Evaluation of a Service-mesh based Service Communication proxy for Future 5G Core Network

White paper

Kris Kim (kris.kim@nokia.com)
Nokia Software GBC, Nokia

Satvinder Bawa (satvinder.bawa@nokia.com)
Nokia Software CSD Product, Nokia

DongJin Lee (dongjin@sktelecom.com)
B5G/6G Core R&D Architect, ICT Infra Tech, Infra Strategy & Tech Center, SK Telecom

JoongGunn Park (clark.park@sktelecom.com)
Core Architect, ICT Infra Tech, Infra Strategy & Tech Center, SK Telecom

Contents

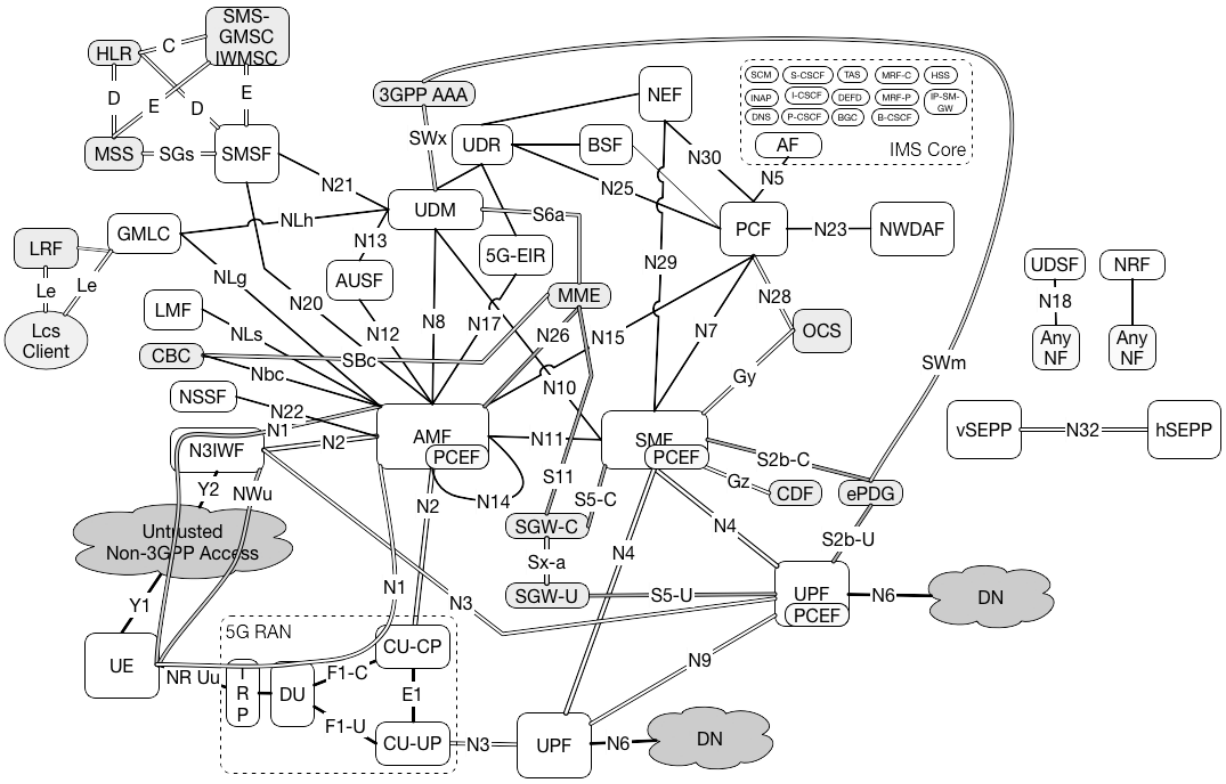
Executive Summary

3GPP defines the service-based architecture starting in Rel.15 system for 5G standalone Core network. This 5G Core has many Network Functions (NFs) that require service routing, scalability, security and interworking functions to support end-to-end telecommunication services. The service-based architecture (SBA) is enhanced in 3GPP Rel.16.

The 5G Core introduces a vast variety of new NFs where the versions are continuously updated at a fast pace as shown in Figure 1. These NFs are expected to be based on cloud-native architecture and are now developed with microservice architecture with a high degree of modularity, such that applications are composed from easily managed and deployed services across a multitude of infrastructures with flexible and minimal coupling. However, with increasing number of such microservice NFs, the complexity of a distributed service-based architecture is further increased. These challenges are the key reasons which lead to the need of a services interworking framework and systems.

Service Communication Proxy (SCP) is a newly standardized 3GPP Rel.16 system in 5G Core which provides service-level load distribution, QoS control, resiliency and monitoring functions for all 5G NF signals.1) In addition to this, the main purpose of the SCP is to reduce complexity of interconnection between multiple 5G NFs. The SCP is a service-mesh based system designed to take control of the service-based interface (SBI), so that NFs may have another option not to communicate directly with each other, and instead communicate via the SCP. The injection of the SCP in the flow of control plane provides the SCP with awareness of content, performance and trouble spots in the control plane. This awareness is used to bring a rich variety of dashboards to the operators of 5G Core network.

Figure 1. Diagram showing the NFs and related interfaces within 5G Core



As indicated, the SCP is built using service-mesh components. A service-mesh is a set of comprehensive tools to facilitate microservice architectures by providing a secure, reliable, and resilient communications layer. This solution fits the requirements of 3GPP Rel.16 5G Cores and can handle the delivery of signaling requests for the cloud-native 5G deployment.

In this joint whitepaper by Nokia and SK Telecom, the SCP's service-mesh concepts, capabilities and implementations are described for cloud-native 5G deployment. Also, result of findings for SCP's interoperability with SK Telecom's 5G PoC network are documented. The findings are focused on the following areas:

- 5G Core signaling controls in the cloud-native/service-mesh architecture
- Telco-grade Reliability/Resilience for 5G cloud-native service communications
- Service Quality Observation in the 5G cloud-native/service-mesh architecture

In summary, this paper is about:

- Simplifying, scaling and optimizing the signaling communications for 5G services across cloud infrastructures.
- E2E visibility of communications at the service level: single pane of contextual observability of all service-based interface flows to the operator to assist in operational proficiency.
- Telco-grade 5G signaling plane for better inter-service communication.

However, this paper is not about:

- explanation of network technologies such as 5G, nor necessarily exclusive about 5G only.
- extensions or projections of SK Telecom's commercial 5G Core.
- explanation of Nokia CSD products as Service Communication Proxy.

