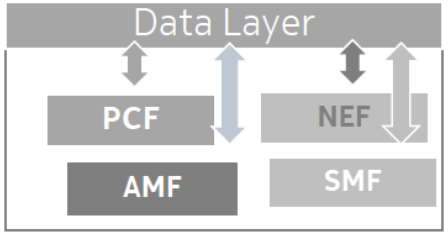
S2-166770

Review of Data Layer Framework
(Discussion in response to S2-166590)

Cisco Systems, Inc., Ericsson, ZTE

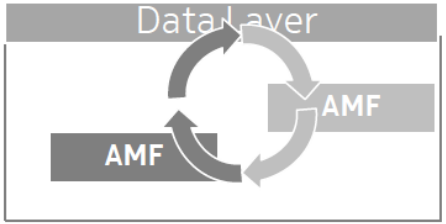
Decoupling “compute” and “storage” – What is proposed?
Data Layer becoming an integral part of NextGen

#1 - Multivendor opaque data store



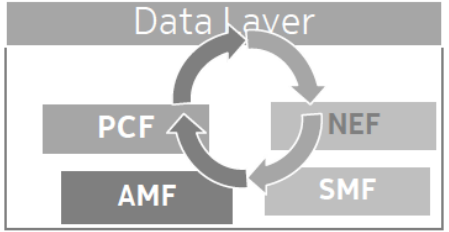
- Enables NF resiliency, stateless NF but not data sharing for inter vendor AMF mobility
- DL I/f standardization
- Opaque data - No Data Model standardization required

#2 – Structured data store for sharing across same NF (hand-over/mobility, resiliency)



- Enables NF resiliency, ease context retrieval during NF mobility, enables stateless NF
- DL I/f standardization
- Standardization of UE MM context data model like in S10 today (TS 29.274 – Context response message)
- Stage 2 call flows should incorporate DL for context storage/retrieval

#3- Multi-NF selective data sharing (AMF, SMF, NEF, PCF)



- Enables capability/data exposure
- DL I/f standardization
- Data model standardization
  - selective data shared between different NFs. (TS 29.128 – Monitoring report message).
- Stage 2 call flows should incorporate DL for context storage/notification

All use cases proposed for standardization

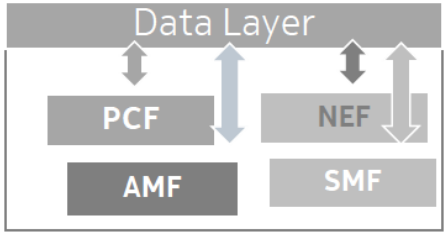
© Nokia 2015
NOKIA

Could be potentially done to reduce number of backend DBs used for opaque data (if used by an NF).

No obvious benefits while introducing additional signaling and delaying many system procedures (see following slides)

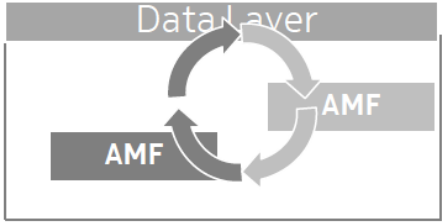
Already supported based on agreed NF capability exposure. No need for an additional solution (see S2-166721)

#1 - Multivend que data store



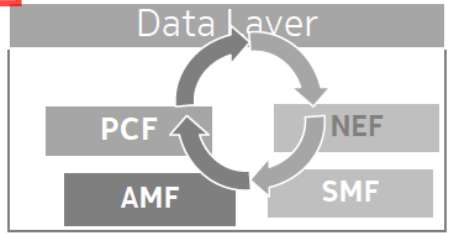
- Enables NF resiliency, stateless NF but not data sharing for inter vendor AMF mobility
- DL I/f standardization
- Opaque data - No Data Model standardization required

#2 – Structur data store for sharing across same NF
(hand-over/mobility, resiliency)



- Enables NF resiliency, ease context retrieval during NF mobility, enables stateless NF
- DL I/f standardization
- Standardization of UE MM context data model like in S10 today (TS 29.274 – Context response message)
- Stage 2 call flows should incorporate DL for context storage/retrieval

#3- Multi-NF selecti sharing
(AMF, SMF, N , PCF)



- Enables capability/data exposure
- DL I/f standardization
- Data model standardization
  - selective data shared between different NFs. (TS 29.128 – Monitoring report message).
- Stage 2 call flows should incorporate DL for context storage/notification

Whether single interface/single DB is sufficient and what should be re-used from other SDOs needs to be studied

ses proposed for standardization

Cisco, ZTE,

NOKIA

Mapping between use cases and NextGen questions in S2-66588

- **Use case #1 maps to question 7/2a –**
- Should 3GPP recommend or specify an optional interface from network functions to a storage service that network functions may use to store selected state as opaque (= vendor specific) data? (☑)
- **Use case #3 maps to question 7/2b –**
- Should Rel15 support Information exchange (exposure) of selected information like ULI (specific info to be decided on a case by case basis) between different NF via a "data layer" in a standardized manner? ☒
- **Use case #2 maps to question 7/2c –**
- Should Rel-15 support Network function with standardized interface to Information repository for UE context? ☒

Mapping between use cases a

Exposure is already supported based on agreed capability exposure (no need for an additional solution).

- **Use case #1 maps to question 7/2a –**
- Should 3GPP recommend or specify an o terface from network functions to a storage service that network functions n e to store selected state as opaque (= vendor specific) data?
- **Use case #3 maps to question 7/2b –**
- Should Rel15 support Information exchange (exposure) of selected information like ULI (specific info to be decided on a case by case basis) between different NF via a "data layer" in a standardized manner?
- **Use case #2 maps to question 7/2c –**

See also question #7/4: Should selected information like ULI be exposed in a standardized manner to other network functions as a capability (e.g. ULI reporting), in line with the existing interim agreement (KI#7) "the capability (s) of NFs are exposed as a service to other NF, wherever applicable"?

