates**
* **Mutual TLS** *requires* **server certificates**
* **All SBA APIs** *are often coupled with* **OAuth authorization**

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

* **IPsec tunnels (NWu/NWt)** *protect* **NAS signaling**
* **IPsec tunnels (NWu/NWt)** *operate over* **non-3GPP access**
* **Optional NULL encryption** *is used for* **trusted networks**

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

* **SEPP** *secures* **inter-PLMN signaling on N32 via TLS**
* **SEPP** *secures* **inter-PLMN signaling on N32 via PRINS**
* **Securing inter-PLMN signaling via TLS or PRINS** *ensures* **integrity**
* **Securing inter-PLMN signaling via TLS or PRINS** *ensures* **confidentiality**

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

* **IPUPS** *secures* **N9 GTP-U traffic between UPFs**
* **IPUPS** *uses* **IPsec-based filtering**
* **IPsec-based filtering** *avoids* **tunnel spoofing**

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

* **NSSAAF** *enables* **slice-level authentication**
* **Slice-level authentication** *uses* **EAP-based credentials**
* **EAP-based credentials** *are used for* **each network slice**

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

* **TLS 1.2/1.3 configurations** *require* **AEAD cipher suites**
* **TLS 1.2/1.3 configurations** *require* **OCSP**
* **TLS 1.2/1.3 configurations** *remove* **legacy weak cipher support**

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

* **Zero-Trust security** *demands* **explicit flow definitions**
* **Zero-Trust security** *demands* **strict policies**
* **Zero-Trust security** *demands* **continuous monitoring**

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

* **Slice management interfaces** *leverage* **OAuth  
  Slice management interfaces** *use* **mTLS**
* **OAuth** *provides* **access control  
  mTLS** *secures* **management messaging**

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

* **UE identity** *is encrypted as* **SUCI**
* **Encryption as SUCI** *uses* **public-key encryption**
* **Encryption as SUCI** *protects* **SUPI from over-the-air exposure**

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

* **SEAF**, **AUSF**, **UDM/ARPF**, and **SIDF** *coordinate to* **authenticate UE**
* **SEAF**, **AUSF**, **UDM/ARPF**, and **SIDF** *coordinate to* **manage key derivation**

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

* **5G key hierarchy** *includes* **KAUSF**
* **5G key hierarchy** *includes* **KSEAF**
* **5G key hierarchy** *includes* **KAMF**
* **5G key hierarchy** *offers* **deeper security separation than 4G**

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

* **5G authentication session** *establishes* **multiple security contexts**
* **Security contexts** *span* **different access types**

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

* **EAP-TLS** *leverages* **X.509 certificates**
* **EAP-TLS** *authenticates* **without requiring USIM**
* **EAP-TLS** *suits* **BYOD devices**
* **EAP-TLS** *suits* **enterprise devices**

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

* **EAP-AKA′** *is* **symmetric key-based EAP method**
* **EAP-AKA′** *offers* **trust equivalent to 5G-AKA**
* **Trust equivalent to 5G-AKA** *is delivered via* **EAP exchange**

\_\_\_\_\_\_\_ BATCH 5 GEMINI \_\_\_\_\_\_\_\_

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

* **SIDF** *decrypts* **SUCI** to **SUPI**
* **Decrypting SUCI to SUPI** *enables* **identifier confidentiality**
* **Identifier confidentiality** *is protected by* **public-key protection**

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

* **SEAF** *uses* **mutual TLS**
* **SEAF** *uses* **PRINS**
* **Using mutual TLS and PRINS** *secures* **inter-PLMN communication**
* **Using mutual TLS and PRINS** *prevents* **downgrade attacks**

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

* **KAUSF** *derives* **KSEAF**
* **KSEAF** *derives* **KAMF  
  Mapping of keys (KAUSF → KSEAF → KAMF)** *ensures* **layered trust**
* **Mapping of keys (KAUSF → KSEAF → KAMF)** *ensures* **key separation in 5G**

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

* **SUPI encryption** *enhances* **privacy**
* **SBA authentication functions** *enhance* **privacy**
* **Deeper key derivation** *enhances* **privacy**
* **SUPI encryption** *enhances* **home-network control**
* **SBA authentication functions** *enhance* **home-network control**
* **Deeper key derivation** *enhances* **home-network control**
* **Enhancing privacy and home-network control** *occurs in* **5G**

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

* **EAP-TLS** *eliminates* **symmetric key dependency**
* **EAP-TLS** *introduces* **certificate lifecycle overhead**
* **Eliminating symmetric key dependency** *trades* **key management** for **lifecycle complexity**
* **Introducing certificate lifecycle overhead** *trades* **key management** for **lifecycle complexity**

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

* **5G-AKA** *is a mandatory primary authentication protocol across* **N1 interface**
* **5G-AKA** *is a mandatory primary authentication protocol across* **N12 interface**
* **5G-AKA** *is a mandatory primary authentication protocol across* **N13 interface  
  EAP-AKA′** *is a mandatory primary authentication protocol across* **N1 interface**
* **EAP-AKA′** *is a mandatory primary authentication protocol across* **N12 interface**
* **EAP-AKA′** *is a mandatory primary authentication protocol across* **N13 interface**

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

* **EAP** *enables* **unified authentication**
* **Unified authentication** *applies regardless of UE access via* **3GPP RAT**
* **Unified authentication** *applies regardless of UE access via* **non-3GPP RAT**

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

* **Security threats** *include* **replay attacks**
* **Security threats** *include* **MitM attacks**
* **Security threats** *include* **downgrade attacks**
* **Security threats** *target* **N1 interface** *if* **proper cryptographic protections are absent**
* **Security threats** *target* **N12 interface** *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.

* **N1 interface** *must be protected using* **IPsec over non-3GPP access**
* **N1 interface** *must be protected using* **NAS integrity algorithms over 3GPP access**
* **N2 interface** *must be protected using* **IPsec over non-3GPP access**
* **N2 interface** *must be protected using* **NAS integrity algorithms over 3GPP access**

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

* **N12 interface** *requires* **mutual TLS**
* **N13 interface** *requires* **mutual TLS**
* **Mutual TLS** *enforces* **authentication**
* **Mutual TLS** *enforces* **confidentiality**

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

* **Inter-domain interface N32** *needs* **TLS with PRINS or equivalent**
* **Inter-domain interface N16** *needs* **TLS with PRINS or equivalent**
* **TLS with PRINS or equivalent** *ensures* **end-to-end signaling integrity**

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

* **Formal verification tools like Tamarin** *uncovered* **potential linkability vulnerabilities**
* **Potential linkability vulnerabilities** *occur in* **5G-AKA**
* **Potential linkability vulnerabilities** *remain unless* **mitigations are implemented**

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

* **Authentication exchanges** *can be* **secure  
  Implementations** *must ensure* **serving network binding via proper key derivation**
* **Proper key derivation for serving network binding** *prevents* **impersonation**

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

* **Non-3GPP access (e.g., Wi-Fi)** *requires* **IPsec tunnels (e.g., NWu)  
  IPsec tunnels (e.g., NWu)** *secure* **NAS traffic**
* **IPsec tunnels (e.g., NWu)** *secure* **user-plane traffic**

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

* **Upgrade or downgrade prevention** *is* **critical**
* **NAS protections** *must enforce* **version awareness**
* **Interface-level protections** *must enforce* **context awareness**
* **Enforcing version and context awareness** *avoids* **downgrade exploits**

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

* **UE** *establishes* **IPsec signaling SA (NWu)**
* **IPsec signaling SA (NWu)** *is established with* **N3IWF**
* **IPsec signaling SA (NWu)** *uses* **IKEv2**
* **IPsec signaling SA (NWu)** *uses* **EAP-5G**
* **UE** *is connected via* **untrusted WLAN**

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

* **EAP-5G** *encapsulates* **NAS-based authentication (5G-AKA/EAP-AKA′)**
* **NAS-based authentication** *includes* **5G-AKA**
* **NAS-based authentication** *includes* **EAP-AKA′**
* **Encapsulation within IKEv2 exchanges** *occurs 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.

* **User-plane over Wi-Fi** *uses* **separate IPsec child SA(s)**
* **Separate IPsec child SA(s)** *are established after* **PDU session establishment**

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

* **Trusted WLAN setups** *use* **layer-2 authentication (802.1x via TNAP)**
* **Layer-2 authentication (802.1x via TNAP)** *is followed by* **IPsec signaling SA**
* **IPsec signaling SA** *employs* **NULL encryption**
* **NULL encryption** *avoids* **double encryption**

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

* **Ta/Yw interfaces** *connect* **WLAN APs** to **TNGF/TWIF gateways**
* **Ta/Yw interfaces** *exist outside* **3GPP scope**
* **Ta/Yw interfaces** *are 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.

* **Wi-Fi only devices without USIM** *rely on* **certificate-based EAP-TLS**
* **Wi-Fi only devices without USIM** *rely on* **certificate-based EAP-TTLS**
* **Certificate-based EAP-TLS/EAP-TTLS** *are used in* **private SNPN environments**

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

* **ATSSS** *enables* **steering or splitting of traffic between 3GPP and Wi-Fi**
* **Steering or splitting of traffic** *is based on* **central policy**
* **Central policy** *is provided by* **SMF** and **PCF**

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

* **NAS messages over Wi-Fi** *rely on* **TCP/IP transport**
* **TCP/IP transport** *occurs within* **IPsec tunnels**
* **IPsec tunnels** *ensure* **message reliability**
* **IPsec tunnels** *ensure* **message ordering**

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

* **Building trust zones in Wi-Fi** *determines* **IPsec encryption type  
  IPsec encryption type** *is* **NULL encryption** *in* **trusted zones**
* **IPsec encryption type** *is* **standard encryption** *in* **untrusted zones**
* **Using NULL encryption** *prevents* **redundant encryption**

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

* **Integration of EAP-5G, IKEv2, and IPsec** *ensures* **secure control-plane transport**
* **Integration of EAP-5G, IKEv2, and IPsec** *ensures* **secure data-plane transport**
* **Secure control-plane transport** *occurs across* **Wi-Fi links**
* **Secure data-plane transport** *occurs across* **Wi-Fi links**

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

* **Primary authentication in industrial 5G** *mandates use of* **5G-AKA**
* **Primary authentication in industrial 5G** *mandates use of* **EAP-AKA**
* **Primary authentication in industrial 5G** *mandates use of* **EAP-TLS**
* **Use of 5G-AKA, EAP-AKA, or EAP-TLS** *verifies* **device identity**

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

* **Secondary EAP authentication** *enables* **access to external services**
* **Secondary EAP authentication** *enables* **access to slice-specific networks**
* **Access to external services** *uses* **alternate credentials**
* **Access to slice-specific networks** *uses* **alternate credentials**

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

* **UDM/HSS** *securely stores* **USIM credentials**
* **UDM/HSS** *supports* **provisioning workflows for industrial devices**

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

* **SIM provisioning in industrial environments** *requires* **secure lifecycle procedures**
* **Secure lifecycle procedures** *include* **revocation**
* **Secure lifecycle procedures** *include* **updates**

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

* **Industrial RAN** *requires* **IPsec tunnels (NWu/NWt)**
* **IPsec tunnels (NWu/NWt)** *protect* **NAS signaling**
* **IPsec tunnels (NWu/NWt)** *protect* **data-plane**
* **IPsec tunnels (NWu/NWt)** *include* **SPI protections**

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

* **Trusted industrial deployments** *may use* **NULL-encryption under IPsec tunnels**
* **NULL-encryption under IPsec tunnels** *avoids* **double-layer encryption overhead**

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

* **Industrial 5G** *prioritizes* **availability** over **confidentiality**
* **Industrial 5G** *prioritizes* **deterministic latency** over **confidentiality**
* **Prioritizing availability and deterministic latency over confidentiality** *supports* **maintaining real-time control**

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

* **TSN traffic** *must be mapped into* **5G slices**
* **TSN traffic mapping into 5G slices** *requires* **real-time scheduling**
* **TSN traffic mapping into 5G slices** *requires* **IPsec security**
* **TSN traffic mapping into 5G slices** *is used for* **industrial applications**

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

* **Industrial 5G** *must mitigate* **wireless-specific threats**
* **Wireless-specific threats** *include* **jamming**
* **Wireless-specific threats** *include* **spoofing**
* **Mitigation of wireless-specific threats** *is achieved via* **hardened SIM provisioning**
* **Mitigation of wireless-specific threats** *is achieved via* **tunnel integrity**

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

* **Slice-specific EAP authentication** *is critical for* **multiple industrial operations sharing the same 5G infrastructure**
* **Multiple industrial operations** *share* **the same 5G infrastructure**

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

* **Intent-Based Networking (IBN)** *enables* **high-level operator intent to be translated into network policy**
* **Translation of high-level operator intent into network policy** *occurs via* **NIT**
* **Translation via NIT** *occurs 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.

* **Architectural framework** *integrates* **NWDAF as IBN Analyzer**
* **Integration of NWDAF as IBN Analyzer** *creates* **closed-loop system**
* **Closed-loop system** *includes* **monitoring**
* **Closed-loop system** *includes* **validation**
* **Closed-loop system** *includes* **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.

* **NIT** *uses* **data-model mapping**
* **NIT** *optionally uses* **NLP**
* **Data-model mapping and NLP** *convert* **user intents** *into* **NF-specific configurations**
* **User intents** *include* **slice QoS**
* **User intents** *include* **IoT data collection**

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

* **IBN Controller** *dispatches* **translated policy rules** to **VNFs**
* **IBN Controller** *dispatches* **translated policy rules** to **CNFs**
* **IBN Controller** *dispatches* **translated policy rules** to **PNFs**
* **Dispatching translated policy rules to VNFs, CNFs, or PNFs** *uses* **NF-facing interface**

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

* **Network telemetry** *is collected via* **monitoring interface** from **Network Functions (NFs)**
* **Network telemetry** *is analyzed by* **NWDAF**
* **Analysis of network telemetry by NWDAF** *is used to audit* **intent success**

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

* **Consumer-facing interface** *accepts* **user intent**
* **User intent** *is 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.

* **Use cases** *include* **IoT aggregation**
* **Use cases** *include* **V2X QoS**
* **IoT aggregation use case** *demonstrates* **intent-driven slice creation  
  V2X QoS use case** *demonstrates* **intent-driven slice creation  
  IoT aggregation use case** *demonstrates* **intent-driven SLA enforcement end-to-end**
* **V2X QoS use case** *demonstrates* **intent-driven SLA enforcement end-to-end**
* **Intent** *drives* **slice creation**
* **Intent** *drives* **SLA enforcement end-to-end**

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

* **Closed-loop automation** *ensures* **intent enforcement accuracy**
* **Closed-loop automation** *verifies* **intent enforcement accuracy** via **telemetry**
* **Closed-loop automation** *updates* **policies as needed**

**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.

* **TS 28.312** *defines* **SBMA-based Intent-Driven Management Service (MnS)**
* **SBMA-based Intent-Driven Management Service (MnS)** *allows intent producers to accept* **network or service intents**
* **SBMA-based Intent-Driven Management Service (MnS)** *allows intent producers to fulfill* **network or service intents**
* **SBMA-based Intent-Driven Management Service (MnS)** *allows intent producers to 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.

