KOREA UNIVERSITY

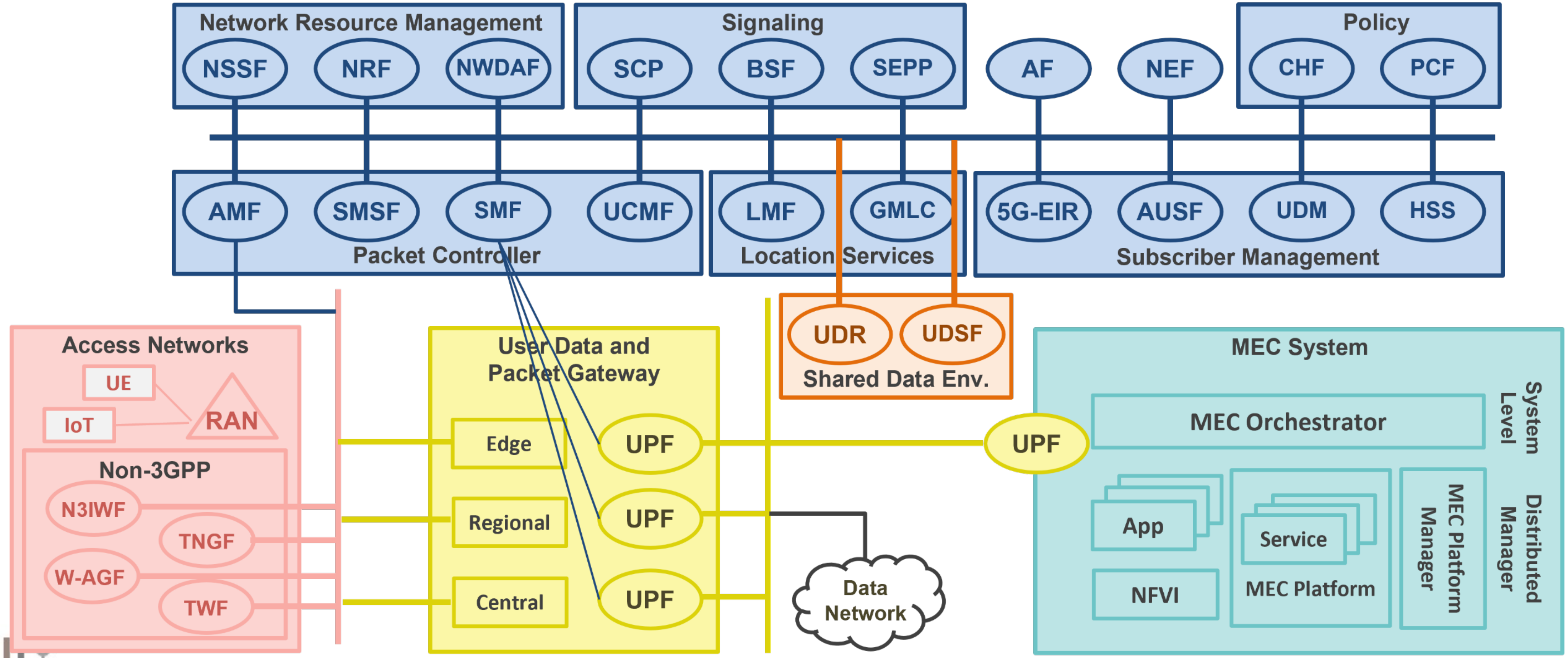
Intelligent and Robust 6G Mobile Core Networks

2024.01.18.

Sangheon Pack
Korea University
(Joint work with Y. Jeon and H. Jeong)

Introduction to 5G (1/3)

