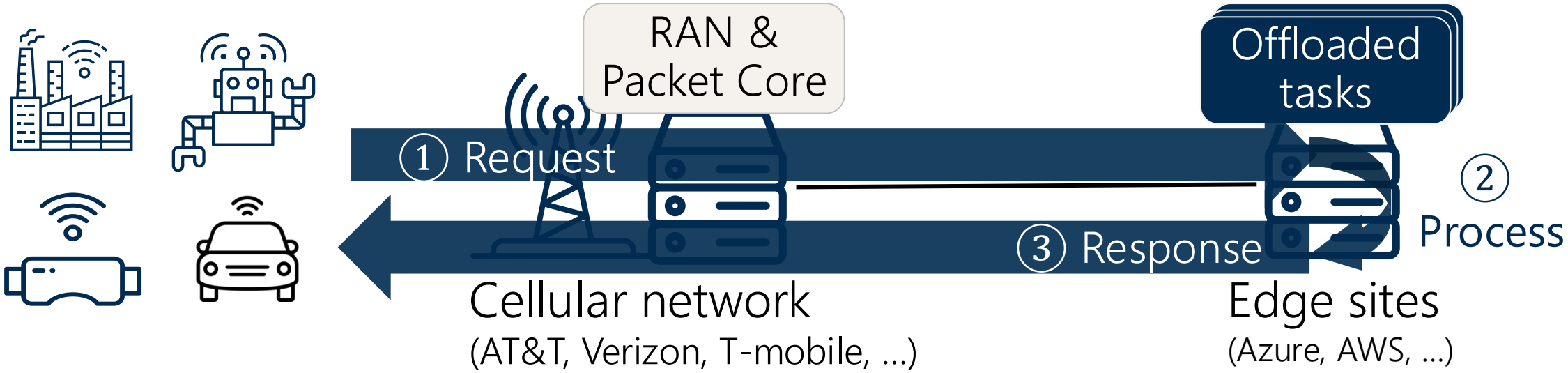
SLO-Aware Multi-Access Edge Computing with SMEC

Xiao Zhang Daehyeok Kim

The University of Texas at Austin

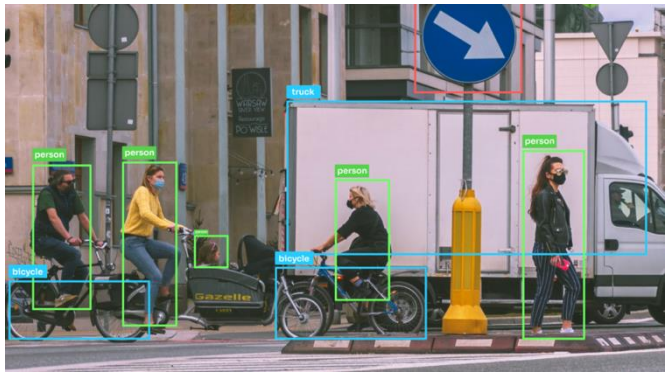
Multi-access edge computing (MEC)



- MEC allows app developers to offload compute-intensive tasks to the edge
- Request-response pattern: **Round-trip latency SLO** (Service Level Objective)

Multi-access edge computing (MEC)

Smart Stadium (100 ms)



Augmented Reality (100 ms)

Video Conferencing (150 ms)

- MEC allows app developers to offload compute-intensive tasks to the edge
- Request-response pattern: **Round-trip latency SLO** (Service Level Objective)
  - E.g., 10-100s ms for smart stadium, AR, video conference, etc.

Trend: Real-time applications move to the edge

Verizon positions 5G and edge as backbone for AI evolution

AT&T combines with AWS in metro, Ericsson in RAN, Azure at edge

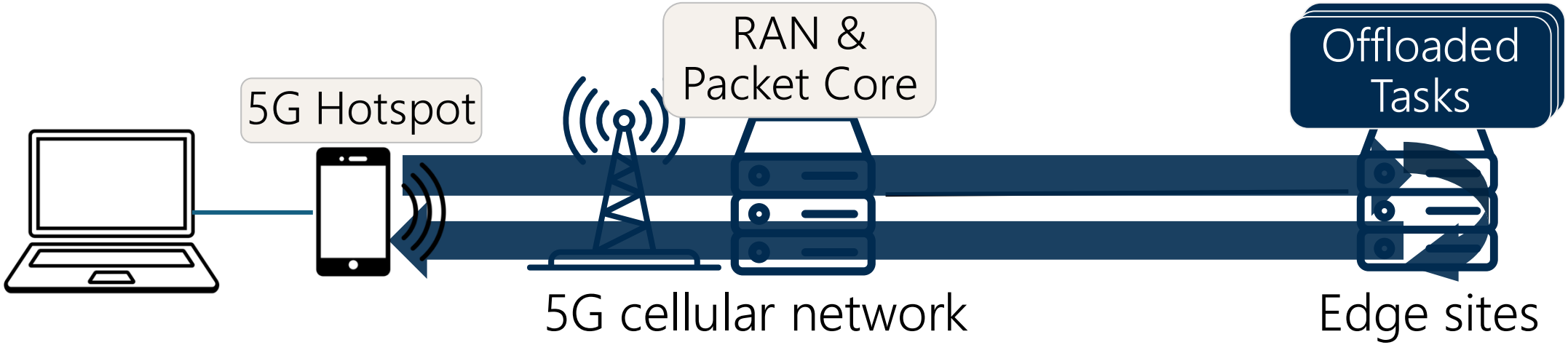
NVIDIA, T-Mobile and Partners Integrate Physical AI Applications on AI-RAN-Ready Infrastructure

AT&T, Cisco and Nvidia advance network-based edge AI

Massive investment in infrastructure and emerging use cases!

But can it actually meet application requirements in practice?

Measurement study on MEC deployments



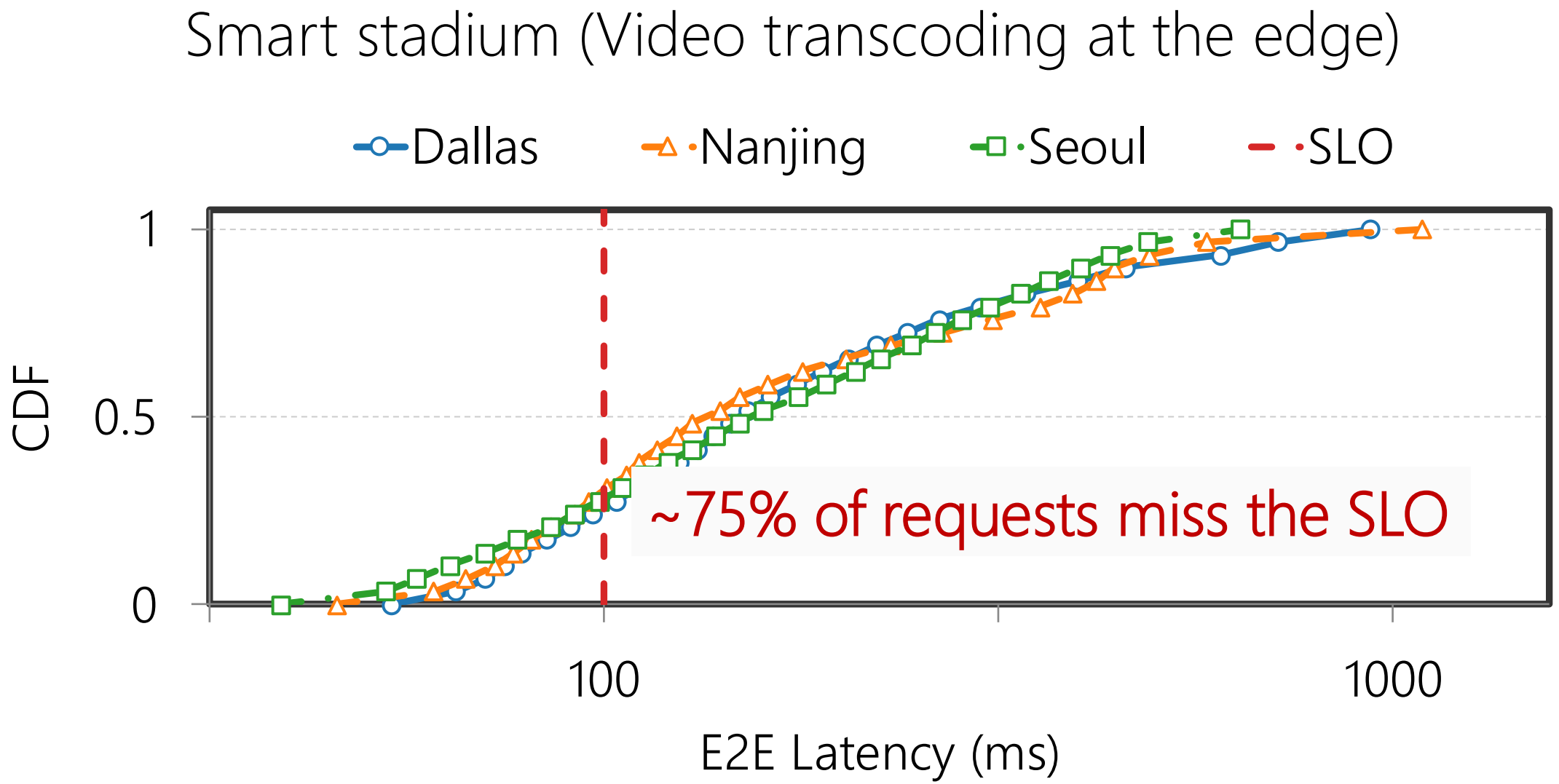
5G networks: Dallas (US), Nanjing (China), and Seoul (S. Korea)