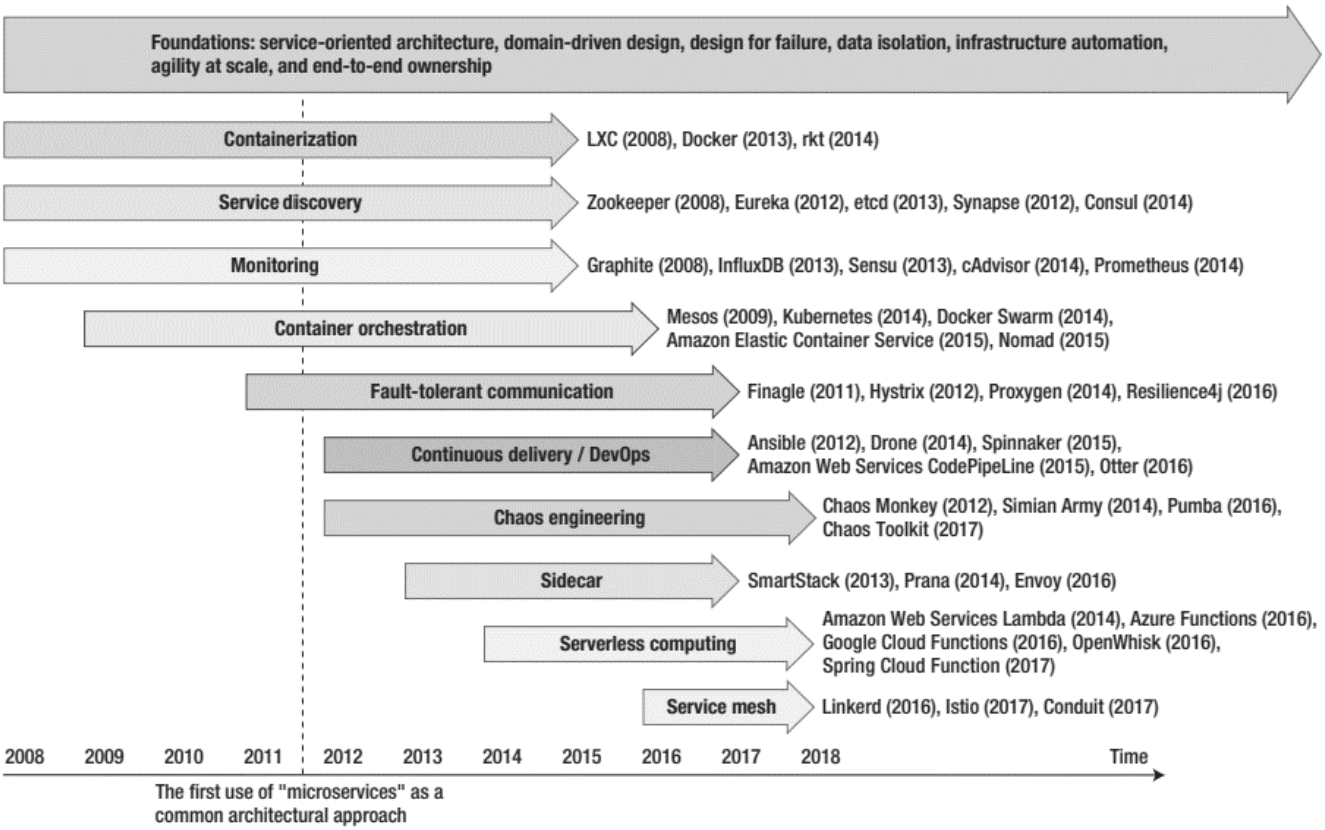
Introduction and Background

In the 5G standalone network defined by 3GPP, the 5G cloud-native core signaling concepts are roughly modelled on the widely adopted patterns of development used by web scale solutions. These solutions are based on highly decomposed and decoupled services coming together to dynamically discover each other, and utilize the services offered by each other, to realize the overall goals of a domain. Specifically, in today's telecommunications industry, the goal of cloud-native architectures is to make 5G network functionality (NF) highly modular and reusable so that NF specific microservices developers can focus more on their applications and business logic.

In such a microservices infrastructure, operators should be able to dynamically deploy services across the network to meet the scale such as Key Performance Indicators (KPIs), and Service Level Agreements (SLAs) requirements of end-to-end services. On the downside, of course, in some respects, the complexity of a distributed service-based architecture is further increased by microservices. This is due to the possibilities of incompatibilities in interactions and various performance degradations and unpredictable failure modes.

Many of these issues have been traditionally encountered by web-scale companies, and they have had various architectural evolutions from first generation microservices that were packed using containerization (2008), Service discovery (2008) and finally come up with service-mesh concepts (2016) in which highly distributed proxies offload a lot of the common tasks as shown in Figure 2.

Figure 2. A microservice technologies timeline[2]

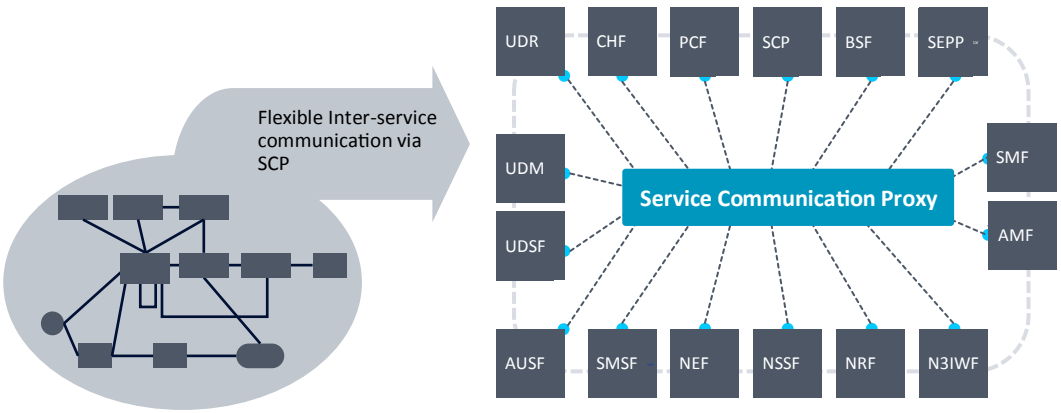


This aspect of service-mesh roughly parallels the task assigned to the Service Communication Proxy. A SCP built using service-mesh architecture provides a secure, reliable, and resilient communication layer that can handle the delivery of signaling requests through the complex, dynamic, and ever-changing service topology that forms the latest cloud-native 5G deployment. This task is largely facilitated by other elements of 5G Core. The NRF (NF Repository Function), and the dynamic registration of different NFs which provide services, and constant updates of state and utilization aspects can be uniformly consumed and applied to all consumers of the services.

For cloud-native 5G deployment, automating, securing, scaling, optimizing and simplifying signal communications for 5G services in several unreliable cloud infrastructures is an important challenge. Communication between NFs in 3GPP 5G Core does not simply mean signaling connections between previously virtualized network elements, but rather services that dynamically support numerous NF services by various vendors or service providers. The 3GPP Rel.16 5G Core service-based architecture provides constructs such as NF service sets and NF sets, which facilitate service registration, discovery, selection and re-selection aspects with full knowledge of shared persistence layer amongst different types of sets of NFs. The consistent approach to the realization of the result is a key delivery of the SCP.

The concept of proxies has been formalized in 3GPP Rel.16. According to 3GPP, the SCP is the next generation of standard services capable of providing load sharing, control, management and monitoring capabilities for all 5G NF services. As shown in Figure 3, the SCP is in the middle of the NF communication channel for all NF services, connecting each NF service as the communication brain.

Figure 3. Inter-service communications between microservices using SCP



The below item lists some of use cases enabled for visibility of control plane traffic supported by the SCP.

Key SCP Functions	Description
Mediation	SCP enables JSON format mediation between incompatible implementations of service APIs, as well as inter protocol mediation (Diameter/HTTP2-JSON)
Scale Mismatch	SCP allows pooling of servers and message by message distribution based on dynamic or static criteria
Scaling Up or Down	SCP allows consistent introduction of new NFs into server pools, or removal of NFs from such server pools
Upgrade Facilitation	SCP allows for per message mediation, or per message exposure to newer NFs
Message Validation	SCP allows for plausibility as well as strict applicability tests in specific deployments. Such validations typically result in a narrowing of allowed values for a specific attribute
Rate Limiting	SCP can orchestrate rate limitations in a consistent manner, and prevent or help remove congestion scenarios

Service-mesh based SCP for 5G NF

"A service-mesh is a dedicated infrastructure layer for making service-to-service communication safe, fast, and reliable. If you're building a cloud-native application, you need a service-mesh3)." The primary role of a service-mesh is the management of communications between services. In the cloud-native network, as the network functions (or services) grow, this means that inter-service communications also increase and become more complicated. One of the main ideas is to move the common functionality from each service to the service-mesh area.

Although not mandatory, the 3GPP Service Communication Proxy structure is recommended to support the service-mesh like architecture.1) Considering the 5G cloud-native network services, we can obtain a better network efficiency and utilization with service-mesh based Service Communication Proxy (SCP) architecture. Service-mesh SCP supports these by maximizing the benefits of the Service-mesh technology referred below.

The service agent of Service-mesh SCP1) is a lightweight, transparent proxy with support for HTTP/2 and gRPC 5G Core signaling terminations. One of its primary roles is to abstract the network by providing a common feature set necessary for every 5G Core NF service in order to communicate safely, rapidly, and reliably. We call this service agent of SCP as 'sidecar container' which runs alongside 5G NF service

containers. All the logic required for inter-service communication is abstracted out of the 5G NF microservice and put into the sidecar. Together, microservices and sidecars form a service-mesh network.[4]

Each of the sidecar is autonomously injected as a sidecar container alongside the instance (or Pod) of a 5G NF service. In scenarios where a 5G Core signaling network comprises physical or virtualized NFs, the SCP sidecar is deployed as a shared proxy NF to provide the same capabilities offered by a cloud-native servicemesh deployment. Once a sidecar has been deployed, it interacts with the service-mesh control plane over its APIs to implement and apply the necessary policies, for example, observability and security to the 5G Core's traffic flows and signals.[5]

Figure 4. service-mesh SCP architecture with service agent (3GPP Rel.16 23.501)[1]