Comparison - Use case #1 versus Use case #2

#1 Opaque data store in the DL:

- Enables stateless NF, NF resiliency
- **Inter NF Mobility** between stateless NFs (i.e. TAU in a new AMF) will require old (stateless) AMF to be instantiated solely for the purpose of reading the DL and sending the UE MM context via messaging to the new AMF instance.
- **Standards impact:** NF <-> DL interface specification; No Data model standardization required

#2 Structured data for sharing across same NFs for NF resiliency, mobility:

- Enables stateless NF, inter vendor NF resiliency
- **Inter NF mobility** (e.g. AMF mobility due to TAU in a new AMF) does not involve old AMF to be instantiated rather new AMF can retrieve UE context directly from the DL.
- **Standards impact:** Incorporating DL as integral part of the architecture, Stage 2 flows, NF <-> DL interface specification, Data model standardization.

© Nokia 2015
NOKIA

Triggers for storing context in the Data Layer

Generally NF is expected to have the knowledge when it is appropriate to store UE context information in the data layer (i.e. when UP function is processing traffic and CP function has not much activity for the given UE and/or UE remains idle for a long duration). Following are some illustrations:

- Completion of registration (e.g. attach/TAU) procedure.
- When the UE transitions from active to idle mode (i.e. release of NG2/1 signalling connection).
- Completion of HO procedure

The main requirement:

- NF should ensure that the UE context stored in the data layer is up to date (stable state is expected to stored in the Data Layer)

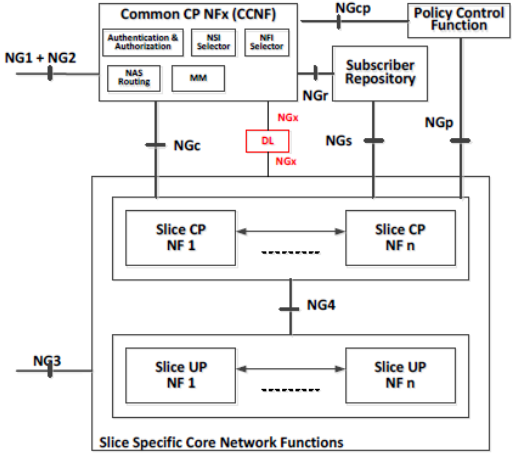
This vague rule only works for use case #1.
For other use cases e.g. data transfer via DL or selection of NF based on information from DL, data needs to be synchronized before direct messaging (i.e. clear rule is required).

NOKIA

Flexibility in storage

Intrinsic capabilities of the Data Layer shared across network functions:

- Data may be stored in a distributed manner or in a centralized manner.
- The data can be replicated across multiple locations transparently for the application.
- Data layer can either be shared across network slices or support isolation requirement for isolated network slices (based on operator policies)



© Nokia 2015

Latency and system performance

In a virtualized environment, there are 2 ways to store data:

- Cache within Network functions.
- Independent storage layer.

- Having storage capabilities in each and every control plane functions is disadvantageous considering cost and it is not flexible/elastic. Furthermore, it takes away the storage space in a transaction processing compute machine (= network function).
- Especially in a virtualized environment, an independent storage (data layer) improves the resiliency of the network by decoupling the “compute” from “storage”. The latency for data retrieval is just a consequence of virtualization while providing the benefit of independent scalability.
- Even if there is no 3GPP defined standard interface for this purpose, this I/O still exists. On the other hand, benefit of standardized interface allows multi-vendor capability in operator’s environment.
- NF stores the context in the DL when there is no critical transaction ongoing. In the middle of a transaction, the network function is expected to store the context within NF cache. **Thus there should be no impact to system performance due to context storage in the Data layer**

© Nokia 2015
NOKIA

Latenc

This is incorrect. In virtualized deployments it does not matter where state is stored since both the NFs and the state repository run on the same NFVI (typically X86 blades). Thus, cost of memory is always the same regardless where the information is stored.
Instead, separating state into a separate DB implies additional cost for I/O (to convey the state from/to the state DB) and CPU (to remove and re-instate the state for a given session).

In a virtu

- Cache within Network functions.
- Independent storage layer.

- Having storage capabilities in each and every control plane functions is disadvantageous considering cost and it is not flexible/elastic. Furthermore, it takes away the storage space in a transaction processing compute machine (= network function).
- Especially in a virtualized environment, an independent storage (data layer) improves the resiliency of the network by decoupling the “compute” from “storage”. The latency for data retrieval is just a consequence of virtualization while providing the benefit of independent scalability.
- Even if there is no 3GPP defined standard interface for this purpose, this I/O still exists. On the other hand, benefit of standardized interface allows multi-vendor capability in operator’s environment.
- NF stores the context in the DL when there is no critical transaction ongoing. In the middle of a transaction, the network function is expected to store the context within NF cache. **Thus there should be no impact to system performance due to context storage in the Data layer**

NOKIA

La

Various solutions exist to support resiliency (separating state from processing is just one tool that may or may not be used). Implementations can separate state from compute today if they wish so (no need for a standardized state DB).

In a

- Cache within Network functions.
- Independent storage layer.