* **Intent** *is decoupled from* **execution**
* **Intent** *specifies* **intent expectations**
* **Intent** *does not specify* **policies/actions**
* **Decoupling of intent from execution** *aligns with* **model-driven SBMA approach**

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

* **Intent operations** *include* **create**
* **Intent operations** *include* **modify**
* **Intent operations** *include* **query**
* **Intent operations** *include* **delete**
* **Intent operations** *include* **activate**
* **Intent operations** *include* **deactivate**
* **Intent operations** *manage* **intent lifecycle states**
* **Managing intent lifecycle states** *applies to* **each intent operation**

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

* **Intent content** *includes* **expectations**
* **Expectations** *include* **throughput targets**
* **Expectations** *include* **area coverage**
* **Intent content** *is linked to* **context objects**
* **Context objects** *include* **PLMN**
* **Context objects** *include* **TAC**

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

* **TR 28.912** *enhances* **TS 28.312**
* **TR 28.912** *provides* **verification reports**
* **TR 28.912** *provides* **conflict resolution**
* **TR 28.912** *provides* **feasibility checks**
* **TR 28.912** *provides* **AI/ML mapping**
* **TR 28.912** *provides* **intent-driven SON orchestration**

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

* **Intent producers** *implement actions via* **rule-based mechanisms**
* **Intent producers** *implement actions via* **closed-loop mechanisms**
* **Intent producers** *implement actions via* **AI/ML-driven mechanisms**

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

* **3GPP intent services** *are interoperable with* **TM-Forum intent frameworks**
* **3GPP intent services** *are interoperable with* **O-RAN intent frameworks**
* **Interoperability** *enables* **cross-domain automation**

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

* **Intent expectations** *can include* **energy savings targets**
* **Intent expectations** *can include* **service performance metrics**
* **Energy savings targets** *are balanced against* **service performance metrics**

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

* **Release 19 (TR 28.914)** *extends* **intent models**
* **Extended intent models** *include* **RAN-level intent autopolicies**
* **RAN-level intent autopolicies** *support* **6G readiness**

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

* **RadioNetworkExpectation** *includes* **coverageAreaPolygon**
* **RadioNetworkExpectation** *includes* **RAT types**
* **RadioNetworkExpectation** *includes* **target throughput**
* **RadioNetworkExpectation** *includes* **latency thresholds**

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

* **Network intents** *are* **high-level objectives**
* **Network intents** *are defined as* **IntentTargets**
* **Network intents** *are defined as* **IntentExpectations**
* **Network intents** *are defined as* **context constraints**

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

* **End-to-end IDM architecture** *includes* **ingestion components**
* **End-to-end IDM architecture** *includes* **translation components**
* **End-to-end IDM architecture** *includes* **orchestration components**
* **End-to-end IDM architecture** *includes* **assurance components**
* **End-to-end IDM architecture** *includes* **reporting components**
* **Translation components** *are implemented via* **ILUs/ILL**

**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.

* **Inner loop** *automates* **fulfillment**
* **Inner loop** *automates* **assurance**
* **Outer loop** *involves* **user refinement**
* **Outer loop** *involves* **intent updates**

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

* **Conflict detection mechanisms** *identify* **overlapping intents**
* **Conflict detection mechanisms** *identify* **contradictory intents**
* **Identifying overlapping or contradictory intents** *requires* **resolution rules**

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

* **Intent assurance** *semantically maps* **network telemetry** to **intent expectations  
  Semantic mapping of network telemetry to intent expectations** *detects* **drift**

\_\_\_\_\_\_\_ BATCH 6 GEMINI \_\_\_\_\_\_\_

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

* **Assurance systems** *may deploy* **LLMs**
* **LLMs** *recommend* **policy changes**
* **Recommendation of policy changes** *occurs when* **intent drift arises**

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

* **Intent frameworks** *align with* **TM Forum**
* **Intent frameworks** *align with* **3GPP**
* **Intent frameworks** *align with* **ETSI ZSM**
* **Alignment with TM Forum, 3GPP, and ETSI ZSM** *enables* **multi-domain interoperability**

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

* **Intent ingestion** *supports* **user-interactive refinement**
* **Intent ingestion** *supports* **automated processing**
* **Automated processing** *targets* **machine-actionable formats**

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

* **Intent translation** *uses* **reusable Intent Logic Units (ILUs)**
* **Reusable Intent Logic Units (ILUs)** *are sourced from* **Intent Logic Library**
* **Intent Logic Units (ILUs)** *map* **abstract intent** *to* **policies**

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

* **Reporting** *abstracts* **low-level telemetry**
* **Reporting** *generates* **intent-aligned summaries**
* **Intent-aligned summaries** *support* **operator decision-making**

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

* **Assurance loop** *may trigger* **corrective workflows**
* **Assurance loop** *may escalate to* **user** when **semantic compliance falls below thresholds**

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

* **Natural-language intent interface** *parses* **English statements**
* **Natural-language intent interface** *translates* **English statements** into **formal policy definitions**
* **Formal policy definitions** *enable* **private 5G management**

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

* **5G-CLARITY platform** *includes* **Intent Engine**
* **5G-CLARITY platform** *includes* **MLFO**
* **5G-CLARITY platform** *includes* **Data Lake**
* **Intent Engine** *is used for* **telemetry-driven intent mapping**
* **MLFO** *is used for* **telemetry-driven intent mapping**
* **Data Lake** *is used for* **telemetry-driven intent mapping**

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

* **ML models** *convert* **high-level intents** into **API workflows**
* **API workflows** *target* **NFV orchestrators**
* **API workflows** *target* **RAN orchestrators**

**Factoid**: Use cases—slice provisioning, indoor positioning, and service deployment—were benchmarked for provisioning latency.

* **Use cases** *include* **slice provisioning**
* **Use cases** *include* **indoor positioning**
* **Use cases** *include* **service deployment**
* **Use cases** *were benchmarked for* **provisioning latency**

**Factoid**: Platform aligns with 3GPP TS 28.312, ETSI ZSM, and TM Forum intent frameworks for cross-SDO integration.

* **Platform** *aligns with* **3GPP TS 28.312**
* **Platform** *aligns with* **ETSI ZSM**
* **Platform** *aligns with* **TM Forum intent frameworks**
* **Alignment with 3GPP TS 28.312, ETSI ZSM, and TM Forum intent frameworks** *enables* **cross-SDO integration**

**Factoid**: Future work includes embedding LLMs to automate intent-to-workflow translation and reduce manual configuration effort.

* **Future work** *includes embedding* **LLMs**
* **Embedding LLMs** *automates* **intent-to-workflow translation**
* **Automating intent-to-workflow translation** *reduces* **manual configuration effort**

**Factoid**: Intent systems govern RAN, core, and data-center domains, adjusting parameters in RUs, DUs, CUs to meet performance or power objectives.

* **Intent systems** *govern* **RAN domain**
* **Intent systems** *govern* **core domain**
* **Intent systems** *govern* **data-center domain**
* **Intent systems** *adjust* **parameters in RUs**
* **Intent systems** *adjust* **parameters in DUs**
* **Intent systems** *adjust* **parameters in CUs**
* **Adjusting parameters in RUs, DUs, CUs** *aims to meet* **performance objectives**
* **Adjusting parameters in RUs, DUs, CUs** *aims to meet* **power objectives**

**Factoid**: Intents are constrained to pre-defined capabilities; they cannot create new functionality beyond the data model.

* **Intents** *are constrained to* **pre-defined capabilities**
* **Intents** *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.

* **ETSI ENI**, **ETSI ZSM**, **TM Forum IG1253**, and **3GPP TS 28.312** *are* **primary SDOs driving intent-driven network standards**

**Factoid**: Intent architectures require a policy/action engine, monitoring/assurance subsystem, and optional AI/ML-enhanced mapping.

* **Intent architectures** *require* **policy/action engine**
* **Intent architectures** *require* **monitoring/assurance subsystem**
* **Intent architectures** *include* **optional AI/ML-enhanced mapping**

**Factoid**: Policy translation regenerates and updates when intents change to keep intent-data models and actions aligned.

* **Policy translation** *regenerates* **intent-data models** when **intents change**
* **Policy translation** *updates* **actions** when **intents change**
* **Regeneration and updates** *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.

* **Gen-2 orchestration** *must span* **PNFs**
* **Gen-2 orchestration** *must span* **VNFs**
* **Gen-2 orchestration** *must span* **CNFs**
* **Gen-2 orchestration** *must span* **VMs**
* **Gen-2 orchestration** *must span* **containers**
* **Gen-2 orchestration** *must serve across* **private clouds**
* **Gen-2 orchestration** *must serve across* **public clouds**
* **Gen-2 orchestration** *must serve across* **edge clouds**

**Factoid**: Closed-loop telemetry is mandatory for dynamic reconfiguration and SLA compliance within intent-driven systems.

* **Closed-loop telemetry** *is mandatory for* **dynamic reconfiguration**
* **Closed-loop telemetry** *is mandatory for* **SLA compliance**
* **Dynamic reconfiguration and SLA compliance** *occur 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.

* **IETF’s intent model** *uses* **nested loops**
* **Nested loops** *include* **inner autonomous control loop**
* **Nested loops** *include* **outer user-in-the-loop refinement loop**

**Factoid**: Vendor data models vary, so intent systems need abstraction bridges across polymorphic device APIs and configurations.

* **Vendor data models** *vary*
* **Intent systems** *need* **abstraction bridges**
* **Abstraction bridges** *span* **polymorphic device APIs**
* **Abstraction bridges** *span* **configurations**

**Factoid**: AI/ML can enhance intent mapping and assurance but remains bounded by intended expressiveness of policy and data models.

* **AI/ML** *can enhance* **intent mapping**
* **AI/ML** *can enhance* **assurance**
* **AI/ML** *remains bounded by* **intended expressiveness of policy and data models**

**Factoid**: A slice is a customer SLA-backed end-to-end logical network running on shared infrastructure with performance bounds.

* **Slice** *is a* **customer SLA-backed end-to-end logical network**
* **End-to-end logical network** *runs on* **shared infrastructure**
* **End-to-end logical network** *has* **performance bounds**

**Factoid**: Slices comprise both dedicated and shared resources and must be logically isolated from one another.

* **Slices** *comprise* **dedicated resources**
* **Slices** *comprise* **shared resources**
* **Dedicated and shared resources** *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.

* **Business bundles** *allow* **operators to combine multiple slice types under a unified SLA**
* **Operators** *combine* **URLLC** and **eMBB** slice types
* **Combination of multiple slice types** *occurs under* **unified SLA**

**Factoid**: Generic Slice Templates define attribute domains (e.g., latency range, reliability targets) for network slices.

* **Generic Slice Templates** *define* **attribute domains**
* **Attribute domains** *include* **latency range**
* **Attribute domains** *include* **reliability targets**
* **Attribute domains** *apply to* **network slices**

**Factoid**: Network Slice Type instantiates a GST with specific values tailored to a vertical use case’s requirements.

* **Network Slice Type** *instantiates* **Generic Slice Template (GST)**
* **Network Slice Type** *applies* **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.

* **Slices** *can span* **RAN**, **Core**, and **Transport domains**
* **Slices** *can be shared across* **multiple operators**
* **Sharing slices across multiple operators** *supports* **roaming support**

**Factoid**: Operators publish standardized slice blueprints to visited PLMNs to preserve SLAs and enable slice continuity.

* **Operators** *publish* **standardized slice blueprints**
* **Standardized slice blueprints** *are published to* **visited PLMNs**
* **Publishing standardized slice blueprints to visited PLMNs** *preserves* **SLAs**
* **Publishing standardized slice blueprints to visited PLMNs** *enables* **slice continuity**

**Factoid**: Slice customization is capped by GST/NEST—customers cannot exceed operator-defined attribute ranges.

* **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.

* **Use cases like V2X and industrial IoT** *require* **elasticity**
* **Use cases like V2X and industrial IoT** *require* **strict traffic isolation from slices**

**Factoid**: Regulatory models must handle neutrality, data sovereignty, and cross-border slicing SLAs.

* **Regulatory models** *must handle* **neutrality**
* **Regulatory models** *must handle* **data sovereignty**
* **Regulatory models** *must handle* **cross-border slicing SLAs**

**Factoid**: Slice descriptors and SLAs are central to operator interoperability and roaming in sliced 5G environments.

* **Slice descriptors** *are central to* **operator interoperability**
* **Slice descriptors** *are central to* **roaming in sliced 5G environments**
* **SLAs** *are central to* **operator interoperability**
* **SLAs** *are central to* **roaming in sliced 5G environments**
* **Operator interoperability** *occurs in* **sliced 5G environments**
* **Roaming** *occurs in* **sliced 5G environments**

**Factoid**: 5G slicing enables dynamic per-slice traffic treatment, creating potential information asymmetries vis-à-vis regulators.

* **5G slicing** *enables* **dynamic per-slice traffic treatment**
* **Dynamic per-slice traffic treatment** *creates* **potential information asymmetries**
* **Potential information asymmetries** *occur vis-à-vis* **regulators**

**Factoid**: NWDAF can expose per-slice KPIs (packet loss, delay, QoS metrics) to external regulators via NEF.

* **NWDAF** *exposes* **per-slice KPIs**
* **Per-slice KPIs** *include* **packet loss**
* **Per-slice KPIs** *include* **delay**
* **Per-slice KPIs** *include* **QoS metrics**
* **NWDAF** *exposes per-slice KPIs to* **external regulators**
* **Exposure to external regulators** *occurs via* **NEF**

**Factoid**: NWDAF employs a consumer–producer model with standardized interfaces for periodic KPI and analytics distribution.

* **NWDAF** *employs* **consumer–producer model**
* **Consumer–producer model** *uses* **standardized interfaces**
* **Standardized interfaces** *support* **periodic KPI distribution**
* **Standardized interfaces** *support* **analytics distribution**

**Factoid**: Analytics functions include historical trend analysis, predictive modeling, anomaly detection, and QoS sustainability reports.

* **Analytics functions** *include* **historical trend analysis**
* **Analytics functions** *include* **predictive modeling**
* **Analytics functions** *include* **anomaly detection**
* **Analytics functions** *include* **QoS sustainability reports**

**Factoid**: A proof-of-concept showed eMBB slice analytics can be used by regulators to detect traffic differentiation.

* **Proof-of-concept** *showed* **eMBB slice analytics** *can be used by* **regulators**
* **eMBB slice analytics** *enables* **detection of traffic differentiation**
* **Regulators** *detect* **traffic differentiation**

**Factoid**: NWDAF enables real-time regulatory monitoring of net neutrality compliance based on slice-level transparency.

* **NWDAF** *enables* **real-time regulatory monitoring of net neutrality compliance**
* **Real-time regulatory monitoring of net neutrality compliance** *is based on* **slice-level transparency**

**Factoid**: Deployment options include SNPN, PNI‑NPN, and hybrid models with shared RAN and TSN integration.

* **Deployment options** *include* **SNPN  
  Deployment options** *include* **PNI-NPN**
* **Deployment options** *include* **hybrid models**
* **Hybrid models** *use* **shared RAN**
* **Hybrid models** *use* **TSN integration**

**Factoid**: Rel‑16 slicing lacks native support for industrial Ethernet integration; enhancements planned in Rel‑17.

* **Rel-16 slicing** *lacks* **native support for industrial Ethernet integration**
* **Enhancements for industrial Ethernet integration** *are planned in* **Rel-17**

**Factoid**: Security zones and VLAN-based segmentation enable dynamic isolation of traffic within slices.

* **Security zones** *enable* **dynamic isolation of traffic within slices**
* **VLAN-based segmentation** *enables* **dynamic isolation of traffic within slices**

**Factoid**: Operator policy defines tenant-customizable parameters like coverage area and TSN support levels.

* **Operator policy** *defines* **tenant-customizable parameters**
* **Tenant-customizable parameters** *include* **coverage area**
* **Tenant-customizable parameters** *include* **TSN support levels**

**Factoid**: Qualitative mapping of use-case requirements (latency, reliability) to NPN architecture types informs deployment choice.

* **Use-case requirements** *include* **latency**
* **Use-case requirements** *include* **reliability**
* **Use-case requirements** *are qualitatively mapped to* **NPN architecture types**
* **Qualitative mapping of use-case requirements to NPN architecture types** *informs* **deployment choice**

**Factoid**: Slice access can be geographically constrained using TAC or other location identifiers.

* **Slice access** *can be geographically constrained using* **TAC**
* **Slice access** *can be geographically constrained using* **other location identifiers**

**Factoid**: Edge cloud and TSN functions are co-located based on geo-proximity to meet tight latency targets.

* **Edge cloud and TSN functions** *are co-located based on* **geo-proximity**
* **Co-location based on geo-proximity** *enables* **meeting tight latency targets**

**Factoid**: Dynamic traffic-driven forwarding rules support adaptive slice isolation and performance optimization.

* **Dynamic traffic-driven forwarding rules** *support* **adaptive slice isolation**
* **Dynamic traffic-driven forwarding rules** *support* **performance optimization**

**Factoid**: A Maximum Slice Data Rate (UL/DL) can be configured by the operator per S‑NSSAI to cap slice-wide throughput.