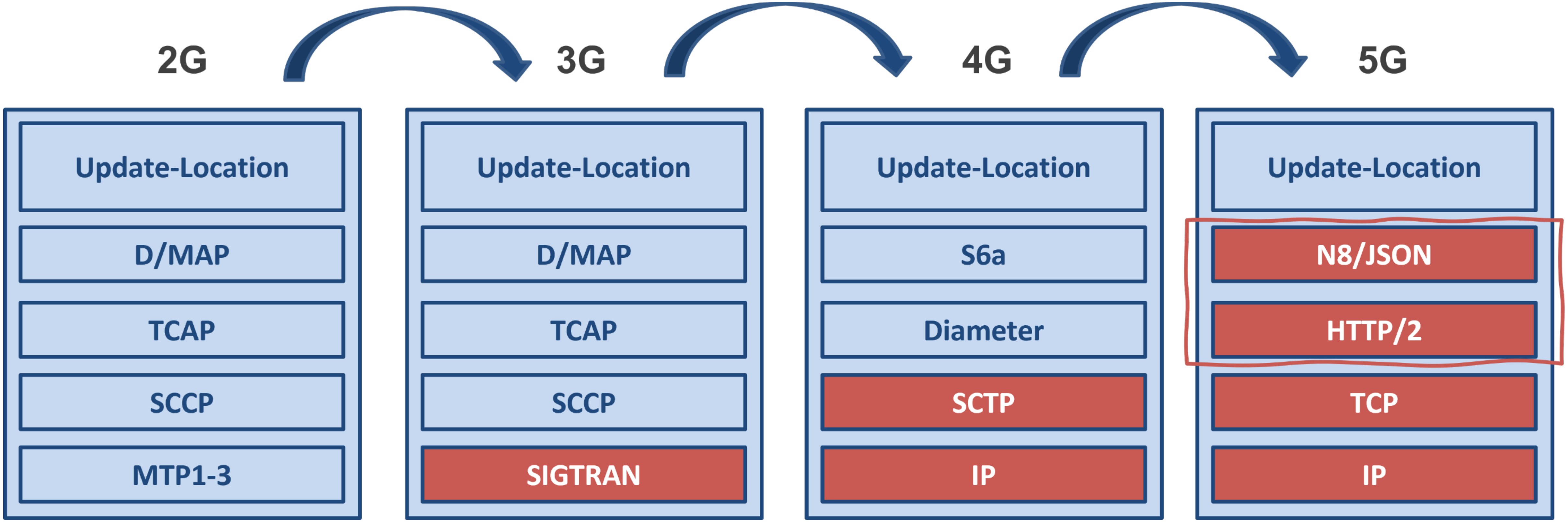
- **5G Architecture**



https://devopedia.org/5g-service-based-architecture

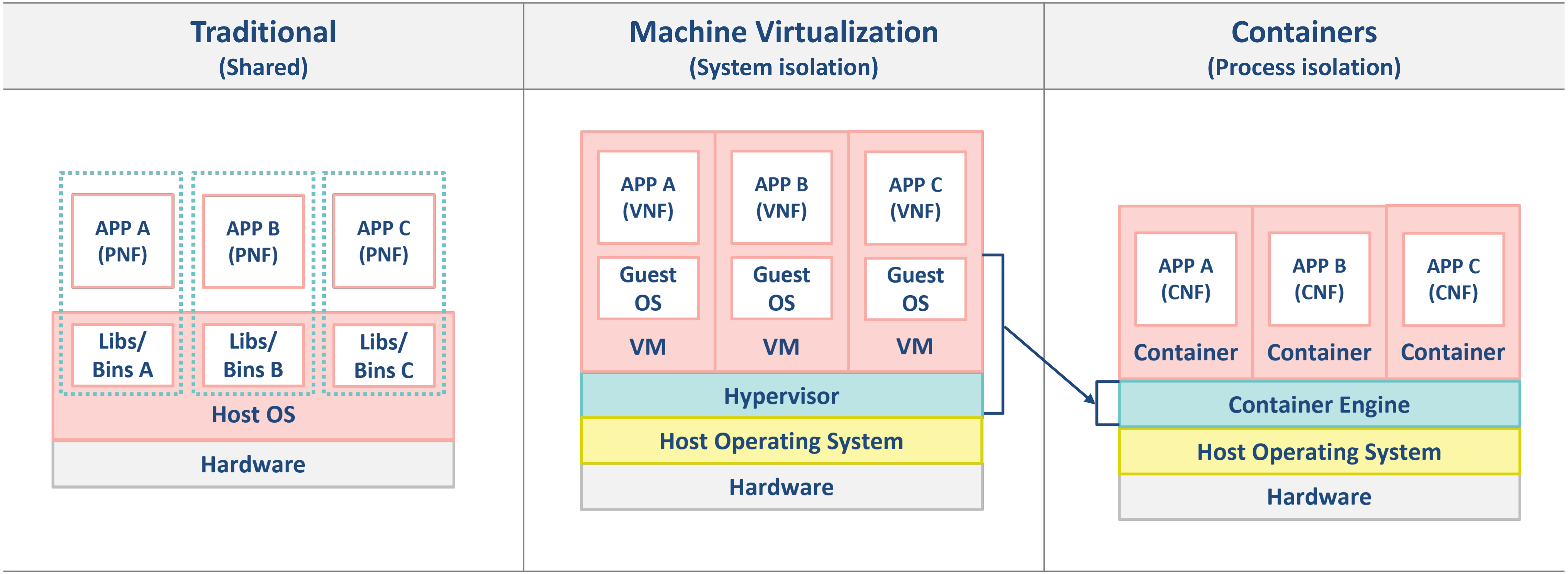
Introduction to 5G (2/3)

- Protocol Evolution

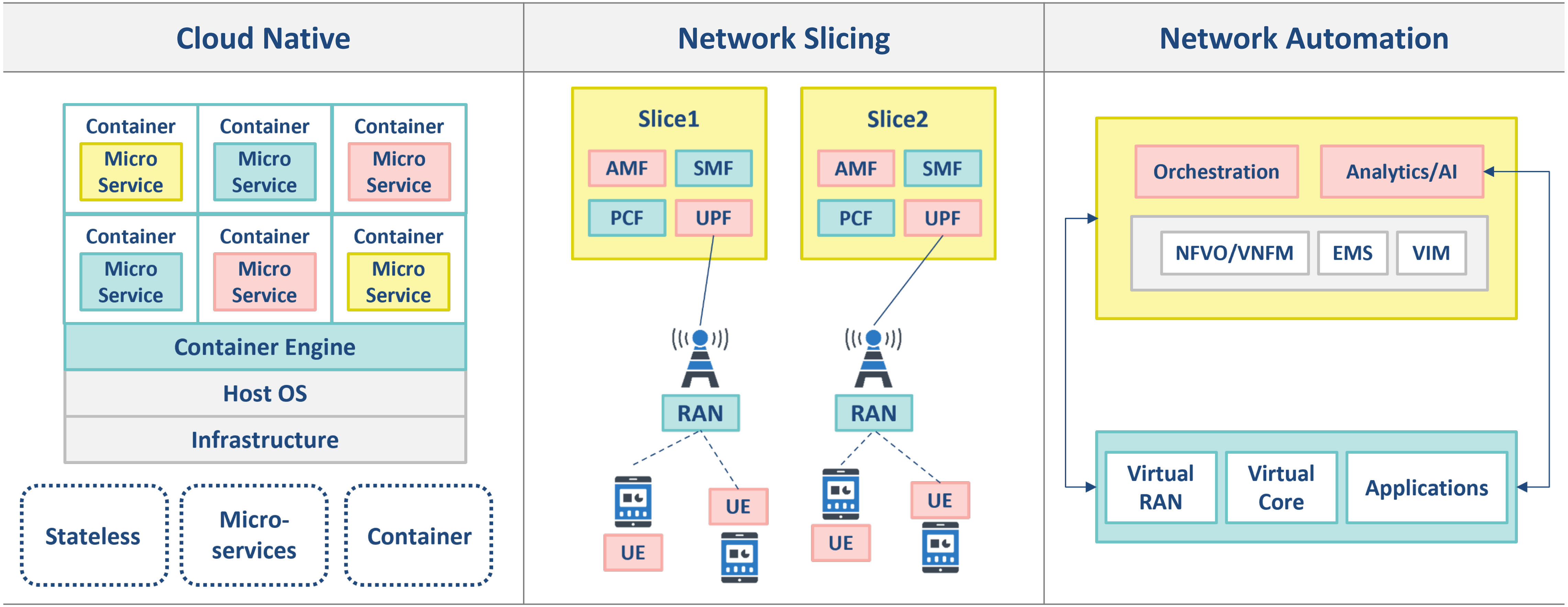


Introduction to 5G (3/3)

- PNF → VNF → CNF



Key Characteristic of 5GC



But, one more important thing in 6G core networks

Resilient 6G System (1/2)

