5G Service Based Architecture enables universal core

Hannu Flinck, 18.09.2018

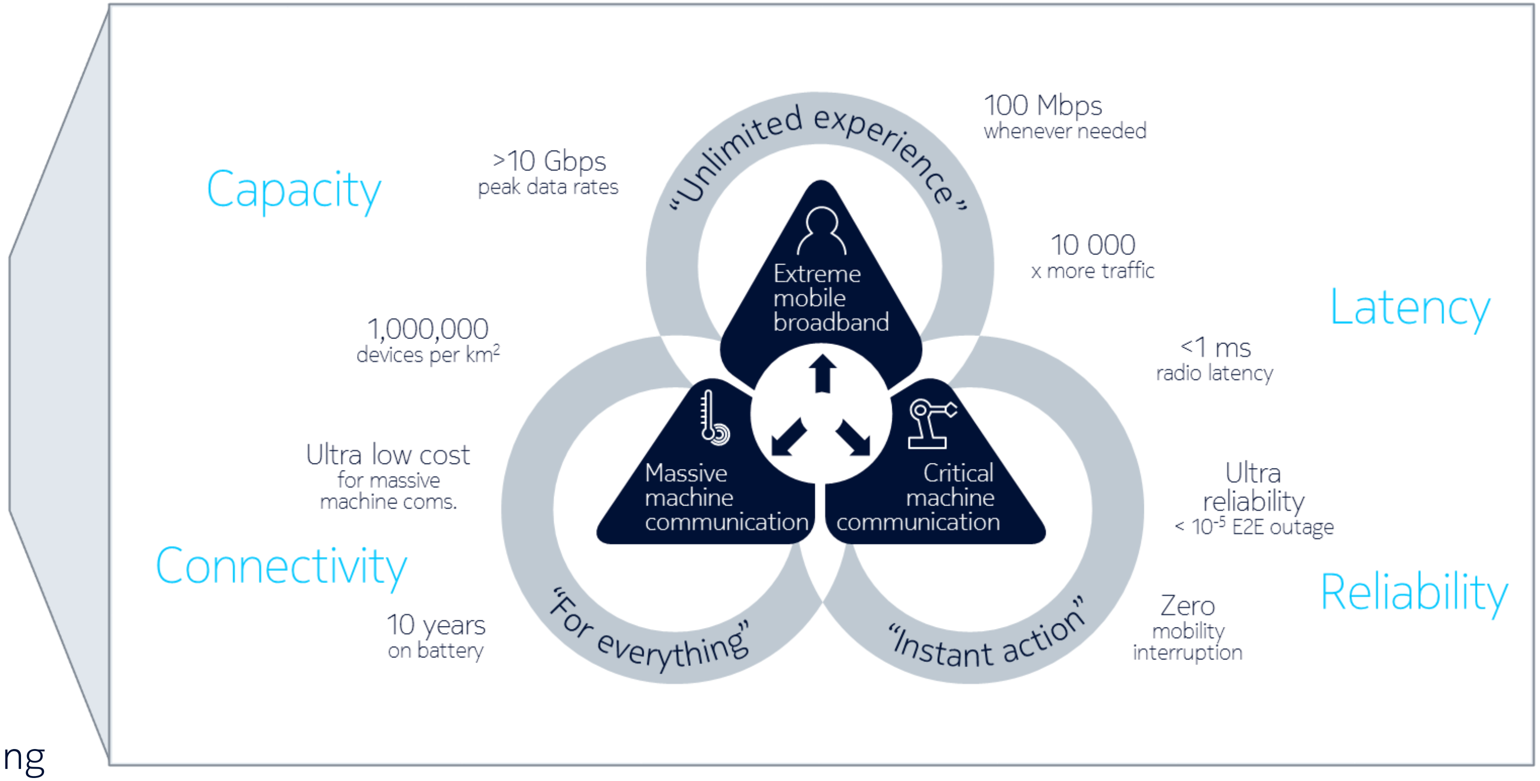
Why 5G? New user demands – with extremely diverse requirements

Devices
1.5 GB/day

Smart Factories
1 PB/day

Billions of sensors connected

Autonomous driving
1ms latency



Capacity

>10 Gbps
peak data rates

1,000,000
devices per km^2

Ultra low cost
for massive
machine coms.

Connectivity

10 years
on battery

100 Mbps
whenever needed

10 000
x more traffic

Latency

<1 ms
radio latency

Ultra
reliability
< 10^{-5} E2E outage

Reliability

Zero
mobility
interruption

Design and architecture principles:
flexible | scalable | automated | cloud native
software centric | dynamic network slicing