Edge services: AWS Wavelength, Tencent Cloud

Three UDP-based low latency applications with different latency SLOs

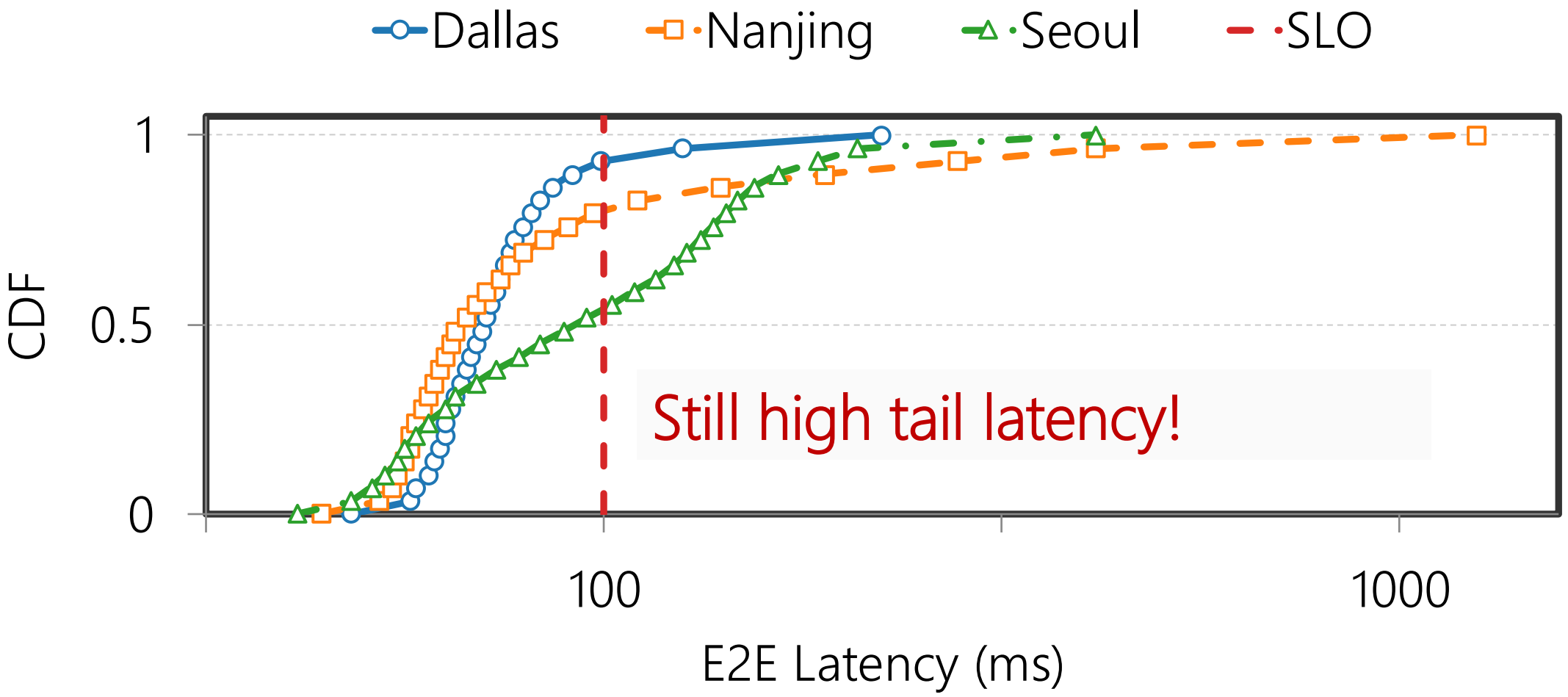
Measured the "request-response latency" for each application