- **What has to be provided for resilient 6G**

Strong security, from design to deployment and operations

Strong privacy protection

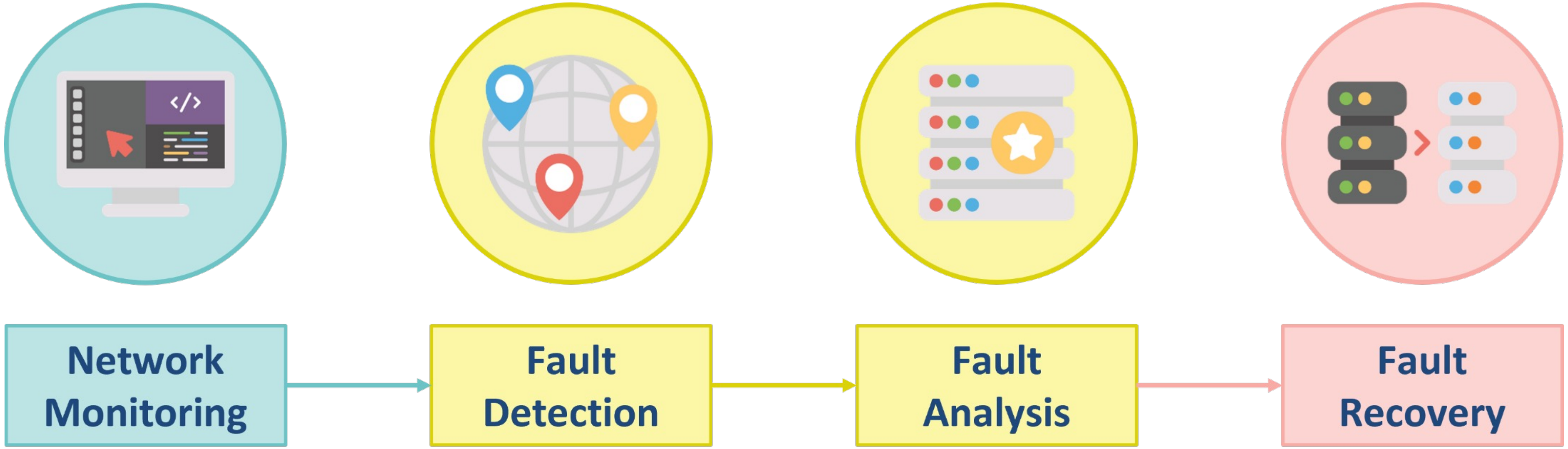
Reliability, low latency service

Availability, service availability/up time

Resilience, fast network recovery

Resilient 6G System (2/2)

- **Fault Management Process**



- Network traffic, mobility monitoring, CPU usage etc.
- Link packet delay, unexpected traffic loss, node failure etc.
- Inconsistent state, incorrect event processing etc.
- Rollback, repair etc.

FAILURE MANAGEMENT ON MOBILE CORE

ECHO / Neutrino / L25GC (+) / CellClone / CoreKube

Failure Management on Mobile Core

- Summary

Research	Keywords	Main goal	Open source	Reliability	Availability	Resilience
ECHO	fast failure recovery, state consistency, low latency	A distributed network architecture for the EPC on the public cloud	OpenEPC	O	O	X
Neutrino		Abstraction of reliable access to cellular services for ensuring low latency	OpenAirInterface, FlatBuffers	O	X	O
L25GC		NFV-based low-latency 5GC network solution	Free5GC	X	O	O
L25GC+		Newly shared-memory-based networking stack to support synchronous I/O between CP NFs.	Free5GC	O	O	X
CellClone		Fast and fault-tolerant control plane processing	OpenAirInterface, FlatBuffers	O	X	O
CoreKube		A novel message focused and cloud-native mobile core system design	Open5GS, NextEPC	X	O	O

ECHO: A Reliable Distributed Cellular Core Network for Hyper-scale Public Clouds (1/4)

- **ECHO Challenge & Solution**

Challenge 1: Remaining 99.999% uptime despite VM/container crashes and network partition

Solution 1: Fast malfunctioning replacement and scaling through NF replication

Challenge 2: 10x slower fault detection time in the public cloud compared to cellular core

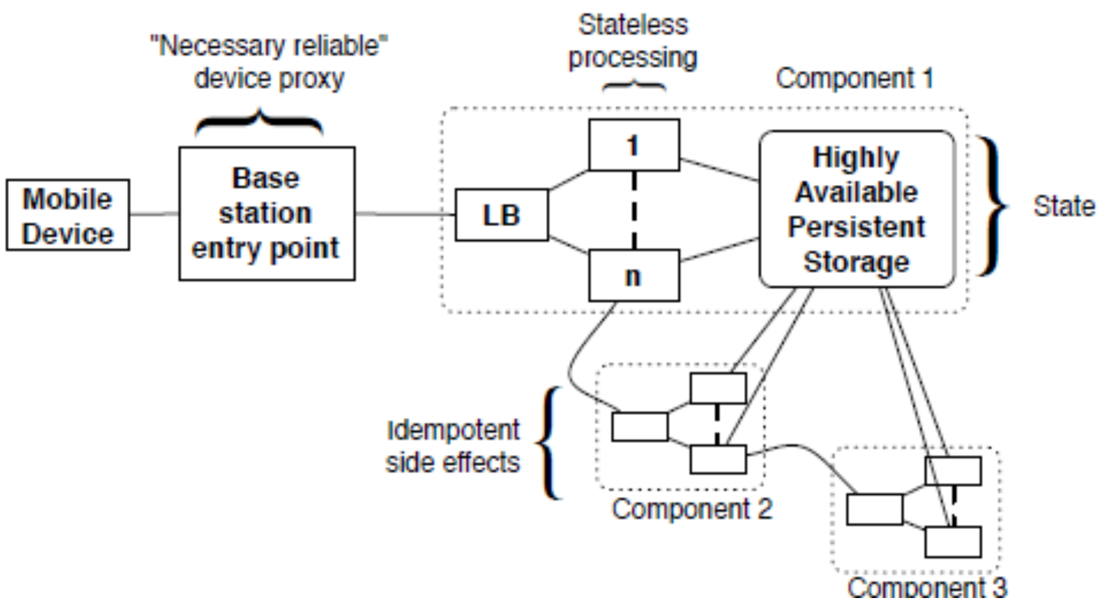
Solution 2: Operate serializable and in FIFO order

Challenge 3: Maintain consistency of mobile clients' session state

Solution 3: Guarantee components atomicity and in-order execution

ECHO: A Reliable Distributed Cellular Core Network for Hyper-scale Public Clouds (2/4)

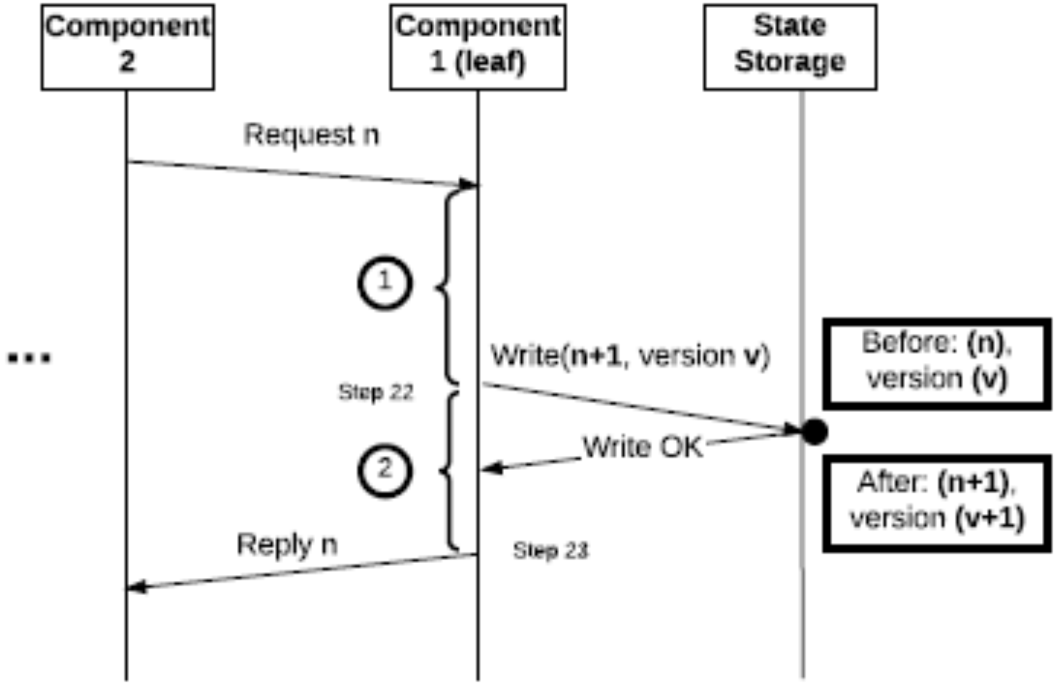
- **ECHO Overview**
  - **Replication of control-plane components** (e.g., MME, PGW) behind a load balancer
  - A **high availability persistent storage** that maintains state for all replicas
    - ➢ **(Solution 1)** possible to malfunctioning component quick replacement & scaling
  - A Necessarily reliable **BS entry point**



