Istio Usage in 5G Core CNFs

Faseela Kundattil and Ingo Meirick
Ericsson

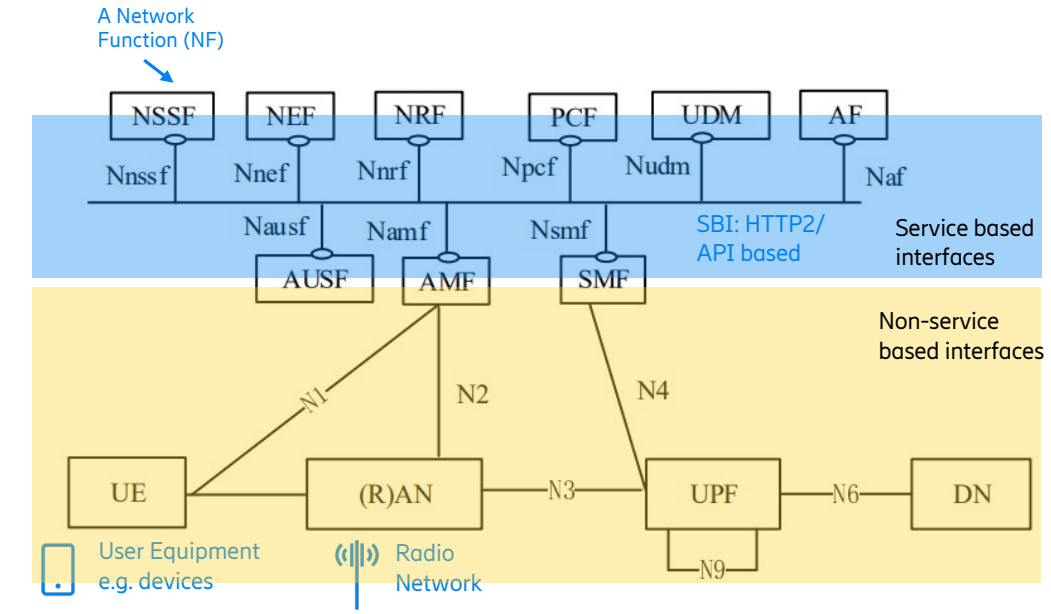
#IstioCon

Agenda

- 5G System Architecture
- Why using service mesh in 5G Core Cloud-native Network Functions (CNFs)
- Service mesh in a CNF and related requirements
- Technical details on service mesh usage in a CNF
- What's next

5G System Architecture

- Separate Control Plane and User Plane
- Network Functions (NFs) are defined by 3GPP standard and consist of smaller unit functions called NF services
- NFs implemented using cloud-native design principles are called Cloud-native NFs (CNFs)
- Different NFs connect to each other via uniform interface, called service-based interface (SBI), HTTP2/API based
- Service mesh is most useful for Control Plane NFs (HTTP2 based, less strict requirements on latency)



AUSF	Authentication Server Function
AMF	Access and Mobility Management Function
AF	Application Function
DN	Data Networks
SMF	Session Management Function
UDM	Unified Data Management

NEF	Network Exposure Function
NSSF	Network Slice Selection Function
NRF	Network Repository Function
PCF	Policy Control Function
UPF	User Plane Function
(R)AN	(Radio) Access Network

Reference: 3GPP TS 123 501 V15.3.0

Why using Istio in 5G Core Network

Security

mTLS for internal and external communication
Authentication and authorization of workloads
Automatic certificate provisioning and rotation