Today's offerings incur high latency variability

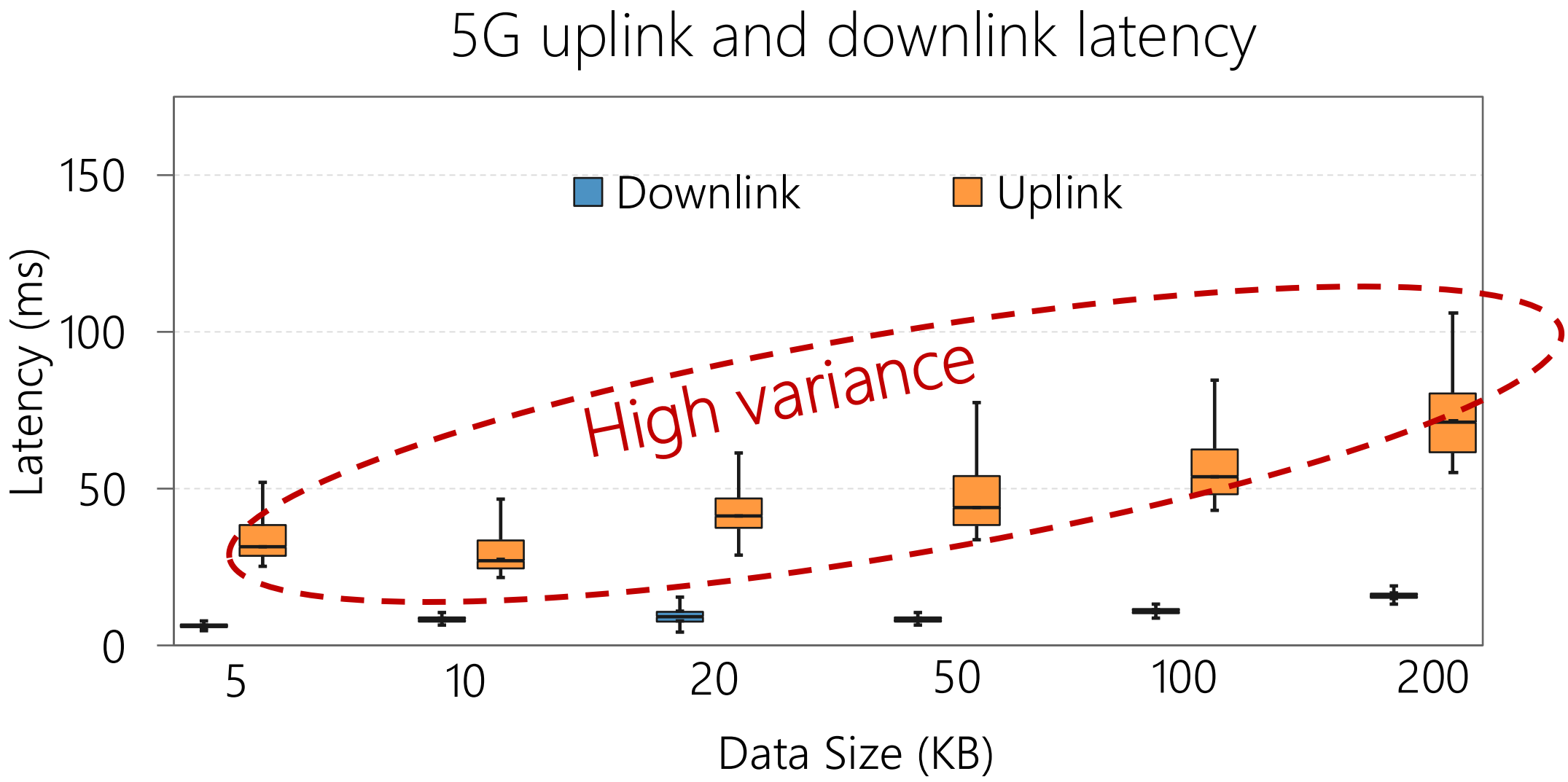


Source 1: 5G wireless resource contention

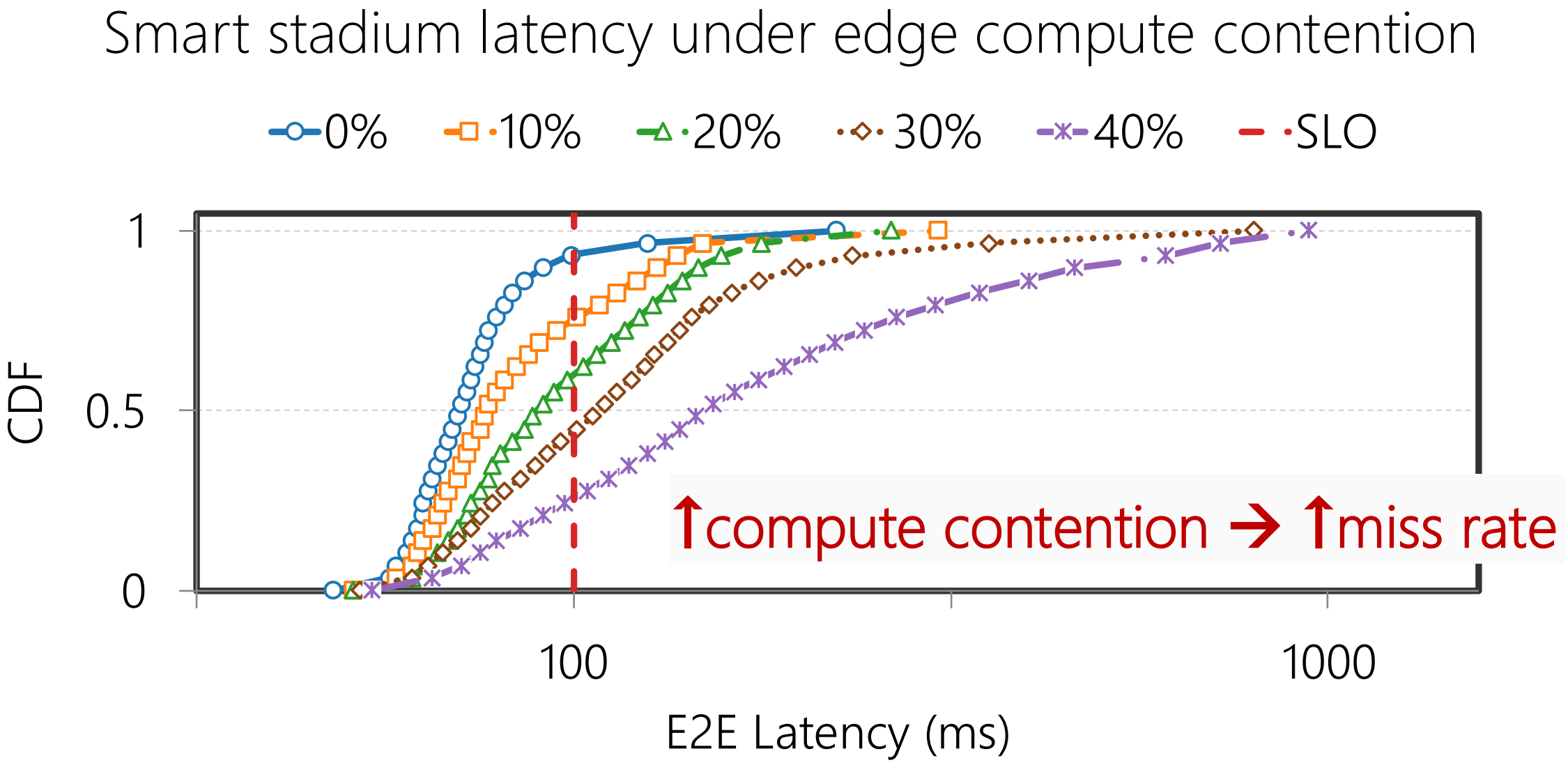
Smart stadium latency *without* edge compute contention



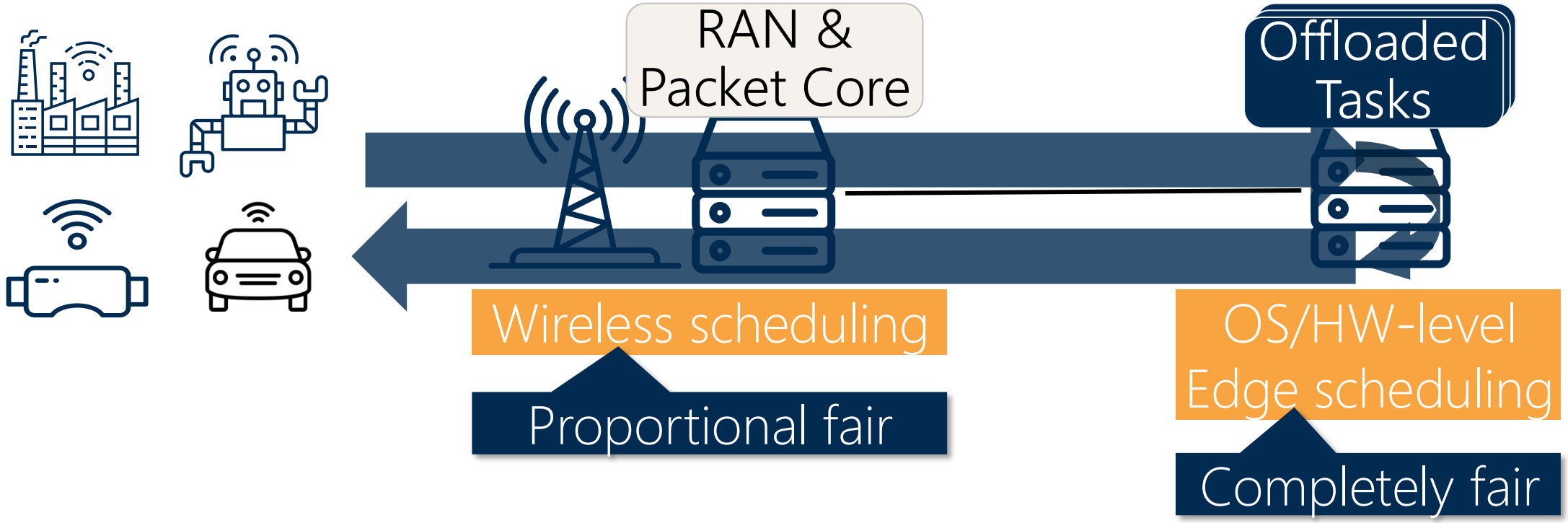
Uplink contention is the main factor



Source 2: Edge resource contention



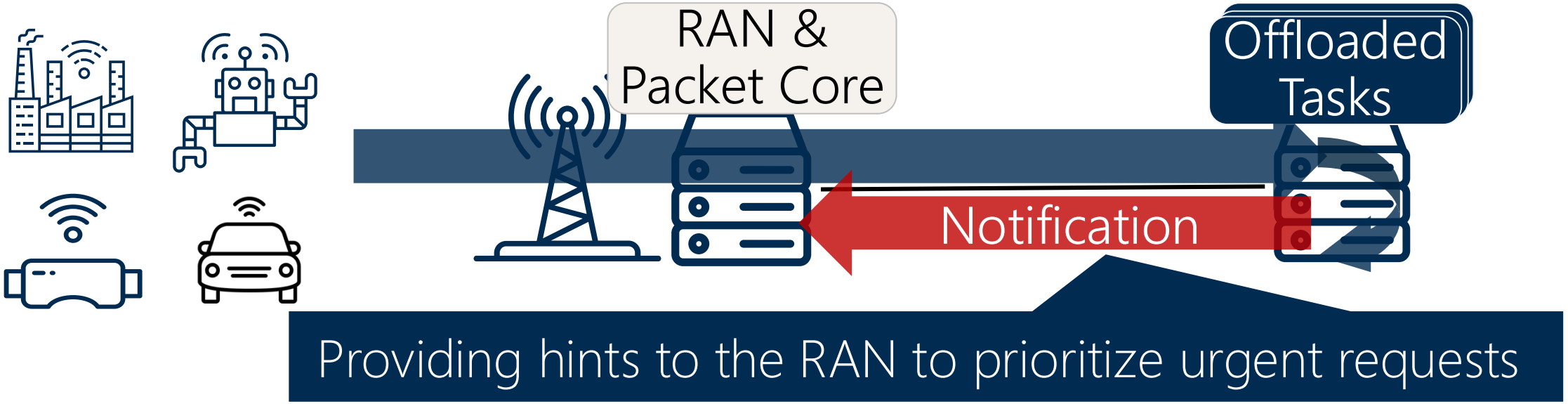
Root cause:
SLO-unaware resource management



Resource schedulers typically consider fairness and/or resource utilization, but not application-level latency SLOs

RAN-Edge coordination does not help!