ECHO Overview

ECHO: A Reliable Distributed Cellular Core Network for Hyper-scale Public Clouds (3/4)

- **Solution 2: Operate serializable and in FIFO order**
  - Components non-blocking even in redundant and failure
  - Operating **linearizable** (i.e., serializable and in FIFO order) leaf component
  - Linearizable results: aborted, successfully, crashed before the update, crashed after the update



ECHO's leaf component is linearizable

ECHO: A Reliable Distributed Cellular Core Network for Hyper-scale Public Clouds (4/4)

- **Solution 3. Guarantee components atomicity and in-order execution**
  - Stateless instances that perform non-blocking algorithms in parallel
  - Concurrent retries of request at entry point → occurrence of inconsistency
  - **Component's atomicity:** atomic conditional writes provided by the persistent storage
  - **Component's in-order execution:** delete old request ID

11:01:57 mme_sm():1725> [2:NAS__Attach_request]
11:01:58 mme_sm():1725> [1:NAS__Attach_complete]
{UE attached}
{UE switches OFF, triggers a Detach procedure}
11:03:45 mme_sm():1725> [6:NAS__Detach_request] <delayed 60s>
{**MME thread #1** received Detach Request, and holds for 60s without a progress}
{UE switches ON, triggers an Attach procedure}
11:03:58 mme_sm():1725> [2:NAS__Attach_request]
11:03:59 mme_sm():1725> [1:NAS__Attach_complete] <suceeded>
{**MME thread #2** received and processed the Attach Request sucessfully}
11:04:45 mme_sm():1725> [6:NAS__Detach_accept] <suceeded>
{After 60s, **MME thread #1** processed the stale Detach Request, and suceeded}
{UE is detached from the network}
11:06:05 mme_sm():1925> [09:EMM__Service_request] <failed>
{UE has no service for 54 minutes}

Caused state inconsistency

Neutrino: A Low Latency and Consistent Cellular CP (1/3)

- **Neutrino Challenge & Solution**

Challenge 1: UE-Core state inconsistency

Solution 1: Consistent UE processing

Primary-backup state replication scheme

Challenge 2: Slow state updates

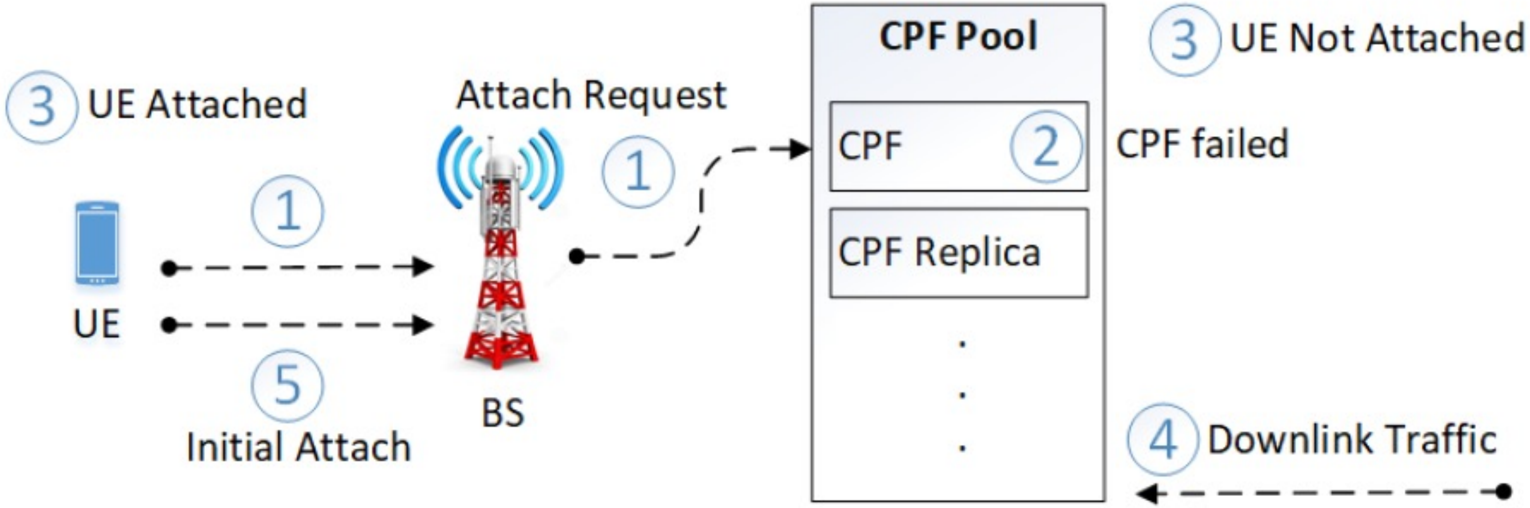
Solution 2: Fast serialization engine

Challenge 3: Frequent control handovers

Solution 3: Proactive geo-replication

Neutrino: A Low Latency and Consistent Cellular CP (2/3)