Traffic management

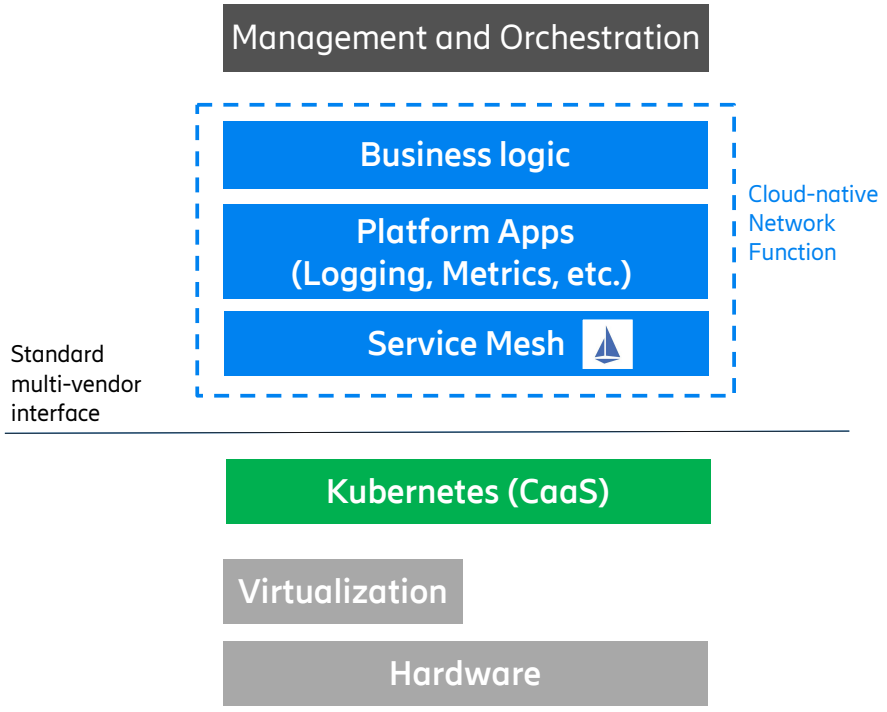
L7 based load balancing and traffic steering
Handling of ingress and egress traffic
Network resilience and robustness

Observability

Metrics, traces, logging

Service Mesh in Cloud Native Telco Stack

- CNFs are operated by Communication Service Providers on a container as a service (CaaS) platform based on Kubernetes
- CNF and Kubernetes platform could be from different vendors
- Multiple CNFs might run on the same Kubernetes cluster
- Ericsson adapted Istio service mesh included in the CNF
- Avoid dependency on service mesh to be present on the Kubernetes platform