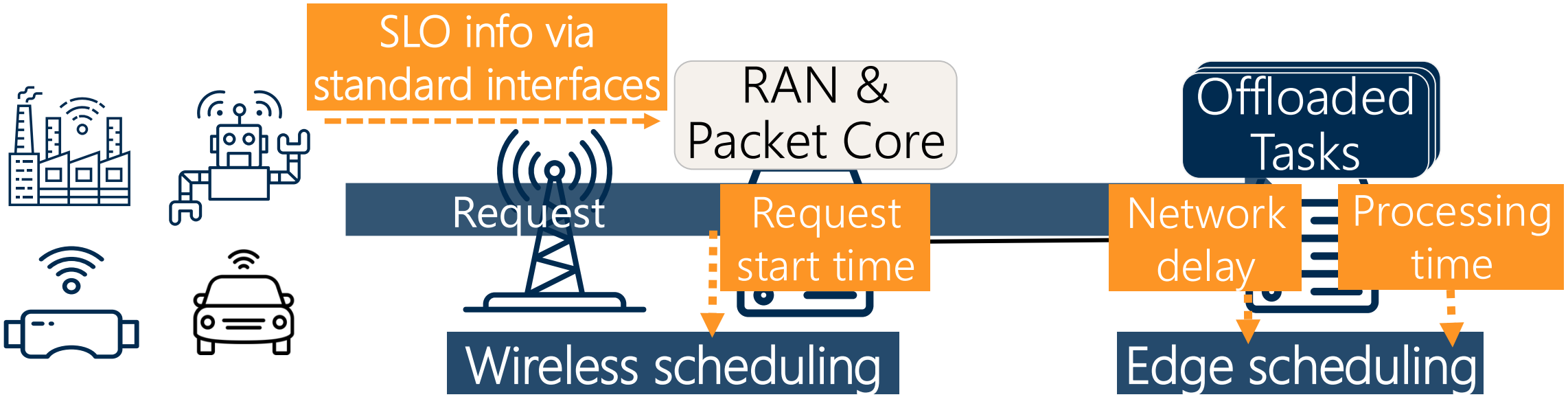
Prior systems rely on RAN-Edge coordination to make an informed decision
E.g., Tutti (Mobicom'22) and ARMA (Mobisys'25)



- Coordination delays hinder timely resource allocation for urgent requests
- Impractical as the RAN and edge are managed by different operators

SMEC: SLO-aware resource management for MEC

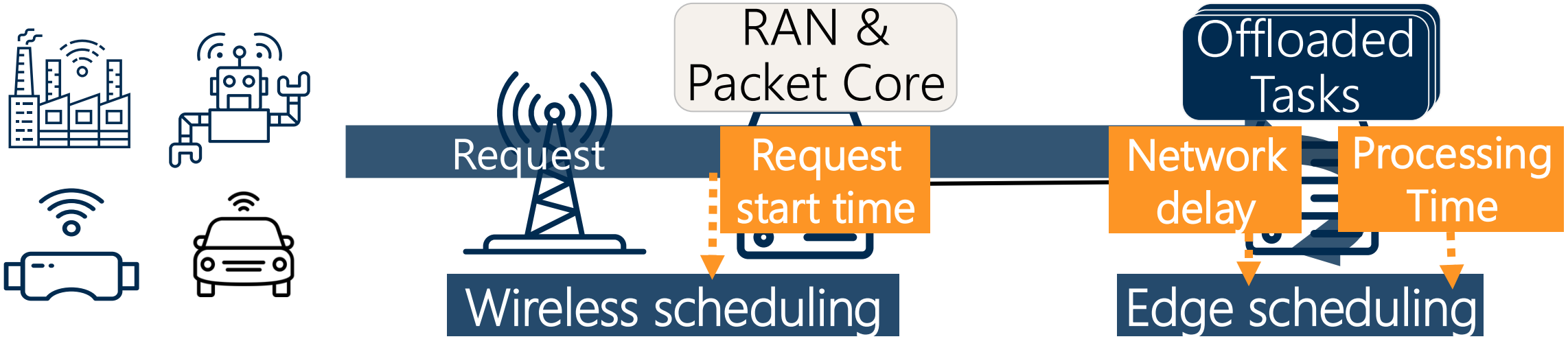
Goal: Improving SLO satisfaction rate *without* RAN-edge coordination



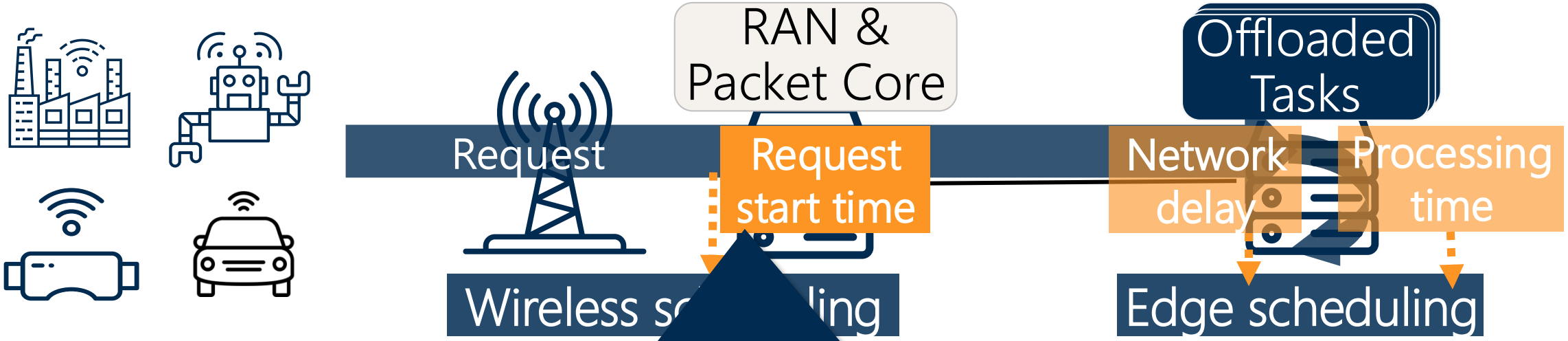
Our approach: RAN and Edge estimate the deadline of requests *locally* and accelerate near-deadline requests!

Is this vision achievable?

