Additional CATS requirements Consideration for Service Segmentation-related use cases

draft-dcn-cats-req-service-segmentation-02

Minh-Ngoc Tran (Soongsil University), Younghan Kim (Soongsil University)

This Document's Target

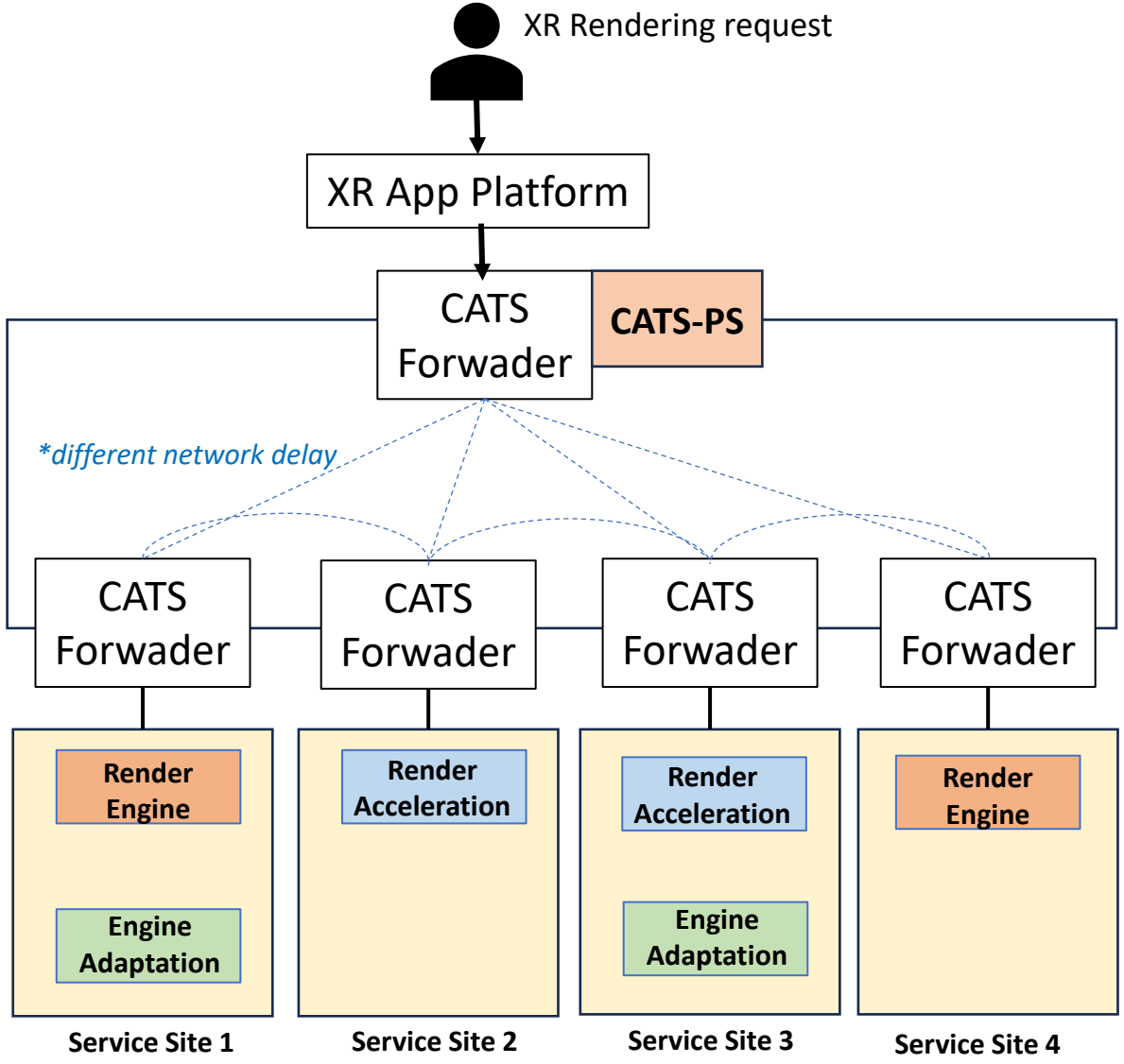
- **Service Segmentation[1]** is a service deployment option
  [1]draft-li-cats-task-segmentation-framework
  - A Service **is split into multiple subtasks** which can run in different computing sites.
  - A Service request is served by the **combination of these subtasks.**
  - These subtasks can run **in parallel or in sequence/chain.**