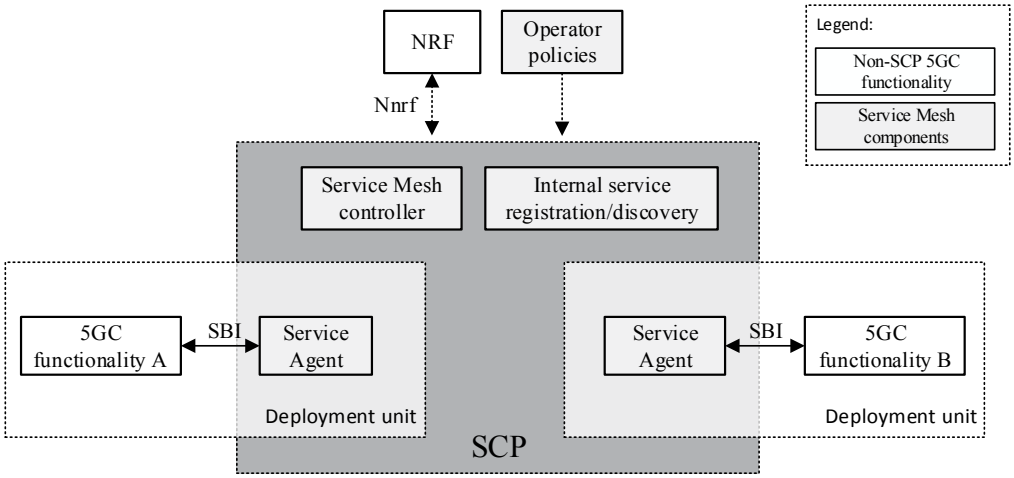


Figure 4 shows a service-mesh SCP like architecture where the sidecar referred as 'Service Agent' intercepts all requests entering and exiting the connected pod services aka 5GC functionalities. It applies policy on how the in-and-out traffic is handled. All requests are routed between 5G NFs via sidecar. Without a service-mesh, developers would be less focused on a business logic, and instead they have to spend time and resources on individual microservice's logic for inter-service communications. The most complex challenge in realizing microservice architecture is not building the services themselves, but the communication between services. It also means communication failures are harder to diagnose because the logic that governs inter-service communication is hidden within each service.

Examples of two technologies enabling the service-mesh SCP are Kubernetes and Istio open-source project. Kubernetes is a container orchestration system for Docker containers. Istio is a platform on top of Kubernetes that manages, and controls distributed microservices. Istio operates at the application layers such as HTTP and gRPC to perform more advanced load balancing and traffic routing with various pluggable policies. Istio uses the sidecar – a proxy instance added in an application pod either manually or automatically during service deployment. The sidecar container intercepts all the inbound and outbound traffic of the service instance pod. It also applies any network policies without the knowledge of application container. The controlling entity of the service-mesh, the Istio pilot provides the policies to a sidecar. Whenever an application container is deployed, an additional container is injected to it. The application container contains the core logic for the application. The sidecar and the application containers share a number of resources including parts of filesystem, hostname, network and other namespaces.[6]

Figure 5. how the service-mesh SCP controls the traffic

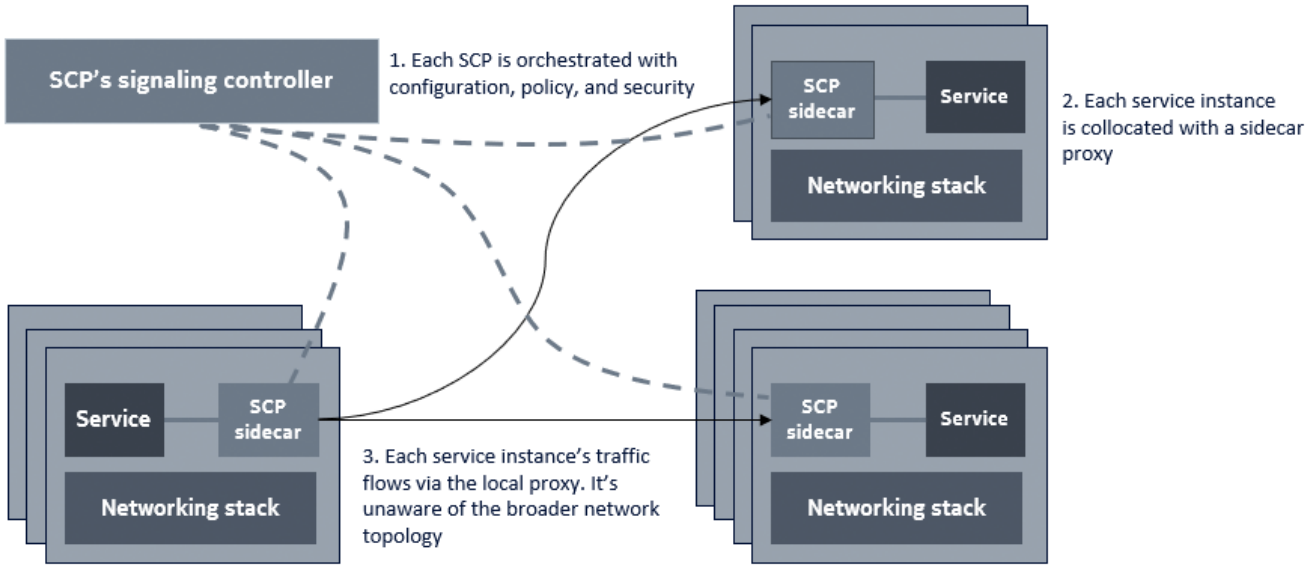


Figure 5 explains how the service-mesh SCP controls the traffic. By default, the controller of service-mesh SCP will send the related configuration of all services in the namespace where the sidecar is located, including inbound and outbound listeners, clusters, routes, etc. With SCP sidecar, it is possible to fine-tune the configuration, for example, only the relevant outbound configuration of those external services that the SCP sidecar service needs to access.

With all the advantages of above referred service-mesh technologies in the cloud native environments, we'll address the following items to review as a main benefit of service-mesh SCP.

- Traffic controls
- Observability
- Failure handling and platform resiliency
- Security

Traffic Controls

In a traditional approach, it is typical for application logic to implement, establish, and control the flow of communications traffic, which is based on the topology of established connections with the peer entities forming a group or cluster. With a service-mesh, the traffic flows can be decoupled from the infrastructure topology and provide a variety of traffic management features and use cases.[7]

The current 3GPP Rel.15 5G Core provides a capability of registration and discovery of services via NRF. In this 'direct communication', 5G NFs or 5G NF services talk directly with each other. This means that some aspects of server availability and loading are always notified to the NRF and conveyed along if client NFs are properly registered for such notifications. With this architecture, every NF shall know what to do, and every NF shall do what it needs to do correctly. This could be an architectural risk in the direct communication of 3GPP Rel. 15 which places a lot of burden on every single NF to do a lot of common redundant work, in diverse implementations, consistently.

The newly introduced SCP in 3GPP Rel.16 5G Core provides more capability of in-service registration and discovery of services via SCP with NRF. Because the SCP is composed of service based interface(SBI)/ service-mesh component, it controls and understands which microservices are active and how they are related for the end-to-end functional point of view. It also provides a policy enforcement at a granular level by deciding how workloads should be balanced. For example, if a microservice is upgraded, the service-mesh decides which requests should be routed to the microservices running the stable version and which requests should be routed to the microservices running the upgraded version. These service level traffic controls as examples in the preceding sentence cannot be achieved easily without service-mesh SCP.

Observability

Troubleshooting a distributed application is one of the more challenging activities in a production deployment. With the advent of cloud-native applications, these activities have become more complex and labor intensive. Debugging 5G NF microservice applications has become harder because the 5G NF microservices are now distributed. In the context of service-to-service interactions, the HTTP/2-based traffic is secured and headers are modified, deleted and added, which makes it difficult to compose a full picture of end-to-end 5G signaling.[5] Thus, it is nearly impossible to use traditional techniques for capturing, monitoring traffic and perform behavior analysis.

Because the service-mesh SCP is a dedicated service infrastructure layer allowing all service-to-service communication pass, it is uniquely positioned in the middle to provide uniform telemetry at the service call level. SCP captures all passed traffic such as source, destination, protocol, URL, status codes, latency, duration and so on. Once captured, metrics and logs are collected by the SCP's control plane and passed along to the monitoring tools such as Prometheus and Grafana for storage and visualization which is frequently used in cloud-native open source environment. This way, it brings inter-service 5G NF communication into a full visibility of 5G signaling. The term often associated with such distributed visibility is "observability."

For example, the signaling telemetry could be feed into BSS/OSS analytics systems for improving operations and the creation of new offers. Deploying service traffic via SCP means the automatic collection of fine-grained telemetry data. The signaling telemetry is automatically collected from any 5G NF in the microservices, and service-mesh SCP provides a consistent view by generating uniform metrics throughout. The telemetry data collected from the service-mesh SCP includes Success Rates, Volume, Duration, Size, Latency and HTTP error codes etc.

Another example of observability is a flexible tracing support. Engineers and operators can troubleshoot compositional problems such as NF's flow sequencing issues, service call tree abnormalities, and consumer and producer NF's request/response and subscription/notification specific issues. i.e., to identify which microservices are involved in some specific services. Considering the microservices architecture, it is true that it is difficult to understand the dependencies across the services. Tracing is possible with service-meshes because of the use of span identifiers and forwarded context headers. To make it work, the service needs to be modified to read tracing headers upon input, pass them along to all the related threads of execution and then add them to every call to other services. With this type of flexible tracing, it is easy to understand what is happening and troubleshoot the problems in service.