Challenge: Inferring required timing information

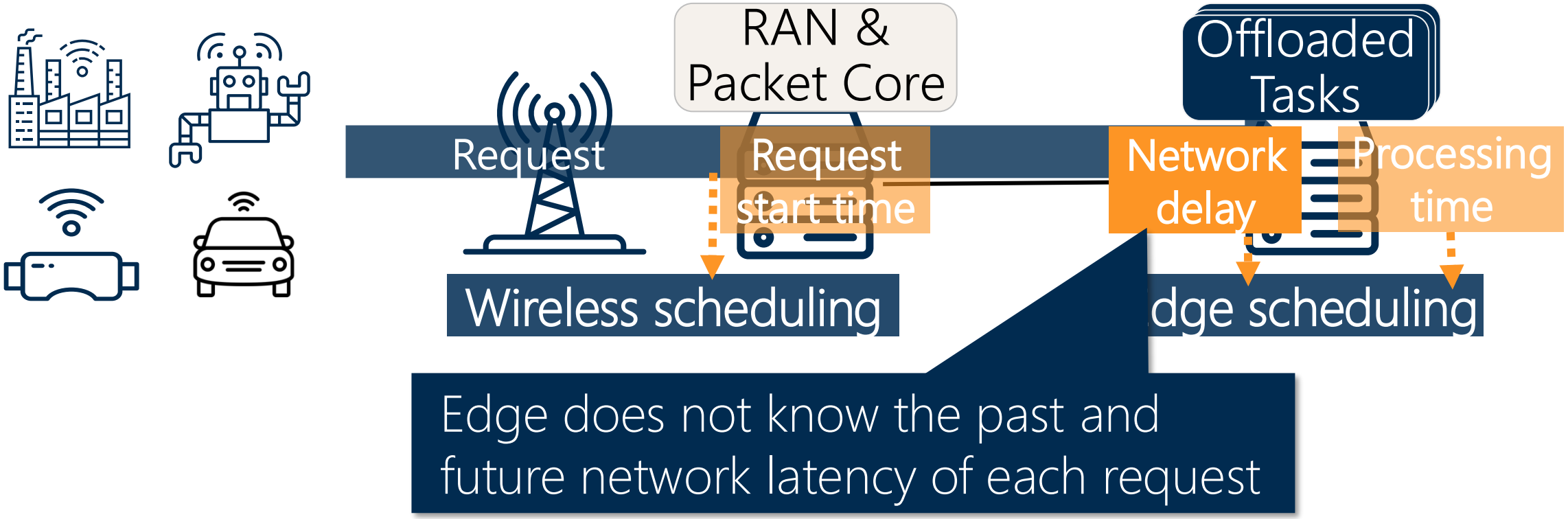


Challenge: Inferring required timing information

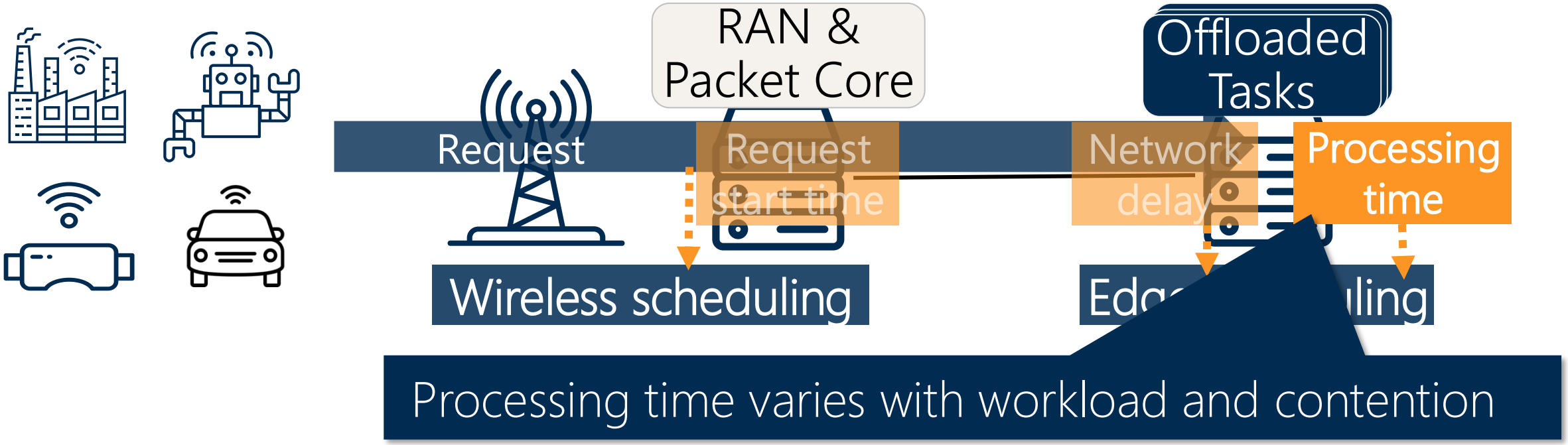


- RAN lacks applications' request-level visibility
- Tight timing (500 µs) requirements for scheduling

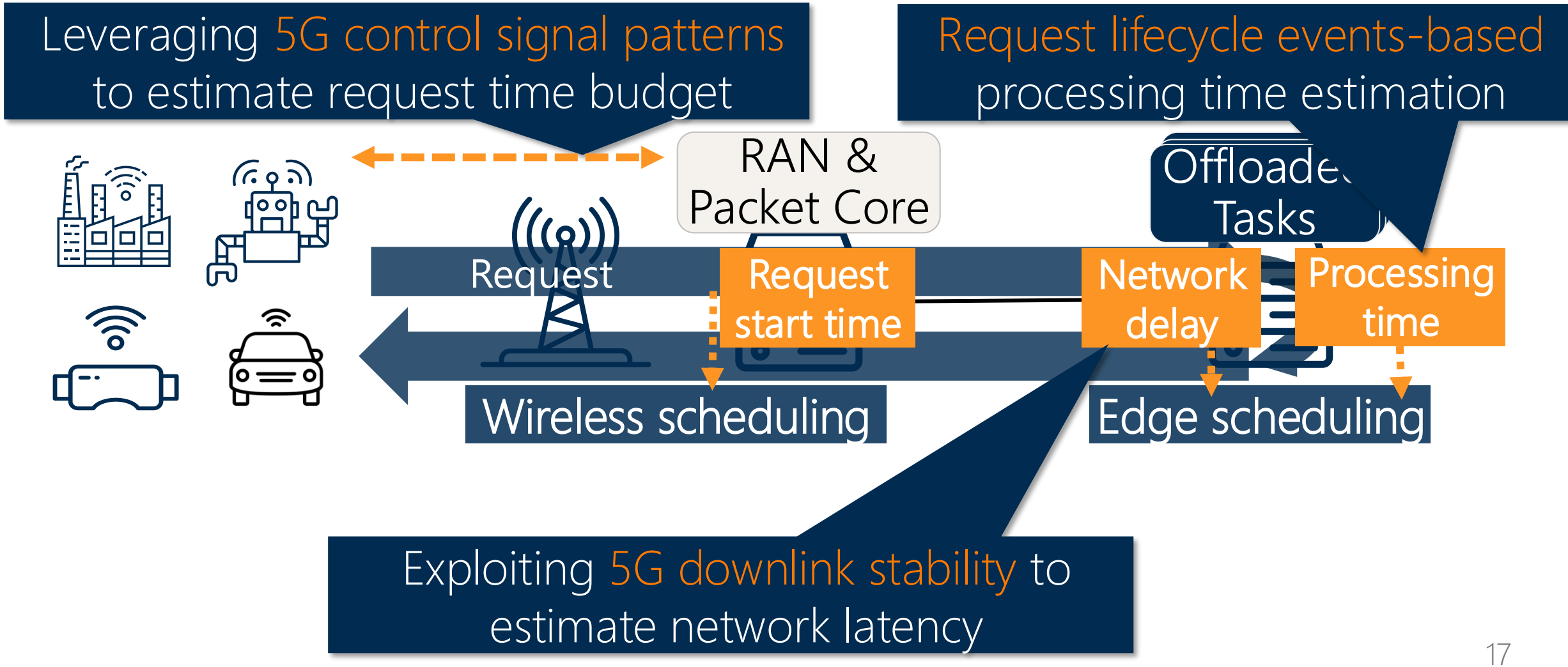
Challenge: Inferring required timing information