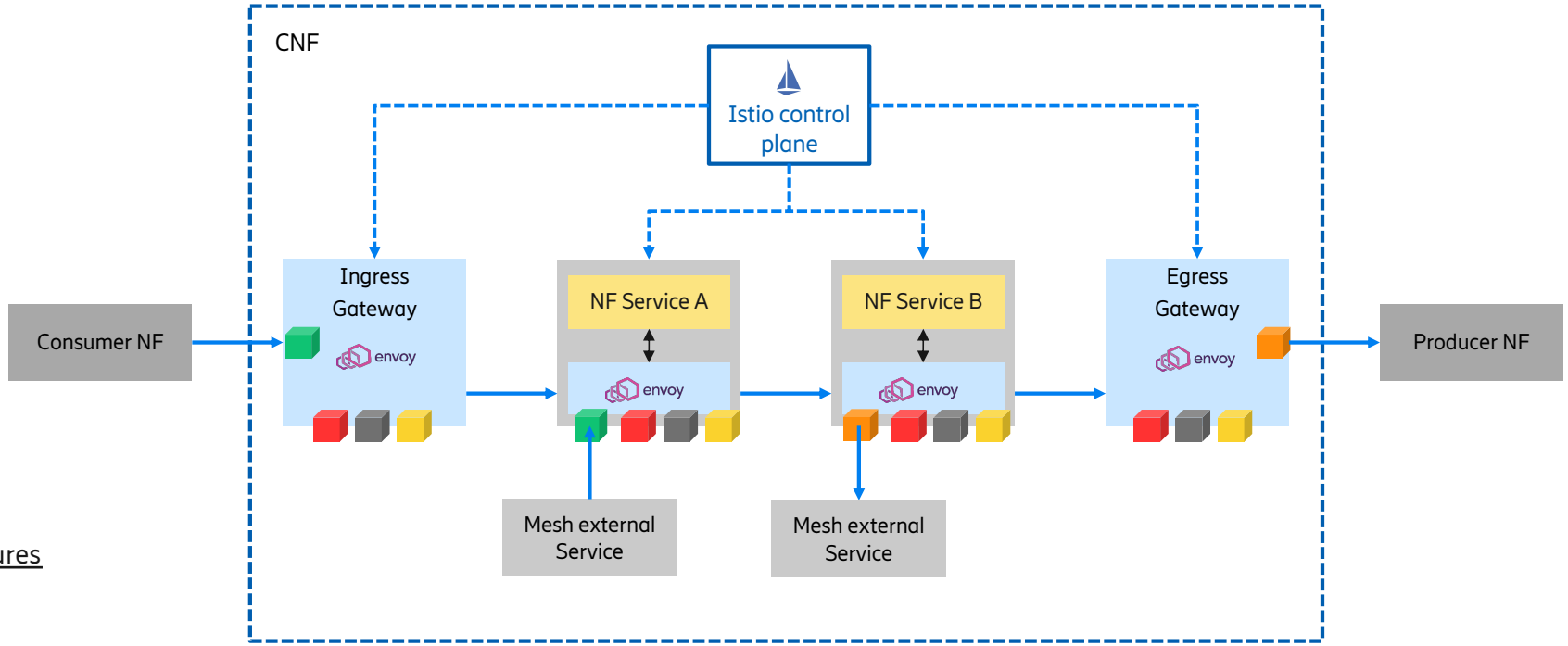


Cloud native telco stack

Further requirements

- Support for multi-tenancy since service mesh is part of CNF
- Communication with services outside the mesh
  - Not all services in a CNF can be integrated to the service mesh
- Support of specific 3GPP requirements for robustness and resilience
- Lifecyle Management (LCM)
  - Coupled LCM of CNF and service mesh
  - Support of long upgrade window (> 6 months)
  - Support of helm
- Support for dual-stack
- Adaptation of sidecar resource requests and limits according to CNF dimensioning
- Hardening and vulnerability testing

Istio Service Mesh usage in a CNF



Service Mesh features

- = Ingres
- = Egress
- = Traffic Management
- = Security
- = Observability

Service mesh features in detail

Security

- mTLS for internal communications
- Authentication and authorization of workloads
- Automatic certificate provisioning & rotation

Traffic Management

- L7 Load Balancing
- L7 Traffic Steering
- Resilience (Timeout, retries, etc.)
- Robustness (Connection pooling, Circuit breaker, etc.)

Observability

- Prometheus metrics (bytes sent/rcvd, connection failures, HTTP error codes)
- Jaeger-compatible Distributed tracing

Ingress

- Termination of external TLS and mTLS traffic
- Hitless rotation of certificates
- Optional Validation of user identities / JWT
- Overload handling
- Load Balancing

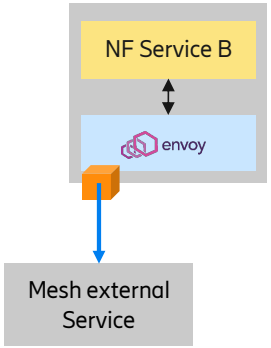
Egress

- Origination of external TLS and mTLS traffic
- Hitless rotation of certificates
- Enhanced security for egress traffic when using Egress Gateway on dedicated nodes

Technical Details

Egress (m)TLS handling by sidecar

- Why: Avoid additional hop through Egress GW
- Current configuration
  - Sidecar annotations to mount certificates
  - DestinationRule with cert/key paths
  - Note: DestinationRule gets applied to all workloads




```
(snip)
metadata:
  annotations:
    sidecar.istio.io/userVolume:
      '{"egress-secret":{"secret":{"secretName":"sbi-client-certs"}},
        "egress-ca-secret":{"secret":{"secretName":"sbi-client-certs-cacert"}}}'
    sidecar.istio.io/userVolumeMount:
      '{"egress-secret":{"mountPath":"/etc/istio/egress-certs/","readOnly":true},
        "egress-ca-secret":{"mountPath":"/etc/istio/egress-ca-certs/","readOnly":true}}'
(snip)
```

```
kind: DestinationRule
metadata:
  name: egress-mtls-example
spec:
  host: mynginx.example.com
  trafficPolicy:
    portLevelSettings:
    - port:
        number: 443
      tls:
        mode: MUTUAL
        clientCertificate: /etc/istio/egress-certs/tls.crt
        privateKey: /etc/istio/egress-certs/tls.key
        caCertificates: /etc/istio/egress-ca-certs/ca-chain.cert.pem
```

Egress (m)TLS handling by sidecar (cont.)

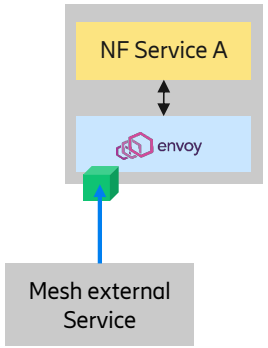
Proposed enhancements

- Allow reference to Kubernetes secret with credentials in DestinationRule
- Use workloadSelector in DestinationRule to add configuration only to specific workload
- RFC and PRs proposed in community
  - RFC: Simplify sidecar Egress for mTLS
  - Workload Selector support in DR
  - Support CredentialName configuration in DR for sidecars

```
kind: DestinationRule
metadata:
  name: egress-mtls-example
spec:
  host: mynginx.example.com
  workloadSelector:
    labels:
      app: client-service
  trafficPolicy:
    portLevelSettings:
    - port:
        number: 443
      tls:
        mode: MUTUAL
        CredentialName: client-credential
```