The service-mesh SCP is provided with cloud-native traffic monitoring applications (HTTP/2 detailed attributes, KPI, Alert, Log/Trace, and Visualization), and can provide quality of service measurement for the 5G microservices.

Failure handling and platform resiliency

Microservices are composed of small module units (container/pod) and are activated/deactivated in a very short time. Service-mesh SCP needs to understand cloud-native characteristics and how to achieve telco-grade reliability for the microservice architecture. There are some telco-grade requirements that must be considered even for the Cloud-native network.

- Network Availability, Latency, and Bandwidth: Carrier-grade (99.999%) availability is a critical requirement for telco operations. Unless solutions are implemented with these requirements in mind, cloud-native applications may not be able to meet such availability requirements.
- Unique Protocols and Network Design: Telco operations are characterized by specific protocols and a network design that consist of many components that interact with each other.

When designing a full microservice based 5G NF, failure of a microservice pod should not affect the overall functionality and availability of the 5G NF, which means that each microservice pod must be able to be isolated from each other, rather than tightly coupled. This is one of the key advantages of microservice design. And this can achieve graceful service degradation. If there's any issue is service A, other remaining services such as B, C... Z continue to work properly.

In a microservices architecture, microservice pods are loosely coupled with each other and distributed independently. And these microservice pods may start, restart and stop because of failures, deployments or autoscaling. Self-healing of microservice pods can also help to recover 5G NF.

The service-mesh SCP is capable of failure recovery features such as timeouts, active health checks, bounded retries, and others. Several capabilities as part of traffic control like circuit breaking, latency aware load balancing, consistent service discovery etc. enhance the ability of the service-mesh SCP to avoid some common failure pitfalls. For example, based on the overall view of the traffic flows and the observability between 5G cloud-native services, a dynamic load balancing policy profile can be applied to avoid the signaling imbalance in service capacity during an execution of scaling/recovering of 5G cloud-native services.

Security

The service-mesh SCP provides policy based security at three levels: the service-mesh platform level, the namespace level and NF service level. Ingress gateways in the service-mesh SCP allow verified traffic to come from outside the service-mesh network to inside the service-mesh. This solution can, and typically does provide the end-to-end encryption for incoming traffic from the outside. The traffic is authenticated with TLS termination. Once this traffic enters the service-mesh SCP, it will be re-encrypted and will be sent to the target NF. Within the Istio service-mesh, Citadel pod of Istio control plane provides the key and certificate management function. The Citadel handles the certificate creation and rotation for traffic encryption between services. And the service handles it on a service account level of Kubernetes namespaces.

It is also very important to keep the same security guidelines for communications between 5G NFs and for their communications with the external network. The service-mesh SCP enforces security measures and policies without affecting the code of application pod. The service-mesh SCP also supports these security related measures and policies to both inbound and outbound traffics flows to the external 5G NF services.

The security-related benefits of service-mesh SCP can be defined as:

- The 5G NF service authentication
- The encryption of traffic between 5G NF services
- Security-specific policy enforcement to inbound and outbound traffic flows

SCP PoC for 5G Core network

To further understand the concept of SCP as defined in 3GPP Rel.16, we defined the following goals to be verified in whitepaper.

- 5G Core signaling controls in the cloud-native/service-mesh architecture
- Telco-grade Reliability/Resilience for 5G cloud-native service communications
- Service quality observation in the 5G cloud-native/service-mesh architecture

For 3GPP compliancy, the following assumptions and configurations are made for the technical evaluation.

- 3GPP pre-Rel. 16 injection of SCP are targeting compliancy to 3GPP Rel. 15 23.501 (2018. 12) with NRF flows compliant to 3GPP Rel.15 29.510 (2019. 03), mimicking Model D-Indirect Communication with delegated discovery. Note that the major improvement which 3GPP did in 5G vs. 4G is the dynamic discovery aspect. In this, we expect to observe an improvement of the current 3GPP Rel. 15's discovery with the 3GPP pre-Rel. 16 of SCP's seamless injection.
- Service-mesh SCP's co-location with 5GC functionality is configured so that the service-mesh SCP manages NF's registration, discovery and (re)selection for communication within the service-mesh. It interacts with an external NRF for service exposure and communication across service-mesh boundaries. For non-containerized NF, a shared proxy is applied.
- All of consumer and producer NF operations functions which follow 3GPP Rel.15 and pre-Rel.16 NFs are not aware of SCP. Thus, all of consumer and producer NF's traffics are intercepted transparently. by the SCP. This transparent function known as NRF-Proxy acts to control overall NF interactions including service registration, discovery, (re)selection and specific-NF's traffic. Any state changes, scaling, healing procedures can also be monitored by this transparent function.
- Within the service-mesh SCP, NRF-Proxy pod is configured so that the registration request is forwarded to the internal registry as well as forwarding to the external NRF. The internal registration is used to store the address to service identifier and the Service Deployment Cluster location as 5GC services are running as microservices on cloud in deployment clusters.

SCP Evaluation

In this technical evaluation, the following setup has been configured as below 'table' to construct our service-mesh SCP, so that it acts as a dedicated infrastructure layer to handle service-to-service communication.

Project name	Description
Kubernetes	This refers to the container orchestration system for Docker containers.
Istio	This is an open-source service mesh to connect, monitor, and secure microservices.
Grafana	This is an open-source analytics and monitoring solution.
Prometheus	This is a common metrics collection and alerting system. It is also available as a data source for Grafana.
Jaeger	This is an open-source tracing system used for monitoring and troubleshooting microservice-based distributed systems.
Kiali	This is an observability console for Istio with service-mesh configuration capabilities.
Kibana	This is an open-source data visualization dashboard for the Elasticsearch data.
fluentd	This is an open-source data collector for unified logging layer.

This service-mesh SCP is then configured and tested with 3GPP defined 5GC NFs such as PCF (Policy Charging Function), BSF (Binding Support Function), AF (Application Function) and NRF (Network Repository Function) as shown in Figure 6. In the setup, 3 all-active SCP pods, 2 redundant Istio Ingress-gateway pods, 2 redundant Istio Egress-gateway pods are configured like in Figure 7.

Figure 6. Service-mesh SCP architectures

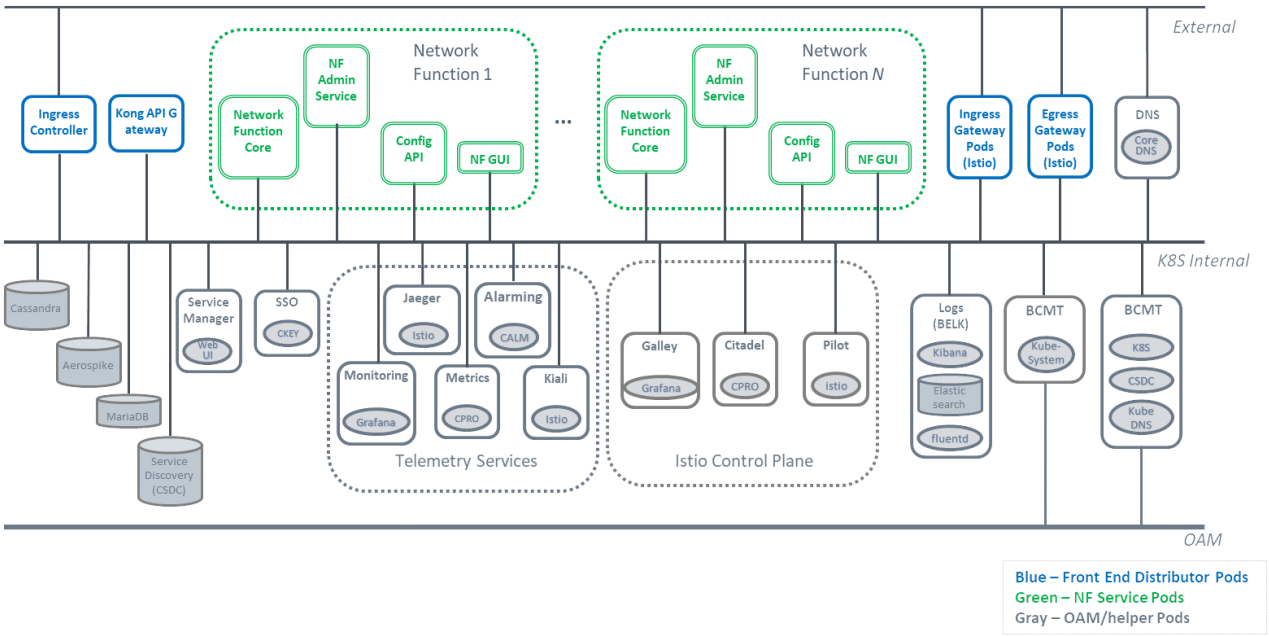
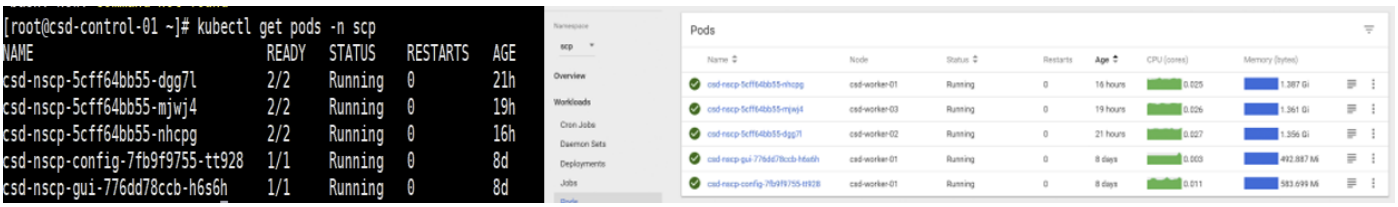


Figure 7. Serviced SCP pods configured service-mesh SCP



The following service-mesh SCP test has been done for Load test with the below configuration. In order to measure the max throughput of each service Pods, we configured differently that the above PoC configuration.