- Having storage capabilities in each and every control plane fun sadvantageous considering cost and it is not flexible/elastic. Furthermore, it takes away the sto pace in a transaction processing compute machine (= network function).
- Especially in a virtualized environment, an independent storage (data layer) improves the resiliency of the network by decoupling the "compute" from "storage". The latency for data retrieval is just a consequence of virtualization while providing the benefit of independent scalability.
- Even if there is no 3GPP defined standard interface for this purpose, this I/O still exists. On the other hand, benefit of standardized interface allows multi-vendor capability in operator's environment.
- NF stores the context in the DL when there is no critica ongoing. In the middle of a transaction, the network function is expected to store th **Thus there**

Need for multi-vendor fail-over is not obvious: Due to inevitable proprietary state, inter-vendor fail-over will by definition not work as well as intra-vendor fail-over.
Also, even in multi-vendor deployments, SPs would deploy multiple instances of the same function from the same vendor anyway, so that intra-vendor fail-over can easily be used for resiliency.

This is incorrect for use cases 2 and 3:

- **<u>When using Data Layer to pass information, every procedure is delayed</u>** (data transfer via Data Layer needs to be completed and acknowledged and different Data Layer replicas need to be synchronized before direct signaling can be sent)
  - [see slides 21/22/23]
- When using DL to support stateless NFs, **<u>transactions are delayed due to forwarding of requests from "wrongly" selected NFs (AMF in MO case and SMF in MT case) to correct NF</u>**
  - [see slide 32]
- **<u>Using Data Layer to store serving NF IDs (e.g. AMF IF) delays system procedures</u>** (shifts race conditions into the Data Layer; implies need for lengthy three phase commits across DB replicas)

consequence o[illegible] ding the benefit of independent scalability.

- Even if there is no 3GPP [illegible] dard interface for this purpose, this I/O still exists. On the other hand, benefit of standardized in[illegible]ace allows multi-vendor capability in operator's environment.
- NF stores the context in the DL when there is no critical transaction ongoing. In the middle of a transaction, the network function is expected to store the context within NF cache. **Thus there should be no impact to system performance due to context storage in the Data layer**

NOKIA

Main benefits of supporting Data Layer framework in the standards

- Allows “compute” (Network function) to be decoupled from the “storage” (data layer) functionality and this helps any network function to perform the processing for a certain UE thus improving the resiliency of a Network Function. Data Layer can offer resiliency independent of Network Function.
- Network Function can save its resources for massive IoT devices.
- Support for data analytics (KPIs) and network capability exposure (within and outside MNO).
- Provides a unified way how UE MM context and session context is stored across network functions.
- Data Layer exposes one interface for any network function that needs to leverage the services of Data Layer. “Data as a service” eases the introduction of new network functions as the same interface can be reused (i.e. a new interface need not be introduced).
- In standards, it embeds and expands the UDC architecture as part of the 5G architecture.

© Nokia 2015
NOKIA

Main b

Unclear what is being saved in this case: In virtualized deployments it does not matter where state is stored since both the NFs and the state repository run on the same NFVI. Thus, cost for storing IoT device state is always the same regardless where the information is stored.

- Allow
  and this helps any network function to perform the processin thus improving the resiliency of a Network Function. Data Layer can offer resili independent of Network Function.
- Network Function can save its resources for massive IoT devices.
- Support for data analytics (KPIs) a network capability exposure (within and outside MNO).
- Provides a unified way how U ntext and session context is stored across network functions.
- Data Layer exposes one i any network function that needs to leverage the services of Data Layer. "Data as the introduction of new network functions as the same interface can be r rface need not be introduced).
-

Offloading state from NFs to a state DB actually increases cost (I/O cost for storing and reading the context) from the state DB.
If at all, then a different database technology may be useful for state from IoT devices (e.g. disk-based memory). Standardizing a single Data Layer would actually prevent using different database technologies for different NFs.

Cisco, ZTE, Ericsson

NOKIA

Main benefits of supporting Data Layer framework in the standards

- Allows “compute” (Network function) to be decoupled from the “storage” (data layer) functionality and this helps any network function to perform the processing for a certain UE thus improving the resiliency of a Network Function. Data Layer can offer resiliency independent of Network Function.
- Network Function can save its resources for massive IoT devices.
- Support for data analytics (KPIs) and network capability exposure (within and outside MNO).
- Provides a unified way how UE MM context and session context is stored across network functions.
- Data Layer exposes op network function that needs to leverage the services of Dat int
- In

No Data Layer needed for this: This is already supported by the agreement to expose NF capabilities as a service.

An NF implementation (e.g. NEF) could decide to also locally store information.

Conclusion: Data Layer (as a basis for analytics) is an implementation option on top of the already agred functionality. No standardized Data Layer needed to support this.

Cisco, ZTE, Ericsson

NOKIA

Value proposition for operators

- Data layer framew rdization opens up new opportunities:
  - Newer R odels:
    - N+m cy, rather than 1:1 Mated Pair (really 1+1 : 1+1) Geo-Redundancy
    - Ena twork Function resiliency
    - Abi liability with less hardware footprint

t)

Repetition of previous slides; please see related comments.

Decoupling **"compute"** from **"storage" with a standardized interface opens up new opportunities for operators.**
Cloud native technology, improves NF resiliency and enables data sharing

Cisco, ZTE, Ericsson

NOKIA

Why should DL be standardized and Why NOW (in phase 1)?

- Why?
  - Without standardized data layer, every VNF vendor supplying Network Function would support its own data layer. Standardized DL eases [illegible] port for Data analytics, Network Capability exposure.
  - Standardization of DL en[illegible] to deploy it across Network Functions (same type or different type) provided by di[illegible] out major integration effort.
  - [illegible]
  - [illegible]
- Why [illegible]
  - [illegible] needs to be used as the primary storage of context by a NF.
- Has this been done before?
  - This is similar to UDC architecture standardized with Ud interface in CT4.

Repetition of earlier slides. See related comments on earlier slides and S2-166721.