Unleashing the potential of 5G – driven by Service Based Architecture

Powerful

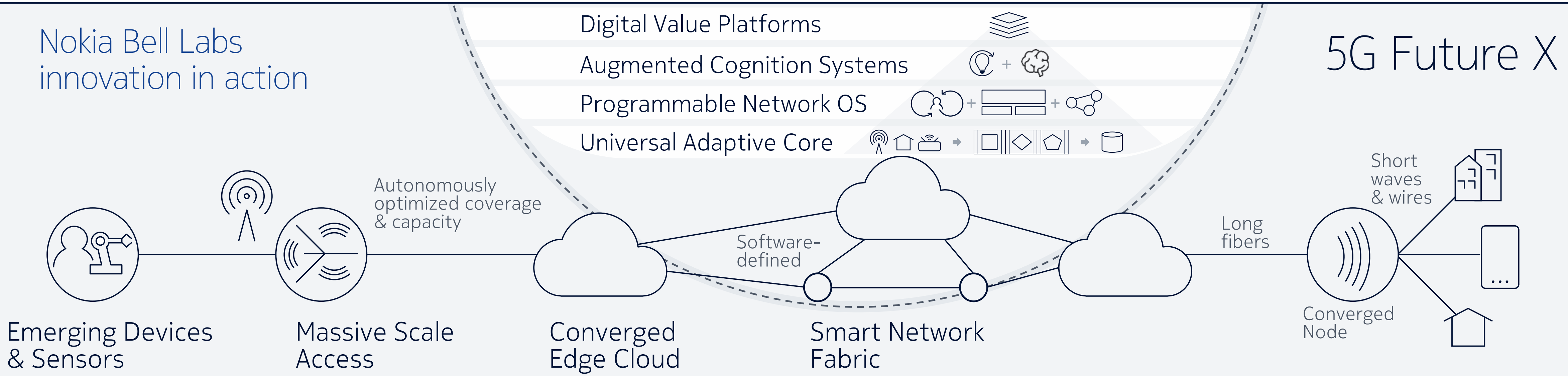
Efficient

Intelligent

Flexible

Nokia Bell Labs
innovation in action

5G Future X



Digital Value Platforms

Augmented Cognition Systems

Programmable Network OS

Universal Adaptive Core

Autonomously optimized coverage & capacity

Software-defined

Long fibers

Short waves & wires

Converged Node

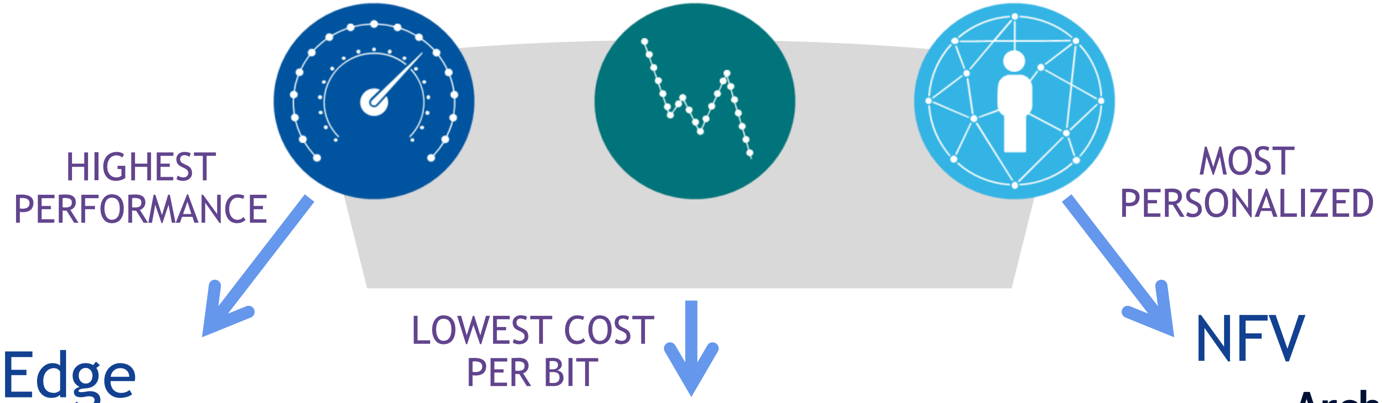
Emerging Devices & Sensors

Massive Scale Access

Converged Edge Cloud

Smart Network Fabric

Architectural shifts are underway…



Architectural shift 3:

Distributing the Core Cloud… in the Network

Architectural shift 4:

Distributing the Access Network

Edge Cloud/Access

SDN

Architectural shift 2:

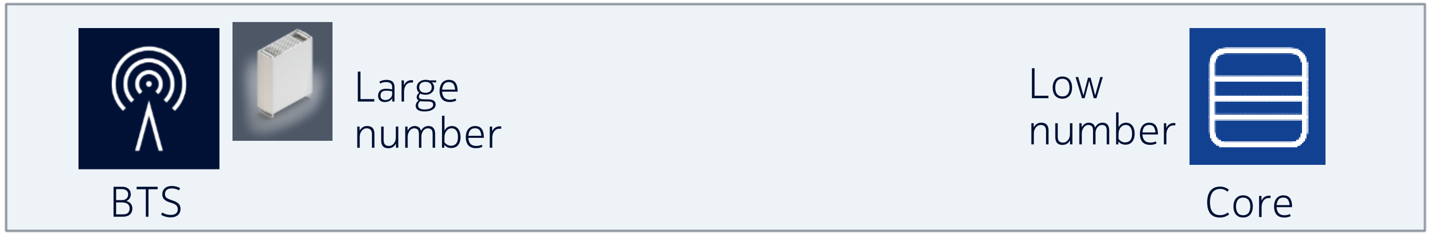
Software-Defining the Network

NFV

Architectural shift 1:

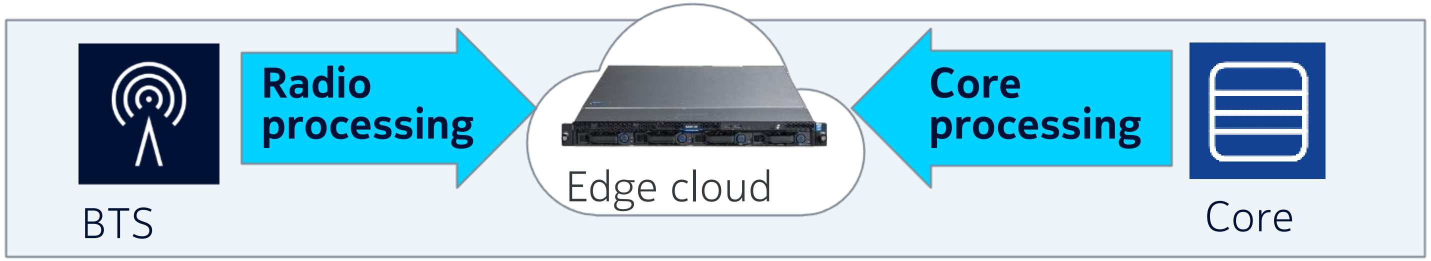
Virtualizing the Network

Today



- Current radio processing and control is distributed.
- Current core is centralized.