Number of pods:	
Component Name	**Number of Pods**
SCP	8
Istio Ingress gateway	1
Istio Egress gateway	1

Resource Allocation For Single Container:				
Container Name	**CPU (limits)**	**Memory (limits**	**CPU (requests)**	**Memory (requests**
SCP Application container	2	2GB	2	2GB
SCP Sidecar proxy	2	1GB	100m	128MB
Istio Ingress gateway	12	2GB	2	512MB
Istio Egress gateway	8	2GB	2	1GB

With the single service-mesh SCP with the above configuration, we could test 99.99% success ratio for 16K TPS load. The P90 (the average latency for the slowest 10% of request over the last 10s) was under 5ms.

We tested three main items for our whitepaper which are, 1) SCP Function (Service discovery and exposure, and Load Balancing for traffic controls), 2) SCP's Resiliency and High Availability (High Availability, Platform Resiliency) and 3) Traceability and Observability (E2E tracing, KPI observability).

SCP Function Test and Output

In this test, we wanted to see '5G Core signaling controls in the cloud-native/service-mesh architecture'. We tested the below items over PoC to see how SCP acts over 5G Core network.

Test Criteria	Functions	Test Items	Description
Service discovery and exposure	3GPP deployment: inter-operability with NRF	1) NF registration through SCP	Integration of network elements as NRF-P
		2) Get NF details [Discover]	request to consume a service as NRF-P
		3) Subscribe for an NF notification	request to notify changes in a service as NRF-P
		4) Update sent from an NF	request indicating changes in service as NRF-P
		5) Notification	responded to notify a service change occurred as NRF-P
		6) Deregistration of NF	Service is no longer available as NRF-P
	Inter-service operability with NFs in 5GC	7) NF registration through CSD	Integration of network elements
		8) Get NF details [Discover]	request to consume a service
		9) Subscribe for an NF	request to notify changes in a service
		10) Update sent from an NF	request indicating changes in service
		11) Notification	responded to notify a service change occurred
		12) Deregistration of NF	Service is no longer available
	Binding Support	13) Create PDU session	User initiated data session
		14) Get PDU session details	finding an ongoing session
		15) Delete PDU session	tearing down of a session
Load Balancing for traffic controls	Stateless Load Balancing	16) Stateless Balancing with priority	Multiple requests for same service
		17) Stateless Balancing with weight	Multiple requests for same service
	Stateful Routing	18) Stateful Routing based on NF type definition	sticky routing of a service request
		19) Stateful Routing based on NF service type definition	sticky routing of a service request
		20) Stateful Routing based on URI definition	sticky routing of a service request

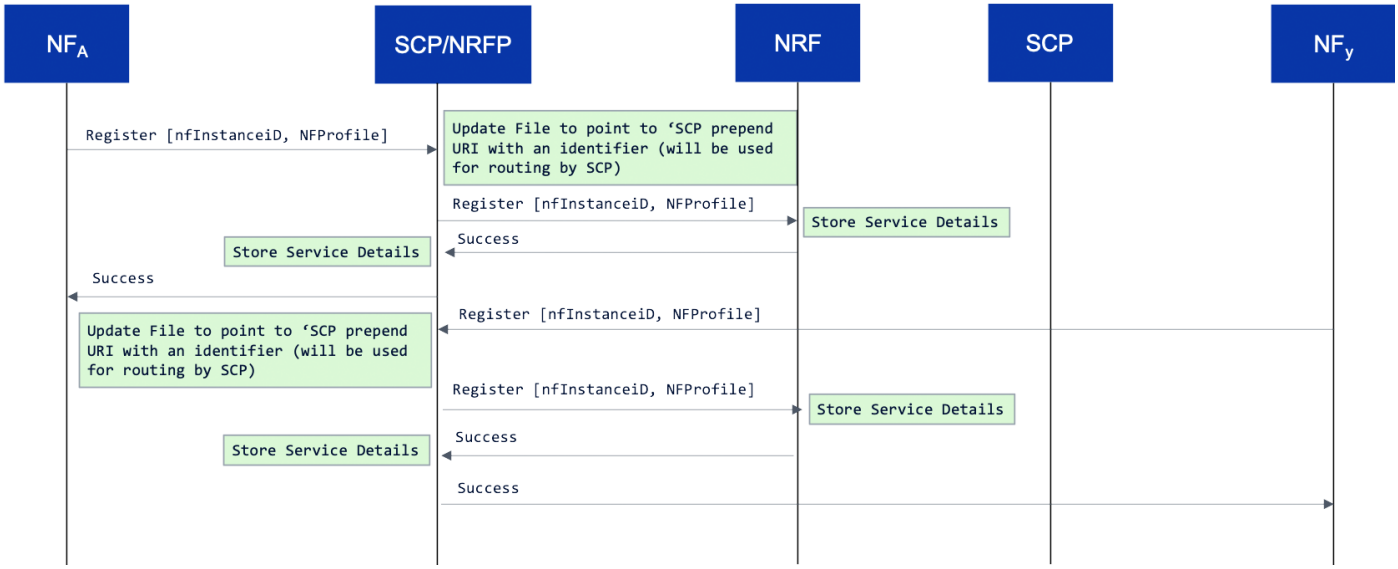
As example, the following scenario is tested with SCP.

1. PCF/AF register itself to NRF via SCP.
2. AF sends a binding request to PCF via SCP. Session binding is created.
3. AF to PCF Service calls will be done via SCP.
4. Make one of BSF fail to see SCP load balancing.

The below call flows of Figure 8 to 11 explain of the detailed test scenarios.

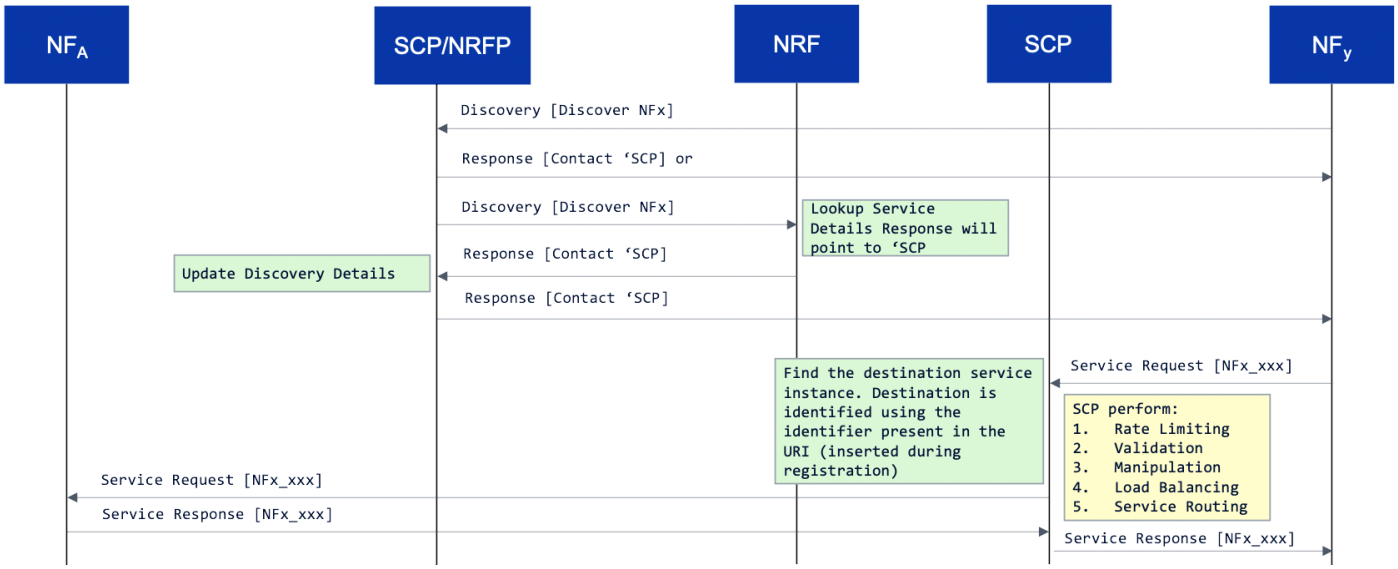
The following Figure 8 illustrates the tested registration call flow of the 5G network function over PoC test configuration. Each 5G NF registers to the target NRF to indicate its availability and to get discovered by any NF. The registration flow is proxied by the NRFP pod. Prior to routing of registration request to the actual target NRF, the NRFP pod modifies certain parameters in the NF profile to ensure that the discovery of NF profile service, routes the traffic to flow through the Service Communication Proxy (SCP) function. Upon receiving a successful registration response, the NRFP pod saves the NF Profile to be used by SCP pod.