* **Maximum Slice Data Rate (UL/DL)** *can be configured by* **operator**
* **Maximum Slice Data Rate (UL/DL)** *is configured per* **S-NSSAI**
* **Configuring Maximum Slice Data Rate (UL/DL)** *caps* **slice-wide throughput**

**Factoid**: PCF enforces Maximum Slice Data Rate by rejecting SM Policy or PDU session establishment if slice budget is exceeded.

* **PCF** *enforces* **Maximum Slice Data Rate**
* **PCF** *rejects* **SM Policy establishment**
* **PCF** *rejects* **PDU session establishment**
* **Rejection of SM Policy or PDU session establishment** *occurs if* **slice budget is exceeded**

**Factoid**: With NWDAF integration, PCF uses slice usage data (volume and duration) to apply or relax constraints dynamically.

* **PCF** *uses* **slice usage data (volume and duration)**
* **Slice usage data** *is provided by* **NWDAF integration**
* **PCF** *applies or relaxes* **constraints dynamically**

**Factoid**: Without NWDAF, the PCF deducts used capacity from UDR via Session-AMBR and MBR during flow provisioning.

* **PCF** *deducts* **used capacity** from **UDR** via **Session-AMBR**
* **PCF** *deducts* **used capacity** from **UDR** via **MBR**
* **Deducting used capacity via Session-AMBR and MBR** *occurs during* **flow provisioning**

**Factoid**: If PCF rejects due to exceeded data rate, SMF returns HTTP 403 'EXCEEDED\_SLICE\_DATA\_RATE' to UE.

* **PCF** *rejects* **flow setup 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.

* **Slice capacity** *is restored to* **UDR**
* **Restoration of slice capacity to UDR** *occurs when* **sessions terminate**
* **Restoration of slice capacity to UDR** *occurs when* **GBR rules are modified downward**

**Factoid**: Multiple PCFs can synchronously enforce slice caps using conditional UDR updates with etags.

* **Multiple PCFs** *can synchronously enforce* **slice caps**
* **Slice cap enforcement** *uses* **conditional UDR updates with etags**

**Factoid**: Maximum Group Data Rate control is extended to VN groups similarly to slice-based rate limiting.

* **Maximum Group Data Rate control** *is extended to* **VN groups**
* **Extension of Maximum Group Data Rate control to VN groups** *is similar to* **slice-based rate limiting**

**Factoid**: Emergency or prioritized services can bypass slice data rate limits based on operator policy or regulation.

* **Emergency services** *can bypass* **slice data rate limits**
* **Prioritized services** *can bypass* **slice data rate limits**
* **Bypassing slice data rate limits** *is based on* **operator policy**
* **Bypassing slice data rate limits** *is based on* **regulation**

**Factoid**: Slice caps apply across both GBR and non-GBR flows—Non‑GBR via Session-AMBR, GBR via MBR in PCC rules.

* **Slice caps** *apply across* **non-GBR flows**
* **Slice caps** *apply across* **GBR flows**
* **Non-GBR flows** *are enforced via* **Session-AMBR**
* **GBR flows** *are enforced via* **MBR in PCC rules**

**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.

* **ENISA** *identifies* **AUSF** as **critical asset requiring trust zone isolation**
* **ENISA** *identifies* **SEAF** as **critical asset requiring trust zone isolation**
* **ENISA** *identifies* **SEPP** as **critical asset requiring trust zone isolation**
* **ENISA** *identifies* **SIDP** as **critical asset requiring trust zone isolation**
* **ENISA** *identifies* **AMF** as **critical asset requiring trust zone isolation**
* **ENISA** *identifies* **UDM** as **critical asset requiring trust zone isolation**
* **ENISA** *identifies* **NSSAAF** as **critical asset requiring trust zone isolation**
* **ENISA** *identifies* **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.

* **ENISA’s threat landscape** *uses* **nine architectural ‘zoom-ins’**
* **Nine architectural ‘zoom-ins’** *map* **vulnerabilities**
* **Mapping vulnerabilities** *occurs across* **control-plane and user-plane (CP–UP)**
* **Mapping vulnerabilities** *occurs across* **NFV domain**
* **Mapping vulnerabilities** *occurs across* **SDN domain**
* **Mapping vulnerabilities** *occurs across* **MEC domain**
* **Mapping vulnerabilities** *occurs across* **slicing domain**
* **Mapping vulnerabilities** *occurs across* **orchestration domain**

**Factoid**: EU regulations (NISD, Telecom Security Act, Article 13a) require strict operational controls: IAM, configuration management, logging, continuity planning.

* **EU regulations (NISD, Telecom Security Act, Article 13a)** *require* **IAM**
* **EU regulations (NISD, Telecom Security Act, Article 13a)** *require* **configuration management**
* **EU regulations (NISD, Telecom Security Act, Article 13a)** *require* **logging**
* **EU regulations (NISD, Telecom Security Act, Article 13a)** *require* **continuity planning**

**Factoid**: Control‑plane functions are architecturally isolated from user‑plane, mitigating CP-targeted attacks without affecting UP traffic.

* **Control-plane functions** *are architecturally isolated from* **user-plane**
* **Architectural isolation** *mitigates* **CP-targeted attacks**
* **Mitigating CP-targeted attacks** *occurs without affecting* **UP traffic**

\_\_\_\_\_\_\_\_\_ BATCH 7 GEMINI \_\_\_\_\_\_\_\_\_\_

**Factoid**: MANO, NFVO, and VNF/VIM orchestrators are critical CP assets vulnerable to supply chain and virtualization-layer attacks.

* **MANO** *is* **critical CP asset**
* **NFVO** *is* **critical CP asset**
* **VNF orchestrators** *are* **critical CP assets**
* **VIM orchestrators** *are* **critical CP assets**
* **Critical CP assets** *are vulnerable to* **supply chain attacks**
* **Critical CP assets** *are vulnerable to* **virtualization-layer attacks**

**Factoid**: Attacks on CP APIs and slice orchestration functions can compromise multiple domains—ENISA recommends hardened authentication and policy enforcement.

* **Attacks on CP APIs and slice orchestration functions** *can compromise* **multiple domains**
* **ENISA** *recommends* **hardened authentication**
* **ENISA** *recommends* **policy enforcement**

**Factoid**: Operational resilience requirements include incident reporting, network audits, disaster recovery, and service continuity under EU telecom law.

* **Operational resilience requirements** *include* **incident reporting**
* **Operational resilience requirements** *include* **network audits**
* **Operational resilience requirements** *include* **disaster recovery**
* **Operational resilience requirements** *include* **service continuity**
* **Operational resilience requirements** *are mandated 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.

* **NFV systems** *centralize* **CP control**
* **NFV systems** *centralize* **UP control**
* **SDN systems** *centralize* **CP control  
  SDN systems** *centralize* **UP control  
  Centralization of CP and UP control** *makes* **NFV and SDN systems** *high-value attack vectors*
* **High-value attack vectors** *include* **DoS**
* **High-value attack vectors** *include* **configuration tampering**

**Factoid**: Trust boundaries are enforced using certificate-based isolation, role-based access, and dedicated API profiles per NF in control plane.

* **Trust boundaries** *are enforced using* **certificate-based isolation**
* **Trust boundaries** *are enforced using* **role-based access**
* **Trust boundaries** *are enforced using* **dedicated API profiles**
* **Dedicated API profiles** *apply 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.

* **Threat actors** *may exploit* **cross-domain trust misuse in orchestration**
* **Threat actors** *may exploit* **cross-domain trust misuse in roaming**
* **Exploitation of cross-domain trust misuse** *is part of* **multi-stage control-plane attacks**

**Factoid**: UPF serves as the primary User‑Plane NF in 5GC, handling packet forwarding, QoS, buffering, usage reporting, and mobility anchoring.

* **UPF** *serves as* **primary User-Plane NF in 5GC**
* **UPF** *handles* **packet forwarding**
* **UPF** *handles* **QoS**
* **UPF** *handles* **buffering**
* **UPF** *handles* **usage reporting**
* **UPF** *handles* **mobility anchoring**

**Factoid**: The UPF interconnects RAN and Data Networks, and performs reflective QoS marking and application-specific flow detection.

* **UPF** *interconnects* **RAN** and **Data Networks**
* **UPF** *performs* **reflective QoS marking**
* **UPF** *performs* **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).

* **UPF** *uses* **N3 reference point**
* **N3 reference point** *connects* **UPF** and **gNB**
* **UPF** *uses* **N4 reference point**
* **N4 reference point** *connects* **UPF** and **SMF** via **PFCP**
* **UPF** *uses* **N6 reference point**
* **N6 reference point** *connects* **UPF** and **Data Network**
* **UPF** *uses* **N9 reference point**
* **N9 reference point** *supports* **UPF-to-UPF chaining**

**Factoid**: PFCP over N4 installs forwarding (FAR), buffering (BAR), QoS (QER), usage (URR), and detection (PDR) rules in UPF.

* **PFCP over N4** *installs* **forwarding rules (FAR)** in **UPF**
* **PFCP over N4** *installs* **buffering rules (BAR)** in **UPF**
* **PFCP over N4** *installs* **QoS rules (QER)** in **UPF**
* **PFCP over N4** *installs* **usage rules (URR)** in **UPF**
* **PFCP over N4** *installs* **detection rules (PDR)** in **UPF**

**Factoid**: GTP‑U encapsulated traffic traverses N3/N9 for user-plane data exchange between RAN and/or UPF instances.

* **GTP-U encapsulated traffic** *traverses* **N3 interface**
* **GTP-U encapsulated traffic** *traverses* **N9 interface**
* **Traversal of N3 and N9** *facilitates* **user-plane data exchange**
* **User-plane data exchange** *occurs between* **RAN** and **UPF instances**

**Factoid**: UPF fulfills the CUPS model by decoupling data-plane processing from control-plane SMF logic and enabling edge deployment.

* **UPF** *fulfills* **Control and User Plane Separation (CUPS) model**
* **Control and User Plane Separation (CUPS) model** *decouples* **data-plane processing** from **control-plane SMF logic**
* **Decoupling of data-plane processing from control-plane SMF logic** *enables* **edge deployment**

**Factoid**: Cloud‑native UPF architectures using microservices and Kubernetes enable elastic scaling for diverse throughput demands.

* **Cloud-native UPF architectures** *use* **microservices**
* **Cloud-native UPF architectures** *use* **Kubernetes**
* **Use of microservices and Kubernetes** *enables* **elastic scaling**
* **Elastic scaling** *supports* **diverse throughput demands**

**Factoid**: SmartNIC offload (Napatech) enables UPFs to process dual 100 Gbps data-plane loads with zero CPU usage.

* **SmartNIC offload (Napatech)** *enables* **UPFs** to process **dual 100 Gbps data-plane loads**
* **Processing dual 100 Gbps data-plane loads** *occurs with* **zero CPU usage**

**Factoid**: Branching via N9 or uplink classification allows UPF to intelligently split or redirect traffic flows.

* **Branching via N9** *allows* **UPF** to split or redirect traffic flows intelligently
* **Uplink classification** *allows* **UPF** to split or redirect traffic flows intelligently

**Factoid**: UPF supports IPv4/IPv6, NAT at N6, multi-tenancy, session anchoring, and reflective QoS for end-to-end service compliance.

* **UPF** *supports* **IPv4/IPv6**
* **UPF** *supports* **NAT at N6**
* **UPF** *supports* **multi-tenancy**
* **UPF** *supports* **session anchoring**
* **UPF** *supports* **reflective QoS**
* **Reflective QoS** *enables* **end-to-end service compliance**

**Factoid**: UPF is the CUPS-based user-plane NF in 5GC, enabling separation from SMF and independent scaling/deployment.

* **UPF** *is* **the CUPS-based user-plane NF in 5GC**
* **UPF** *enables* **separation from SMF**
* **UPF** *enables* **independent scaling**
* **UPF** *enables* **independent deployment**

**Factoid**: CUPS allows UPF to evolve and scale independently, enabling edge deployment for low-latency use cases.

* **CUPS** *allows* **UPF to evolve independently**
* **CUPS** *allows* **UPF to scale independently**
* **Independent evolution and scaling of UPF** *enables* **edge deployment**
* **Edge deployment** *supports* **low-latency use cases**

**Factoid**: UPF handles mobility anchoring, IP allocation, routing, classification, branching, buffering, and PFD-based application detection.

* **UPF** *handles* **mobility anchoring**
* **UPF** *handles* **IP allocation**
* **UPF** *handles* **routing**
* **UPF** *handles* **classification**
* **UPF** *handles* **branching**
* **UPF** *handles* **buffering**
* **UPF** *handles* **PFD-based application detection**

**Factoid**: Packet-level policy enforcement in UPF includes gating, redirection, traffic steering, reflective QoS marking, and rate control.

* **Packet-level policy enforcement in UPF** *includes* **gating**
* **Packet-level policy enforcement in UPF** *includes* **redirection**
* **Packet-level policy enforcement in UPF** *includes* **traffic steering**
* **Packet-level policy enforcement in UPF** *includes* **reflective QoS marking**
* **Packet-level policy enforcement in UPF** *includes* **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.

* **N3 interface** *uses* **GTP-U**
* **N4 interface** *is used for* **PFCP rule installation**
* **N6 interface** *connects* **Data Network**
* **N9 interface** *enables* **UPF-to-UPF chaining**

**Factoid**: PFCP enables UPF to install PDR, FAR, BAR, QER, and URR via N4 under SMF control.

* **PFCP** *enables* **UPF** to install **PDR** via **N4** under **SMF control**
* **PFCP** *enables* **UPF** to install **FAR** via **N4** under **SMF control**
* **PFCP** *enables* **UPF** to install **BAR** via **N4** under **SMF control**
* **PFCP** *enables* **UPF** to install **QER** via **N4** under **SMF control**
* **PFCP** *enables* **UPF** to install **URR** via **N4** under **SMF control**

**Factoid**: N9 chaining supports branching, multi-anchoring, and breakout of traffic flows in complex deployment scenarios.

* **N9 chaining** *supports* **branching**
* **N9 chaining** *supports* **multi-anchoring**
* **N9 chaining** *supports* **breakout of traffic flows**
* **Branching, multi-anchoring, and breakout** *apply in* **complex deployment scenarios**

**Factoid**: Edge-deployed UPFs and cloud-native architecture support elastic scaling managed via orchestration platforms.

* **Edge-deployed UPFs** *support* **elastic scaling**
* **Cloud-native architecture** *supports* **elastic scaling**
* **Elastic scaling** *is managed via* **orchestration platforms**

**Factoid**: UPF supports GTP-U-level packet duplication and elimination for handover and redundancy.

* **UPF** *supports* **GTP-U-level packet duplication**
* **UPF** *supports* **GTP-U-level packet elimination**
* **Packet duplication and elimination** *are used for* **handover**
* **Packet duplication and elimination** *are used for* **redundancy**

**Factoid**: Intent-driven orchestration of UPF includes lifecycle automated deployment, test validation, and performance assurance using DevOps aligned tools.

* **Intent-driven orchestration of UPF** *includes* **automated lifecycle deployment**
* **Intent-driven orchestration of UPF** *includes* **test validation**
* **Intent-driven orchestration of UPF** *includes* **performance assurance**
* **Automated deployment, validation, and assurance** *use* **DevOps-aligned tools**

**Factoid**: GTP octet counters on N3 (GTP.In/OutDataOctetsN3) monitor transport bandwidth usage per QoS class.

* **GTP octet counters on N3** *include* **GTP.InDataOctetsN3** and **GTP.OutDataOctetsN3**
* **GTP.In/OutDataOctetsN3** *monitor* **transport bandwidth usage**
* **Transport bandwidth usage** *is tracked per* **QoS class**

**Factoid**: Packet loss on N3 is tracked via GTP.InDataPktLossN3QoS, enabling detection of per-class service degradation.

* **Packet loss on N3** *is tracked via* **GTP.InDataPktLossN3QoS**
* **GTP.InDataPktLossN3QoS** *enables 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.

* **Average and distribution metrics of N3 round-trip delay per DSCP** *are measured using* **GTP.RttDelayN3DlPsaUpfMean.DSCP**
* **GTP.RttDelayN3DlPsaUpfMean.DSCP** *supports* **QoS-level latency modelling**

**Factoid**: N4 PFCP session metrics measure N4SessionEstabReq/Fail and N4SessionReport/ReportSucc to assess control-plane signaling performance.