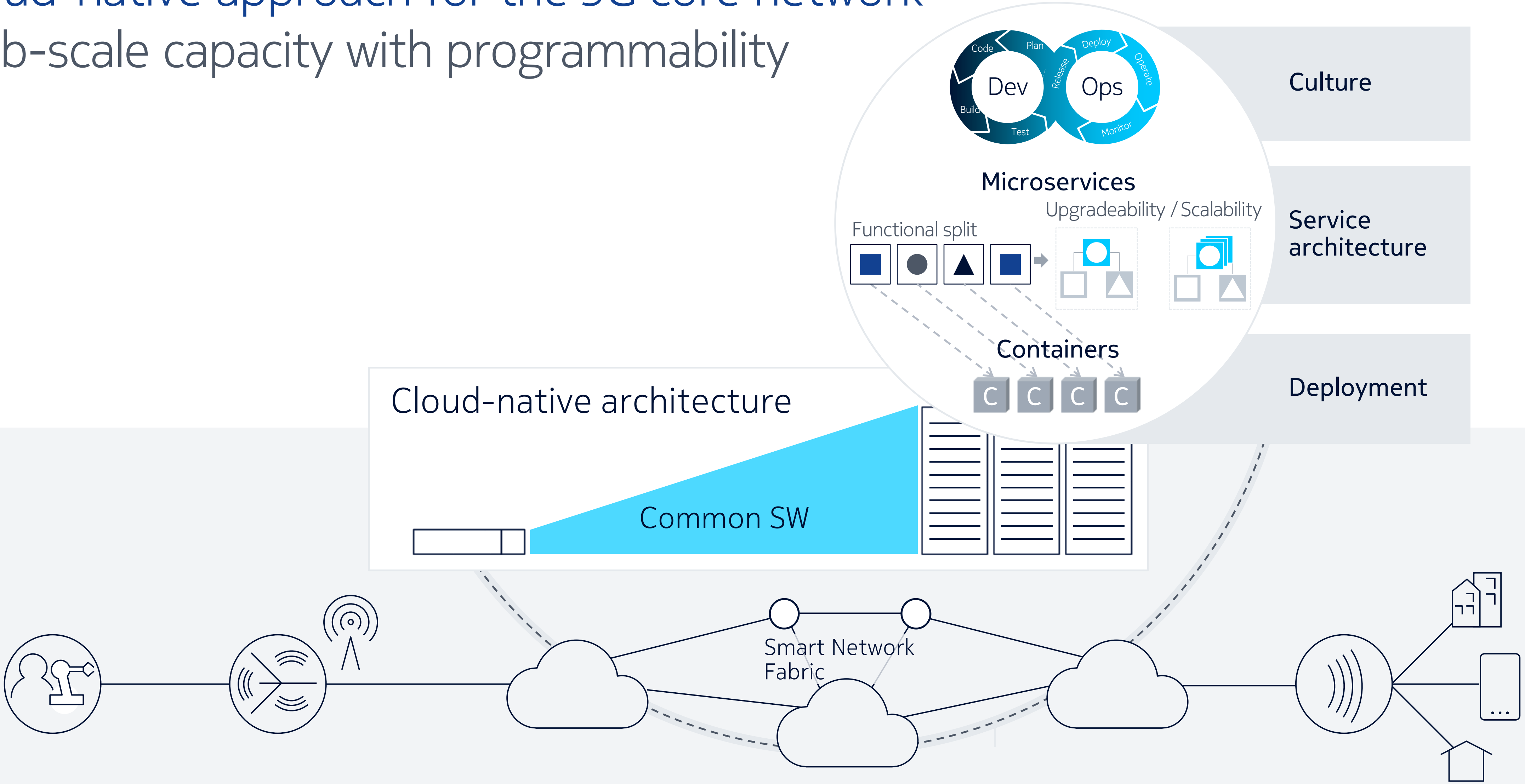
Target



- Radio processing and control more centralized for scalability.
- Core more distributed for low latency.