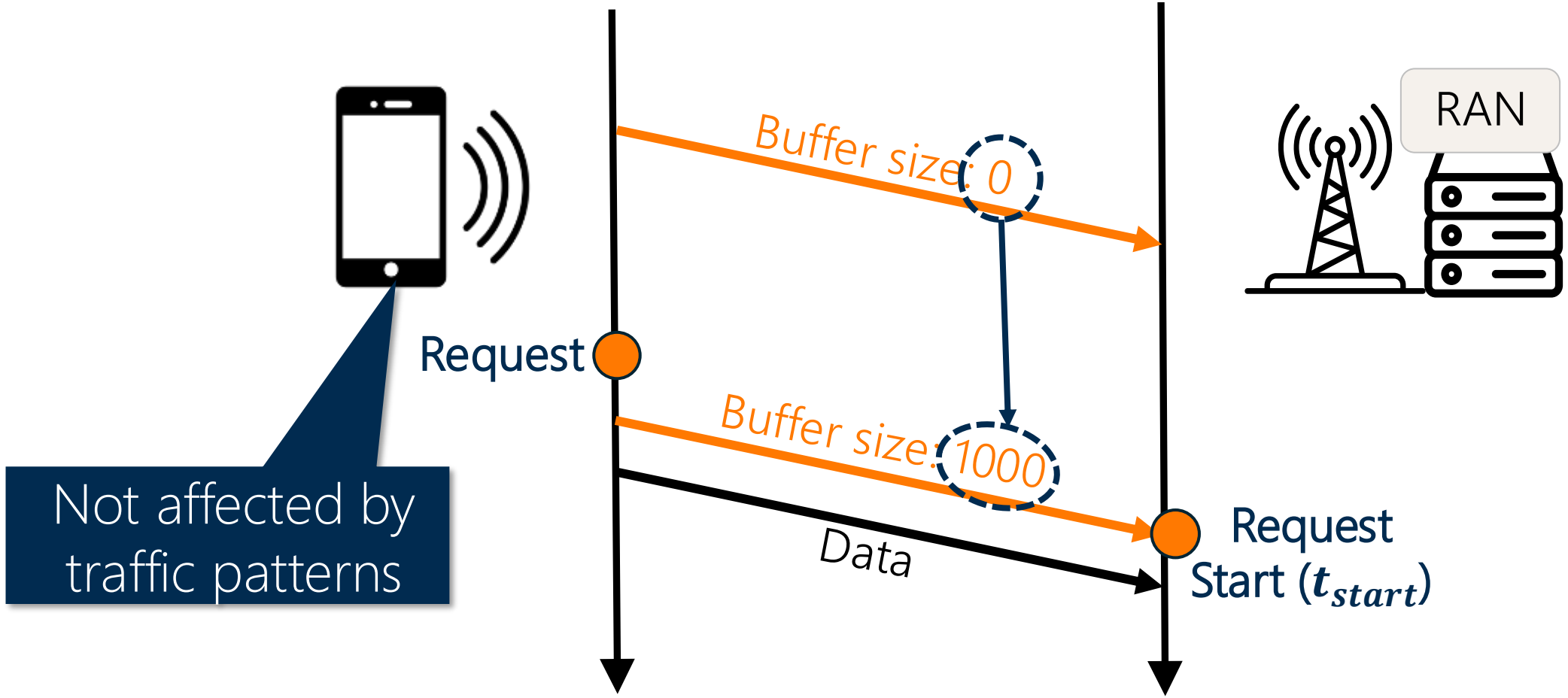
Challenge: Inferring required timing information