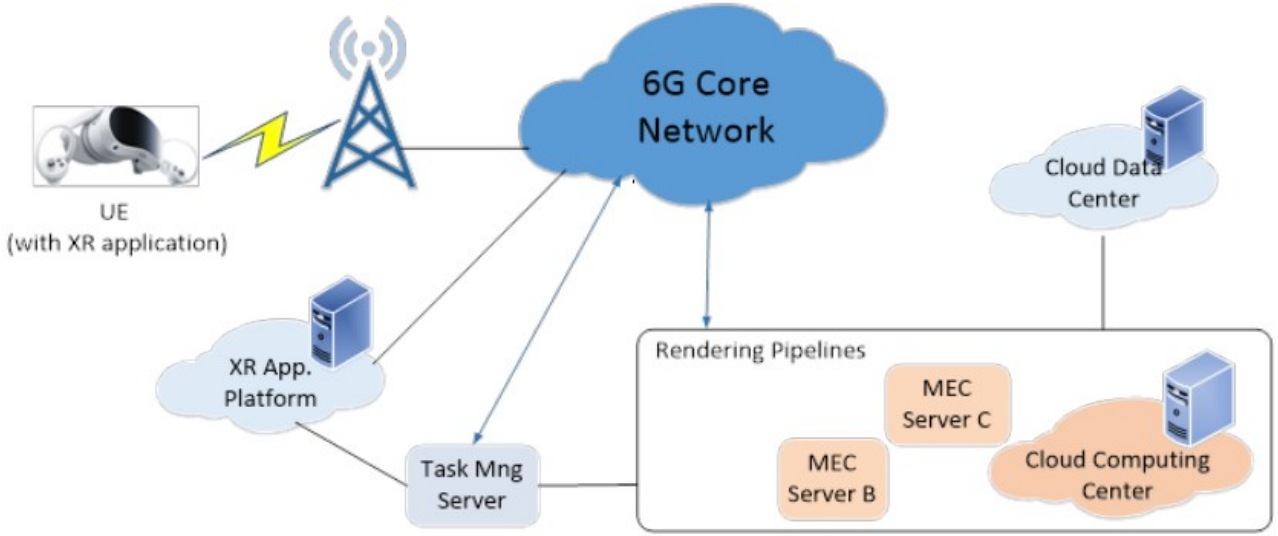
- **Service Segmentation is a deployment option for 2 Use Cases** in CATS Use Cases I-D (draft-ietf-cats-usecases-requirements)

  (2 more realistic examples compared with presentation at previous meeting)
  - AR-VR
    - Example: AR-VR(XR) Rendering service is deployed as a pipeline of XR rendering subtasks
  - Distributed AI Inference
    - Example: A ML model is vertically split into several slices that can processed split input in parallel before merging into one final inference result

- **In this time presentation:** Based on 2 CATS use case examples where service segmentation is applied
  - Propose CATS system design consideration points for supporting CATS service segmentation to enhance current WG documents