Cloud-native approach for the 5G core network

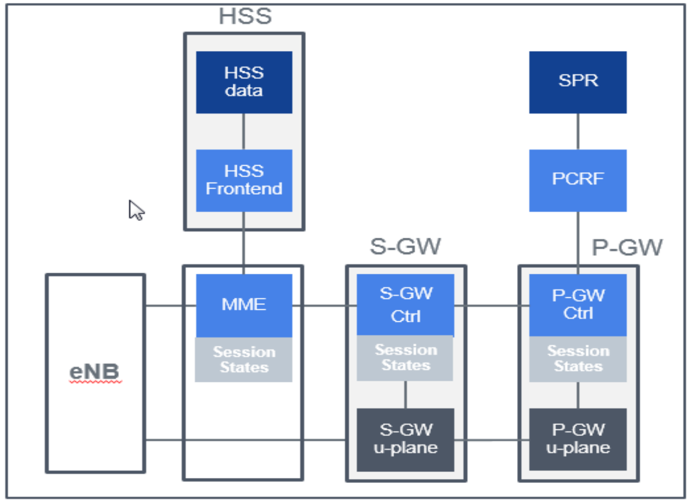
Web-scale capacity with programmability



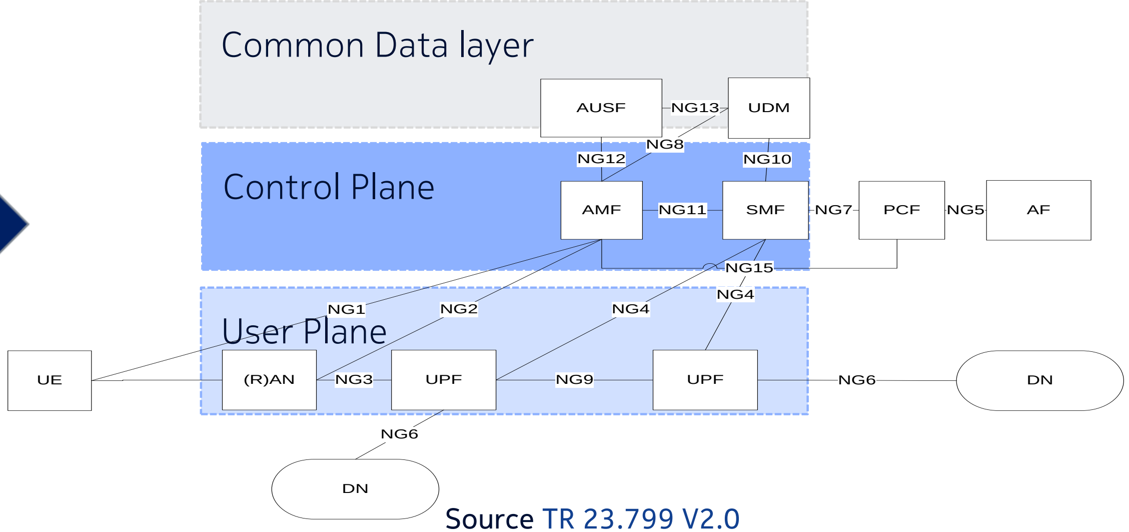
3GPP Control Plane evolution

from boxes to cloud native Network Functions and services

EPC: box-driven function split

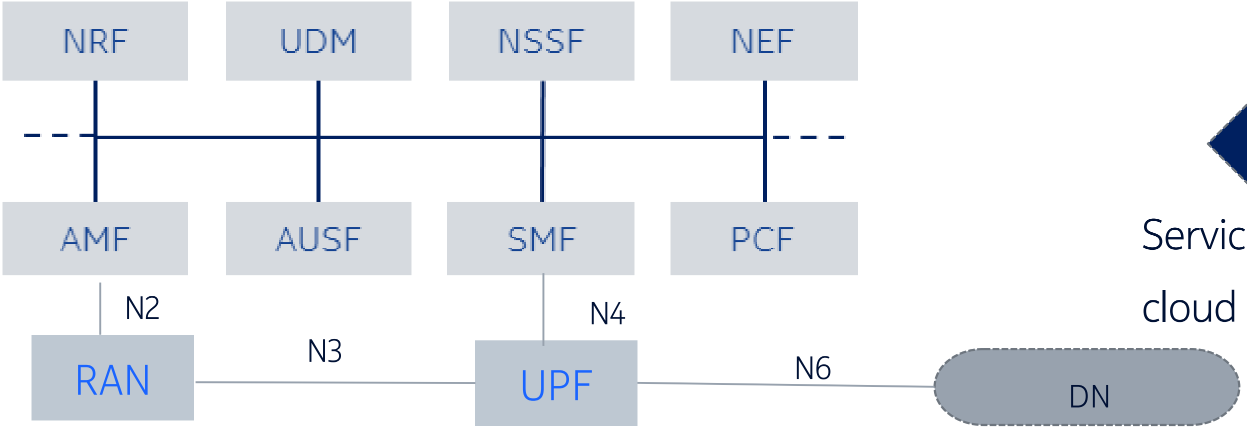


NG core: reference point based



Source TR 23.799 V2.0

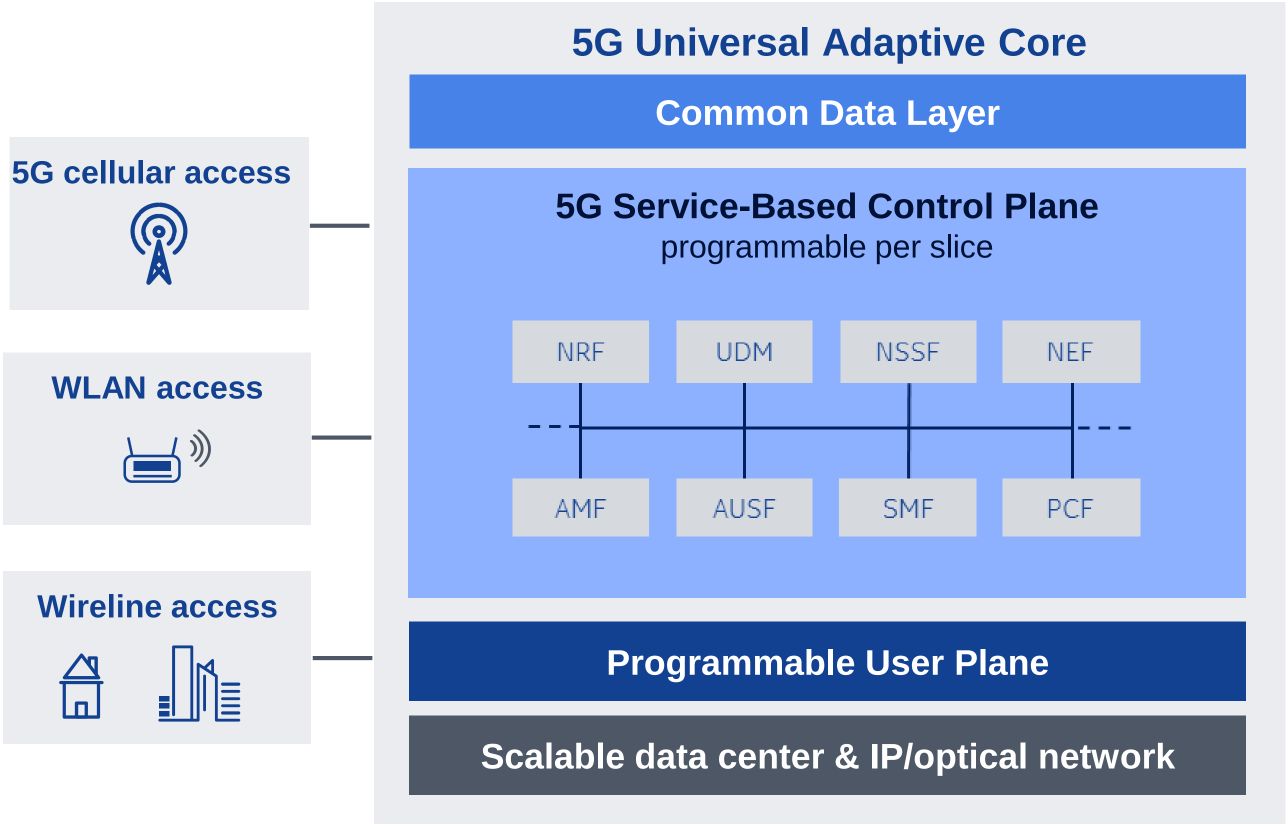
NG core: Service Based Architecture