NOKIA

Open points resolution

Is there a synchronization issue due to DL framework?

- Is there is a negative consequence due to communication between 2 NFs that involves the DL (write followed by read operation):
- a) NF1 stores data in DL
- b) NF1 sends a message to NF2 to trigger a certain request.
- c) NF2 reads data from DL in order to perform some procedure
- How can it be ensured that NF2 has data available from DL in order to process the message from NF1?

Proposed Resolution

Depending on the type of procedure, Network Function 1 can store the data and **request for acknowledgement** prior to sending a message to NF2. This ensures that NF2 has the data available from DL for processing the message from NF1.

 © Nokia 2016

NOKIA

Is there a synchronization issue due to DL framework?

- Is there is a negative consequence due to communication between 2 NFs that involves the DL (write followed by read operation):
- a) NF1 stores data in DL
- b) NF1 sends a message to NF2 to trigger a certain request.
- c) NF2 reads data from DL in order to perform some procedure
- How can it be ensured that NF2 has data available from DL in order to process the message from NF1?

Potentially a solution, but…

Proposed Resolution

Depending on the type of procedure, Network Function 1 can store the data and **request for acknowledgement** prior to sending a message to NF2. This ensures that NF2 has the data available from DL for processing the message from NF1.

…NF1 actually has to wait for the ACK from NF2 before NF1 can send the request to NF2.
(which comes at quite a price, see next slide)

NOKIA

Passing data via Data layer increases signaling overhead drastically and delays every procedure unnecessarily

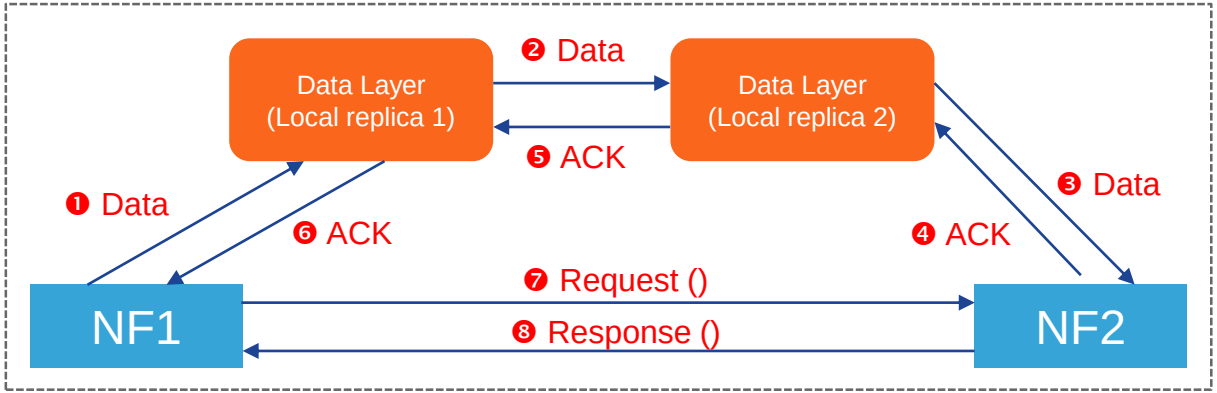


Figure 1: With Data layer

Signaling overhead for every request

- 6 additional messages (4 if both NFs use same DL replica)

Every procedure gets delayed

- Request can only be sent once ACK has been received at NF1

Figure 2: Without Data layer

No additional messages needed

Request can be sent immediately

Overhead and delay increases even more if Data Layer is used for return data as well.

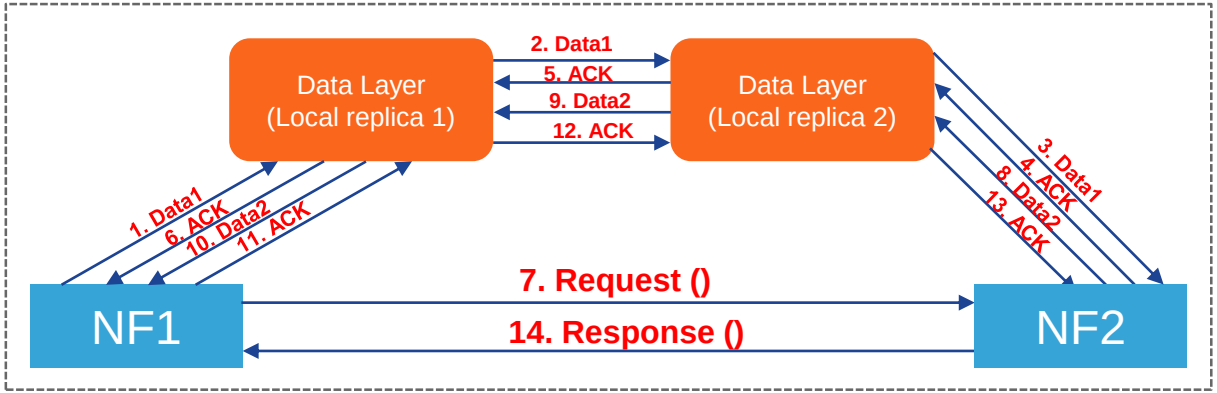


Figure 1: With Data layer

Signaling overhead for every procedure

- 12 additional messages (8 if both NFs use same DL replica)

Every procedure gets delayed

- Request can only be sent once ACK has been received at NF1
- Respose can only be send once ACK from NF1 reaches NF2