Example 1: XR Sequential Rendering Pipeline



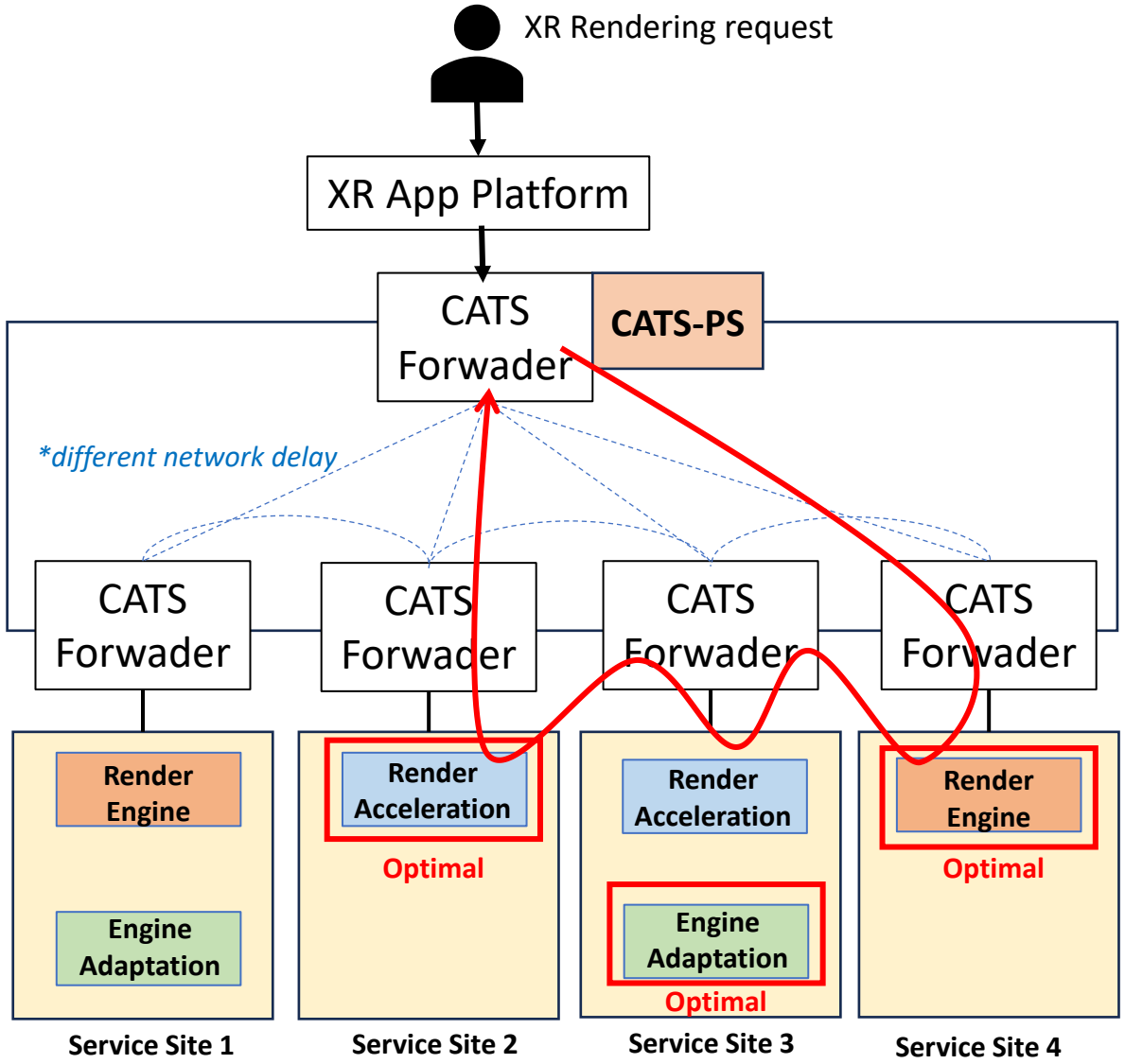


***XR Rendering Pipeline example* In 3GPP Technical Report 22.870 (Mar 2025) about 6G use cases and services**

XR Rendering service is deployed as pipeline of 3 subtasks across multiple MEC servers:

- Render Engine
- Engine Adaptation
- Render Acceleration

Example 1: XR Sequential Rendering Pipeline



CATS Objective

*Instead of **selecting an optimal XR Rendering service instance***

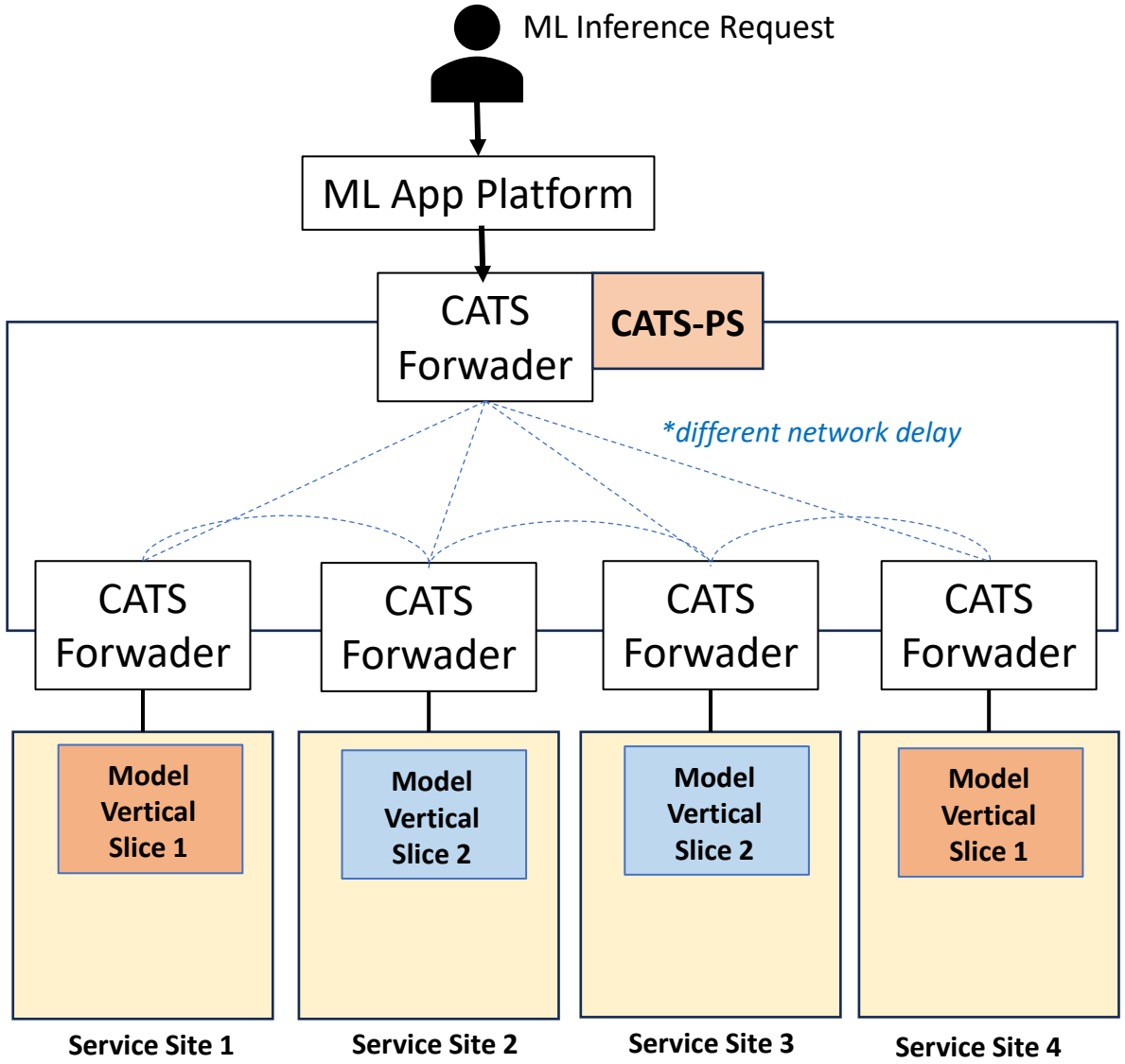
➔ *Select an **optimal Sequential XR Rendering pipeline***

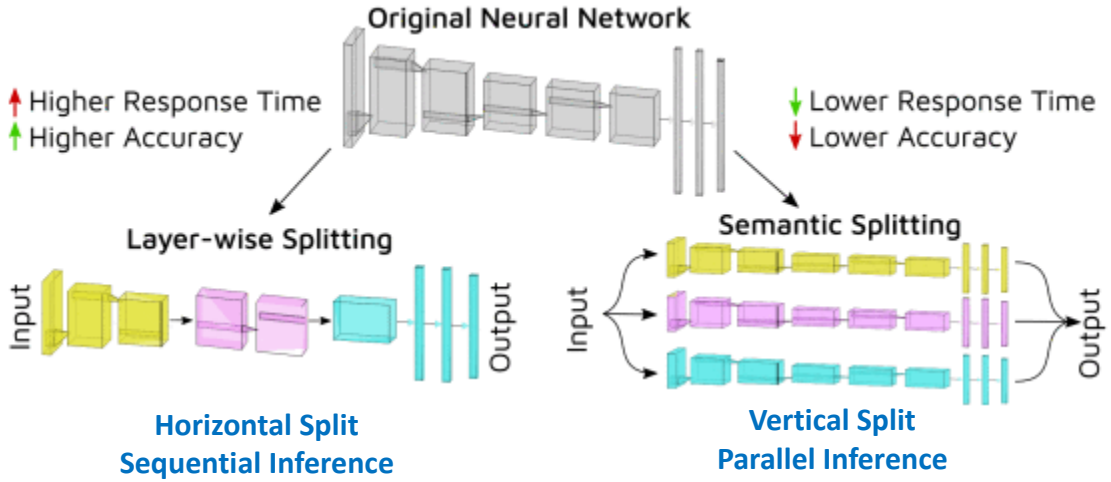
- ➔ ***1 Optimal instance for each subtask/sub-service in Pipeline***
- ➔ ***Subtasks are routed in Sequential order Render Engine (RE) -> Engine Adaptation (EA) -> Render Acceleration (RA)***