Service Based Architcture using cloud native Network Functions

Service Based Architecture (SBA)

Scalable core architecture for the 5G era



Major Changes

- Control Plane – User Plane Separation
- Service Based Architecture (SBA)
- Compute Storage Separation

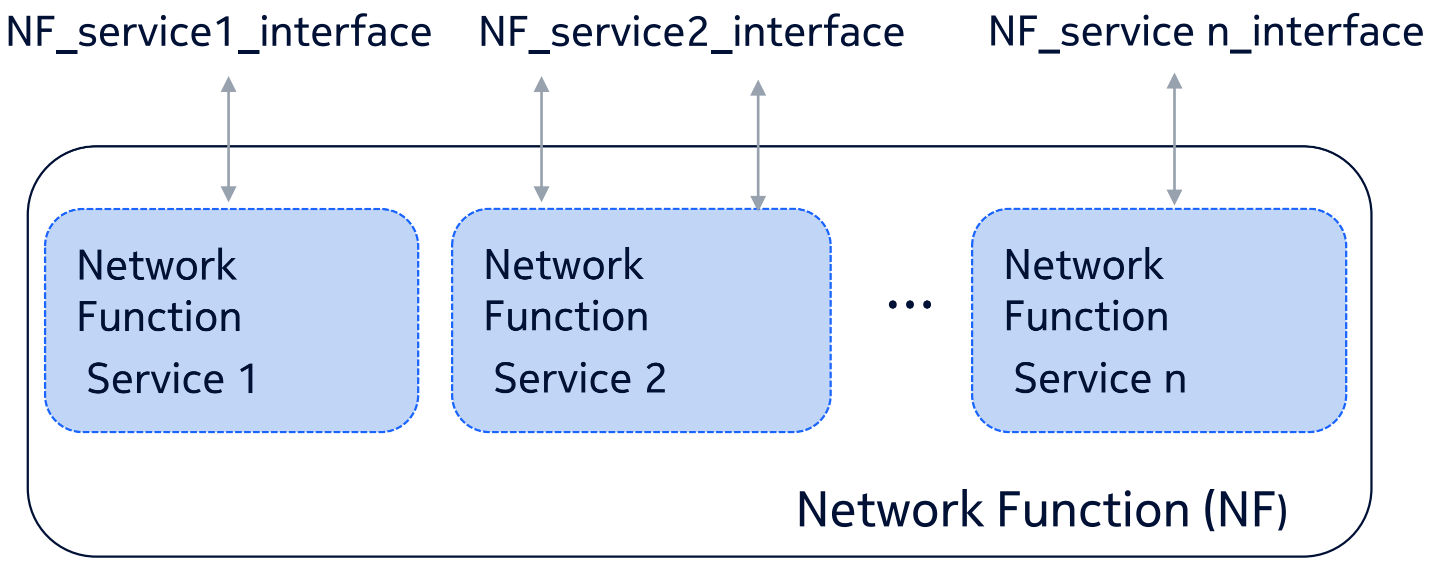
Agile Virtual Edge/Core

- Flexible distribution, scaling of edge and core functions
- Access specific control functions minimized, and contained in edge functions