Figure 2: Without Data layer

No additional messages needed

Request and response can be sent immediately

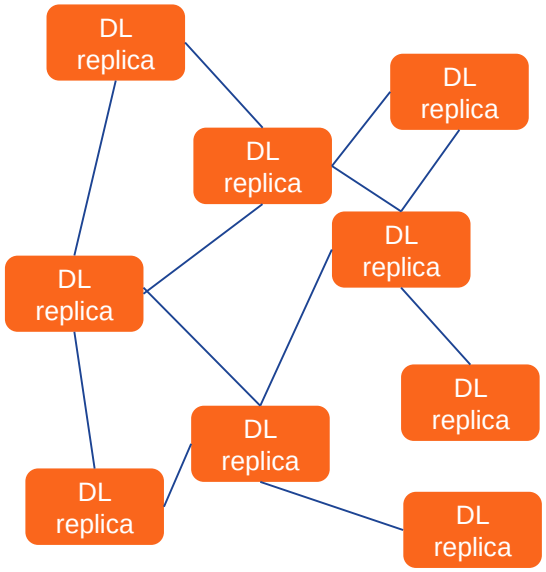
Consistency model required by Data Layer not recommended for cloud applications

Background

- Strong consistency = all changes are atomic across replicas

Analysis

- As shown before, data needs to reach target function before target function gets called via direct messaging
- Data and ACK exchange between DL replica 1 and 2 shown on previous two slides is actually a simplification
- In reality, strong consistency across many more replicas needs to be ensured (which requires more complex algorithms)
- Strong consistency comes at the price of lower system performance and is recommended to be avoided in distributed cloud systems
  - See also *Microsoft Developer Network: Data Consistency Primer. https://msdn.microsoft.com/en-gb/library/dn589800.aspx*



Conclusion: Passing information via Data Layer instead of as part of direct signaling increases signaling need drastically and delays all procedures that rely on Data Layer.

What are the contexts stored in the Data Layer?

- **Mainly UE contexts.**
- Exact set of parameters that is stored in data layer should be determined based on analysis of all relevant scenarios/key issues, outcome of the NextGen TR, work during the normative phase.

- **For Use case #1:**
- NF can store opaque (= vendor specific) data in the data layer

- **For Use case #2 (Structured data for sharing across same NFs for NF resiliency, mobility):**
- AMF can store standardized UE MM context in the DL. MM Contexts include information such as the following –
- **IMSI, MSISDN (optional), MM state, TAI, UE radio capability, UE network capability, DRX parameters, Mobility Restriction parameters, Access restriction parameters, RFSP index, NSSAI.**
- In addition, AMF can also store opaque data in the DL

- **For Use case #3 (Multi-NF selective data sharing across different NFs):**
- AMF can store UE contexts (**TAI, Cell ID, UE reachability event notification)** to share with other network functions
- SMF can store UE contexts (**Standardized PDU session contexts, User plane function topology information**)

 © Nokia 2016

NOKIA

What are the contexts stored in the Data Layer?

Inefficient: Those need to be separate contexts (i.e. information is stored multiple times) since for #2 the solution allows "lazy write" (NF can cache, write information later) while for #3 information needs to be written immediately before other NF is triggered

should be determined based on analysis of all relevant
ork during the normative phase.

ata layer

- **For Us #2 (Structured data g across same NFs for NF resiliency, mobility):**
- AMF c store standardized UE MM in the DL. MM Contexts include information such as the following –
  IMSI, MSISDN (optional), MM state, TA radio capability, UE network capability, DRX parameters, Mobility Restriction parameters, Access restrict parameters, RFSP index, NSSAI.
- In addition, AMF can also store opaque data in the DL

- **For Use case #3 (Multi-NF selective data sharing across different NFs):**
- AMF can store UE contexts (**TAI, Cell ID, UE reachability event notification)** to share with other network functions
- SMF can store UE contexts (**Standardized PDU session contexts, User plane function topology information**)

Cisco, ZTE, Ericsson

NOKIA

Stateless Network Function Selection – MO transaction

Applicable only for use case #2

- When the initial message is received from UE/RAN, Network function is selected based on load balancing algorithm.
- NG2AP association is created between RAN and CCF/AMF for a given UE when the UE moves to CN connected state and the NG2 association remains active as long as UE is in connected mode.
- So, for subsequent transactions, when the UE is in CN connected state, RAN selects the same AMF pool and eventually the entry point (e.g. load balancer) in the core network may select an AMF instance based on NG2AP association.
- Ensures selection of same AMF for subsequent transaction.

- RAN selects the CCF/AMF pool and eventually AMF instance is selected based on load balancing algorithm and there is no previously established NG2AP association. For subsequent transactions RAN routes the NAS signalling from the UE to a CCF/AMF in the serving network slice based on the NSSAI provided by the UE and according the same logic, e.g. based on load balancing algorithms. .
- Potential for race conditions (e.g. consecutive messages sent to different AMFs) exists

 © Nokia 2016

NOKIA

Stateless Network Function Selection – MO transaction

Applicable only for use case #2

- When the initial message is received from UE/RAN, Network function is selected based on
- association remains active a UE is in connected mode.
- So, for subsequent transactions, en the UE is in CN connected state, RAN selects the same AMF pool and eventually the entry point (e.g. load balancer) in the core network may select an AMF instance based on NG2AP association.
- Ensures selection of same AMF for subsequent transaction.

Data Layer relies on a proprietary function ("entry point to CN (e.g. load balancer)") to ensure stickiness (same AMF instance being selected for subsequent transactions).

- RAN selects the CCF/AMF pool and eventually AMF instance is selected based on load
- based on load balancing algorithms. .
- Potential for race conditions (e.g. consecutive messages sent to different AMFs) exists

Unclear how this solution works, e.g. how does "entry point" learn whether an AMF still holds the context for a given UE (or whether it has become stateless already)?

<u>Conclusion: Will not work across implementations.</u> Thus, race condition issue still open (incl. case when UE is in CN connected state).