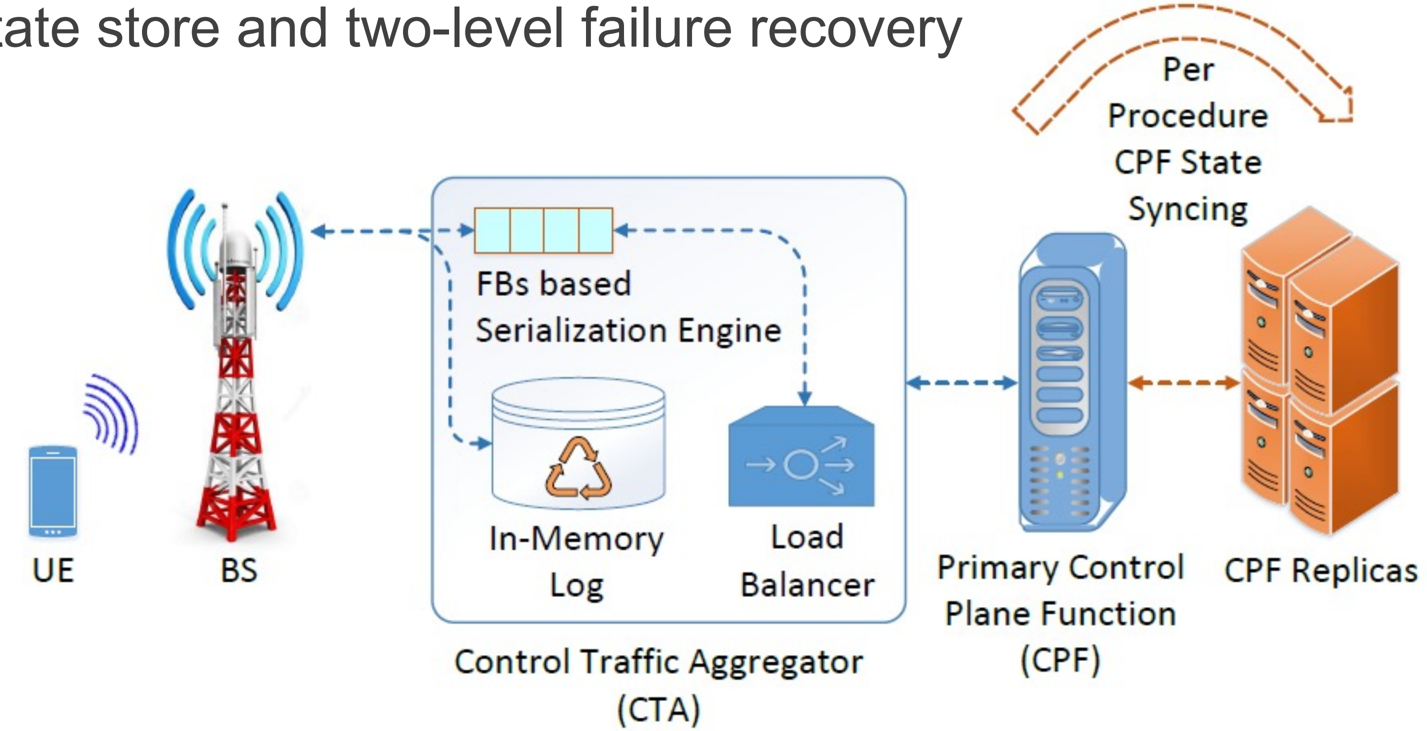
- **Challenge 1: UE-Core state inconsistency**
  - Replicating UE state across multiple CPFs to provide fault tolerance
  - Inability to provide state consistency and availability between replicas



An example scenario in inconsistent user state

Neutrino: A Low Latency and Consistent Cellular CP (3/3)

- **Solution 1: Consistent UE processing**
  - Replication with two-level of failure recovery
    - ➢ Replicated UE state store and two-level failure recovery



Neutrino's system architecture diagram

L25GC: A Low Latency 5G Core Network (1/3)

- **L25GC Challenge & Solution**

Challenge 1: 3GPP-recommended Service Based Interface (SBI)

Solution 1: NF consolidation through careful placement + shared memory communication

5G CP

Challenge 2: Complex Handover Procedure

Solution 2: Reduce latency through smart buffering at 5G core for handovers

Challenge 3: 5G UPF likely to have more PDRs in a single user session

Solution 3: Fast PDR lookup in UPF through improved data structures and packet classification

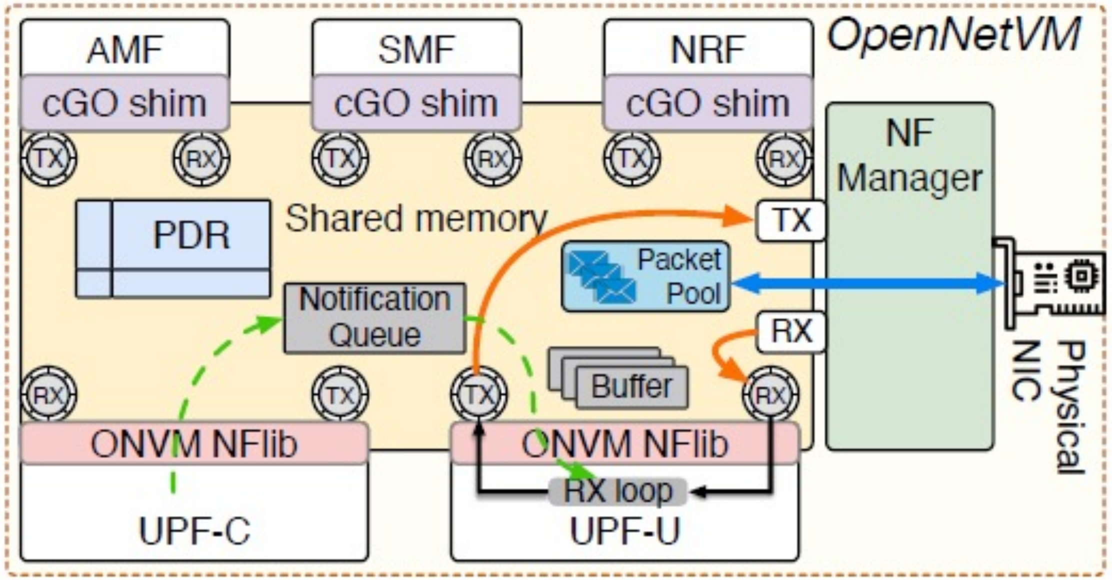
Challenge 4: Sub-optimal NF resiliency and recovery

Solution 4: Resiliency through improved state replication to backup NFs

NF resiliency

L25GC: A Low Latency 5G Core Network (2/3)

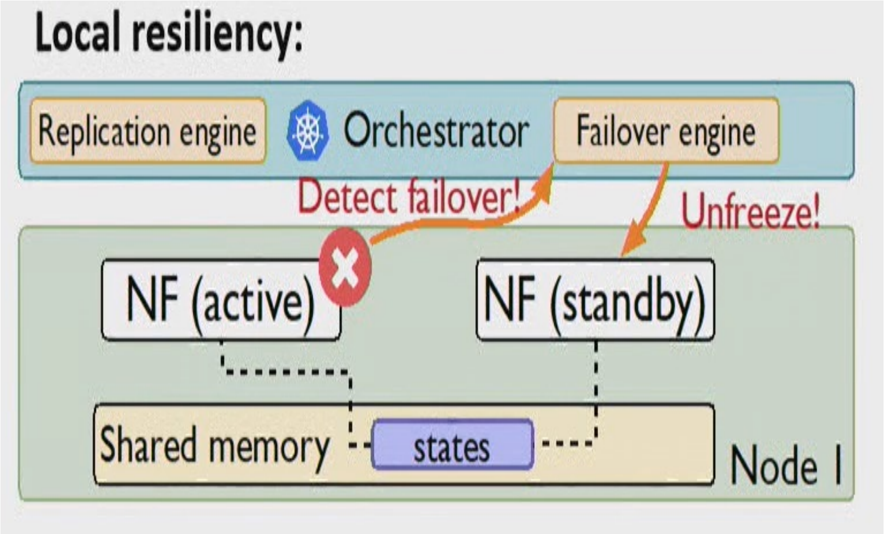
- **Solution 1: NF consolidation through careful placement & shared memory communication**
  - Shared memory for communication between NFs in a 5GC unit on same node
  - Flat memory access: no serialization cost
  - Information changed directly in user space: no kernel overheads or protocol processing
  - Zero-copy packet delivery between NFs: no data movement