Common core platforms deliver all services over all forms of access

Network Functions are made out of Network Function Services

Service consumers use services over well defined REST-interfaces

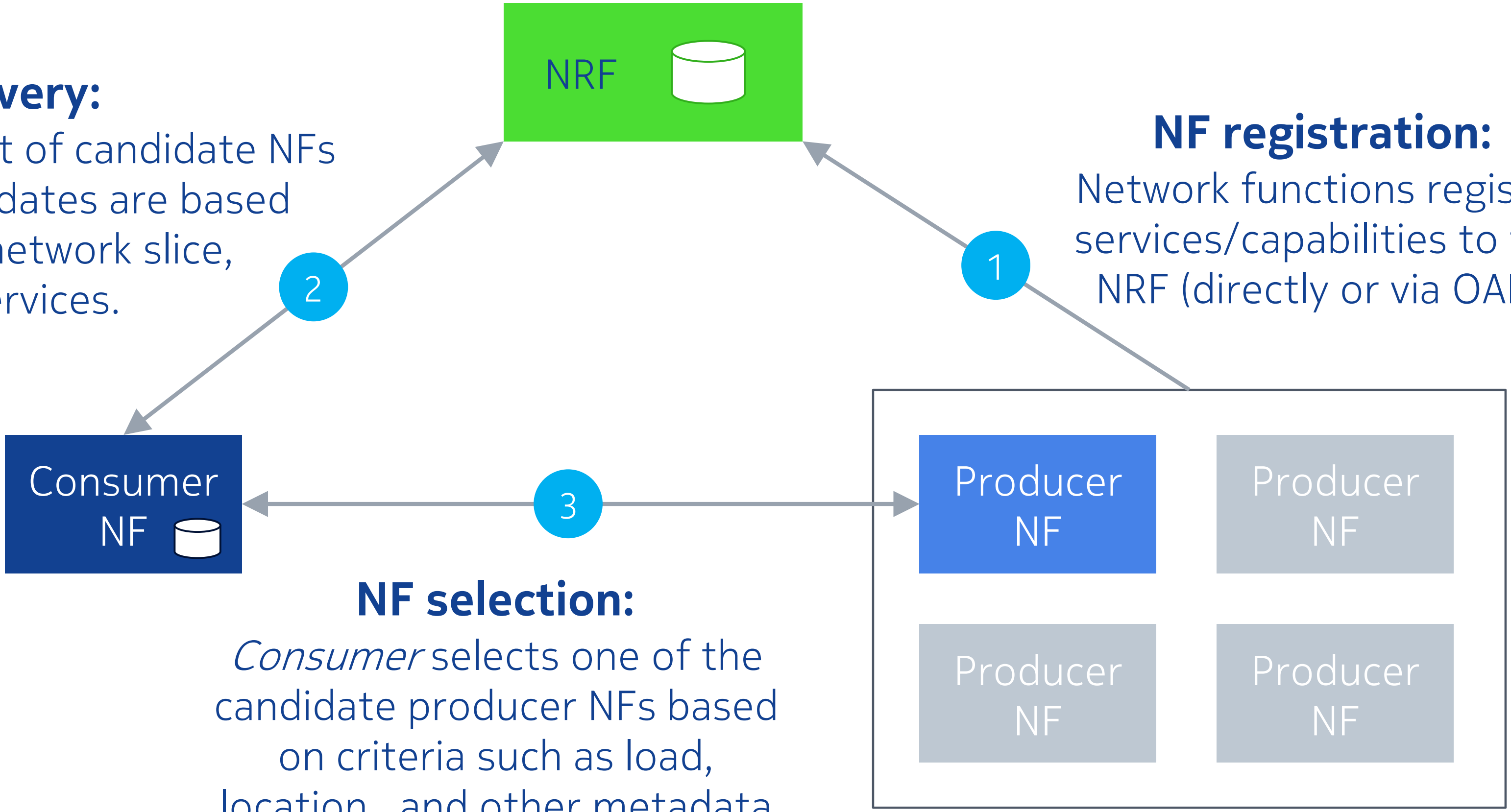


- NF service: a functionality exposed by a NF through a service based interface.
- NF services should be self-contained, reusable and independent.
- Within a given communication context, a service may take the role of either service consumer or service producer.

© Nokia 2018

Service Based Architecture

Principles of network function and service discovery



NF discovery:
Consumer obtains a list of candidate NFs via NRF query; candidates are based on NF type, set, network slice, required services.

NF registration:
Network functions register services/capabilities to the NRF (directly or via OAM)

Consumers should cache NRF responses to keep NRF load low
→ subscribe to NRF updates

NF selection:
Consumer selects one of the candidate producer NFs based on criteria such as load, location, and other metadata