Example CATS candidate paths

- ***RE Site 4, EA Site 1, RA Site 3***
- ***RE Site 1, EA Site 1, RA Site 2***
- ***RE Site 4, EA Site 3, RA Site 2***
- ***....***

Example 2: Vertical ML Model Split Parallel Inference





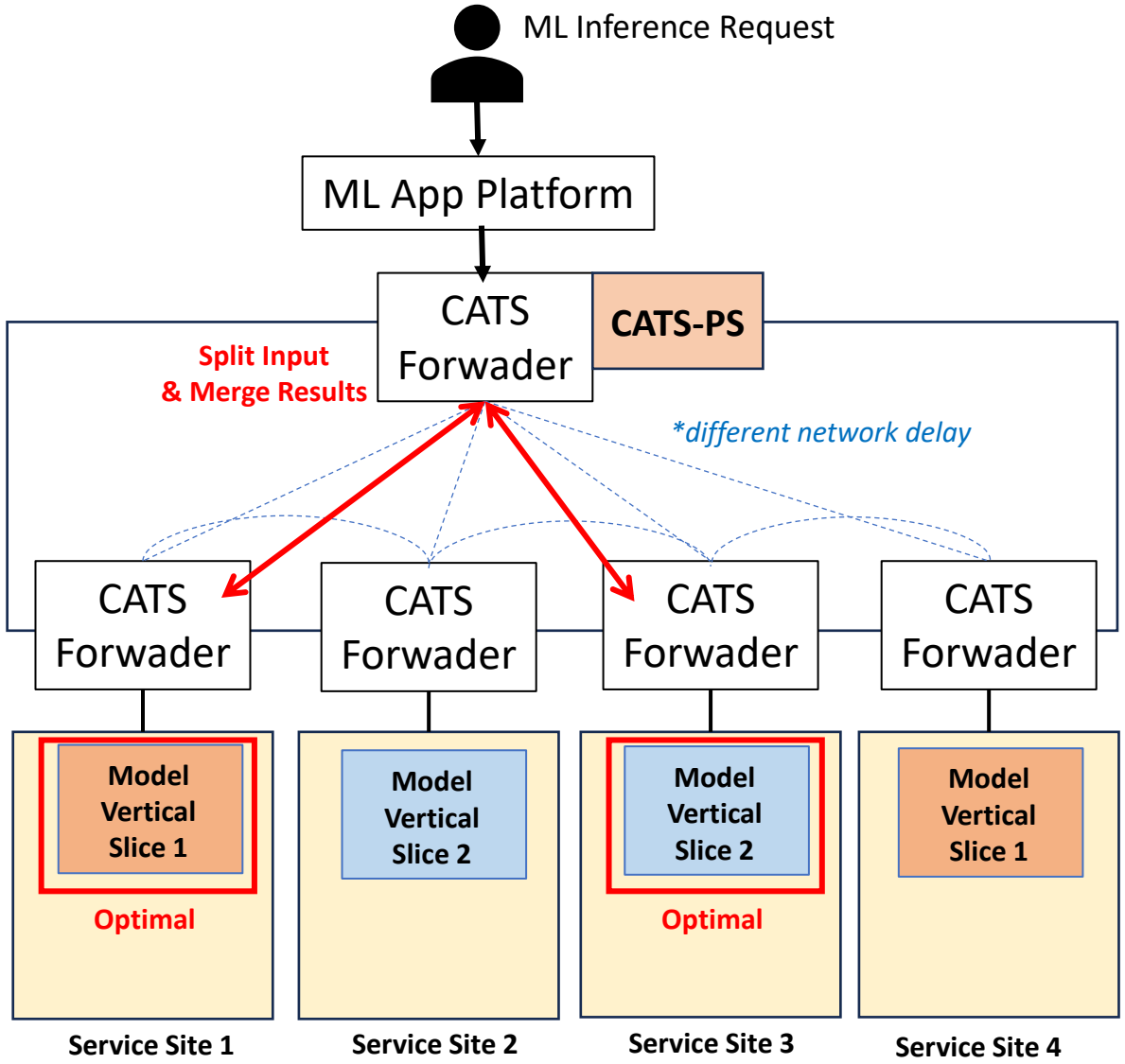
Horizontal and Vertical Split ML Model Inference Techniques