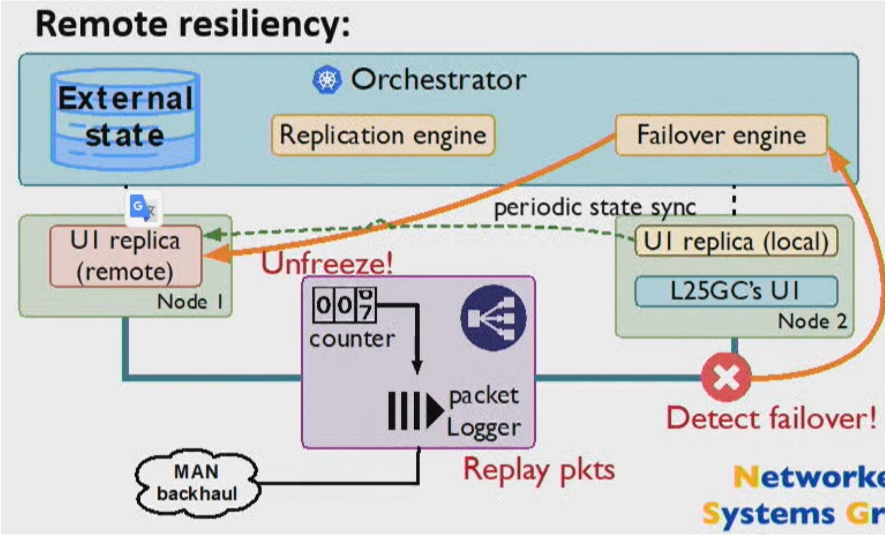


L25GC architecture

L25GC: A Low Latency 5G Core Network (3/3)

- **Solution 4: Resiliency through improved state replication to backup NFs**





✓ 2 levels of resiliency to support software failure (local resiliency) and node/link failure (remote resiliency)

✓ **Local resiliency:** state stored in shared memory

✓ **Remote resiliency:** use reinforce (uses external synchrony) to continue the speculative execution of user events

L25GC+: An Improved, 3GPP-compliant 5G Core for Low-latency Control Plane Operations (1/4)

- **L25GC+ Challenge & Solution**

Challenge 1: Compatibility issues with HTTP/REST-based SBI

Solution 1: A newly designed shared memory I/O interface

Synchronous I/O over shared memory

Challenge 2: Supporting a limited number of user sessions

Solution 2: Keep the state in a state map maintained in the shared memory networking stack

Concurrent connection

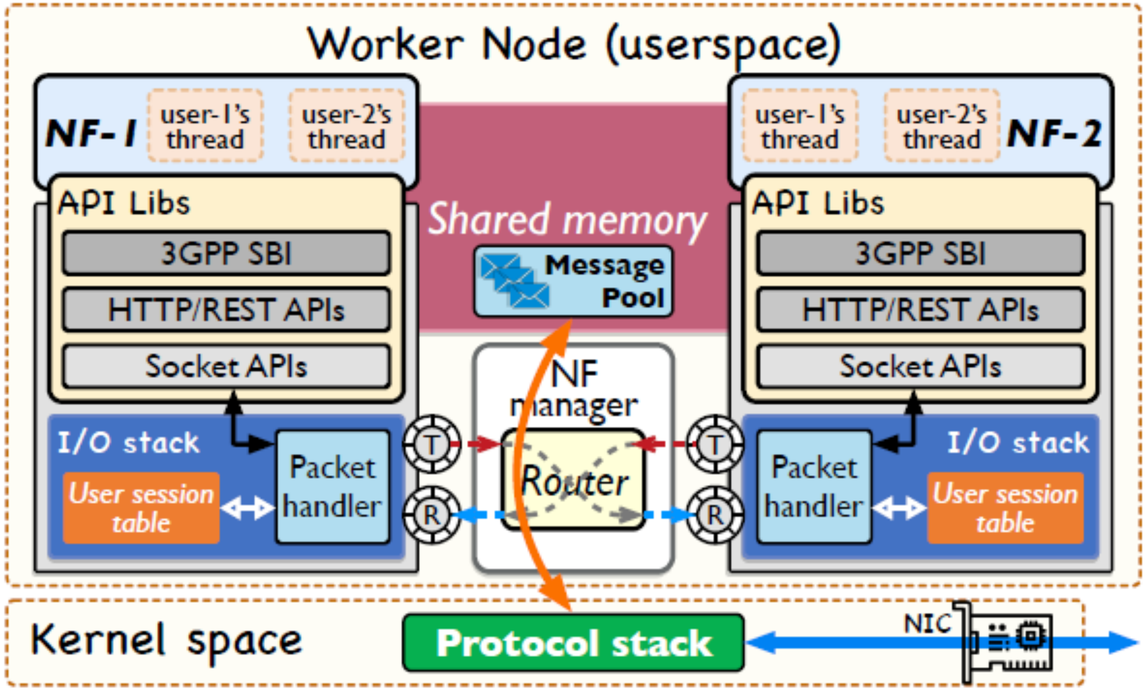
Challenge 3: Code refactoring time is required when porting source code

Solution 3: The cross-language support provided by the GO interface

L25GC+: An Improved, 3GPP-compliant 5G Core for Low-latency Control Plane Operations (2/4)

- **Solution 1: A newly designed shared memory I/O interface (Unified Sync/Async communication)**

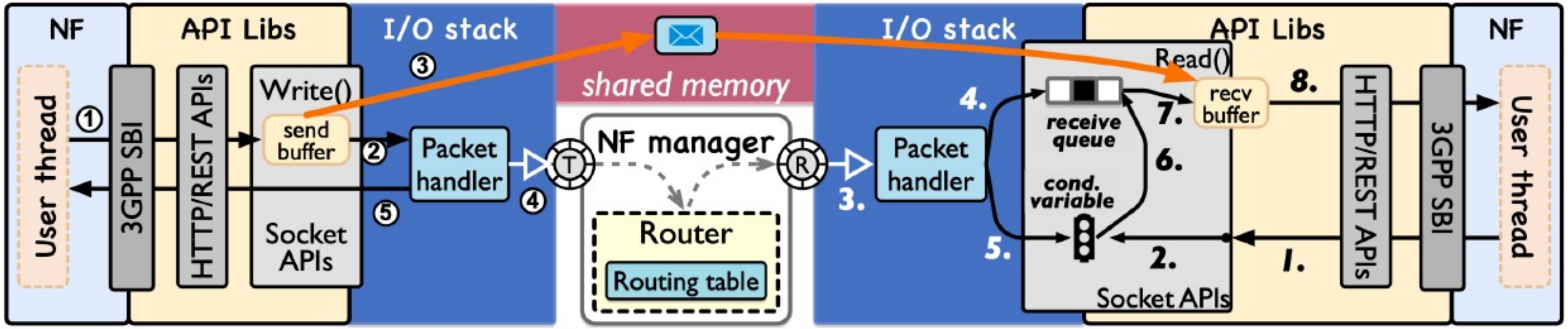
  - Shared memory I/O stack: Shared memory processing w/ lock-free rings
  - API Libs: Synchronous I/O support
  - Concurrent connection management: Using “User session table"
  - Cross-language support: CGo interface in Golang