Key insight:
Exploiting natural signals and properties in MEC

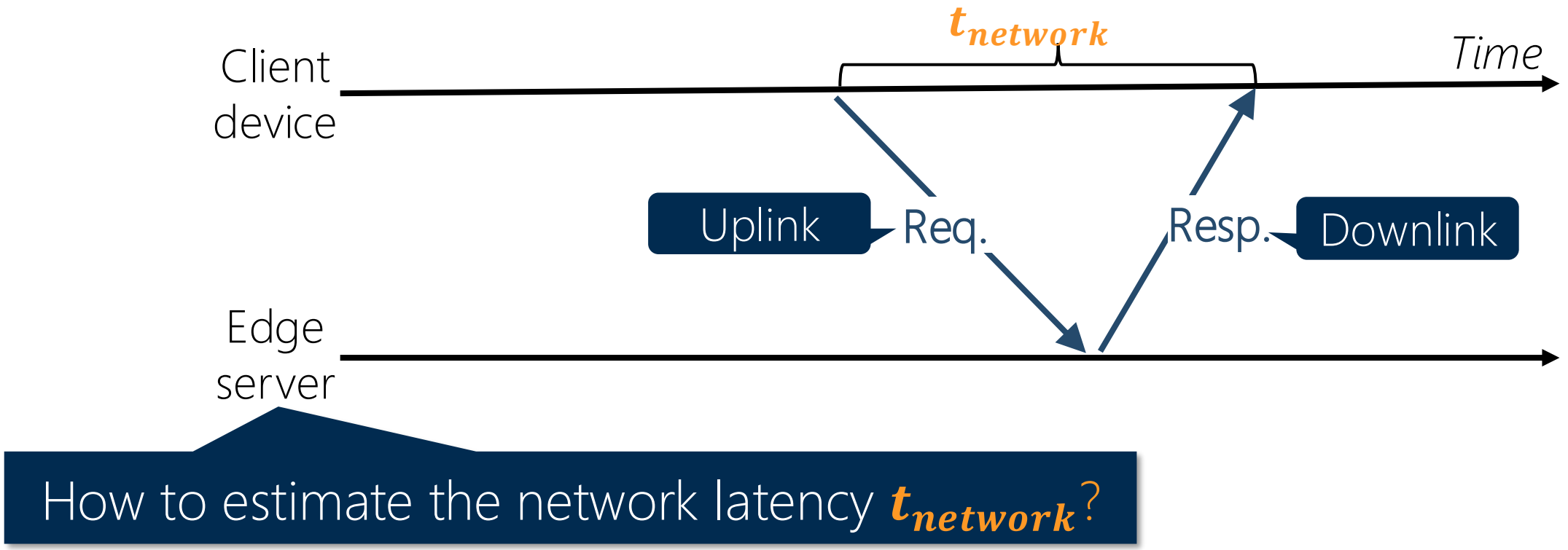


Idea 1: Using standard 5G control signals to infer request start time

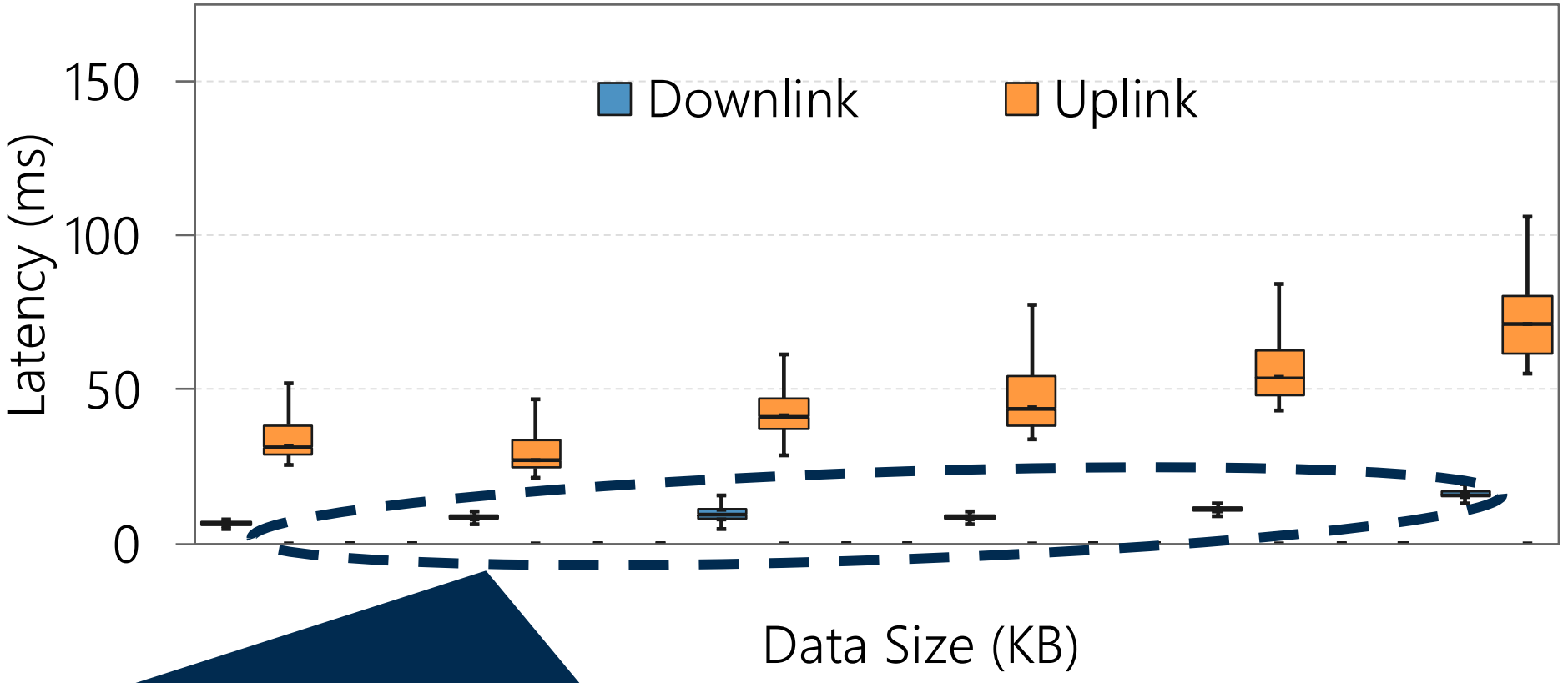


$$t_{budget}^{RAN} = SLO - (t_{current} - t_{start})$$

Idea 2: Exploiting downlink stability for network latency estimation at the edge

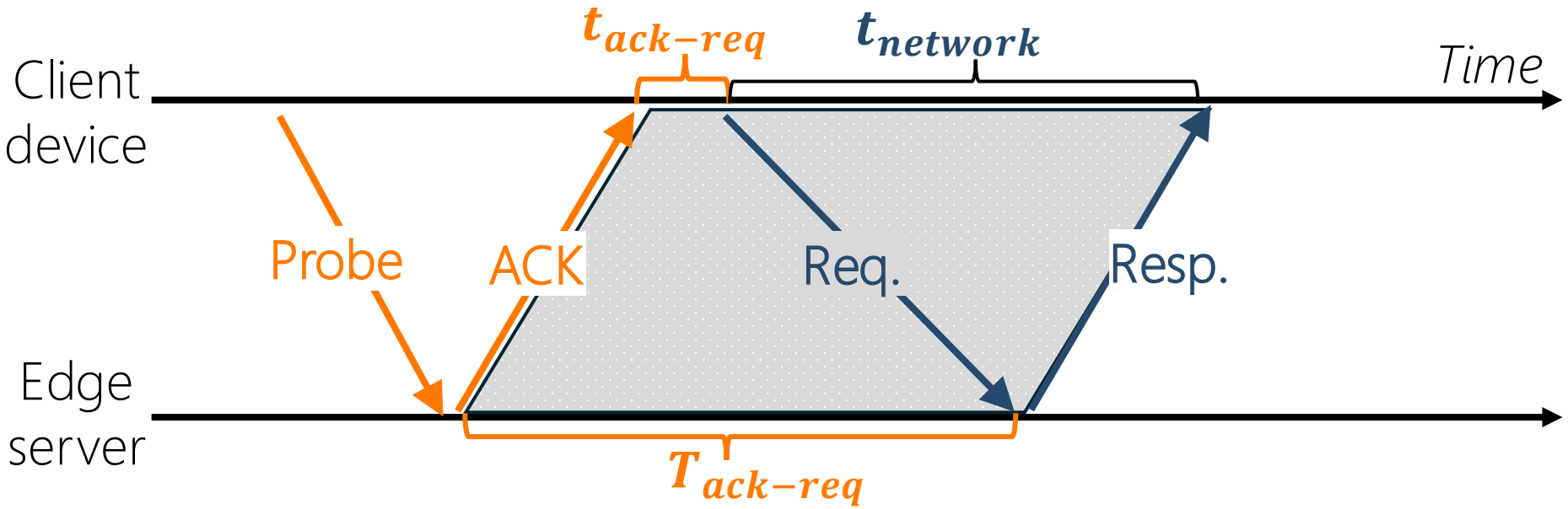


Idea 2: Exploiting downlink stability for network latency estimation at the edge



Key insight: Leverage stable downlink latency as a timing anchor!

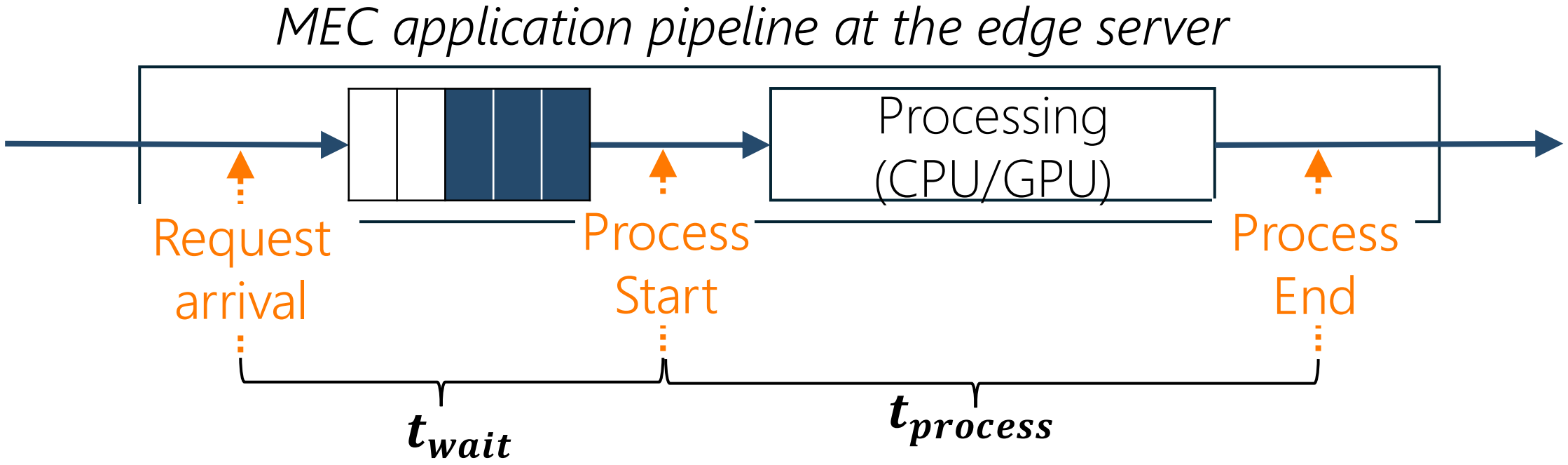
Network latency estimation via periodic probing



$$t_{network} = T_{ack-req} - t_{ack-req}$$

- Low overhead: Probe–ACK exchanges occurring every few seconds
- Applicable to other networks with asymmetric latency, beyond MEC

Idea 3: Leveraging request lifecycle events for processing time estimation

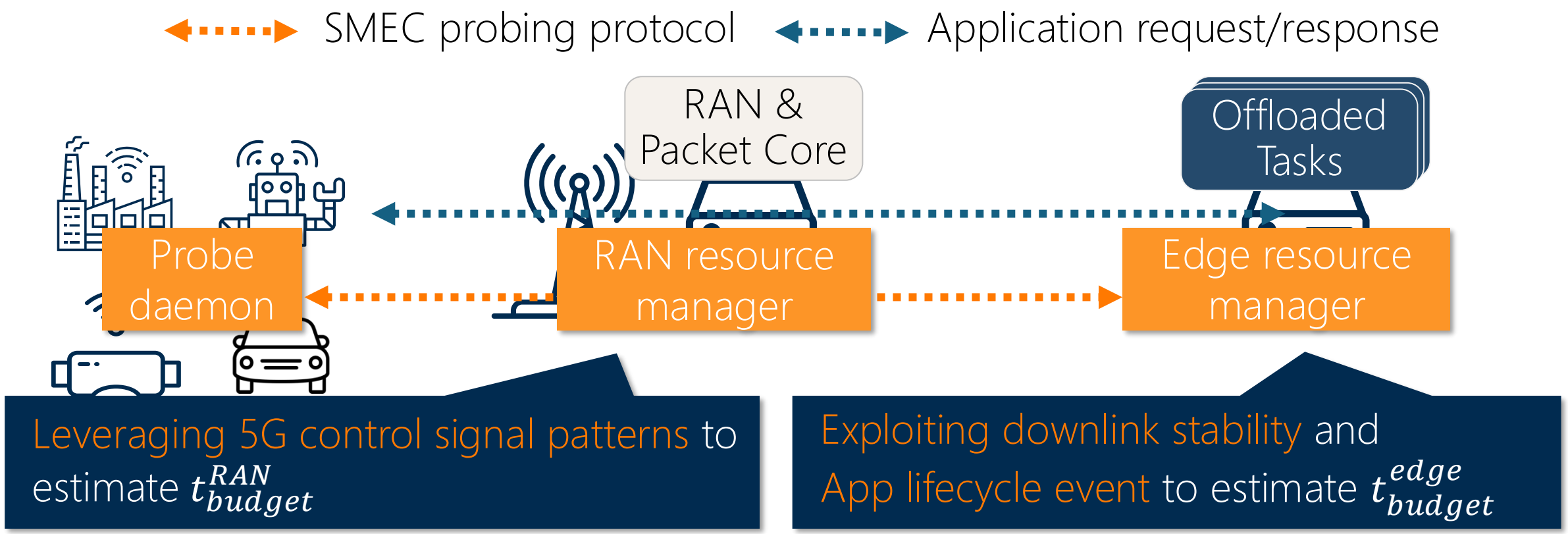


$$t_{process}^{estimate} = median(t_{process}^{0}, t_{process}^{1}, \cdots, t_{process}^{k})$$

$$t_{budget}^{edge} = \text{SLO} - t_{network} - t_{wait} - t_{process}^{estimate}$$