NOKIA

Stateless Network Function Selection – MO transaction

Applicable only for use case #2

- When the initial message is received from UE/RAN, Network function is selected based on
- moves to CN connected state NG2 association remains active as long as connected mode.
- So, for subsequent transactions, when the UE g. ct an AMF instance based on NG2AP association.
- Ensures selection of same AMF for subsequent transaction.

Race condition issue also unresolved for the case when UE enters CN Connected state.

Note: The more AMF instances are deployed the higher the likelihood to select a different ("wrong") AMF for subsequent requests.

- RAN selects the CCF/AMF pool and eventually AMF instance is selected based on load balancing algorithm and there is no previously established NG2AP association. For subsequent transactions RAN routes the NAS signalling from the UE to a CCF/AMF in the serving network slice based on the NSSAI provided by the UE and according the same logic, e.g. based on load balancing algorithms. .
- Potential for race conditions (e.g. consecutive messages sent to different AMFs) exists

Stateless Network Function Selection – MT transaction

Applicable only for use case #2

- In case of MT transaction (e.g. SMF sending a DDN to AMF), the SMF initiating an MT transaction via AMF (that is stateless) can check with the DL function if there is a serving NF for the given UE.
- If there is a serving NF for the UE already, then the NF shall send the MT transaction to the corresponding serving NF.
- With this resolution, we can avoid a race condition i.e. different AMFs being selected for MO/MT transaction.

 © Nokia 2016

NOKIA

Stateless Network Func
Applicable only for use c

Shifts race condition into the Data layer (which comes at quite a price, see slides 32/33)

- In case of MT transaction (e.g. SMF sending a DDN to AMF), the SMF initiating an MT transaction via AMF (that is stateless) can check with the DL function if there is a serving NF for the given UE.
- If there is a serving NF for the UE already, then the NF shall send the MT transaction to the corresponding serving NF.
- With this resolution, we can avoid a race condition i.e. differ[illegible] being selected for MO/MT transa[illegible]n.

Question: Who (and based on which information) selects AMF if there is no AMF listed in the Data Layer?

Question: Who (and based on which information) selects SMF, e.g. in case of an MT transaction?

Assumption (TBC): UPF selects SMF only based on load information (in line with AMF selection by RAN).

Question: What happens if the serving NF changes (e.g. SMF) or becomes stateless after the source NF (e.g. PCF) looked up the serving NFs ID in the Data Layer and while source NF is still busy sending the request to the (former) serving NF (e.g. SMF)?

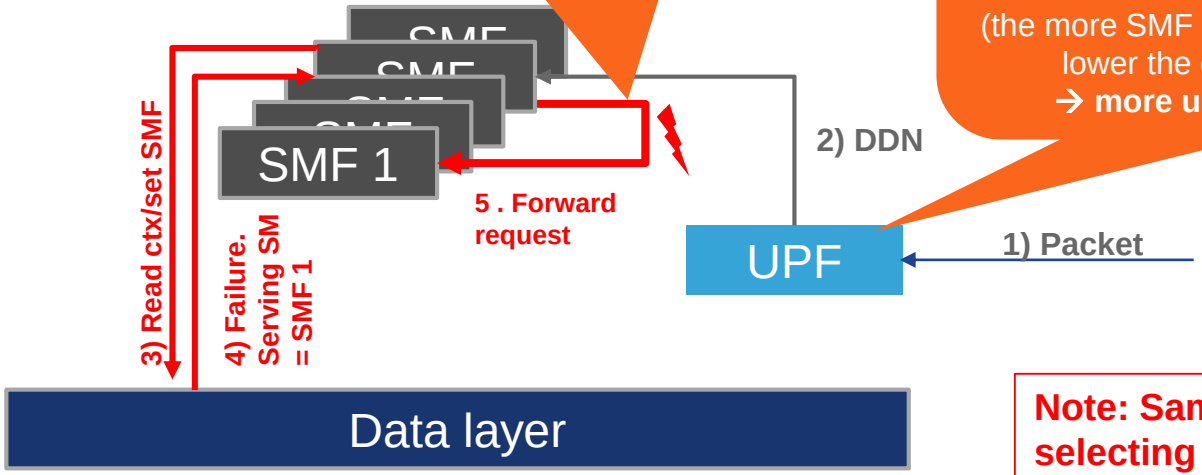
Selection of stateless functions as per DL solution is inefficient and increases system delays

Inefficient: Additional signaling and delays due to required forwarding of requests (can happen per MT transaction)

UPF selects SMF randomly (potentially taking SMF load load into account).