Figure 8. NF registration call flow via SCP



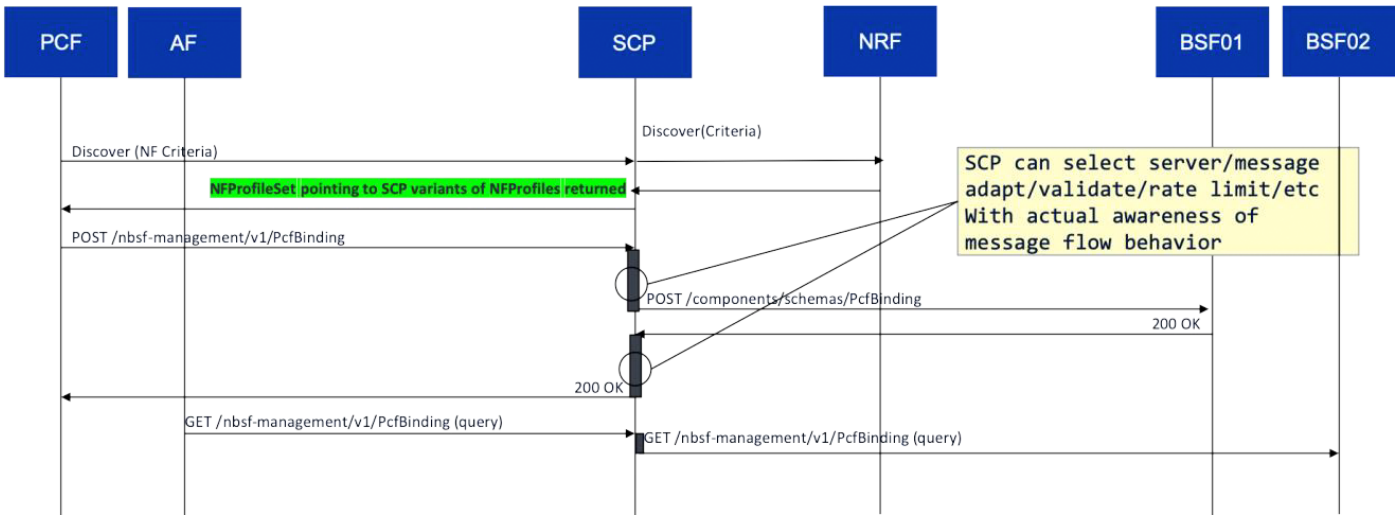
The following Figure 9 illustrates the basic discovery call flow of 5G network functions over PoC test configuration. Each 5G network function queries the target NRF to find the target NF instances and contacts to get an available service. In the following call flow, the basic discovery flow is proxied by the NRFP pod. The NRFP pod proxies the discovery request and response between the consumer service and the NRF. As the NF Profile is modified during the NF registration, the consumer contacts the SCP pod to consume the discovered functionality.

Figure 9. NF discovery call flow via SCP



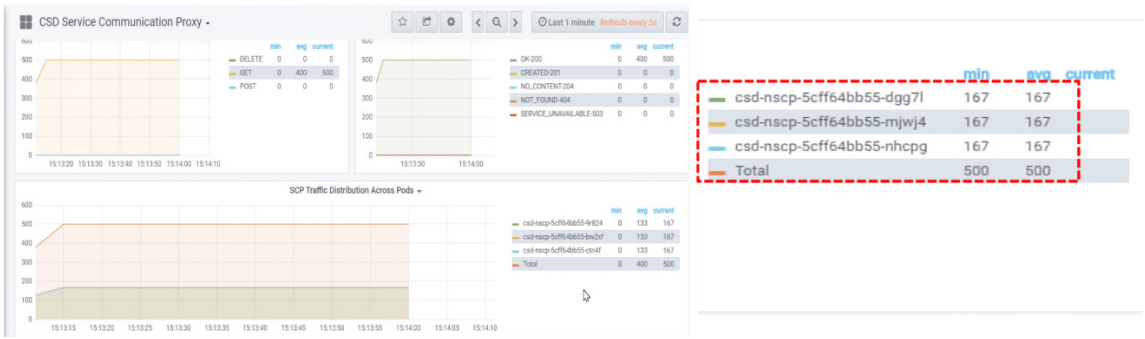
The following Figure 10 illustrates the SCP traffic control call flow between the services over PoC test configuration. Once traffic is entered to the service cluster, the SCP pod sends the traffic to each target NF – BSF1 and BSF2 evenly if both NFs have same weight or priority. If BSF1 of the figure10 has been killed, the service-mesh SCP instantly checked its status and a SCP pod will start to send the traffic to the remained BSF2 only.

Figure 10. PoC call flow via SCP



During the traffic controls in the service-mesh SCP, all active SCP pods handle the traffic equally like the below Figure 11.

Figure 11. SCP load balancing



Resiliency Test and Output

In this test, we wanted to see 'Telco-grade Reliability/Resilience for 5G cloud-native service communications.' Since SCP itself service-mesh based networking, we expected to see high resilience interactions in our testing.

As a main differentiation to the legacy architectures, cloud-native architectures are composed of small module units (container/pod) and are activated or deactivated much more quickly. This speed of activation, re-activation or de-activation implies much faster adjustment to the compute environment. The stateless nature of load balancers as well as application services implies that traffic can be re-directed much faster to other survivor instances of services. For example, a key point in reliability is mean time to recovery. When a much smaller module fails, but is automatically re-started, the mean time to recovery is much smaller.

We tested the below items over PoC to see how service-mesh SCP provides the reliability and resilience for 5G cloud-native service communications.

Test Criteria	Functions	Test Items	Description
High Availability	SCP N-tier architecture resilience	1) switchover cases - EDGE	Edge resilience test
		2) switchover cases - control	Control resilience test
		3) Worker shutdown (Environment Block)	worker resilience test
		4) shutdown (Environment Block)	App resilience test
		5) Pod restart	Container HA
eSBA platform resiliency	Telco-grade web scale framework	6) Telco-grade resilience test (Environment Block)	Platform resilience
		7) Single sign on	Single sign on test
		8) Automatic scaling	Microservice automatic scaling
	Microservices deployment	9) Registry	Microservice POD registry
	Orchestration in K8-clusters/ dashboard	10) K8-cluster and dashboards	Kubernetes dashboard
	Usage pf sidecars in eSBA architecture	11) Sidecars	Envoy sidecars usage
	Self-healing of microservices	12) selfheal of microservices	Pod self-healing
	Enterprise level docker registry for 5G services	13) registry	Docker registration

The following has been tested to see the Resiliency and high availability functions.

1. AF to PCF Service calls is handled by all active SCP Pods.
2. Make 1 Pod restart.
   2-1. Pod self-healing check.
3. Make a serviced Pod unavailable for some time.
   3-1. Check traffic is load balanced to other 2 SCP Pods.
4. Make Auto Scaling of the serviced POD.

While this test configuration is deployed with single mesh only as referred in the beginning of this chapter, it can support distributed across multiple clusters and multiple mesh network from the beginning in any actual use.

As a telco-grade SCP architecture, it should work in a manner to ensure no-single point of failure. When distributing the active pods to the deployed node, the following rules shall be kept ensuring no-single point of failure.

All singleton nodes should operate in active-standby pairs with floating IP addresses. These are load balancer nodes and OAM nodes. The floating IP addresses are pinned to the active nodes. Whenever a failure occurs, the standby takes over the role of active node, and while taking over this role, the newly activating node also assumes the floating IP address.

The compute layer of the system is provided by the routing nodes. The routing nodes run as stateless active nodes. N+K nodes are used for a performance requirement of N nodes. This ensures that K failures can be tolerated. Traffic is distributed to all working nodes. Any failed nodes are simply excluded from new routing request distribution.

The database nodes hold the persistent information. L+M nodes are used for storage requirement of L/2 nodes. Database records are distributed on a record by record basis to 2 database nodes, one node holding the primary record, and the other holding the secondary record. Unless a failure happens, all reads happen from the primary record.