* **N4 PFCP session metrics** *measure* **N4SessionEstabReq**
* **N4 PFCP session metrics** *measure* **N4SessionEstabFail**
* **N4 PFCP session metrics** *measure* **N4SessionReport**
* **N4 PFCP session metrics** *measure* **N4SessionReportSucc**
* **These metrics** *assess* **control-plane signaling performance**

**Factoid**: N6 interface metrics (IP.N6IncLinkUsage, IP.N6OutLinkUsage) record IP-layer data volume, consistent with RFC 5136 counters.

* **N6 interface metrics** *include* **IP.N6IncLinkUsage**
* **N6 interface metrics** *include* **IP.N6OutLinkUsage**
* **IP.N6IncLinkUsage** and **IP.N6OutLinkUsage** *record* **IP-layer data volume**
* **These metrics** *are consistent with* **RFC 5136 counters**

**Factoid**: Per-QoS-class and per-slice measurements enable slice-specific SLA validation across UPF interfaces.

* **Per-QoS-class measurements** *enable* **slice-specific SLA validation**
* **Per-slice measurements** *enable* **slice-specific SLA validation**
* **Slice-specific SLA validation** *occurs across* **UPF interfaces**

**Factoid**: N3, N4, and N6 measurement data provide key inputs to dynamic QoS and scaling actions in intent-driven orchestration.

* **N3 measurement data** *provides* **key inputs to dynamic QoS and scaling actions**
* **N4 measurement data** *provides* **key inputs to dynamic QoS and scaling actions**
* **N6 measurement data** *provides* **key inputs to dynamic QoS and scaling actions**
* **Dynamic QoS and scaling actions** *are used in* **intent-driven orchestration**

**Factoid**: Monitoring of GTP RTT and packet loss informs real-time traffic steering or re-orchestration decisions.

* **Monitoring of GTP RTT** *informs* **real-time traffic steering**
* **Monitoring of GTP RTT** *informs* **re-orchestration decisions**
* **Monitoring of packet loss** *informs* **real-time traffic steering**
* **Monitoring of packet loss** *informs* **re-orchestration decisions**

**Factoid**: Control-plane PFCP session metrics support monitoring of SMF‑UPF coordination health.

* **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.

* **UESPLIT per-interface metrics** *feed* **operators’ QoS dashboards**
* **UESPLIT per-interface metrics** *feed* **NWDAF analytics**
* **Operators’ QoS dashboards and NWDAF analytics** *support* **performance assurance**

**Factoid**: UPCR (UPF placement and chaining reconfiguration) addresses UPF placement in MEC environments to handle user mobility.

* **UPCR (UPF placement and chaining reconfiguration)** *addresses* **UPF placement in MEC environments**
* **UPCR** *handles* **user mobility** through **UPF placement and chaining**

**Factoid**: An ILP-based model optimizes UPF placement to minimize deployment and migration costs under QoS constraints.

* **ILP-based model** *optimizes* **UPF placement**
* **Optimization of UPF placement** *minimizes* **deployment costs**
* **Optimization of UPF placement** *minimizes* **migration costs**
* **Optimization of UPF placement** *operates under* **QoS constraints**

**Factoid**: The DPC‑UPCR heuristic achieves within 15% of optimal placement outcomes with significantly reduced computation time.

* **DPC-UPCR heuristic** *achieves* **within 15% of optimal placement outcomes**
* **DPC-UPCR heuristic** *provides* **significantly reduced computation time**

**Factoid**: An optimal stopping theory–based scheduler triggers UPF reconfiguration when latency thresholds are breached.

* **Optimal stopping theory–based scheduler** *triggers* **UPF reconfiguration**
* **UPF reconfiguration** *is triggered when* **latency thresholds are breached**

**Factoid**: DPC‑UPCR outperforms baseline schedulers by reducing reconfig event count and QoS violations.

* **DPC-UPCR** *outperforms* **baseline schedulers**
* **DPC-UPCR** *reduces* **reconfiguration event count**
* **DPC-UPCR** *reduces* **QoS violations**

**Factoid**: UPF repositioning supports proximity anchoring to minimize latency during access node handovers.

* **UPF repositioning** *supports* **proximity anchoring**
* **Proximity anchoring** *minimizes* **latency during access node handovers**

**Factoid**: Cost-QoS trade-offs are balanced via dynamic UPF chaining instead of static placement clusters.

* **Cost-QoS trade-offs** *are balanced via* **dynamic UPF chaining**
* **Dynamic UPF chaining** *is used instead of* **static placement clusters**

**Factoid**: User mobility patterns drive dynamic reconfiguration decisions rather than purely periodic adjustments.

* **User mobility patterns** *drive* **dynamic reconfiguration decisions**
* **Dynamic reconfiguration decisions** *are preferred over* **purely periodic adjustments**

**Factoid**: Scheduling parameters—e.g., latency violation rate and QoS thresholds—govern when to deploy UPF migrations.

* **Scheduling parameters** *include* **latency violation rate**
* **Scheduling parameters** *include* **QoS thresholds**
* **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.

1. **Heuristic scheduler** *aims to minimize* **cost of migration**
2. **Heuristic scheduler** *preserves* **slice-level performance**
3. **Slice-level performance** *is preserved during* **reconfigurations**

**Factoid**: The 5G‑PPP architecture integrates multi-domain orchestration through ETSI NFV-MANO and SDN agents across RAN, transport, core, and edge.

* **5G-PPP architecture** *integrates* **multi-domain orchestration**
* **Multi-domain orchestration** *is enabled through* **ETSI NFV-MANO**
* **Multi-domain orchestration** *is enabled through* **SDN agents**
* **SDN agents** *operate across* **RAN**, **transport**, **core**, and **edge** domains

**Factoid**: RAN functions are split: DU/PHY/MAC run at edge for performance, while PDCP/RRC run centrally as VNFs.

* **RAN functions** *are split*
* **DU/PHY/MAC** *run at* **edge** *for performance*\*
* **PDCP/RRC** *run* **centrally** as **VNFs**

**Factoid**: Core network topology includes NFVI in core and edge, fronthaul/backhaul, and WLAN/small-cell components.

* **Core network topology** *includes* **NFVI in core**
* **Core network topology** *includes* **NFVI in edge**
* **Core network topology** *includes* **fronthaul**
* **Core network topology** *includes* **backhaul**
* **Core network topology** *includes* **WLAN components**
* **Core network topology** *includes* **small-cell components**

**Factoid**: CUPS architecture enables independent scaling by separating CU-CP from CU-UP and using SMF–UPF control-plane splits.

* **CUPS architecture** *enables* **independent scaling**
* **CUPS architecture** *separates* **CU-CP** from **CU-UP**
* **CUPS architecture** *uses* **SMF–UPF control-plane splits**

**Factoid**: Edge deployment of DU, CU-UP, and UPF ensures low-latency, bandwidth-intensive data handling close to users.

* **Edge deployment of DU, CU-UP, and UPF** *ensures* **low-latency data handling**
* **Edge deployment of DU, CU-UP, and UPF** *ensures* **bandwidth-intensive data handling**
* **Low-latency and bandwidth-intensive data handling** *occurs close to* **users**

**Factoid**: SDN agents co-locate with domain NFs and are managed via NFVO/VNFM, enabling programmable domain control.

* **SDN agents** *co-locate with* **domain NFs**
* **SDN agents** *are managed via* **NFVO/VNFM**
* **Management via NFVO/VNFM** *enables* **programmable domain control**

**Factoid**: Intent ingestion, telemetry, and AI/ML closed-loop orchestration are best-practice control methods for slice and service SLAs.

* **Intent ingestion** *is a* **best-practice control method** for **slice and service SLAs**
* **Telemetry** *is a* **best-practice control method** for **slice and service SLAs**
* **AI/ML closed-loop orchestration** *is a* **best-practice control method** for **slice and service SLAs**

**Factoid**: PHY and MAC layers benefit from hardware acceleration at the DU, while PDCP/RRC benefit from centralized programmability.

* **PHY and MAC layers** *benefit from* **hardware acceleration at the DU**
* **PDCP/RRC layers** *benefit from* **centralized programmability**

**Factoid**: Transport-level programmability leveraged via T‑API and intent-based NBIs integrated with ONF/SDN frameworks.

* **Transport-level programmability** *is leveraged via* **T-API**
* **Transport-level programmability** *is leveraged via* **intent-based NBIs**
* **T-API and intent-based NBIs** *are integrated with* **ONF/SDN frameworks**

**Factoid**: Application-aware orchestration includes plug-in service and function managers integrated into MANO for vertical-specific services.

* **Application-aware orchestration** *includes* **plug-in service managers**
* **Application-aware orchestration** *includes* **plug-in function managers**
* **Plug-in service and function managers** *are integrated into* **MANO**
* **Application-aware orchestration** *supports* **vertical-specific services**

**Factoid**: Kubernetes is used for edge orchestration to support sub‑20 ms latency use cases in IoT and real-time applications.

* **Kubernetes** *is used for* **edge orchestration**
* **Edge orchestration with Kubernetes** *supports* **sub-20 ms latency use cases**
* **Sub-20 ms latency use cases** *include* **IoT applications**
* **Sub-20 ms latency use cases** *include* **real-time applications**

**Factoid**: Native Kubernetes lacks network‑aware scheduling and topology‑based placement essential for edge scenarios.

* **Native Kubernetes** *lacks* **network-aware scheduling**
* **Native Kubernetes** *lacks* **topology-based placement**
* **Network-aware scheduling and topology-based placement** *are essential for* **edge scenarios**

**Factoid**: Custom edge Kubernetes deployments use virtual kubelets, multi‑cluster federation, and custom schedulers.

* **Custom edge Kubernetes deployments** *use* **virtual kubelets**
* **Custom edge Kubernetes deployments** *use* **multi-cluster federation**
* **Custom edge Kubernetes deployments** *use* **custom schedulers**

**Factoid**: Edge K8s implementations struggle with real-time metric processing, fault-tolerance, and container registry placement.

* **Edge Kubernetes implementations** *struggle with* **real-time metric processing**
* **Edge Kubernetes implementations** *struggle with* **fault-tolerance**
* **Edge Kubernetes implementations** *struggle with* **container registry placement**

\_\_\_\_\_\_\_\_\_ BATCH 8 GEMINI \_\_\_\_\_\_\_\_\_\_

**Factoid**: Edge-focused improvements include installing local metrics servers, topology-aware custom schedulers, and edge-located container registries.

* **Edge-focused improvements** *include* **installing local metrics servers**
* **Edge-focused improvements** *include* **topology-aware custom schedulers**
* **Edge-focused improvements** *include* **edge-located container registries**

**Factoid**: Edge orchestration requires orchestration decisions based on realtime network measurements and locality knowledge.

* **Edge orchestration** *requires* **orchestration decisions based on real-time network measurements**
* **Edge orchestration** *requires* **orchestration decisions based on locality knowledge**

**Factoid**: Fault tolerance at the edge demands resilient control planes and backup mechanisms for remote node failures.

* **Fault tolerance at the edge** *demands* **resilient control planes**
* **Fault tolerance at the edge** *demands* **backup mechanisms for remote node failures**

**Factoid**: Placing container registries on edge nodes reduces network usage and speeds up workload deployment.

* **Placing container registries on edge nodes** *reduces* **network usage**
* **Placing container registries on edge nodes** *speeds up* **workload deployment**

**Factoid**: Virtual kubelets enable edge nodes to publish as Kubernetes nodes without running full control plane components.

* **Virtual kubelets** *enable* **edge nodes to publish as Kubernetes nodes**
* **Virtual kubelets** *operate without* **running full control plane components**

**Factoid**: Topology‑aware scheduling uses knowledge of node network connectivity to prioritize workload placement.

* **Topology-aware scheduling** *uses* **knowledge of node network connectivity**
* **Knowledge of node network connectivity** *is used to prioritize* **workload placement**

**Factoid**: Container placement spans cloud, fog, and edge domains, optimizing resource use, energy consumption, and fault tolerance.

* **Container placement** *spans* **cloud domain**
* **Container placement** *spans* **fog domain**
* **Container placement** *spans* **edge domain**
* **Container placement** *optimizes* **resource use**
* **Container placement** *optimizes* **energy consumption**
* **Container placement** *optimizes* **fault tolerance**

**Factoid**: Placement algorithms include optimization-based (e.g. LP, bin-packing), meta-heuristics (ACO, genetic), and reinforcement learning (MDP).

* **Placement algorithms** *include* **optimization-based methods**
* **Optimization-based methods** *include* **Linear Programming (LP)** and **bin-packing**
* **Placement algorithms** *include* **meta-heuristics**
* **Meta-heuristics** *include* **Ant Colony Optimization (ACO)** and **genetic algorithms**
* **Placement algorithms** *include* **reinforcement learning methods**
* **Reinforcement learning methods** *include* **Markov Decision Process (MDP)**

**Factoid**: Migration techniques include cold, pre-copy, post-copy, and hybrid methods, each with different downtime/resource overhead profiles.

* **Migration techniques** *include* **cold migration**
* **Migration techniques** *include* **pre-copy migration**
* **Migration techniques** *include* **post-copy migration**
* **Migration techniques** *include* **hybrid migration**
* **Each migration method** *has different* **downtime and resource overhead profiles**

**Factoid**: Edge/fog placements consider latency, energy, and resource constraints in trade-off-aware optimization.

* **Edge/fog placements** *consider* **latency constraints**
* **Edge/fog placements** *consider* **energy constraints**
* **Edge/fog placements** *consider* **resource constraints**
* **Latency, energy, and resource constraints** *are balanced 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.

* **Pre-copy migration** *reduces* **downtime** *via* **iterative state replication**
* **Post-copy migration** *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.

* **Meta-heuristic strategies** *include* **Ant Colony Optimization (ACO)**
* **Meta-heuristic strategies** *include* **Grey Wolf Optimization**
* **ACO and Grey Wolf Optimization** *optimize* **container placement**
* **Container placement optimization** *targets* **load balancing**
* **Container placement optimization** *targets* **energy efficiency**

**Factoid**: Fault tolerance in edge setups is supported via proactive migrations and container replication to avoid single-node failures.

* **Fault tolerance in edge setups** *is supported via* **proactive migrations**
* **Fault tolerance in edge setups** *is supported via* **container replication**
* **Proactive migrations and container replication** *help avoid* **single-node failures**

**Factoid**: Energy-aware placement utilizes bin-packing to minimize the number of active fog nodes, saving power.

* **Energy-aware placement** *utilizes* **bin-packing**
* **Bin-packing** *minimizes* **number of active fog nodes**
* **Minimizing active fog nodes** *saves* **power**

**Factoid**: Placement models often include QoS constraints, formulated using MILP or knapsack to enforce latency and reliability SLAs.

* **Placement models** *often include* **QoS constraints**
* **QoS constraints** *are formulated using* **Mixed Integer Linear Programming (MILP)**
* **QoS constraints** *are formulated using* **knapsack problem models**
* **Formulated constraints** *enforce* **latency SLAs**
* **Formulated constraints** *enforce* **reliability SLAs**

**Factoid**: Reinforcement learning enables adaptive container placement sensitive to real-time metrics and evolving constraints.

* **Reinforcement learning** *enables* **adaptive container placement**
* **Adaptive container placement** *is sensitive to* **real-time metrics**
* **Adaptive container placement** *responds to* **evolving constraints**

**Factoid**: VNF and container placement across cloud, fog, and edge are NP-hard and directly impact latency, cost, energy, and scalability.

* **VNF and container placement** *across* **cloud, fog, and edge** *are* **NP-hard**
* **VNF and container placement** *directly impact* **latency**
* **VNF and container placement** *directly impact* **cost**
* **VNF and container placement** *directly impact* **energy**
* **VNF and container placement** *directly impact* **scalability**

**Factoid**: Placement techniques are classified as optimization-based (e.g., LP, MIQCP), meta-heuristics (ACO, GWO), or ML-based (reinforcement learning, deep RL).