L25GC+ architecture

L25GC+: An Improved, 3GPP-compliant 5G Core for Low-latency Control Plane Operations (3/4)

- **Solution 1: A newly designed shared memory I/O interface**



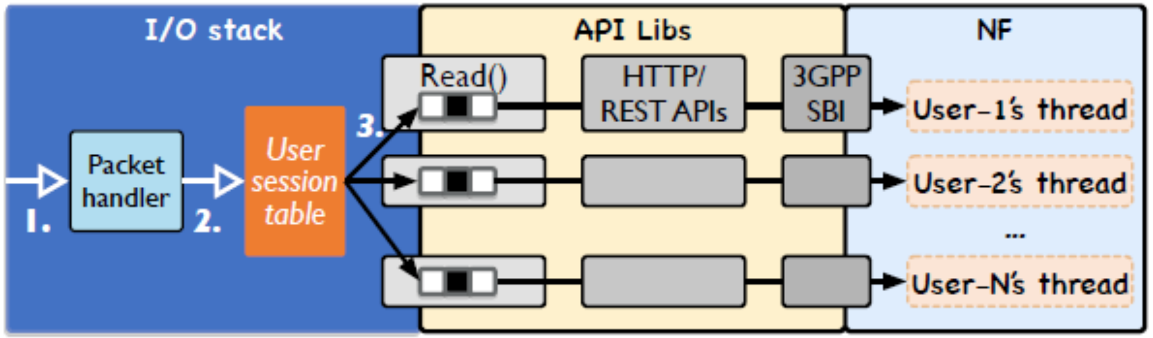
Synchronous I/O primitives from L25GC+'s socket APIs

- Adding blocking primitives to the asynchronous shared memory network stack
  - The caller of Read() is blocked until it receives the request from the I/O stack
  - The caller of Write() is blocked until the data in send buffer is moved to the shm buffer

L25GC+: An Improved, 3GPP-compliant 5G Core for Low-latency Control Plane Operations (4/4)

- **Solution 2: Keep the state in a state map maintained in the shared memory networking stack**

  - Turning “stateless” to “stateful”
  - User session table in I/O stack
  - Dispatch requests to different user sessions via IP 4-tuples lookup



Concurrent user session support in L25GC+.

CellClone: Enabling Emerging Edge Applications Through a 5G CP Intervention (1/3)

- **CellClone Challenge & Solution**

Challenge 1: High Delays with Synchronous Replication

Solution 1: Fast Consistency Protocol

Custom quorum-based consistency protocol

Challenge 2: Inconsistency Due to Non-determinism

Solution 2: Individualized Approach to Non-Determinism

Challenge 3: Failure Detection can be a Potential Bottleneck

Challenge 4: Adverse Impact of Stragglers

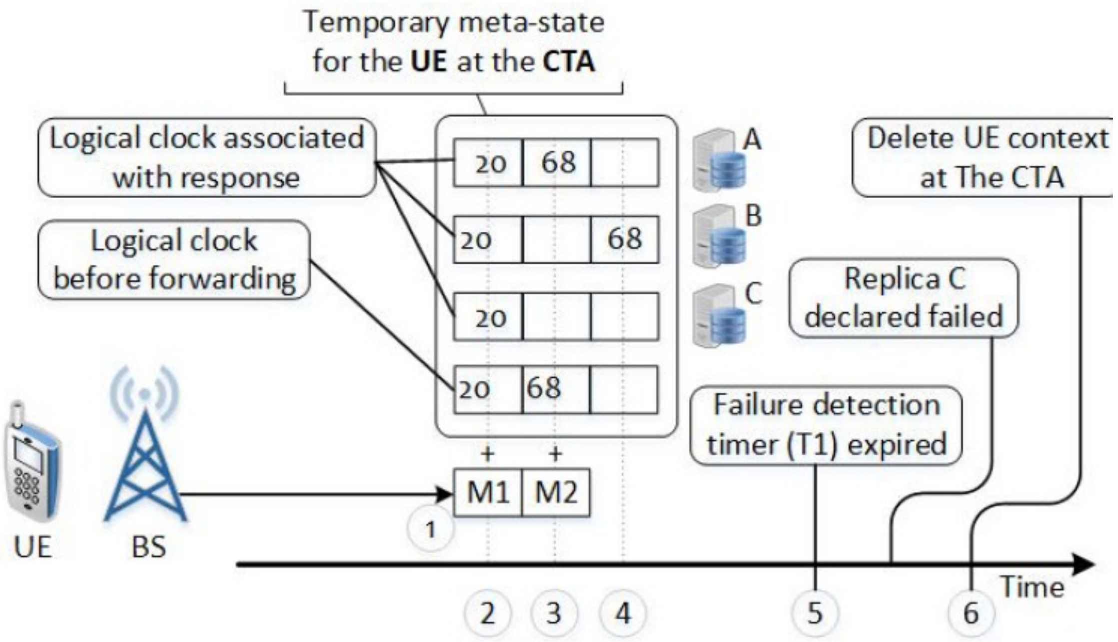
Solution 3: Active Replication

Quorum selection & duplicate filtration at the CTA

CellClone: Enabling Emerging Edge Applications Through a 5G CP Intervention (2/3)