In addition to the above-mentioned resiliency, as a cloud-native configuration, service-mesh SCP itself checks the health of each service pods and restarts automatically if required. Therefore, cloud-native characteristics are inherently resilient to failure. If there's any failure, it instantly moves to another configured node/zone automatically and seamlessly.

With this, we could see no failure in the service-mesh SCP architecture itself.

Observability Tests and Output

With this test, we wanted to see 'Service quality Observation in the 5G cloud-native/service-mesh architecture.'

As a main differentiation to the legacy network element, it is necessary to measure quality of service for NF with a lot of microservice attributes, but it is difficult to capture and monitor traffic in the conventional way. The difficulty arises from the typical use of encryption for HTTP/2 based traffic. With a much higher degree of complexity brought about due to the very high count of services in play, the necessity measure quality of service that can be measured with E2E full picture in 5G Core network becomes much higher.

As a cloud-native SCP is injected and configured to be in the control plane flow, the service-mesh SCP is in a very good position to provide cloud-native traffic monitoring applications (HTTP/2 detailed attributes, KPI, Alert, Log/Trace, and Visualization), and can provide quality of service measurement for the 5G microservices.

We tested the below items over PoC to see how SCP provides the key capabilities of traceability and observability.

Test Criteria	Functions	Test Items	Description
E2E tracing	Traffic visualization	1) Traffic visualization through Kiali	Access Kiali through GUI
		2) KPI Visualization through Grafana	Access Grafana through GUI
		3) E2E HTTP Signal tracing through Jaeger	Access Jaeger through GUI
KPI observability	KPI monitoring	4) KPI monitoring through Prometheus	Access Prometheus through GUI
		5) Alert management through Prometheus	Configure EMAIL and SNMP config during installation in “env.yaml”

The following has been tested to see the traceability and observability functions (Figures 12 to 14)

1. AF to PCF Service calls will be done via SCP.
2. Traffic Visualization via Kiali
3. KPI monitoring through Prometheus
4. KPI Visualization through Grafana
5. E2E HTTP Signal tracing through Jaeger

Figure 12. Web single sign-on for Service-mesh open source environment

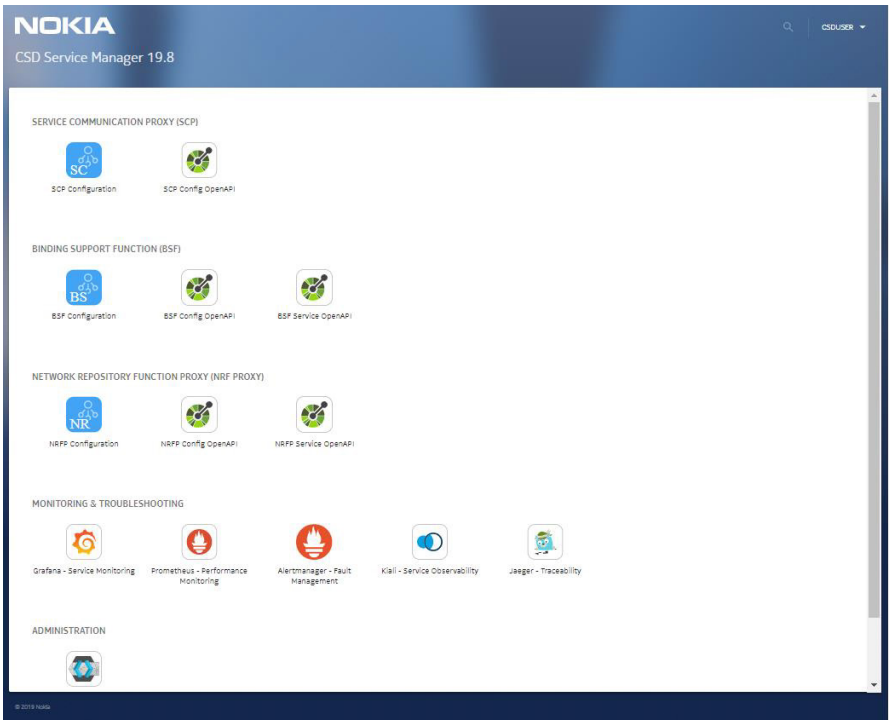


Figure 13. Service Monitoring with service-mesh enabled tools

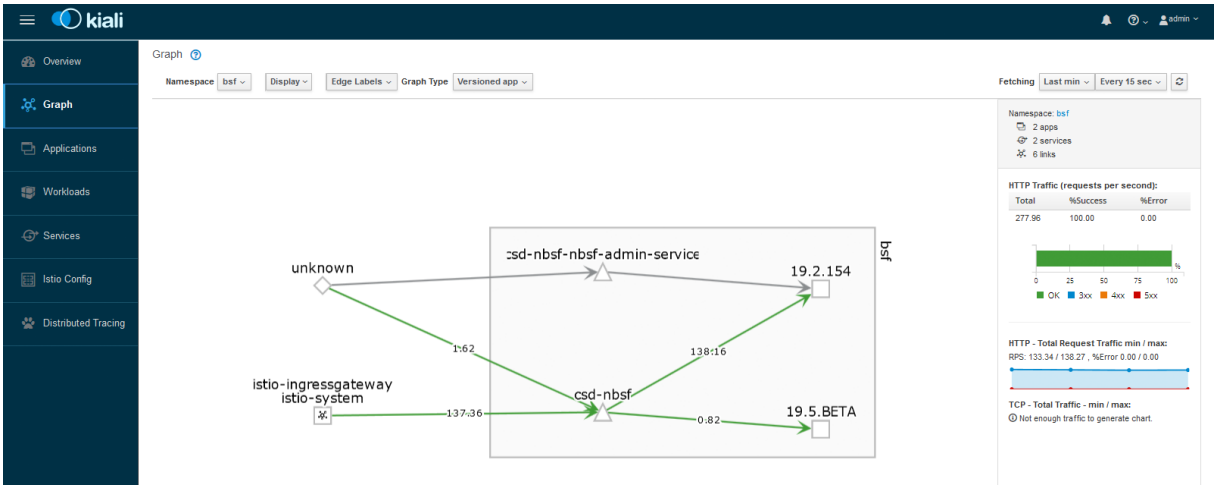
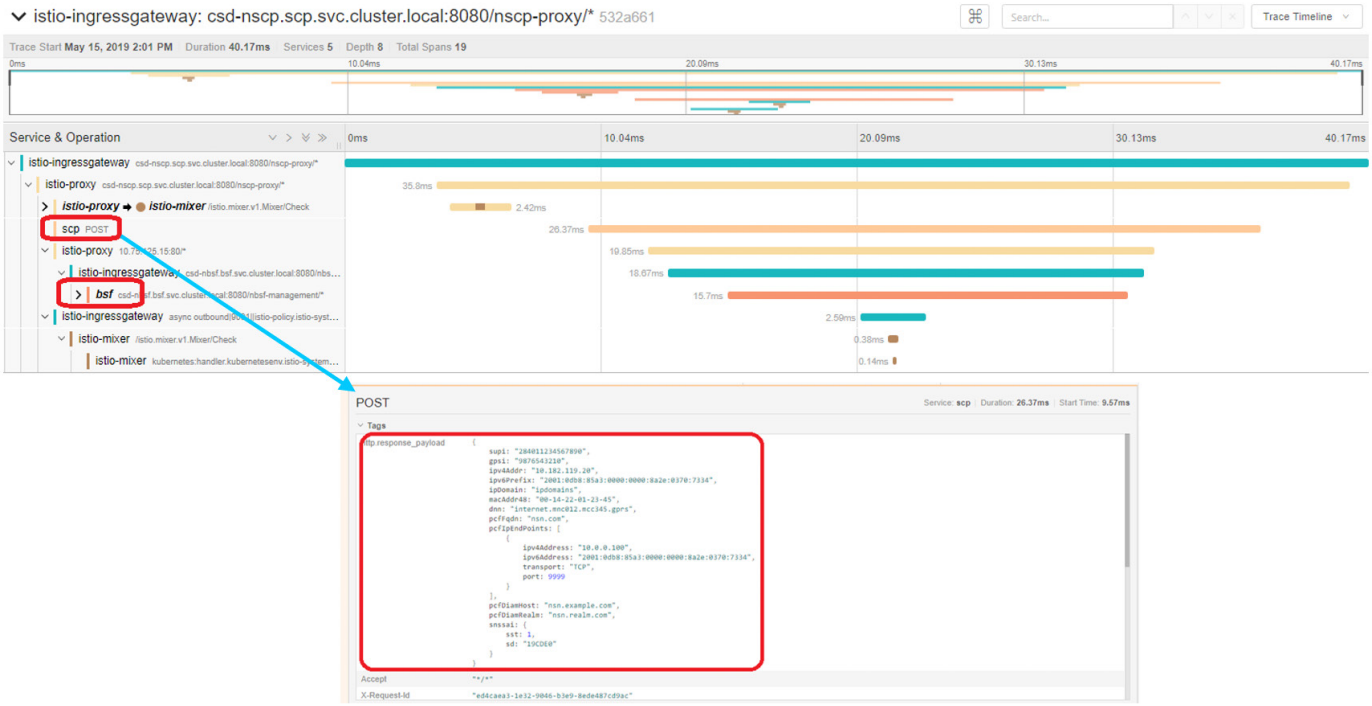


Figure 14. Observing the performance of different segments as well as request and response payloads



Findings

The above tests provide great insights into the capabilities brought into the 5G Core by the service-mesh SCP. The SCP has been standardized in 3GPP Rel.16 to enhance the workings of the service-based interfacing in 5G Core. The typical concerns around the workings of the SCP would be the same. Any learnings from this PoC would apply equally well.