* **Placement techniques** *are classified as* **optimization-based**
* **Optimization-based techniques** *include* **Linear Programming (LP)** and **Mixed-Integer Quadratically Constrained Programming (MIQCP)**
* **Placement techniques** *are classified as* **meta-heuristics**
* **Meta-heuristics** *include* **Ant Colony Optimization (ACO)** and **Grey Wolf Optimization (GWO)**
* **Placement techniques** *are classified as* **ML-based**
* **ML-based techniques** *include* **reinforcement learning** and **deep reinforcement learning (deep RL)**

**Factoid**: Edge placements prioritize low latency, energy efficiency, resource constraints, and intermittent connectivity.

* **Edge placements** *prioritize* **low latency**
* **Edge placements** *prioritize* **energy efficiency**
* **Edge placements** *prioritize* **resource constraints**
* **Edge placements** *prioritize* **intermittent connectivity**

**Factoid**: Multi-objective placement models balance latency, energy, cost, resource use, fault tolerance, and QoS among VNFs/containers.

* **Multi-objective placement models** *balance* **latency**
* **Multi-objective placement models** *balance* **energy**
* **Multi-objective placement models** *balance* **cost**
* **Multi-objective placement models** *balance* **resource use**
* **Multi-objective placement models** *balance* **fault tolerance**
* **Multi-objective placement models** *balance* **QoS**
* **Balanced objectives** *apply to* **VNFs and containers**

**Factoid**: Reinforcement learning enables dynamic adaptation of placement strategies in response to traffic and resource fluctuations.

* **Reinforcement learning** *enables* **dynamic adaptation of placement strategies**
* **Dynamic adaptation of placement strategies** *responds to* **traffic fluctuations**
* **Dynamic adaptation of placement strategies** *responds to* **resource fluctuations**

**Factoid**: Meta-heuristic algorithms like ant-colony and grey wolf are widely used for balancing load, energy, and QoS objectives.

* **Meta-heuristic algorithms** *include* **ant-colony optimization**
* **Meta-heuristic algorithms** *include* **grey wolf optimization**
* **Ant-colony and grey wolf algorithms** *are widely used for* **balancing load**
* **Ant-colony and grey wolf algorithms** *are widely used for* **balancing energy**
* **Ant-colony and grey wolf algorithms** *are widely used for* **balancing QoS objectives**

**Factoid**: Optimization models like MIQCP or bin‑packing address latency and energy constraints, especially in uRLLC scenarios.

* **Optimization models** *include* **MIQCP (Mixed-Integer Quadratically Constrained Programming)**
* **Optimization models** *include* **bin-packing**
* **MIQCP and bin-packing** *address* **latency constraints**
* **MIQCP and bin-packing** *address* **energy constraints**
* **Latency and energy constraints** *are especially relevant in* **uRLLC scenarios**

**Factoid**: Dynamic placement systems migrate VNFs/containers proactively to maintain SLAs in changing network states.

* **Dynamic placement systems** *migrate* **VNFs/containers proactively**
* **Proactive migration of VNFs/containers** *maintains* **SLAs**
* **Proactive migration** *responds to* **changing network states**

**Factoid**: Container placement extends beyond VNFs to include microservices in MEC and private 5G environments.

* **Container placement** *extends beyond* **VNFs**
* **Container placement** *includes* **microservices**
* **Microservices** *are placed in* **MEC environments**
* **Microservices** *are placed in* **private 5G environments**

**Factoid**: Taxonomy connects placement techniques to objectives—single vs multi-objective, static vs dynamic, method class used.

* **Taxonomy** *connects* **placement techniques** to **objectives**
* **Objectives** *include* **single-objective** and **multi-objective**
* **Placement techniques** *are categorized as* **static** or **dynamic**
* **Placement techniques** *are grouped by* **method class used** (e.g., optimization-based, meta-heuristic, ML-based)

**Factoid**: C-RAN base station functions (BBU/RRH) can be containerized using Docker and orchestrated via Kubernetes.

* **C-RAN base station functions** *include* **BBU (Baseband Unit)** and **RRH (Remote Radio Head)**
* **BBU/RRH functions** *can be containerized using* **Docker**
* **Containerized BBU/RRH functions** *can be orchestrated via* **Kubernetes**

**Factoid**: Kubernetes StatefulSets provide pod identity and ordering guarantees for RAN component scaling.

* **Kubernetes StatefulSets** *provide* **pod identity guarantees**
* **Kubernetes StatefulSets** *provide* **pod ordering guarantees**
* **Pod identity and ordering guarantees** *support* **RAN component scaling**

**Factoid**: Calico CNI enables layer-3 networking among RAN containers in orchestration deployments.

* **Calico CNI** *enables* **layer-3 networking**
* **Layer-3 networking** *operates among* **RAN containers**
* **Calico CNI** *is used in* **orchestration deployments**

**Factoid**: etcd acts as a distributed key-value store for dynamic runtime configuration between containerized RRH and BBU modules.

* **etcd** *acts as* **a distributed key-value store**
* **etcd** *supports* **dynamic runtime configuration**
* **Dynamic runtime configuration** *occurs between* **containerized RRH and BBU modules**

**Factoid**: Doubling the number of BBU–RRH pods linearly increases fronthaul data throughput in the containerized RAN testbed.

* **Doubling the number of BBU–RRH pods** *linearly increases* **fronthaul data throughput**
* **Linear increase in fronthaul data throughput** *is observed in* **containerized RAN testbed**

**Factoid**: OAISIM-based RRH emulation consumes over 60% CPU, which is useful for creating autoscaling policies.

* **OAISIM-based RRH emulation** *consumes* **over 60% CPU**
* **High CPU consumption** *is useful for creating* **autoscaling policies**

**Factoid**: Container orchestration enables dynamic resource-driven scaling of RAN baseband functions for performance and efficiency.

* **Container orchestration** *enables* **dynamic resource-driven scaling**
* **Dynamic scaling** *applies to* **RAN baseband functions**
* **Dynamic scaling of RAN baseband functions** *improves* **performance**
* **Dynamic scaling of RAN baseband functions** *improves* **efficiency**

**Factoid**: Resource metrics (CPU, memory) from Kubernetes can trigger scaling based on observed load patterns.

* **Resource metrics (CPU, memory)** *from* **Kubernetes**
* **Kubernetes resource metrics** *can trigger* **scaling actions**
* **Scaling actions** *are based on* **observed load patterns**

**Factoid**: Container-based virtualization (Docker/LXC) in real-time industrial systems incurs latency jitter (≈37–102 ms), posing challenges for hard real-time constraints

* **Container-based virtualization (Docker/LXC)** *in real-time industrial systems* *incurs* **latency jitter (≈37–102 ms)**
* **Latency jitter** *poses 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.

* **Soft real-time deadlines (50–100 ms)** *are feasible with* **containers**
* **Feasibility of soft real-time deadlines** *requires* **RT-kernel tuning on the host**
* **Feasibility of soft real-time deadlines** *requires* **orchestration support for deterministic latency**

**Factoid**: Lack of built‑in real-time scheduling in container runtimes requires use of RT-patched kernels or RT-specific orchestration frameworks.

* **Lack of built-in real-time scheduling in container runtimes** *requires* **use of RT-patched kernels**
* **Lack of built-in real-time scheduling in container runtimes** *requires* **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.

* **Experimental task latency** *varies by* **virtualization type**
* **Virtual PLCs** *experience latency of* **37–102 ms**
* **Kernel-level containers** *experience latency of* **47–54 ms**

**Factoid**: Current orchestration platforms (e.g., Kubernetes) lack real-time awareness—missing scheduling priorities and time-bound placement logic.

* **Current orchestration platforms (e.g., Kubernetes)** *lack* **real-time awareness**
* **Lack of real-time awareness** *includes missing* **scheduling priorities**
* **Lack of real-time awareness** *includes missing* **time-bound placement logic**

**Factoid**: Industrial-grade container platforms need integration of real-time metrics in orchestration controllers to meet strict latency SLAs.

* **Industrial-grade container platforms** *need* **integration of real-time metrics in orchestration controllers**
* **Integration of real-time metrics** *is required 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.

* **Real-time container maturity** *is still limited*
* **Jitter and unpredictable scheduling** *up to* **100 ms**
* **Jitter and scheduling unpredictability** *impede* **hard real-time use**

**Factoid**: Future solution directions include real-time scheduling extensions, metrics‑driven orchestration, and explicit task-level prioritization.

* **Future solution directions** *include* **real-time scheduling extensions**
* **Future solution directions** *include* **metrics-driven orchestration**
* **Future solution directions** *include* **explicit task-level prioritization**

**Factoid**: Time-sensitive container requirements demand enhancements at both runtime (cgroup, RT kernel) and orchestration (placement, scaling) layers.

* **Time-sensitive container requirements** *demand enhancements at* **runtime layer**
* **Runtime layer enhancements** *include* **cgroup** and **RT kernel**
* **Time-sensitive container requirements** *demand enhancements at* **orchestration layer**
* **Orchestration layer enhancements** *include* **placement** and **scaling**

**Factoid**: Systematic surveys identify container overhead as a key bottleneck for industrial control loops, guiding research into latency-aware virtualization.

* **Systematic surveys** *identify* **container overhead** as a **key bottleneck for industrial control loops**
* **Identification of container overhead** *guides* **research into latency-aware virtualization**

**Factoid**: Control-plane (N2) and user-plane (N3) traffic require separate IP subnets, unless using the same VLAN.

* **Control-plane (N2) traffic** *requires* **a separate IP subnet**
* **User-plane (N3) traffic** *requires* **a separate IP subnet**
* **Separate IP subnets** *are not required if* **N2 and N3 use the same VLAN**

**Factoid**: Edge deployments use dedicated subnets or UE IP pools, allocated via N6 interfaces for dynamic and static addresses.

* **Edge deployments** *use* **dedicated subnets**
* **Edge deployments** *use* **UE IP pools**
* **Dedicated subnets and UE IP pools** *are allocated via* **N6 interfaces**
* **N6 interfaces** *support allocation of* **dynamic addresses**
* **N6 interfaces** *support allocation of* **static addresses**

**Factoid**: NAPT must be disabled on N6 when external servers need to initiate connections to UEs.

* **NAPT** *must be disabled on* **N6**
* **Disabling NAPT on N6** *is required 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.

* **Firewalls** *must allow* **routes between corporate networks and UE subnets**
* **Firewalls** *must allow* **port access between corporate networks and UE subnets**
* **UE subnets** *are connected via* **N6 interface**

**Factoid**: HA setups use active‑standby ASE pairs with VRRP and BFD to maintain control- and user-plane continuity within 2.5s failover.

* **HA setups** *use* **active-standby ASE pairs**
* **Active-standby ASE pairs** *utilize* **VRRP** and **BFD**
* **VRRP and BFD** *maintain* **control- and user-plane continuity**
* **Failover continuity** *is maintained within* **2.5 seconds**

**Factoid**: Gateway routers require single static routes per network, with virtual IPs managed across redundant ASE devices.

* **Gateway routers** *require* **single static routes per network**
* **Virtual IPs** *are 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.

* **Separate IP pools** *are needed for* **access traffic**
* **Separate IP pools** *are needed for* **data traffic**
* **Separate IP pools** *are needed for* **management interfaces**
* **Separate IP pools** *are needed for* **cluster nodes**
* **Separate IP pools** *are needed for* **ACS/NFS services**
* **Separate IP pools** *are needed for* **virtual IP services**
* **These IP pools** *are required in* **ASE deployments**

**Factoid**: Azure reserves five private IPs in each subnet, limiting available resources for NF instances.

* **Azure** *reserves* **five private IPs in each subnet**
* **Reserved private IPs** *limit* **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.

* **UE IP address pool** *is specified in* **CIDR block**
* **CIDR block** *is passed to* **AP5GC**
* **CIDR block** *is passed during* **site deployment**
* **Site deployment** *uses* **ARM template**

**Factoid**: Proper segmentation of N2/N3/N6 subnets supports network isolation, QoS, and firewall rule enforcement.

* **Proper segmentation of N2/N3/N6 subnets** *supports* **network isolation**
* **Proper segmentation of N2/N3/N6 subnets** *supports* **QoS enforcement**
* **Proper segmentation of N2/N3/N6 subnets** *supports* **firewall rule enforcement**

**Factoid**: IPv6 is preferred in private 5G deployments for its address space, built-in IPsec, and autoconfiguration (SLAAC).

* **IPv6** *is preferred in* **private 5G deployments**
* **Preference for IPv6** *is due to* **larger address space**
* **Preference for IPv6** *is due to* **built-in IPsec support**
* **Preference for IPv6** *is due to* **autoconfiguration (SLAAC)**

**Factoid**: IPv4 address pools are limited and require NAT, which introduces latency and complexity, especially in URLLC contexts.

* **IPv4 address pools** *are limited*
* **Limited IPv4 pools** *require* **NAT**
* **NAT** *introduces* **latency**
* **NAT** *introduces* **complexity  
  Latency and complexity from NAT** *are problematic in* **URLLC contexts**

**Factoid**: InterLIR recommends a dual-stack approach during migration from IPv4 to IPv6 in private 5G networks.

* **InterLIR** *recommends* **a dual-stack approach**
* **Dual-stack approach** *is used during* **migration from IPv4 to IPv6**
* **Migration from IPv4 to IPv6** *applies to* **private 5G networks**

**Factoid**: RFC 1918 private IPv4 ranges are used to isolate network slices and device classes in private 5G.

* **RFC 1918 private IPv4 ranges** *are used to* **isolate network slices**
* **RFC 1918 private IPv4 ranges** *are used to* **isolate device classes**
* **Isolation of slices and devices** *occurs in* **private 5G networks**

**Factoid**: DHCP is used for dynamic IP assignment; static pools may serve fixed-function devices.

* **DHCP** *is used for* **dynamic IP assignment**
* **Static IP pools** *may serve* **fixed-function devices**

**Factoid**: NAT segmentation across slices enhances isolation and security in multi-slice deployments.

* **NAT segmentation across slices** *enhances* **isolation**
* **NAT segmentation across slices** *enhances* **security**
* **Isolation and security improvements** *apply to* **multi-slice deployments**

**Factoid**: Automated IPAM systems with monitoring capabilities are critical for conflict detection and utilization visibility.

* **Automated IPAM systems** *with monitoring capabilities* *are critical for* **conflict detection**
* **Automated IPAM systems** *are critical for* **utilization visibility**

\_\_\_\_\_\_\_ BATCH 9 GEMINI \_\_\_\_\_\_

**Factoid**: Real-time alerts from IPAM aid in rogue device detection and prevent IP conflicts in private 5G environments.

* **Real-time alerts from IPAM** *aid in* **rogue device detection**
* **Real-time alerts from IPAM** *prevent* **IP conflicts**
* **Rogue device detection and IP conflict prevention** *are essential in* **private 5G environments**

**Factoid**: IPv6 eliminates NAT overhead, improving latency performance in low-latency applications.

* **IPv6** *eliminates* **NAT overhead**
* **Elimination of NAT overhead** *improves* **latency performance**
* **Improved latency performance** *benefits* **low-latency applications**

**Factoid**: Dual-stack deployment supports gradual IPv6 adoption while maintaining IPv4 compatibility with legacy devices.

* **Dual-stack deployment** *supports* **gradual IPv6 adoption**
* **Dual-stack deployment** *maintains* **IPv4 compatibility with legacy devices**

**Factoid**: In CUPS, SMF dynamically assigns IP chunks to UPFs; UPFs allocate individual IPs per UE and report usage back to SMF.

* **In CUPS**, **SMF** *dynamically assigns* **IP chunks to UPFs**
* **UPFs** *allocate* **individual IPs per UE**
* **UPFs** *report* **IP usage back to SMF**

**Factoid**: Cisco’s Cloud‑Native IPAM includes a central IPAM Server and distributed IPAM Caches for multi-cluster consistency.