- **Solution 1: Fast Consistency Protocol**

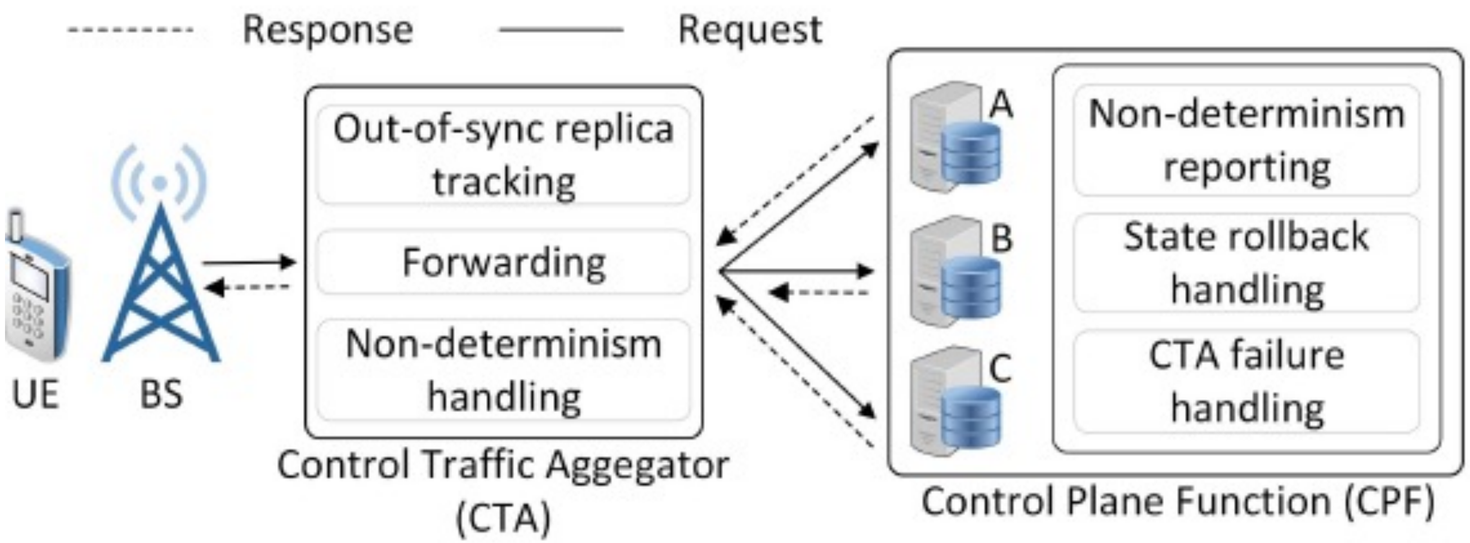
  - **Out-of-sync CPF tracking**
    - P1. create temporary meta-state for UE in CTA
    - P2. respond to CTA with logical clock of M1(20) in quorum
    - P3. for M2, the response from replica A in CTA
    - P4. response to CTA from replica B before timer T1 expires
    - P5. expires failure detection timer T1
    - P6. delete temporary meta-state for UE from CTA



Temporary meta-state for the user at the CTA

CellClone: Enabling Emerging Edge Applications Through a 5G CP Intervention (3/3)

- **Solution 3: Active Replication**
  - Quorum selection
  - Duplicate filtration at the CTA
    - ➢ Using logical clock timestamp



CellClone's system architecture.

CoreKube: An Efficient, Autoscaling and Resilient Mobile Core System (1/3)

- **CoreKube Challenge & Solution**

Challenge 1: The heavily entangled nature of processing and state in standard core functions/events.

Solution 1: Decoupling all the core network states into a separate database

Truly Stateless Workers

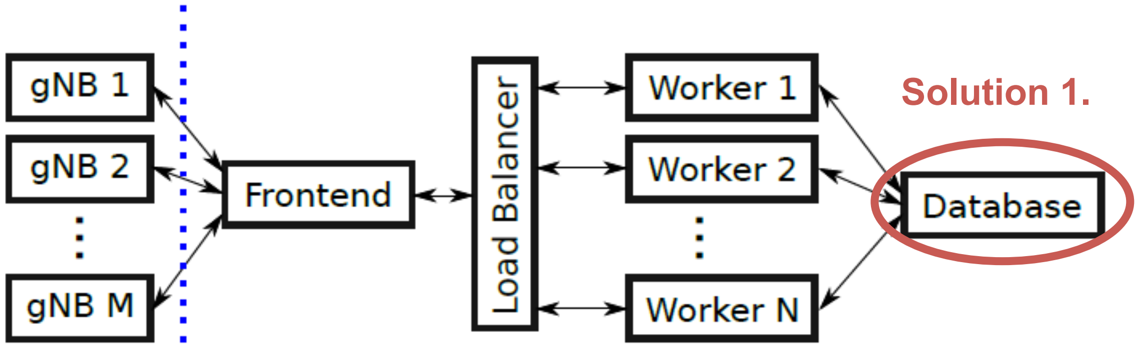
Challenge 2: decouple the RAN-core interface from control plane processing in the core.

Solution 2: A frontend at the RAN-core interface in CoreKube that encapsulates/ decapsulates messages from/to the RAN

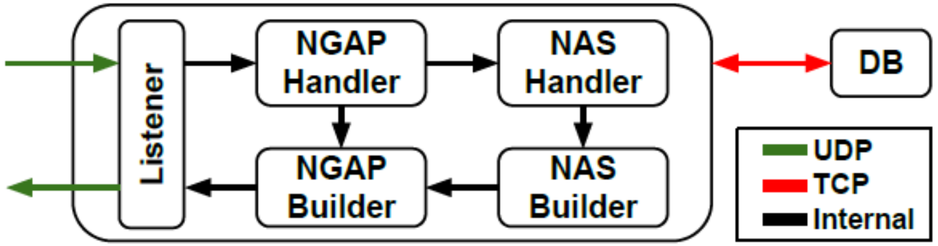
Decoupling RAN-Core interface

CoreKube: An Efficient, Autoscaling and Resilient Mobile Core System (2/3)

- **Solution 1: Decoupling core network state from control plane processing**
  - Three main components: a frontend, a pool of workers, and a database (DB)
  - CoreKube components are containerized → autoscaling and self-healing capabilities
  - Development of standard-compliant
  - Five worker components: Listener, NGAP input/output Handler, NAS input/output Handler



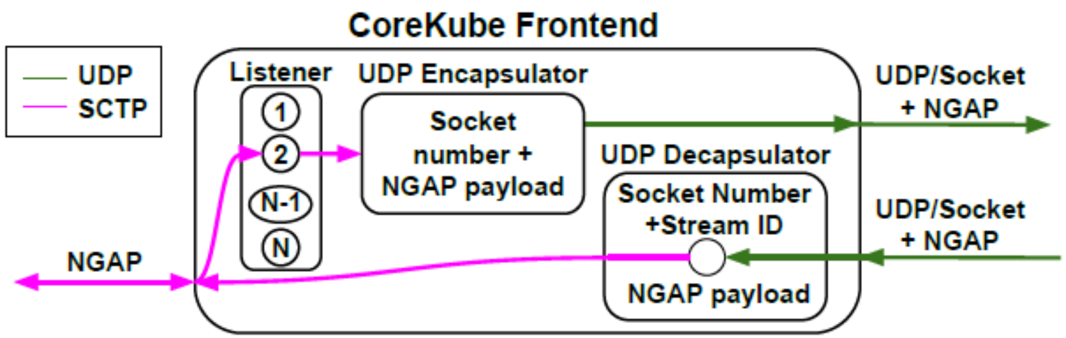
CoreKube architecture



CoreKube Worker Architecture

CoreKube: An Efficient, Autoscaling and Resilient Mobile Core System (3/3)

- **Solution 2: Decoupling control plane processing in the core from RAN interface**
  - Messages are exchanged with RAN through the SCTP protocol according to the standard.
  - Internally communicates with workers using the UDP protocol through the load balancer.



CoreKube Frontend Architecture