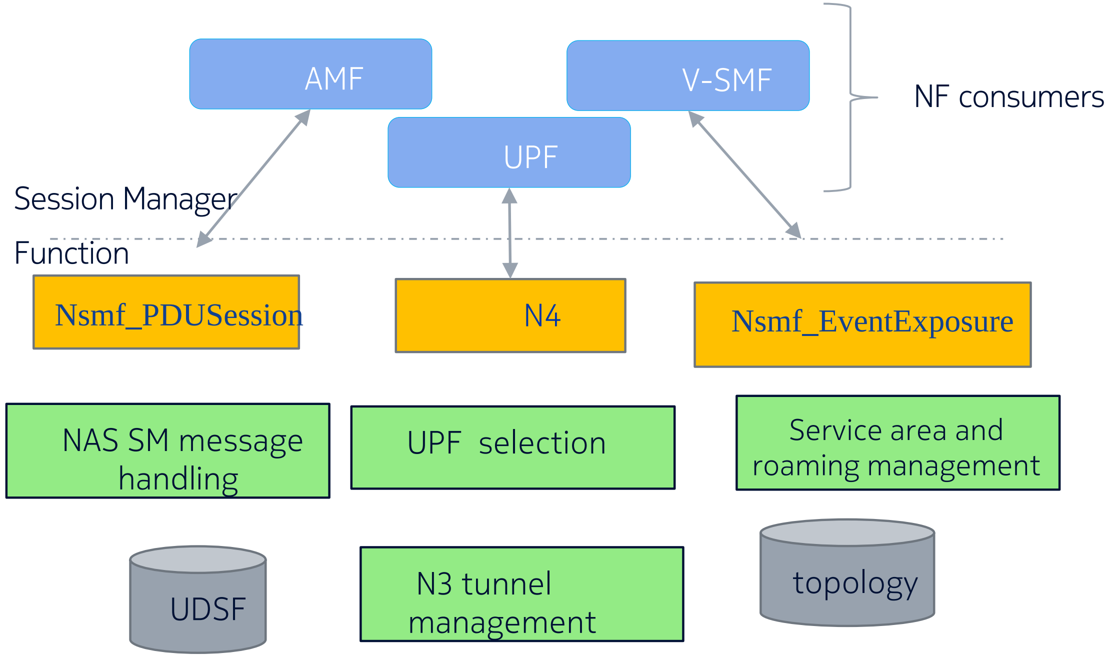
NF selection separated from discovery to allow flexible NF-specific selection methods

© Nokia 2018

Microservices design pattern applied to 5G Network Functions

How to find optimal size for microservices

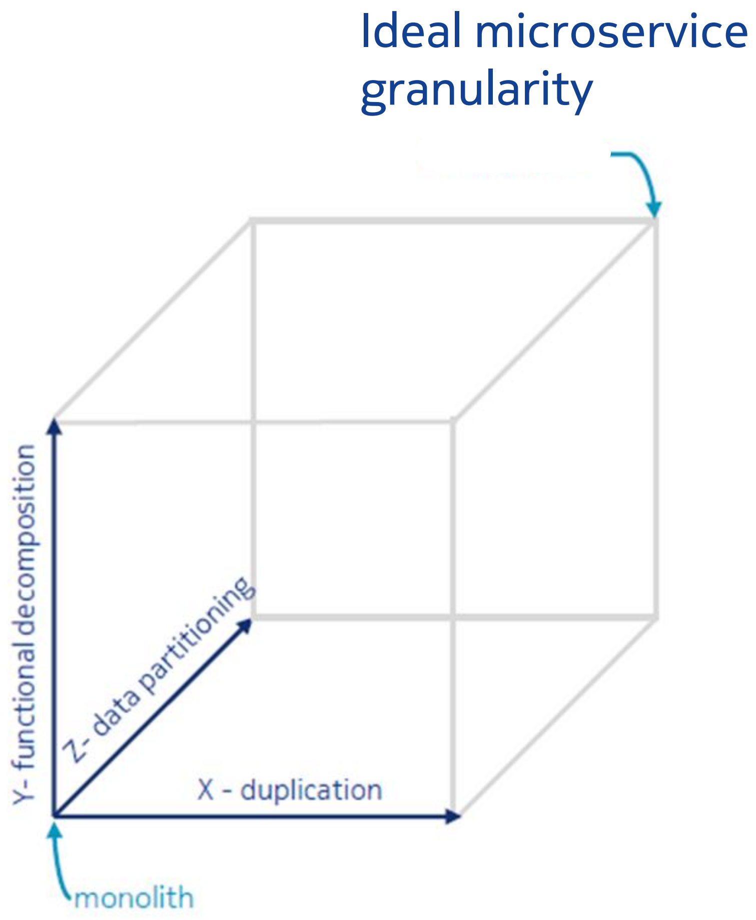
- Microservices are an architectural and organizational style to software development.
- Microservices:
  - Unit of distribution with single responsibility.
  - Part of a distributed system.
  - Are loosely-coupled.
  - Have a single bounded context.
  - Contained in their own server (VM or container).



- Challenges:
  - Finding the optimal size of a microservice is an art.
  - Complexity moves to interactions of the microservices.
  - How to design communication across microservice boundaries?
  - A microservice will often use a combination of sync and async communication styles.

Granularity and scalability of Network Functions and their services

Modelled as microservices: define the bounded context



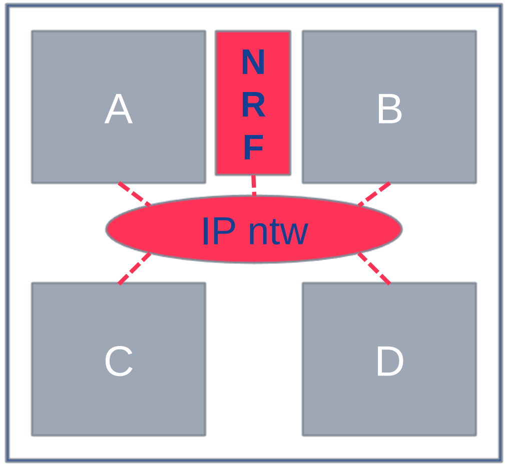
Scale cube

- X-axis scaling:
  - Multiple copies of the service behind a loadbalancer.
  - Provides capacity and high availability.
- Y-axis scaling:
  - Number of microservices.
  - Size measurements: e.g. number of responsibilities, number of files/LOC.
  - Number of interactions.
- Z-axis scaling:
  - Each service/server is responsible for only a subset of the data.