* **Cisco’s Cloud-Native IPAM** *includes* **a central IPAM Server**
* **Cisco’s Cloud-Native IPAM** *includes* **distributed IPAM Caches**
* **Central IPAM Server and distributed IPAM Caches** *ensure* **multi-cluster consistency**

**Factoid**: SMF triggers new IP chunk allocations when UPF usage exceeds 70%, and reclaims underutilized chunks.

* **SMF** *triggers* **new IP chunk allocations** when **UPF usage exceeds 70%**
* **SMF** *reclaims* **underutilized IP chunks**

**Factoid**: Optimal chunk size balances even load distribution against IP exhaustion and provisioning overhead.

* **Optimal chunk size** *balances* **even load distribution**
* **Optimal chunk size** *balances* **IP exhaustion**
* **Optimal chunk size** *balances* **provisioning overhead**

**Factoid**: Maximum UPFs per chunk-limited pool are determined by pool size divided by chunk size (e.g., 1M/65k → ~16 UPFs).

* **Maximum UPFs per chunk-limited pool** *are determined by* **pool size divided by chunk size**
* **Example calculation**: **1M IPs / 65k per chunk** → **~16 UPFs**

**Factoid**: UP Groups tied to APN define chunk associations; dynamic pool updates work without Sx reassociation.

* **UP Groups tied to APN** *define* **chunk associations**
* **Dynamic pool updates** *operate without requiring* **Sx reassociation**

**Factoid**: DNS-based UPF selection uses TAC/RAC to bind UPF and chunk assignment in geo-aware deployments.

* **DNS-based UPF selection** *uses* **TAC (Tracking Area Code)** and **RAC (Routing Area Code)**
* **TAC/RAC** *bind* **UPF and chunk assignment**
* **Binding UPF and chunk assignment** *enables* **geo-aware deployments**

**Factoid**: SMF enforces chunk throttling based on UPF’s advertised max session capacity to prevent over-allocation.

* **SMF** *enforces* **chunk throttling**
* **Chunk throttling** *is based on* **UPF’s advertised max session capacity**
* **Throttling** *prevents* **over-allocation**

**Factoid**: Static IP pools are split and distributed to multiple UPFs; SMF rejects requests outside defined static blocks.

* **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.

* **DHCP-based IP allocation via UPF** *requires* **PFCP N4 NF integration**
* **DHCP-based allocation** *requires* **VLAN ID tagging**
* **DHCP-based allocation** *requires* **DHCP Dora exchange**

**Factoid**: Cisco best practice suggests hierarchical subnet design with Access, Distribution, and Core layers to improve scalability and resilience.

* **Cisco best practice** *suggests* **hierarchical subnet design**
* **Hierarchical subnet design** *includes* **Access layer**
* **Hierarchical subnet design** *includes* **Distribution layer**
* **Hierarchical subnet design** *includes* **Core layer**
* **Access, Distribution, and Core layers** *improve* **scalability and resilience**

**Factoid**: Subnetting divides an IP space into network and host portions, ensuring uniqueness and route summarization.

* **Subnetting** *divides* **an IP space into network and host portions**
* **Subnetting** *ensures* **address uniqueness**
* **Subnetting** *enables* **route summarization**

**Factoid**: VLSM allows subnets of varying sizes within a block—key for allocating optimal address ranges per NF/service.

* **VLSM (Variable Length Subnet Masking)** *allows* **subnets of varying sizes within a block**
* **VLSM** *is 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.

* **IPv6’s 16-bit Subnet ID** *supports* **up to 65,535 subnets** under a **/64 prefix**
* **65,535 subnets under /64** *enable* **zonal network segmentation**

**Factoid**: Cisco recommends using modified EUI-64 interface IDs for stable, unique host addressing in IPv6 subnets.

* **Cisco** *recommends using* **modified EUI-64 interface IDs**
* **Modified EUI-64 interface IDs** *provide* **stable, unique host addressing**
* **Stable, unique host addressing** *is applied in* **IPv6 subnets**

**Factoid**: Hierarchical design localizes network changes, improves fault isolation, and reduces routing table size.

* **Hierarchical design** *localizes* **network changes**
* **Hierarchical design** *improves* **fault isolation**
* **Hierarchical design** *reduces* **routing table size**

**Factoid**: Typical subnet sizes include /28 for NF instances and /24 for core uplinks, providing headroom for growth.

* **Typical subnet size /28** *is used for* **NF instances**
* **Typical subnet size /24** *is used for* **core uplinks**
* **/28 and /24 subnet sizing** *provides* **headroom for growth**

**Factoid**: Hierarchical and CIDR-based subnetting enhances scalability and ease of management in telecom infrastructures.

* **Hierarchical subnetting** *enhances* **scalability**
* **Hierarchical subnetting** *enhances* **ease of management**
* **CIDR-based subnetting** *enhances* **scalability**
* **CIDR-based subnetting** *enhances* **ease of management**
* **Scalability and manageability improvements** *apply to* **telecom infrastructures**

**Factoid**: Separated subnets per network plane (control, user, management) aid in traffic isolation and QoS policy enforcement.

* **Separated subnets per network plane (control, user, management)** *aid in* **traffic isolation**
* **Separated subnets per network plane (control, user, management)** *aid in* **QoS policy enforcement**

**Factoid**: Efficient address planning prevents waste from fixed-length subnetting and traps address exhaustion before it occurs.

* **Efficient address planning** *prevents* **waste from fixed-length subnetting**
* **Efficient address planning** *prevents* **address exhaustion before it occurs**

**Factoid**: ZTE classifies UPFs into Central, Regional, Edge, and Campus deployments, each with defined throughput and latency metrics.

* **ZTE** *classifies* **UPFs** into **Central**, **Regional**, **Edge**, and **Campus** deployments
* **Each UPF deployment type** *has defined* **throughput metrics**
* **Each UPF deployment type** *has defined* **latency metrics**

**Factoid**: Campus UPFs offer ~50 Gbps throughput with sub‑15 ms latency and include TSN, URLLC, and 5G‑LAN enhancements.

* **Campus UPFs** *offer* **~50 Gbps throughput**
* **Campus UPFs** *deliver* **sub‑15 ms latency**
* **Campus UPFs** *include* **TSN enhancements**
* **Campus UPFs** *include* **URLLC enhancements**
* **Campus UPFs** *include* **5G‑LAN enhancements**

**Factoid**: Subnet planning for UPFs defines separate IP ranges per interface: N3, N4, N6, ensuring flow isolation and QoS.

* **Subnet planning for UPFs** *defines* **separate IP ranges per interface**
* **Separate IP ranges** *are assigned for* **N3 interface**
* **Separate IP ranges** *are assigned for* **N4 interface**
* **Separate IP ranges** *are assigned for* **N6 interface**
* **Separate IP ranges per interface** *ensure* **flow isolation**
* **Separate IP ranges per interface** *ensure* **QoS enforcement**

**Factoid**: VLAN or subnet segmentation at edge/campus UPFs enables localized firewall and NAT policies.

* **VLAN or subnet segmentation at edge/campus UPFs** *enables* **localized firewall policies**
* **VLAN or subnet segmentation at edge/campus UPFs** *enables* **localized NAT policies**

**Factoid**: Local firewall/NAT in campus UPFs ensures enterprise-grade security and traffic control at the site level.

* **Local firewall/NAT in campus UPFs** *ensures* **enterprise-grade security**
* **Local firewall/NAT in campus UPFs** *ensures* **traffic control at the site level**

**Factoid**: UPFs apply DNN or slice‑based policy steering, allowing flexible access control across user data flows.

* **UPFs** *apply* **DNN-based policy steering**
* **UPFs** *apply* **slice-based policy steering**
* **DNN or slice-based policy steering** *allows* **flexible access control across user data flows**

**Factoid**: Separate subnets for user, control, and data traffic per plane support QoS isolation and policy enforcement.

* **Separate subnets for user, control, and data traffic per plane** *support* **QoS isolation**
* **Separate subnets for user, control, and data traffic per plane** *support* **policy enforcement**

**Factoid**: Edge and campus UPFs are positioned at county/data-center proximity to offload traffic and reduce backhaul latency (~10–30 ms).

* **Edge and campus UPFs** *are positioned at* **county/data-center proximity**
* **Positioning of edge and campus UPFs** *offloads* **traffic**
* **Positioning of edge and campus UPFs** *reduces* **backhaul latency (~10–30 ms)**

**Factoid**: Campus UPFs integrate simplified O&M and local data storage to satisfy local processing and security compliance.

* **Campus UPFs** *integrate* **simplified O&M**
* **Campus UPFs** *integrate* **local data storage**
* **Simplified O&M and local data storage** *satisfy* **local processing requirements**
* **Simplified O&M and local data storage** *satisfy* **security compliance**

**Factoid**: Central and regional UPFs focus on high throughput and broader connectivity; edge/campus UPFs target performance-sensitive enterprise usage.

* **Central and regional UPFs** *focus on* **high throughput**
* **Central and regional UPFs** *focus on* **broader connectivity**
* **Edge and campus UPFs** *target* **performance-sensitive enterprise usage**

**Factoid**: N3 octet counters (GTP.In/OutDataOctetsN3UPF) measure data volume between gNB and UPF, optionally per QoS or slice.

* **N3 octet counters (GTP.In/OutDataOctetsN3UPF)** *measure* **data volume between gNB and UPF**
* **Measurement** *can be done per* **QoS**
* **Measurement** *can be done per* **slice**

**Factoid**: Packet loss on N3 is captured as GTP.In/OutDataPktLossN3UPF, enabling per-QoS reliability tracking.

* **Packet loss on N3** *is captured as* **GTP.In/OutDataPktLossN3UPF**
* **GTP.In/OutDataPktLossN3UPF** *enables* **per-QoS reliability tracking**

**Factoid**: Average RTT per DSCP on N3 (GTP.RttDelayN3DlPsaUpfMean.DSCP) provides microsecond-level latency metrics for QoS classes.

* **Average RTT per DSCP on N3** *is measured using* **GTP.RttDelayN3DlPsaUpfMean.DSCP**
* **GTP.RttDelayN3DlPsaUpfMean.DSCP** *provides* **microsecond-level latency metrics**
* **Microsecond-level latency metrics** *are provided for* **QoS classes**

**Factoid**: Out-of-order GTP packet counts (GTP.InDataPktOutOfOrderN3UPF) help detect sequence and jitter issues on N3.

* **Out-of-order GTP packet counts (GTP.InDataPktOutOfOrderN3UPF)** *help detect* **sequence issues on N3**
* **Out-of-order GTP packet counts (GTP.InDataPktOutOfOrderN3UPF)** *help detect* **jitter issues on N3**

**Factoid**: N4 interface PFCP session metrics (SessionEstab, ReportSucc) reflect SMF‑UPF control-plane signaling health.

* **N4 interface PFCP session metrics** *include* **SessionEstab**
* **N4 interface PFCP session metrics** *include* **ReportSucc**
* **SessionEstab and ReportSucc metrics** *reflect* **SMF-UPF control-plane signaling health**

**Factoid**: N6 link usage counters (IP.N6IncLinkUsage, IP.N6OutLinkUsage) aggregate user-plane bandwidth data toward external networks.

* **N6 link usage counters** *include* **IP.N6IncLinkUsage**
* **N6 link usage counters** *include* **IP.N6OutLinkUsage**
* **IP.N6IncLinkUsage and IP.N6OutLinkUsage** *aggregate* **user-plane bandwidth data**
* **User-plane bandwidth data** *is toward* **external networks**

**Factoid**: N9 RTT metrics (GTP.RttDelayN9\*) quantify latency across chained UPFs, supporting multi-hop performance assessment.

* *N9 RTT metrics (GTP.RttDelayN9)*\* *quantify* **latency across chained UPFs**
* **Latency quantification via N9 RTT metrics** *supports* **multi-hop performance assessment**

**Factoid**: N9 GTP packet/byte counters support slice-level throughput and control-plane scaling insights.

* **N9 GTP packet counters** *support* **slice-level throughput insights**
* **N9 GTP byte counters** *support* **slice-level throughput insights**
* **N9 GTP packet and byte counters** *support* **control-plane scaling insights**

**Factoid**: Measurement split by DSCP/QoS/S‑NSSAI enables slice-specific SLA monitoring and resource orchestration.

* **Measurement split by DSCP/QoS/S-NSSAI** *enables* **slice-specific SLA monitoring**
* **Measurement split by DSCP/QoS/S-NSSAI** *enables* **resource orchestration**

**Factoid**: These interface-level metrics feed QoS enforcement, auto-scaling, and anomaly detection modules in intent-driven orchestration.

* **Interface-level metrics** *feed* **QoS enforcement modules**
* **Interface-level metrics** *feed* **auto-scaling modules**
* **Interface-level metrics** *feed* **anomaly detection modules**
* **These modules** *operate within* **intent-driven orchestration**

**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.

* **Priority-based packet handling** *yields* **~78% latency improvement**
* **Priority-based packet handling** *yields* **~88% jitter improvement**
* **Latency and jitter improvements** *are relative to* **normal traffic**

**Factoid**: Performance remains stable across CPU load variations, ensuring URLLC resilience in production environments.

* **Performance** *remains stable across* **CPU load variations**
* **Stable performance** *ensures* **URLLC resilience**
* **URLLC resilience** *is critical in* **production environments**

**Factoid**: Hardware-enabled packet steering ensures flow steering to designated cores, optimizing cache and order.

* **Hardware-enabled packet steering** *ensures* **flow steering to designated cores**
* **Flow steering to designated cores** *optimizes* **cache utilization**
* **Flow steering to designated cores** *optimizes* **packet order**

**Factoid**: Throughput and latency separation in UPF enable multi-tier traffic handling (eMBB and URLLC) on same infrastructure.

* **Throughput and latency separation in UPF** *enable* **multi-tier traffic handling**
* **Multi-tier traffic handling** *supports* **eMBB traffic**
* **Multi-tier traffic handling** *supports* **URLLC traffic**
* **eMBB and URLLC traffic** *share* **the same infrastructure**

**Factoid**: Throughput, latency, packet loss, and jitter are the four key QoS metrics defining 5G service quality.

* **Throughput** *is a key QoS metric*
* **Latency** *is a key QoS metric*
* **Packet loss** *is a key QoS metric*
* **Jitter** *is a key QoS metric*
* **These four metrics** *define* **5G service quality**

**Factoid**: Voice, autonomous vehicles, and remote medical services rely critically on low-latency and minimal packet loss.

* **Voice services** *rely critically on* **low latency**
* **Voice services** *rely critically on* **minimal packet loss**
* **Autonomous vehicles** *rely critically on* **low latency**
* **Autonomous vehicles** *rely critically on* **minimal packet loss**
* **Remote medical services** *rely critically on* **low latency**
* **Remote medical services** *rely critically on* **minimal packet loss**

**Factoid**: UPF, AMF, SMF, PCF, and AUSF form a cohesive 5GC suite enabling scalable, reliable, and flexible network operation.

* **UPF, AMF, SMF, PCF, and AUSF** *form* **a cohesive 5GC suite**
* **5GC suite** *enables* **scalable network operation**
* **5GC suite** *enables* **reliable network operation**
* **5GC suite** *enables* **flexible network operation**

**Factoid**: Scalability supports high traffic load, reliability ensures mission-critical service continuity, and flexibility enables smooth 3G/4G to 5G transition.

* **Scalability** *supports* **high traffic load**
* **Reliability** *ensures* **mission-critical service continuity**
* **Flexibility** *enables* **smooth 3G/4G to 5G transition**

**Factoid**: Future-proof architecture ensures readiness for emerging 5G services while balancing performance objectives and cost.

* **Future-proof architecture** *ensures* **readiness for emerging 5G services**
* **Future-proof architecture** *balances* **performance objectives** and **cost**

**Factoid**: A logical slicing architecture partitions 5G infrastructure into QoS-aligned logical networks for tailored service delivery.

* **Logical slicing architecture** *partitions* **5G infrastructure**
* **Partitioning of 5G infrastructure** *creates* **QoS-aligned logical networks**
* **QoS-aligned logical networks** *enable* **tailored service delivery**

**Factoid**: Slice-aware mobility mechanisms are required to support seamless handover at high speeds (up to 500 km/h).

* **Slice-aware mobility mechanisms** *are required to support* **seamless handover**
* **Seamless handover** *occurs 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.

**Joint optimization of power and subchannel allocation** *uses* **ILP (Integer Linear Programming)**

* **ILP-based optimization** *mitigates* **co-tier interference**
* **ILP-based optimization** *mitigates* **cross-tier interference**
* **Interference mitigation** *occurs in* **shared spectrum**

**Factoid**: Dynamic resource allocation among slices supports flexible, demand-driven QoS fulfillment under interference variability.