In SplitPlace: AI Augmented Splitting and Placement of Large-Scale Neural Networks in Mobile Edge Environments (IEEE Transaction on Mobile Computing, Sep 2023)

ML can be vertically split into multiple slices based on different dimensions in a model layer

- Input is split corresponding required dimension of Vertical Model Slice
- Vertical Model Slices process split input in parallel to reduce inference response time
- Results are merged before return to Client

Example 2: Vertical ML Model Split Parallel Inference



CATS Objective

*Instead of **selecting an optimal ML Inference service instance***

➔ *Select an **optimal Parallel ML Inference Pipeline***

- ➔ ***1 Optimal instance for each ML Model Slice***
- ➔ ***Request Input is pre-processed corresponding to each ML Model Slice input requirement, before being steered to these slices in parallel***

Example CATS candidate paths

- ***Slice 1 Site 1, Slice 2 Site 2***
- ***Slice 1 Site 1, Slice 2 Site 3***
- ***Slice 1 Site 2, Slice 2 Site 3***
- ***….***
- ***In some case, more than 1 model split options are available (split to 2,3,4 slices…)***

3 CATS system design consideration points for Service Segmentation scenario

As AR/VR and Distributed AI Inference are among the CATS use cases, the CATS system design should also fully support scenarios where service segmentation is applied to these use cases.

1. Traffic Steering Objective

- **In service segmentation scenario, The optimal service instance is the optimal sequence/parallel pipeline that consists of the optimal instances of each subtask composing the service**
  - Current CATS WG documents is built based on the concept of an steering to a single optimal service instance
    - *"Steering in CATS is about selecting the appropriate service contact instance that will service a request according to a set of network and computing metrics." – CATS Framework document*
  - Instead of introducing subtask concept which might cause a lot of changes, CATS can still cover this Service Segmentation case by:
    - **considering a subtask as an individual service, a pipeline of service might be required to realize an application request**
    - *Then "selecting the appropriate service contact instance(s)" can still cover this case*
      - *1 Optimal service contact instance for each service in a pipeline*
      - *C-PS might require all possible pipeline awareness*

3 CATS system design consideration points for Service Segmentation scenario

As AR/VR and Distributed AI Inference are among the CATS use cases, the CATS system design should also fully support scenarios where service segmentation is applied to these use cases.

2. Traffic Steering Mechanism

- **The CATS system may need to provide a mechanism to steer service requests in a predetermined sequence**
  - ➢ In case of sequential pipeline, packets might need to be routed between Egress CATS Forwarders

3. CATS Metric Aggregation

- **CATS metrics can be aggregated not only by metric category but also by individual service subtasks** (e.g. CATS metrics representing a candidate pipeline is aggregated from the metrics of its component subtask instances)
  - ➢ Current CATS metric document discusses metric aggregation based on different resource category only

Our suggestion

As stated in mailing list discussion thread

- **As Service Segmentation is only an optional service deployment method, these CATS design impacts are only additional implementation requirements for this case, not general requirements**
  - C-PS pipeline awareness, Sequential routing mechanism, CATS pipeline metric aggregation

- **A brief paragraph in CATS Use case & Requirement I.D that mentions certain optional, service-specific deployment models can raise additional CATS system-level design considerations**
  - This might be beneficial to keep this WG I.D to remain both generally applicable and flexible, open for specific implementation requirement in such cases.