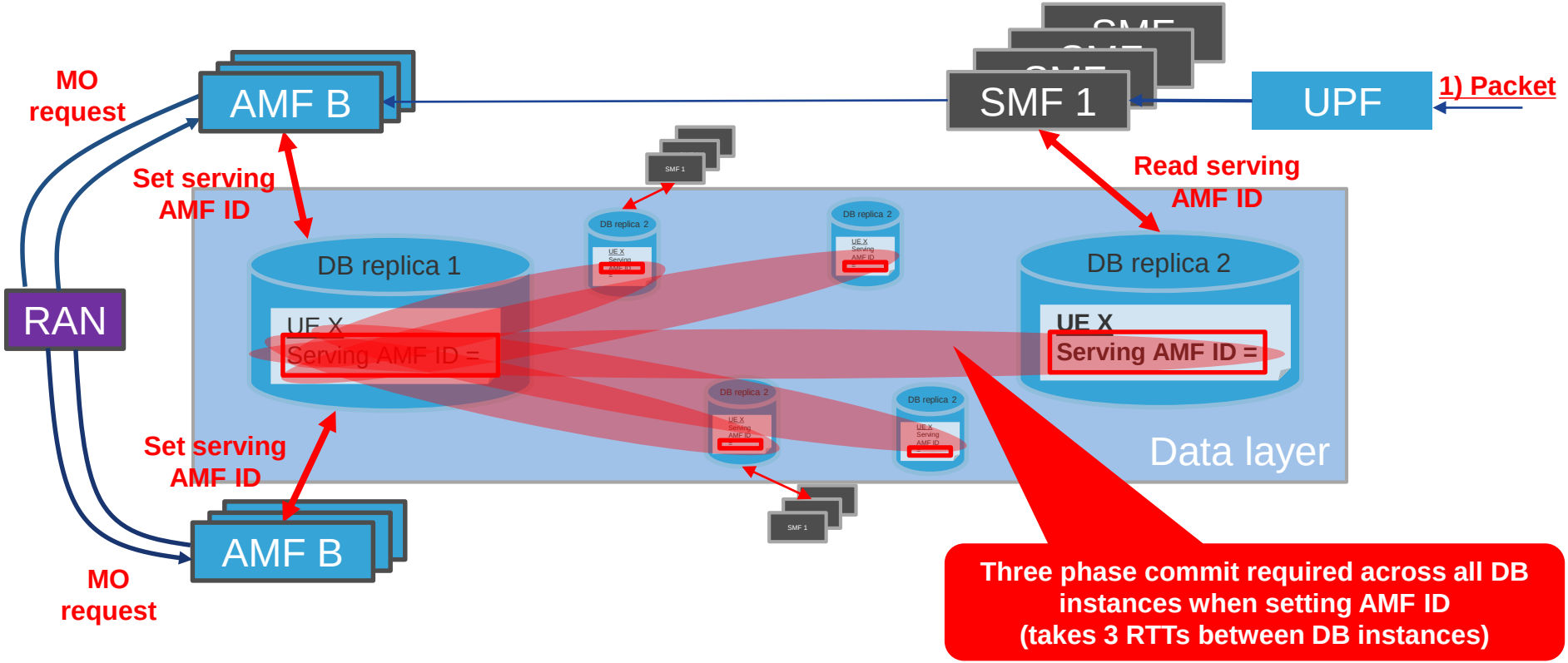
<u>High likelihood to select wrong SMF</u> (SMF which does not hold UE's context).

(the more SMF instances with similar load, the lower the odds to pick correct SMF
→ **more unnecessary forwarding**)



Note: Same issue applies when selecting AMF for MO transactions

Storing serving NF function IDs in Data Layer shifts race condition into DL
Delays all procedures (need for DB sync each time stateless NFs are selected)

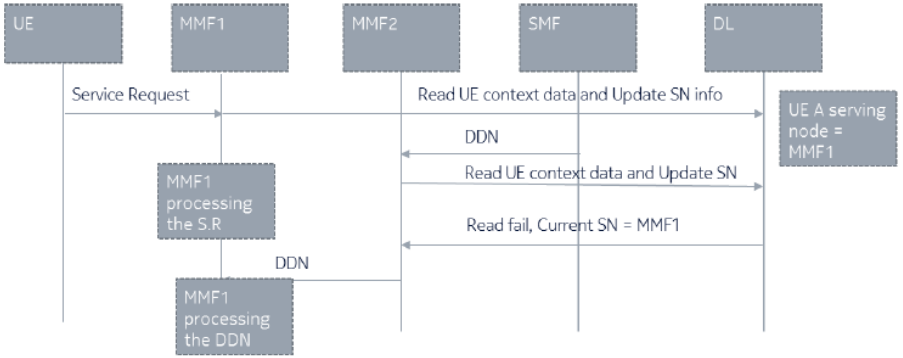


Cisco, ZTE, Ericsson

Race condition #1 - Applicable only for use case #2
Between MO and MT transaction

- **Race condition:**
- Service request is triggered by Uplink data from the UE.
- RAN selects the AMF1 based on load balancing algorithms.
- UPF receives a downlink data for the UE and it routes the Downlink Data and/or Downlink Data Notification message to SMF (depending on where the buffering is supported); SMF eventually forwards it to the AMF2.
- AMF2 reads the UE context from the data layer

Proposed Resolution



- if the scenario happens, the Data layer receives the access request from AMF1 first and when receiving the access request from AMF2, it will reject the access with an indication that AMF1 is the serving node.
- AMF2 should forward the DDN to AMF1. (No need to roll back anything as CCF-2 / AMF-2 hasn't done anything until then).

 © Nokia 2016

NOKIA

Race condition #1 - Applicable only for use case #2

Between MO and MT transaction

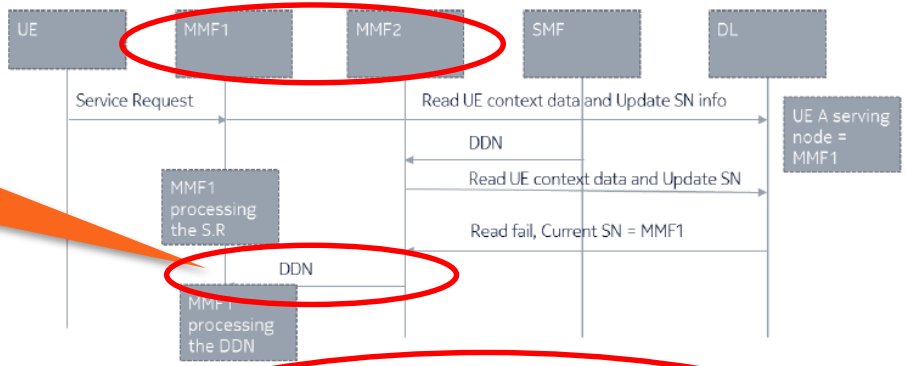
- **Race condition:**
- S … m
  t
- P
  algorithms.
- UPF receives a downlink data for the UE and it … ata

Forwarding of requests between NFs of same type increases signaling and delay unnecessarily.