* **Dynamic resource allocation among slices** *supports* **flexible QoS fulfillment**
* **Dynamic resource allocation among slices** *supports* **demand-driven QoS fulfillment**
* **QoS fulfillment** *occurs under* **interference variability**

**Factoid**: Network slicing introduces open challenges: slice orchestration, mobility coordination, SDN/NFV integration, and infrastructure reconfiguration.

* **Network slicing** *introduces* **open challenges**
* **Open challenges** *include* **slice orchestration**
* **Open challenges** *include* **mobility coordination**
* **Open challenges** *include* **SDN/NFV integration**
* **Open challenges** *include* **infrastructure reconfiguration**

**Factoid**: 5G target is ≤ 1 ms end-to-end latency with 99.99% reliability for use cases like tactile internet.

* **5G target** *is* **≤ 1 ms end-to-end latency**
* **5G target** *is* **99.99% reliability**
* **Latency and reliability targets** *support use cases like* **tactile internet**

**Factoid**: RAN delays comprise ttx, tbsp, tmpt; LTE’s 1 ms TTI must be reduced to ≤0.25 ms.

* **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.

* **Sub-ms TTI (0.25 ms)** *lowers* **latency**
* **Sub-ms TTI (0.25 ms)** *increases* **control overhead**

**Factoid**: Advanced waveforms (GFDM, SC‑FDM), and symbol‑detection (MMSE, ZF) methods reduce RAN processing delays.

* **Advanced waveforms** *include* **GFDM** and **SC-FDM**
* **Symbol-detection methods** *include* **MMSE** and **ZF**
* **Advanced waveforms and symbol-detection methods** *reduce* **RAN processing delays**

**Factoid**: SDN/NFV and MEC in core/backhaul reduce per-packet latency by eliminating centralized processing hops.

* **SDN/NFV and MEC in core/backhaul** *reduce* **per-packet latency**
* **Latency reduction** *occurs by eliminating* **centralized processing hops**

\_\_\_\_\_\_\_\_ BATCH 10 GEMINI \_\_\_\_\_\_\_\_\_

**Factoid**: Dynamic GTP termination and ultra-dense WDM backhaul can contribute < 0.1 ms transport latency improvements.

* **Dynamic GTP termination** *contributes to* **transport latency improvements (< 0.1 ms)**
* **Ultra-dense WDM backhaul** *contributes to* **transport latency improvements (< 0.1 ms)**

**Factoid**: Edge caching of popular content shortens backhaul dependencies and content retrieval latency.

* **Edge caching of popular content** *shortens* **backhaul dependencies**
* **Edge caching of popular content** *shortens* **content retrieval latency**

**Factoid**: NF resource requirements for low latency include fast CPU, low-latency I/O, and real-time orchestration responsiveness.

* **NF resource requirements for low latency** *include* **fast CPU**
* **NF resource requirements for low latency** *include* **low-latency I/O**
* **NF resource requirements for low latency** *include* **real-time orchestration responsiveness**

**Factoid**: RAN optimization combined with transport and caching enables sub-ms latency but demands trade-off in overhead and complexity.

* **RAN optimization** combined with **transport and caching** *enables* **sub-ms latency**
* **Achieving sub-ms latency** *demands trade-offs in* **overhead and complexity**

**Factoid**: Reliable ultra-low latency requires harmonised RAN, core, transport, and caching co-design with real-time resource adaptation.

* **Reliable ultra-low latency** *requires* **harmonised RAN, core, transport, and caching co-design**
* **Reliable ultra-low latency** *requires* **real-time resource adaptation**

**Factoid**: An intent like ‘7 Mbps uplink / ≤ 50 ms latency’ can specify AR/VR service requirements without detailing implementation.

* **Intent ‘7 Mbps uplink / ≤ 50 ms latency’** *specifies* **AR/VR service requirements**
* **Intent specification** *does not detail* **implementation specifics**

**Factoid**: Impacted layers parse intents using RAN identifiers like 5QI and S‑NSSAI to scope service intents.

* **Impacted layers** *parse* **intents** using **RAN identifiers**
* **RAN identifiers** *include* **5QI** and **S-NSSAI**
* **5QI and S-NSSAI** *are used to scope* **service intents**

**Factoid**: The Intent Management Function (IMF) checks feasibility, decomposes intent, and orchestrates domain resources.

* **Intent Management Function (IMF)** *checks* **intent feasibility**
* **IMF** *decomposes* **intent into actionable components**
* **IMF** *orchestrates* **domain resources**

**Factoid**: IMF uses TMF921 and 3GPP APIs for intent lifecycle management, negotiation, and compliance reporting.

* **Intent Management Function (IMF)** *uses* **TMF921 APIs**
* **IMF** *uses* **3GPP APIs**
* **TMF921 and 3GPP APIs** *support* **intent lifecycle management**
* **TMF921 and 3GPP APIs** *support* **intent negotiation**
* **TMF921 and 3GPP APIs** *support* **compliance reporting**

**Factoid**: Utility functions embedded in intents guide decision-making trade-offs, e.g., latency vs. energy.

* **Utility functions embedded in intents** *guide* **decision-making trade-offs**
* **Decision-making trade-offs** *include* **latency vs. energy**

**Factoid**: E2E intent latency targets (e.g., 100 ms) are decomposed to domain-specific contributions (e.g., 60 ms in RAN).

* **End-to-end (E2E) intent latency targets (e.g., 100 ms)** *are decomposed into* **domain-specific contributions**
* **Domain-specific contributions** *include* **60 ms in RAN**

**Factoid**: Closed-loop cognitive operations continuously monitor intent compliance, evaluate metrics, and trigger corrective actions.

* **SDAP sublayer** *maps* **QoS flows** to **DRBs**
* **QoS flows** *are identified by* **given QFI**
* **Mapping** *is supported by* **RRC-configured mapping rules**

**Factoid**: Multi-layer autonomous domains resolve policy conflicts (e.g. energy-saving vs. performance) through intent exchange.

* **Multi-layer autonomous domains** *resolve* **policy conflicts**
* **Policy conflicts** *include* **energy-saving vs. performance**
* **Resolution of policy conflicts** *is achieved through* **intent exchange**

**Factoid**: Intent adoption is phased—from static SLOs to dynamic assurance, to full autonomy integrated into business-service workflows.

* **Intent adoption** *is phased*
* **Phases of intent adoption** *include* **static SLOs**
* **Phases of intent adoption** *include* **dynamic assurance**
* **Phases of intent adoption** *include* **full autonomy integrated into business-service workflows**

**Factoid**: Intent-driven autonomy is expected to materialize in commercial networks between 2025 and 2027.

**Factoid**: MEF uses controlled natural-language DSLs (Allegro, Cantata) to express business-level intents like ‘Skype for Business → mission-critical SLA.’

* **MEF** *uses* **controlled natural-language DSLs (Allegro, Cantata)**
* **Controlled natural-language DSLs (Allegro, Cantata)** *express* **business-level intents**
* **Business-level intents** *include* **‘Skype for Business → mission-critical SLA’**

**Factoid**: Intent expressions from diverse stakeholders are harmonized using MEF models before mapping into enforceable policy rules.

* **Intent expressions from diverse stakeholders** *are harmonized using* **MEF models**
* **Harmonized intent expressions** *are mapped into* **enforceable policy rules**

**Factoid**: Intent-to-policy mapping translates DSL intent into LSO API calls (Legato, Presto, Adagio) for network enforcement.

* **Intent-to-policy mapping** *translates* **DSL intent into LSO API calls**
* **LSO API calls** *include* **Legato**, **Presto**, and **Adagio**
* **LSO API calls** *are used for* **network enforcement**

**Factoid**: MEF enforces declared intents—including performance/security objectives—continuously until manually removed.

* **MEF** *enforces* **declared intents**
* **Declared intents** *include* **performance objectives**
* **Declared intents** *include* **security objectives**
* **Declared intents** *are enforced* **continuously until manually removed**

**Factoid**: The intent processing pipeline roles span Business, System, Admin, and Device layers to ensure end-to-end policy coherence.

* **Intent processing pipeline roles** *span* **Business layer**
* **Intent processing pipeline roles** *span* **System layer**
* **Intent processing pipeline roles** *span* **Admin layer**
* **Intent processing pipeline roles** *span* **Device layer**
* **Spanning all layers** *ensures* **end-to-end policy coherence**

**Factoid**: AI/ML modules assist in interpreting intent, synthesizing policies, and orchestrating LSO-controlled network resources.

* **AI/ML modules** *assist in* **interpreting intent**
* **AI/ML modules** *assist in* **synthesizing policies**
* **AI/ML modules** *assist in* **orchestrating LSO-controlled network resources**

**Factoid**: LSO APIs serve as enforcement endpoints for intent-derived policy rules within automated service lifecycles.

* **LSO APIs** *serve as* **enforcement endpoints**
* **Enforcement endpoints** *apply to* **intent-derived policy rules**
* **Intent-derived policy rules** *operate within* **automated service lifecycles**

**Factoid**: Intent-based automation addresses scaling limitations and policy complexity by abstracting intent from implementation details.

* **Intent-based automation** *addresses* **scaling limitations**
* **Intent-based automation** *addresses* **policy complexity**
* **Intent-based automation** *abstracts* **intent from implementation details**

**Factoid**: Continuous intent validation ensures that performance/security objectives remain enforced over time.

* **Continuous intent validation** *ensures* **performance objectives remain enforced over time**
* **Continuous intent validation** *ensures* **security objectives remain enforced over time**

**Factoid**: MEF’s IBN framework forms a foundation for autonomous networking by linking declarative intent to operational APIs.

* **MEF’s IBN framework** *forms* **a foundation for autonomous networking**
* **MEF’s IBN framework** *links* **declarative intent** to **operational APIs**

**Factoid**: Large-scale programmable 5G/B5G networks require meta-schedulers to coordinate domain-specific schedulers and fulfill user intents.

* **Large-scale programmable 5G/B5G networks** *require* **meta-schedulers**
* **Meta-schedulers** *coordinate* **domain-specific schedulers**
* **Meta-schedulers** *fulfill* **user intents**

**Factoid**: Meta-schedulers issue high-level intent directives, while local schedulers handle domain-specific resource allocations autonomously.

* **Meta-schedulers** *issue* **high-level intent directives**
* **Local schedulers** *handle* **domain-specific resource allocations**
* **Local schedulers** *operate* **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.

**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.

* **The system** *issues* **SDN flow-rule updates**
* **SDN flow-rule updates** *steer* **traffic along the best path**
* **Best path** *matches* **user intent and current metrics**

**Factoid**: Intent mapping rules define how each recognized user intent maps to network adaptation policies (delay thresholds, path priority).

* **Intent mapping rules** *define how* **recognized user intent** *maps to* **network adaptation policies**
* **Network adaptation policies** *include* **delay thresholds**
* **Network adaptation policies** *include* **path priority**

**Factoid**: QoE is maintained through network reconfiguration even as user motion or network load changes.

* **QoE** *is maintained through* **network reconfiguration**
* **Network reconfiguration** *occurs even as* **user motion changes**
* **Network reconfiguration** *occurs even as* **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.

* **SDN-controlled path adaptation** *acts as* **execution layer for intent translation**
* **SDN-controlled path adaptation** *is used 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.

**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.

* **Human feedback loop** *enables* **refinement of intent translation accuracy**
* **Refinement of intent translation accuracy** *occurs over* **system lifetime**

**Factoid**: Effectiveness demonstrated in real-world deployment within the EURECOM 5G facility.

**Factoid**: 5G networks can support IPv6-only user-plane using 464XLAT with CLAT at UE and PLAT in the network.

* **5G networks** *can support* **IPv6-only user-plane**
* **IPv6-only user-plane support** *uses* **464XLAT**
* **464XLAT** *includes* **CLAT at UE**
* **464XLAT** *includes* **PLAT in the network**

**Factoid**: IPv4-only UEs receive their address via CLAT while IPv6-only UEs are fully native on the user-plane.

* **IPv4-only UEs** *receive their address via* **CLAT**
* **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.

* **In roaming scenarios**, the **location of PDU anchor** *(home vs visited 5GC)* *dictates* **CLAT/PLAT placement**
* **CLAT/PLAT placement** *is determined by* **whether the PDU anchor is in the home or visited 5GC**

**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.

* **Policy configurations** *include* **static IP assignment**
* **Policy configurations** *include* **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.

**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.

* **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.

* **IPv6 firewalls** *should use* **stateful implicit-deny policies**
* **IPv6 firewalls** *should not use* **NAT for security**

**Factoid**: Static NAT-PT is deprecated; NAT64/DNS64 is the preferred IPv4-compatibility mechanism.

* **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.

* **DNS64** *synthesizes* **AAAA records when only A records exist**
* **Synthesized AAAA records** *enable* **IPv6-only host access via NAT64**

**Factoid**: SLAAC-only clients should use DNS64/NAT64 to interact with IPv4-only services.

* **SLAAC-only clients** *should use* **DNS64/NAT64**
* **DNS64/NAT64** *enables* **interaction with IPv4-only services**

**Factoid**: IPSec is recommended to secure IPv6 communication channels requiring confidentiality.

* **IPSec** *is recommended to secure* **IPv6 communication channels**
* **Securing IPv6 communication channels** *is required for* **confidentiality**

**Factoid**: Firewalls must limit IPv6 multicast scope and monitor IPv6 transition tunnels (e.g., Teredo).

* **Firewalls** *must limit* **IPv6 multicast scope**
* **Firewalls** *must monitor* **IPv6 transition tunnels (e.g., Teredo)**

**Factoid**: IPv6 compatibility requires testing for IPv6 literals, DNS64 behavior, and supporting dual-stack protocols.

**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.

**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.

* **Micro-segmentation** *implements* **zero-trust control**
* **Software-defined perimeters** *implement* **zero-trust control**
* **Zero-trust control** *limits* **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.

**Factoid**: 5G private network IPAM systems require dynamic automation and conflict resolution to support millions of connected UE and IoT devices.

* **5G private network IPAM systems** *require* **dynamic automation**
* **5G private network IPAM systems** *require* **conflict resolution**
* **Dynamic automation and conflict resolution** *support* **millions of connected UE and IoT devices**

**Factoid**: Operators enforce IPv6-only control-plane policies—no public IPv4 allowed on CP links.

* **Operators** *enforce* **IPv6-only control-plane policies**
* **IPv6-only control-plane policies** *do not allow* **public IPv4 on CP links**

**Factoid**: IPv4 addressing is restricted to external-facing slices or middleboxes, with strict NAT at the UPF.

* **IPv4 addressing** *is restricted to* **external-facing slices**
* **IPv4 addressing** *is restricted to* **middleboxes**
* **Strict NAT** *is enforced 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.

* **IPAM subnets** *are isolated per* **slice (e.g., URLLC, mMTC)**
* **IPAM subnets** *are isolated per* **NF zone (e.g., AMF/SMF vs UPF)**
* **Isolation per slice and NF zone** *enforces* **QoS**
* **Isolation per slice and NF zone** *enforces* **network isolation**

**Factoid**: Real-time IPAM monitoring should alert network managers to conflicts or rogue addresses in dense deployments.

* **Real-time IPAM monitoring** *should alert* **network managers**
* **Alerts** *indicate* **conflicts**
* **Alerts** *indicate* **rogue addresses**
* **Conflicts and rogue addresses** *are common in* **dense deployments**

**Factoid**: IPAM–orchestrator integration enables automated allocation of subnets/IPs during on-the-fly NF instantiation.

* **IPAM–orchestrator integration** *enables* **automated allocation of subnets/IPs**
* **Automated allocation of subnets/IPs** *occurs during* **on-the-fly NF instantiation**

**Factoid**: IPAM must support IPv6-only, dual-stack, and IPv4-translation modes for flexible slice support.

* **IPAM** *must support* **IPv6-only mode**
* **IPAM** *must support* **dual-stack mode**
* **IPAM** *must support* **IPv4-translation mode**
* **Support for these modes** *enables* **flexible slice support**

**Factoid**: IPv6 control-plane networks simplify policy and isolation strategies, improving compliance with operator frameworks.