Ingress (m)TLS handling by sidecar

- Why: Avoid additional hop through Ingress GW for CNF internal services outside the mesh
- Current solution
  - EnvoyFilter to add new filter_chain for inbound traffic
- Enhancements
  - Istio RFC: Support hybrid sidecar mode
  - Extended sidecar to enable TLS termination from outside of mesh
  - Simplify TLS configuration on sidecar proxy
  - Note: Solution under discussion




```
kind: Sidecar
metadata:
  name: httpbin
  namespace: app1
spec:
  workloadSelector:
    labels:
      app: httpbin
  ingress:
    - port:
        number: 80
        name: http
        protocol: HTTPS
      defaultEndpoint: 127.0.0.1:80
      tls:
        mode: SIMPLE
        privateKey: "/etc/certs/tls.key"
        serverCertificate: "/etc/certs/tls.crt"
```

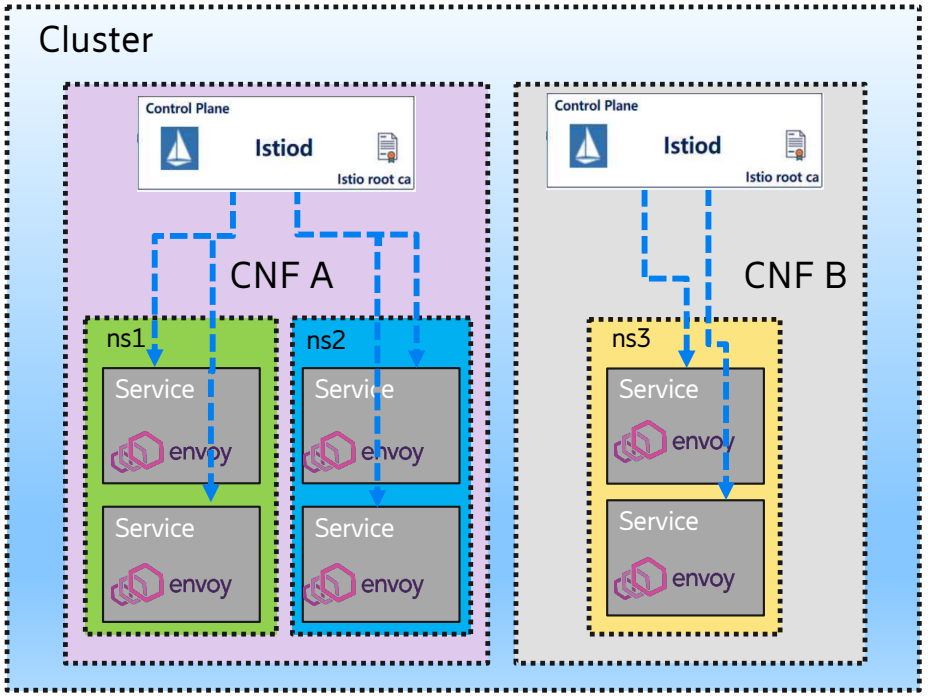
Multi-tenancy support

Why:

- Service mesh included in CNF
- Limit service mesh to specific CNF

How:

- Extended Istio DiscoverySelectors to define tenant boundaries
- Sidecar Injection limited to CNF
- Webhook patching limited to CNF
- Root CA ConfigMaps limited to CNF
  - One Root CA per CNF
- Prevent configuration leakages across CNFs



Support for Dual-stack

- Why: Dual-stack support is crucial for 5G CNFs
- Current solution
  - Use envoy ipv4_compat mode
  - Override IP Family settings and envoy local IP address version
  - Enforce setup of ip6tables on sidecars
  - Proposal made to community, but not suitable for external istiod or multi cluster mesh
- Desired Enhancement
  - Analysis in progress for the new dualstack support proposal upstream

Usage of EnvoyFilters

- Why: For enhanced features
  - Global rate-limiting and better load balancing
  - HTTP Tap Filter
  - TLS 1.3 setting
  - Ingress (m)TLS termination at sidecar
- Support through Istio native APIs preferred

What's next

- Support for multi-tenancy
- Better support for dual-stack
- Better support for non-HTTP based protocols, e.g., databases like Postgres
- Better performance by e.g., eBPF, gRPC proxyless service mesh
- Integration with external CAs
- Better graceful termination logic during node failure/shutdown

Thank you!

@handle
https://mysite.com

#IstioCon