SCP Functional Test

The tests showed delegated discovery and indirect communications at work. The pre-Rel.16 SCP roughly approximates the Rel.16 concept of model D-Indirect interactions. In this scenario, the discovery portion is also delegated. Based on the discovery, the selection and reselection are carried out by the SCP.

We got successful proof points of sticky and non-sticky flows. Both concepts are important as control plane use cases often require that a selected service instance is the one to handle all subsequent requests in the context of an interaction lasting multiple exchanges with the producers.

NF producers report their capacity to the NRFs via their NF profiles, as well as priority and locality. These attributes require weighted load balancing with special regard of locality. Our tests showed proper distribution of traffic across all producers with full awareness of the complete workload presented to the producers available to the SCP.

Resiliency Test

In these set of tests, we induced failures at different layers of the cloud-native solution. We also monitored the effects of control node failures.

Failures were induced at complete node level (equivalent of hardware failure, or VM failure), or pod failure. In total, we cycled through failures of load balancer hosts, SCP domain microservice hosts, as well as different levels of pods. In all cases, the traffic flow was observed to continue with minimal interruption.

We also observed automatic recovery (self-healing), as well as auto-scaling. This test also allowed us to to gain a better understanding of cloud-native environments. Through this test we learned about:

- K8s operations
- Service-Mesh concepts
- Sidecar injection concepts
- Microservice level registry

The success rates of traffic observed considering different types of failures provided us a good understanding of the importance of cloud-native architectures with active-active stateless layers, as well the important role these architectures play in the automation of recovery and scalability of cloud-native solutions.

Observability

These tests were meant to understand the diversity of traffic, the performance of the different components of the SCP, as well as downstream NFs. These tests gave us the proof point of single pane observability and traceability.

We were able to see traces of the un-encrypted traffic with several criteria of performance as well as headers. We were also able to drill down on performance scatter plots to see the time expenditures of different layers of the request response loop. Many aspects of these functions were brought in by the underlying service-mesh architecture and enabled by the service-mesh oriented implementation of the SCP. The results from these tests gave us the confidence of unprecedented insight for the operations personnel of the 5G Core systems.

Summary

This joint whitepaper by Nokia and SK Telecom for the SCP's technical evaluation brought about great understanding of the tremendous potential of cloud-native architecture for the 5G Core. Service-mesh SCP is a newly standardized 3GPP Rel.16 system in 5G Core which provides service-level load distribution, QoS control, resiliency and monitoring functions for all 5G NF signals. This architecture is a big step forward in enabling much higher level of responsiveness in the rollout, continuous monitoring, as well as evolution of the Core network. As a result, developers of the cloud-native NFs can focus more on application and business logic instead and, leave out the rest of traditional features such as load-distributions and resiliency to be offloaded into the service-mesh SCP.

Nokia and SK Telecom believes that the introduction of the 3GPP's SCP as an enhancement to the service based architecture is a step in the right direction. The service-mesh SCP reduces complexity of interconnection between multiple 5G NFs by allowing 5G NFs not to communicate directly with each other, but instead communicate via the SCP. This cloud-native architecture is built using service-mesh components which provides a secure, reliable, and resilient service communications layer to handle the delivery of signaling requests for the 5G Core deployment.

Note that our evaluation test and observations need more rigorous analysis as the cloud-native architecture principles are a big step change in telco-grade 5G signaling plane. Aside from the aforementioned SCP's key elements and benefits, further studies should be performed which are 1) Performance verification for large-scale capacity test with latency impact with SCP, 2) Architectural evaluation for a fully meshed 5G Core NFs/components containerized, 3) Design evaluation for an optimized 5G framework for signal deliveries with assistance from AI/ML based policy automation.

This jointly collaborated evaluation of the SCP for next-generation 5G network also aims to present a need for a de facto technology to smoothly introduce the 5G cloud-native services. As the SCP architecture is changing a communication service environment, telco and vendors should collaborate closely together to create a ecosystem to adopt and contribute guidelines to international standard technologies.

Definitions and Abbreviations

Definitions

Container: This refers to the package software into standardize units for development, shipment and deployment. Docker is one of the de facto containers.

Direct Communication: This refers to the 3GPP defined communication between 5G NFs or 5G NF services without using an SCP.

Indirect Communication: This refers to the 3GPP defined communication between 5G NFs or 5G NF services via an SCP.

Istio: This is an open-source service mesh to connect, monitor, and secure microservices deployed on-premise, in the cloud, or with orchestration platforms like Kubernetes.

Istio Egress Gateway: Istio uses ingress and egress gateways to configure load balancers executing at the edge of a service mesh. Egress gateway is a symmetrical concept; it defines exit points from the mesh.

Istio Ingress Gateway: Istio uses ingress and egress gateways to configure load balancers executing at the edge of a service mesh. Ingress gateway defines entry points into the mesh that all incoming traffic flows through.

Kubernetes: This refers to the container orchestration system for Docker containers.

Kubernetes Pod: This refers to the group of containers that are deployed together on the same host.

Microservice: This refers to the network functions or applications as a collection of loosely coupled services.

Network Function: A 3GPP adopted or 3GPP defined processing function in a network, which has defined functional behavior and 3GPP defined interfaces.

NF service: This refers to the 3GPP defined functionality exposed by a NF through a service-based interface and consumed by other authorized NFs.

SCP Sidecar: This refers to the service agent of SCP that is a lightweight, transparent proxy with support for HTTP/2 and gRPC 5G Core signaling terminations. This can support the service-mesh to handle outbound traffic for each NF service.

Service: This refers to the 3GPP defined functional behavior or process that is independently deployable in the 5G SBA.

Service-mesh SCP: This refers to the dedicated infrastructure layer for facilitating service-to-service communications between 5G NF services, often using a SCP sidecar.

SBA: This refers to the 3GPP defined architecture for 5G NFs.

SBI: This refers to the 3GPP defined interfaces for 5G NFs which communicates by service messages (via HTTP/2)

Abbreviations

3GPP	Third Generation Partnership Project
5G	Fifth Generation
API	Application Programming Interface
BSF	Binding Support Function
DB	Database
eSBA	enhanced Service Based Architecture
FQDN	Fully Qualified Domain Name
gRPC	Remote Procedure Call
GUI	Graphical User Interface
IP	Internet Protocol
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
NE	Network Element
NF	Network Function
NRF	Network Repository Function
PCF	Policy Control Function
SBA	Service Based Architecture
SBI	Service Based Interface
SCP	Service Communication Proxy
SLA	Service Level Agreement
SMF	Session Management Function
SCP	Service Communication Proxy
TCP	Transport Control Protocol
UDP	User Datagram Protocol

Reference

1. 3GPP TS 23.501, (https://www.3gpp.org/ftp/Specs/archive/23_series/23.501)

2. P. Jamshidi et al. “Microservices: The Journey So Far and Challenges Ahead,” in IEEE Software, vol. 35, no. 3, pp. 24-35

3. W. Morgan, “What’s a service mesh?”, (https://buoyant.io/2017/04/25/whats-a-service-mesh-and-why-do-i-need-one)

4. Envoy Proxy, (https://www.envoyproxy.io/)

5. Why does 5G need a service mesh? (https://onestore.nokia.com/asset/202073)

6. Patterns for Composite Containers, (http://blog.kubernetes.io/2015/06/the-distributed-system-toolkit-patterns.html)

7. Istio traffic management, (https://istio.io/docs/tasks/traffic-management/)

About Nokia

At Nokia, we create technology that helps the world act together.

As a B2B technology innovation leader, we are pioneering the future where networks meet cloud to realize the full potential of digital in every industry.

Through networks that sense, think and act, we work with our customers and partners to create the digital services and applications of the future.

Nokia OYJ
Karakaari 7
02610 Espoo
Finland
Tel. +358 (0) 10 44 88 000

Document code: CID 207387 (March)