* **IPv6 control-plane networks** *simplify* **policy strategies**
* **IPv6 control-plane networks** *simplify* **isolation strategies**
* **Simplified policy and isolation strategies** *improve* **compliance with operator frameworks**

**Factoid**: Operator policies mandate no public IPv4 on CP while allowing IPv4 for UPF-based NAT translation.

* **Operator policies** *mandate* **no public IPv4 on control plane (CP)**
* **Operator policies** *allow* **IPv4 for UPF-based NAT translation**

**Factoid**: Slice-specific addressing enhances enforcement of slice-level policies and traffic segregation.

* **Slice-specific addressing** *enhances* **enforcement of slice-level policies**
* **Slice-specific addressing** *enhances* **traffic segregation**

**Factoid**: NIST blueprint supports both aggregated and disaggregated O‑RAN testbeds using srsRAN, FlexRIC, OSC, and Open5GS stacks.

* **NIST blueprint** *supports* **aggregated O-RAN testbeds**
* **NIST blueprint** *supports* **disaggregated O-RAN testbeds**
* **NIST blueprint** *uses* **srsRAN**, **FlexRIC**, **OSC**, and **Open5GS stacks**

**Factoid**: Time synchronization in O‑RAN requires PPS and 10 MHz frequency distribution to USRPs.

* **Time synchronization in O-RAN** *requires* **PPS (Pulse Per Second)**
* **Time synchronization in O-RAN** *requires* **10 MHz frequency distribution**
* **PPS and 10 MHz signals** *are distributed to* **USRPs**

**Factoid**: Testbed servers use Ubuntu (20.04/22.04), low-latency kernels, and BIOS tuning (disable C‑states, HT, secure boot).

* **Testbed servers** *use* **Ubuntu (20.04/22.04)**
* **Testbed servers** *use* **low-latency kernels**
* **BIOS tuning** *includes disabling* **C-states**
* **BIOS tuning** *includes disabling* **Hyper-Threading (HT)**
* **BIOS tuning** *includes disabling* **secure boot**

**Factoid**: The NIST Testbed Automation Tool automates deployment of gNB, UE, RIC, xApps, and 5G Core on bare-metal or virtual hosts.

* **NIST Testbed Automation Tool** *automates deployment of* **gNB**
* **NIST Testbed Automation Tool** *automates deployment of* **UE**
* **NIST Testbed Automation Tool** *automates deployment of* **RIC**
* **NIST Testbed Automation Tool** *automates deployment of* **xApps**
* **NIST Testbed Automation Tool** *automates deployment of* **5G Core**
* **Deployment** *can occur on* **bare-metal hosts**
* **Deployment** *can occur on* **virtual hosts**

**Factoid**: Automation tool supports configuration of ZMQ/E2 messaging and default Docker IP like 10.53.1.2.

* **Automation tool** *supports configuration of* **ZMQ/E2 messaging**
* **Automation tool** *supports configuration of* **default Docker IP (e.g., 10.53.1.2)**

**Factoid**: Modular orchestration with support for multiple RIC and RAN stacks reduces setup complexity and ensures repeatability.

* **Modular orchestration** *supports* **multiple RIC and RAN stacks**
* **Support for multiple RIC and RAN stacks** *reduces* **setup complexity**
* **Support for multiple RIC and RAN stacks** *ensures* **repeatability**

**Factoid**: Testbed requires integration of DNS and clock services to support real-time control and function discovery.

* **Testbed** *requires integration of* **DNS services**
* **Testbed** *requires integration of* **clock services**
* **DNS and clock services** *support* **real-time control**
* **DNS and clock services** *support* **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.

* **xApps (e.g., KPI monitor)** *are deployed on* **near-RT RIC**
* **xApps** *subscribe to* **E2KP metrics**
* **E2KP metrics** *include* **RSRP**
* **Subscription to metrics** *uses* **ZMQ**
* **Subscription to metrics** *can use* **container orchestration**

**Factoid**: Hardware tuning ensures low-latency and deterministic scheduling suitable for RF-based O‑RAN operations.

* **Hardware tuning** *ensures* **low-latency**
* **Hardware tuning** *ensures* **deterministic scheduling**
* **Low-latency and deterministic scheduling** *are suitable for* **RF-based O-RAN operations**

**Factoid**: Testbed uses srsRAN and Open5GS on commodity hardware and USRP NI‑2944R via PCIe for 5G SA operation.

* **Testbed** *uses* **srsRAN**
* **Testbed** *uses* **Open5GS**
* **srsRAN and Open5GS** *run on* **commodity hardware**
* **Testbed** *uses* **USRP NI-2944R via PCIe**
* **srsRAN, Open5GS, and USRP NI-2944R** *support* **5G SA operation**

**Factoid**: Backhaul uses gigabit Ethernet; upgrade to 10 Gbps is recommended for high-throughput experiments.

* **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.

* **Time sync via NTP** *is essential for* **control-plane protocols**
* **DNS** *is essential for* **control-plane protocols**
* **Control-plane protocols** *include* **PFCP**, **NGAP**, and **HTTP2**

**Factoid**: Tuning involves disabling HyperThreading/VT, enabling real-time scheduling, and optimizing buffer sizes and MTU.

* **Tuning** *involves disabling* **HyperThreading/VT**
* **Tuning** *involves enabling* **real-time scheduling**
* **Tuning** *involves optimizing* **buffer sizes**
* **Tuning** *involves optimizing* **MTU (Maximum Transmission Unit)**

**Factoid**: RF planning requires configuring USRP gains, duplex mode, SCS, and matching UE APN and compatible bands.

* **RF planning** *requires configuring* **USRP gains**
* **RF planning** *requires configuring* **duplex mode**
* **RF planning** *requires configuring* **subcarrier spacing (SCS)**
* **RF planning** *requires* **matching UE APN**
* **RF planning** *requires* **matching compatible bands**

**Factoid**: Some consumer UEs must be rooted or APN-modified to function reliably in 5G SA testbeds.

* **Some consumer UEs** *must be* **rooted**
* **Some consumer UEs** *must be* **APN-modified**
* **Rooting and APN modification** *ensure* **reliable function in 5G SA testbeds**

**Factoid**: Open5GS logs (/var/log/open5gs) and srsRAN/ZMQ traces are critical for NGAP, GTP-U, and slice debugging.

* **Open5GS logs (/var/log/open5gs)** *are critical for* **NGAP debugging**
* **Open5GS logs (/var/log/open5gs)** *are critical for* **GTP-U debugging**
* **Open5GS logs (/var/log/open5gs)** *are critical for* **slice debugging**
* **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.

* **Basic network slicing** *uses* **NSSAI (SST/SD)**
* **NSSAI (SST/SD)** *is validated in* **RAN domain**
* **NSSAI (SST/SD)** *is validated in* **Core domain**
* **Validation** *is performed via* **packet trace inspection**

**Factoid**: Consistent slice enforcement is observable in PFCP session setups and NGAP signaling for UE-slice mapping.

* **Consistent slice enforcement** *is observable in* **PFCP session setups**
* **Consistent slice enforcement** *is observable in* **NGAP signaling**
* **PFCP session setups and NGAP signaling** *support* **UE-slice mapping**

**Factoid**: Consumer-grade UE variability necessitates RF tuning, root access, and APN configurations to ensure SA connectivity.

* **Consumer-grade UE variability** *necessitates* **RF tuning**
* **Consumer-grade UE variability** *necessitates* **root access**
* **Consumer-grade UE variability** *necessitates* **APN configurations**
* **RF tuning, root access, and APN configurations** *ensure* **SA connectivity**

**Factoid**: open5gs‑k8s provides both microservice and all‑in‑one Kubernetes manifests for Open5GS.

* **open5gs-k8s** *provides* **microservice Kubernetes manifests for Open5GS**
* **open5gs-k8s** *provides* **all-in-one Kubernetes manifests for Open5GS**

**Factoid**: Multus and OVS‑CNI enable distinct network attachments for N2, N3, N4, preserving plane separation.

* **Multus** *enables* **distinct network attachments for N2, N3, N4**
* **OVS-CNI** *enables* **distinct network attachments for N2, N3, N4**
* **Distinct network attachments** *preserve* **control and user plane separation**

**Factoid**: MongoDB statefulsets with PVCs store subscriber and NF profile data externally.

* **MongoDB statefulsets with PVCs** *store* **subscriber data externally**
* **MongoDB statefulsets with PVCs** *store* **NF profile data externally**

**Factoid**: CLI and Python scripts automate slice and subscriber provisioning via MongoDB for multi-slice deployments.

* **CLI scripts** *automate* **slice and subscriber provisioning via MongoDB**
* **Python scripts** *automate* **slice and subscriber provisioning via MongoDB**
* **Slice and subscriber provisioning** *supports* **multi-slice deployments**

**Factoid**: UERANSIM manifests facilitate gNB and UE emulation, including automated ping tests for AMF connectivity.

* **UERANSIM manifests** *facilitate* **gNB emulation**
* **UERANSIM manifests** *facilitate* **UE emulation**
* **UERANSIM manifests** *include* **automated ping tests**
* **Automated ping tests** *verify* **AMF connectivity**

**Factoid**: Monarch integration supports real-time slice KPI monitoring when enabled in manifest.

* **Monarch integration** *supports* **real-time slice KPI monitoring**
* **Real-time slice KPI monitoring** *is enabled in* **manifest**

**Factoid**: Supported configurations include Kubernetes v1.28, Ubuntu 22.04, containerd 1.6 – documented via release tags.

* **Supported configurations** *include* **Kubernetes v1.28**
* **Supported configurations** *include* **Ubuntu 22.04**
* **Supported configurations** *include* **containerd 1.6**
* **These configurations** *are documented via* **release tags**

**Factoid**: Init containers orchestrate proper startup ordering, ensuring dependency readiness before NF launch.

* **Init containers** *orchestrate* **proper startup ordering**
* **Proper startup ordering** *ensures* **dependency readiness before NF launch**

**Factoid**: HPA-based autoscaling is used for 5G CNFs, while VPA is avoided due to potential service disruptions.

* **HPA-based autoscaling** *is used for* **5G CNFs**
* **VPA** *is avoided due to* **potential service disruptions**

**Factoid**: Cluster Autoscaler dynamically adds/removes nodes based on HPA pod requirements.

* **Cluster Autoscaler** *dynamically adds/removes* **nodes**
* **Node scaling decisions** *are based on* **HPA pod requirements**

**Factoid**: GitOps (via ArgoCD) ensures consistent CNF configuration and prevents drift across clusters.

* **GitOps (via ArgoCD)** *ensures* **consistent CNF configuration**
* **GitOps (via ArgoCD)** *prevents* **configuration drift across clusters**

**Factoid**: Node pools must remain homogeneous in capacity and configuration, enforced by PodDisruptionBudgets and resource requests.

* **Node pools** *must remain* **homogeneous in capacity and configuration**
* **Homogeneity** *is enforced by* **PodDisruptionBudgets**
* **Homogeneity** *is enforced by* **resource requests**

**Factoid**: A scale-down fuse protects critical CNFs (e.g., AMF, SMF) from being evicted during resource scaling events.

* **Scale-down fuse** *protects* **critical CNFs (e.g., AMF, SMF)**
* **Protection from eviction** *occurs during* **resource scaling events**

**Factoid**: SCTP protocol (132) must be permitted at network and OS level to support 5G CNF control-plane communication.

* **SCTP protocol (132)** *must be permitted at* **network level**
* **SCTP protocol (132)** *must be permitted at* **OS level**
* **Permitting SCTP protocol (132)** *supports* **5G CNF control-plane communication**

**Factoid**: Use of external monitoring (metrics API, Istio) provides data-driven trigger inputs for autoscaling decisions.

* **Use of external monitoring** *provides* **data-driven trigger inputs for autoscaling decisions**
* **External monitoring tools** *include* **metrics API**
* **External monitoring tools** *include* **Istio**

**Factoid**: Avoid multiple autoscalers in the same node group and check cloud quotas when configuring scaling policies.

* **Avoiding multiple autoscalers in the same node group** *prevents* **scaling conflicts**
* **Checking cloud quotas** *is important when* **configuring scaling policies**

**Factoid**: GitOps-based management hub enables central control of Day‑0/Day‑2 lifecycle operations for distributed 5G environments.

* **GitOps-based management hub** *enables* **central control of Day-0/Day-2 lifecycle operations**
* **Day-0/Day-2 lifecycle operations** *apply to* **distributed 5G environments**

**Factoid**: Cluster and workload configuration consistency is essential for reliable auto-scaling in telecommunication-grade CNFs.

* **Cluster and workload configuration consistency** *is essential for* **reliable auto-scaling**
* **Reliable auto-scaling** *applies to* **telecommunication-grade CNFs**

**Factoid**: 5G testbeds include remote-area (5G‑RANGE), neutral‑host city deployment (5GCity), and UAV‑assisted RAN with NFV flexibility.

* **5G testbeds** *include* **remote-area deployment (5G-RANGE)**
* **5G testbeds** *include* **neutral-host city deployment (5GCity)**
* **5G testbeds** *include* **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.

* **External DNS** *is mandatory for* **realistic end-to-end testbed operation**
* **Internet connectivity** *is mandatory for* **realistic end-to-end testbed operation**
* **Time sync (PPS/NTP)** *is mandatory for* **realistic end-to-end testbed operation**

**Factoid**: Automated deployment uses open-source toolchains with Docker/K8s for NF instantiation, telemetry, and orchestration.

* **Automated deployment** *uses* **open-source toolchains**
* **Open-source toolchains** *include* **Docker** and **Kubernetes (K8s)**
* **Docker/K8s** *enable* **NF instantiation**
* **Docker/K8s** *enable* **telemetry**
* **Docker/K8s** *enable* **orchestration**

**Factoid**: KPI measurement includes latency, throughput, reliability, cross‑domain sync, collected via telemetry and ML analytics.

* **KPI measurement** *includes* **latency**
* **KPI measurement** *includes* **throughput**
* **KPI measurement** *includes* **reliability**
* **KPI measurement** *includes* **cross-domain sync**
* **KPI data** *is collected via* **telemetry**
* **KPI data** *is analyzed with* **ML analytics**

**Factoid**: Testbeds bridge lab and field environments through consistent resource pipelines, enabling repeatable validation cycles.

* **Testbeds** *bridge* **lab and field environments**
* **Testbeds** *use* **consistent resource pipelines**
* **Consistent resource pipelines** *enable* **repeatable validation cycles**

**Factoid**: Neutral-host frameworks (5GCity) allow operators to share infrastructure dynamically using slicing and multitenancy.

* **Neutral-host frameworks (e.g., 5GCity)** *allow* **operators to share infrastructure dynamically**
* **Infrastructure sharing** *is enabled by* **slicing**
* **Infrastructure sharing** *is enabled by* **multitenancy**

**Factoid**: 5G-RANGE combines fixed RAN with UAVs supported by NFV for sporadic deployments in remote regions.

* **5G-RANGE** *combines* **fixed RAN** with **UAVs**
* **5G-RANGE** *is supported by* **NFV**
* **5G-RANGE** *enables* **sporadic deployments in remote regions**

**Factoid**: End-to-end KPI validation must consider NFV stack maturity, telemetry quality, and orchestration tool chain alignment.

* **End-to-end KPI validation** *must consider* **NFV stack maturity**
* **End-to-end KPI validation** *must consider* **telemetry quality**
* **End-to-end KPI validation** *must consider* **orchestration tool chain alignment**

**Factoid**: Edge and core synchronization in testbeds hinges on coordinated time sync, network path management, and telemetry integration.

* **Edge and core synchronization in testbeds** *hinges on* **coordinated time sync**
* **Edge and core synchronization in testbeds** *hinges on* **network path management**
* **Edge and core synchronization in testbeds** *hinges on* **telemetry integration**

**Factoid**: Lab-to-production deployment is facilitated by standardized virtualization patterns across SDR, core, and orchestration environments.

* **Lab-to-production deployment** *is facilitated by* **standardized virtualization patterns**
* **Standardized virtualization patterns** *apply across* **SDR environments**
* **Standardized virtualization patterns** *apply across* **core environments**
* **Standardized virtualization patterns** *apply across* **orchestration environments**