Proposal moves the race condition into the Data layer (see slide 33).

- AMF2 reads the UE context from the data layer

Proposed Resolution



- If the scenario happens, the Data layer receives the access request from AMF1 first and when receiving the access request from AMF2, it will reject the access with an indication that AMF1 is the serving node.
- AMF2 should forward the DDN to AMF1. (No need to roll back anything as CCF-2 / AMF-2 hasn't done anything until then).

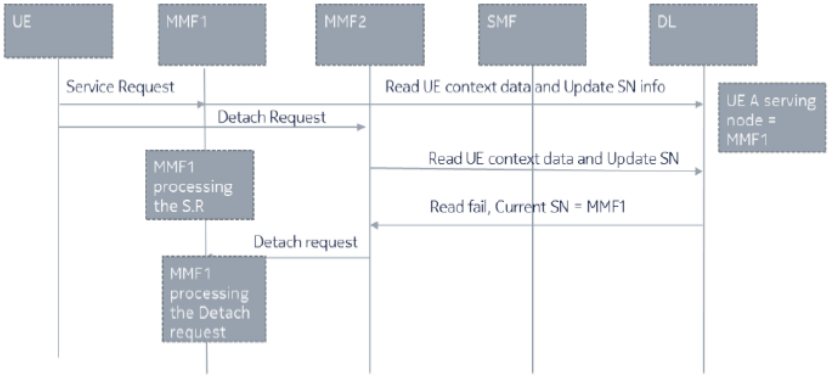
NOKIA

Race condition #2 - Applicable only for use case #2

Between consecutive MO transactions

- **Race condition:**
- Service request is triggered by Uplink data from the UE.
- Detach request is sent by the UE.

Proposed Resolution



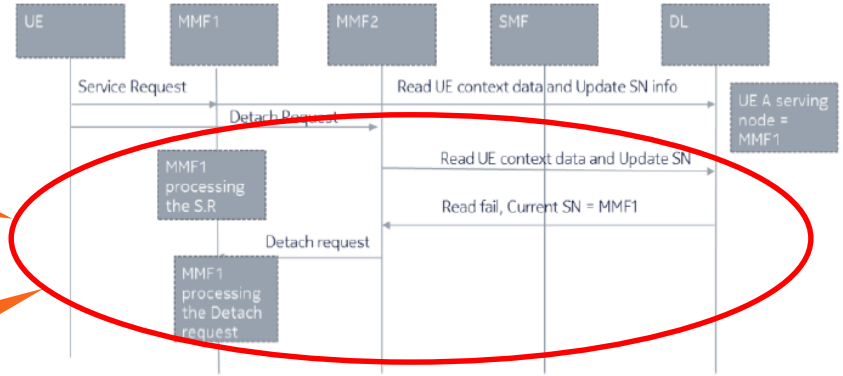
- if the scenario happens, the Data layer receives the access request from AMF1 first and when receiving the access request from AMF2, it will reject the access with an indication that AMF1 is the serving node.
- AMF2 should forward the DDN to AMF1. (No need to roll back anything as CCF-2 / AMF-2 hasn't done anything until then).

Race condition #2 - Applicable only for use case #2

Between consecutive MO transactions

- **Race condition:**
- Forwarding of requests between NFs of same type increases signaling and delay for request execution (for all types of MO requests).
- For subsequent MO requests the likelihood of this happening is high (increases with the number of serving AMFs with similar load).
- Current architecture proposals do not support this; would require an interface per NF to itself (AMF-AMF, SMF-SMF, PCF-PCF)

Proposed Resolution



- if the scenario happens, the Data layer receives the access request from AMF1 first and when receiving the access request from AMF2, it will reject the access with an indication that AMF1 is the serving node.
- AMF2 should forward the DDN to AMF1. (No need to roll back anything as CCF-2 / AMF-2 hasn't done anything until then).

NOKIA

Conclusion: Data Layer relies on proprietary function for handling MO transactions and delays system transactions due to forwarding of requests between NFs and due to need for lengthy three phase DB commits during NF selection.

Recovery and Restoration

How recovery and restoration can be supported while a network function holds state (i.e. before it writes its state to the data layer) for scenarios where multiple network functions are involved (e.g. as part of a procedure involving multiple network functions)?

Proposed Resolution

- If a network function fails during a certain procedure, some data created in the middle of a transaction may be lost.
- This is true with or without the Data Layer. But the Data Layer allows the service/UE context to be restored from the last stable state.
- This is similar to the existing EPC restoration procedure, which also relies on storing state information in a centralized repository, and for which a failure may also happen in the middle of a transaction.
- The Data Layer provides a framework that inherently supports the restoration feature.

 © Nokia 2016

NOKIA

Recovery and Restoration

How recovery and restoration can be supported while a network function holds state (i.e. before it writes its state to the data layer) for scenarios where multiple network functions lure involving multiple network functions)?

Implementations typically support better resiliency mechanisms than just relying on EPC restoration (the last resort), which do not necessarily use centralized repositories.

posed Resolution

- If a network function fails du rtain procedure, some data created in the middle of a transaction may be lost.
- This is true with or without the Data Layer. t the Data Layer allows the service/UE context to be restored from the last stable state.
- This is similar to the existing EPC restoration procedure, which also relies on storing state information in a centralized repository, and for which a failure may also happen in the middle of a transaction.
- The Data Layer provides a framework that inherently supports the restoration feature.

NOKIA