Role of Service Framework in 5G

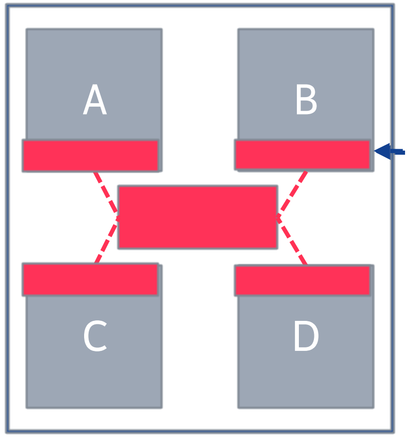
Discovery, selection and routing

3GPP model for Rel. 15 – "Centralized discovery"



Centralized discovery and availability monitoring by NRF (Network Repository Function), with distributed selection and message routing.

Service Framework discussions for Rel. 16
– Service Mesh for micro-services



Service Framework. Could be a sidecar proxy (e.g. a utility in Kubernetes Pod) **loosely coupled** to the main application container

Not agreed in 3GPP, but under discussions.

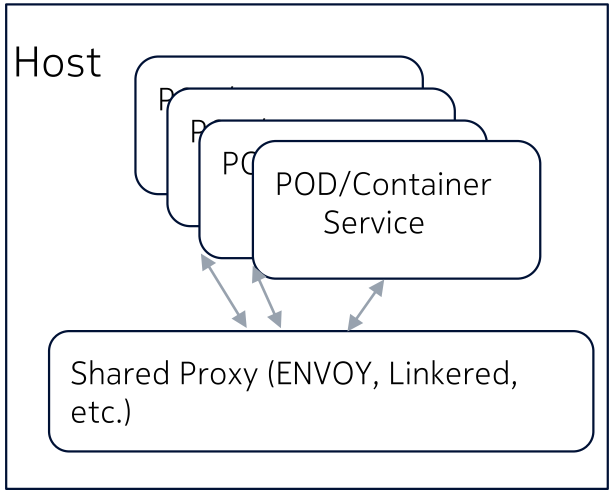
Centralized control plane for discovery, availability monitoring, selection and message routing with distributed user plane ("sidecar")

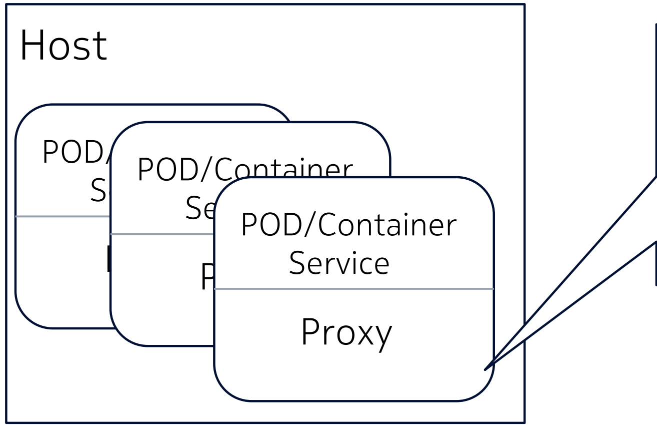
© Nokia 2018

Use of microservices leads to Service Mesh approach

Microservice = unit of distribution with bounded context

- Microservices approach leads to hundreds to thousands of small service instances that may be rescheduling from moment to moment by the orchestrator.
- Each micro service can be written in a different language with different libraries leading to different versions and behavior of protocols.
- Service mesh is **a networking abstraction layer** above TCP/IP to handle service to service communications.





In case of more capable proxy like Linkerd, this deployment will cost few hundred MBs of memory per pod.

QUIC vs. HTTP/2

Does QUIC bring any advantages over HTTP/2 in the SBA or microserve settings?

- Connection Setup delay ~ latency
  - 3 RTTs and 1 RTT for reconnect with HTTP/2.
  - QUIC can achieve faster connection establishment by combining encryption and connection handshakes: 1 RTT and 0-RTT.
  - BUT for 0-RTT Data is limited to idempotent requests.
- Stream Multiplexing in both.
  - Helps to avoid head of line blocking. But needs mapping of request–responses to their own parallel streams.
  - BUT what about the bounded context? A shared connection seems to create a shared context!
- Connection Migration.
  - QUIC allows connection migration while a session is in progress. BUT only for client side.
  - Maybe MPQUIC would be helpful here.
- Pluggable Sender Side Congestion Control in QUIC at the application level.
  - Interference with other traffic? May vary between implementations.
- Improved header compression and Improved Recovery and Acknowledgement..
  - Are these useful inside a DC?

© Nokia 2018

Conclusions

- 5G core is being re-designed to be cloud native in the Service Based Architecture.
- Services are currently grouped into Network Functions that expose their contained services to NF consumers. A centralized Network Repository Function offers service discovery.
- The interaction model of the services follows REST client – server model that has its own limitations.
- In micro-service philosophy bounded context defines the service granularity. But how to factor in optimal use of HTTP2 and QUIC multiplexing and connections?
- Not clear if QUIC really provides justifiable benefits in this use case.
- How would transition to QUIC happen?
  - HTTP2 over QUIC? A proxy GW?