SMEC API to report events without intrusive app changes

Putting it all together



- RAN/Edge resource managers prioritize near-deadline requests with higher scheduling priority
- SMEC realizes SLO-aware MEC with minimal application changes

Evaluation setup

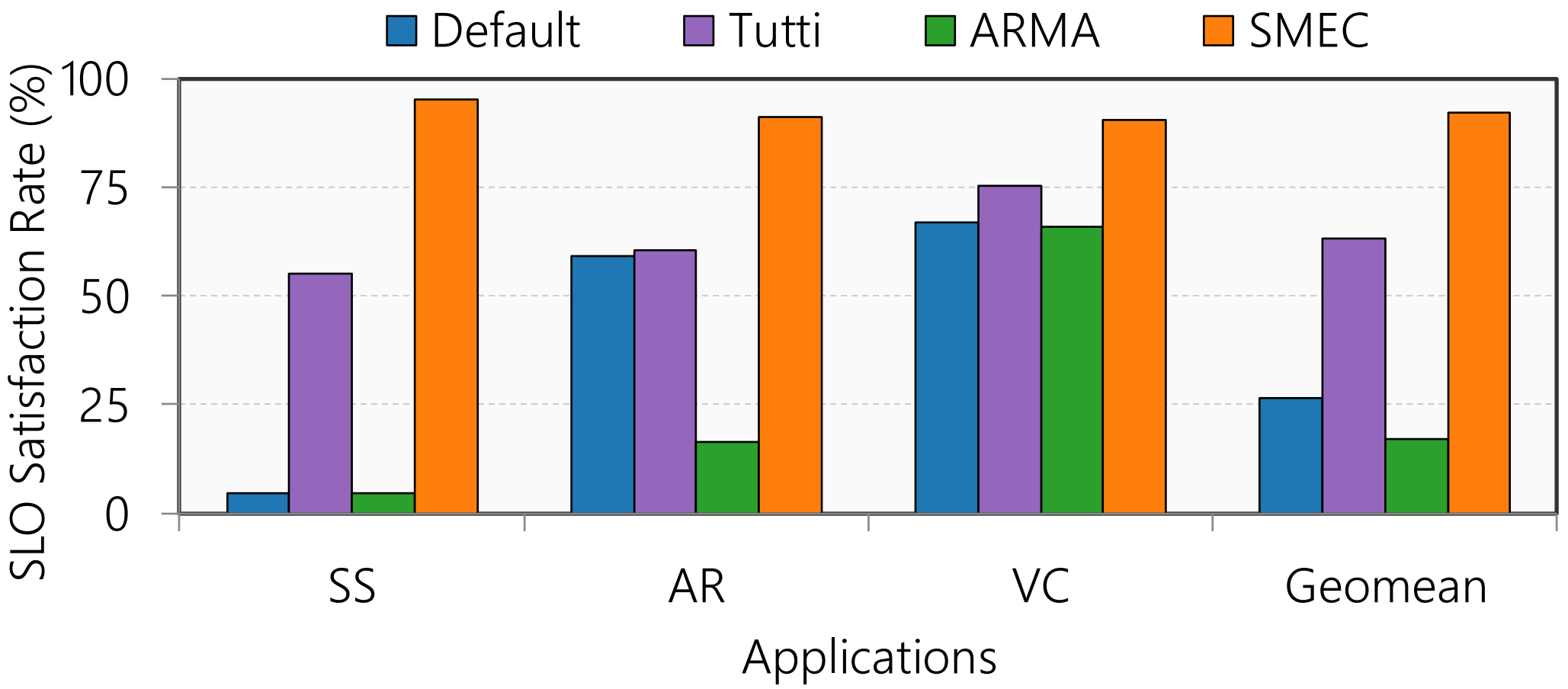
Testbed

- **5G network:** srsRAN 5G stack
- **Edge server:** Intel Xeon CPUs + NVIDIA L4 GPU
- **UE devices:** 12 user devices (AmariSoft UE simbox)

Applications (3 x latency-critical & 1 x best-effort)

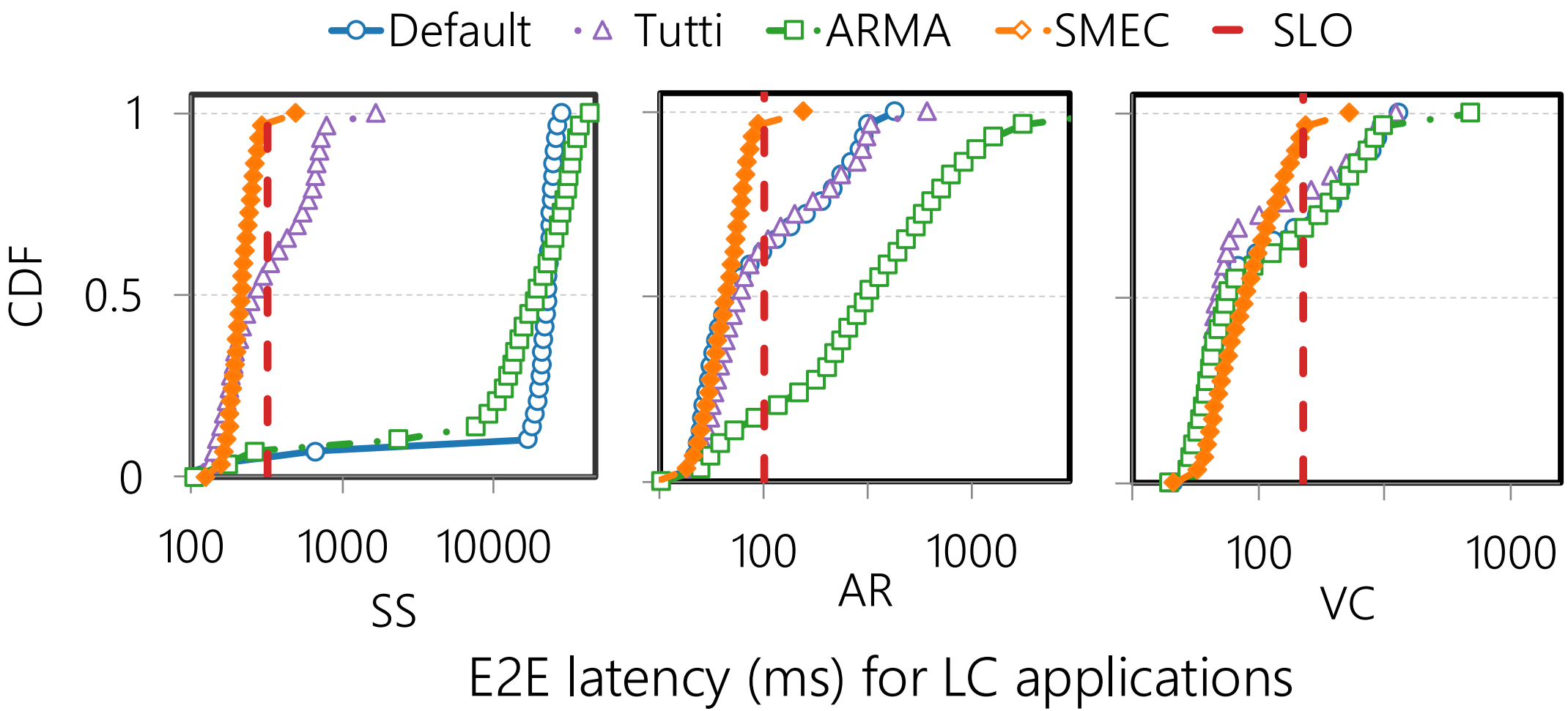
- Smart Stadium (Video transcoding; CPU-heavy),
- Augmented Reality (Object detection; GPU-heavy)
- Video Conferencing (Super-resolution; GPU-heavy)
- File transfer (Best effort)

SMEC achieves consistently high SLO satisfaction



SMEC achieves 90-96% SLO satisfaction rate

SMEC achieves predictable latency



SMEC reduces tail latency by up to 122x → Predictable latency!

Summary

- Today's MEC deployments suffer **from high and unpredictable latency**
- **SMEC:** an SLO-aware resource management framework for MEC
- Achieves **high SLO satisfaction** and **low tail latency**
- Opens new opportunities for **ultra-low-latency edge applications**
  (e.g., remote XR rendering, robotic/vehicle teleoperations)

Check out our paper and GitHub repo!
https://smec-project.github.io

Implementing resource allocation policies

Operators can implement deadline-aware policy using t_{budget}^{RAN} and t_{budget}^{edge}

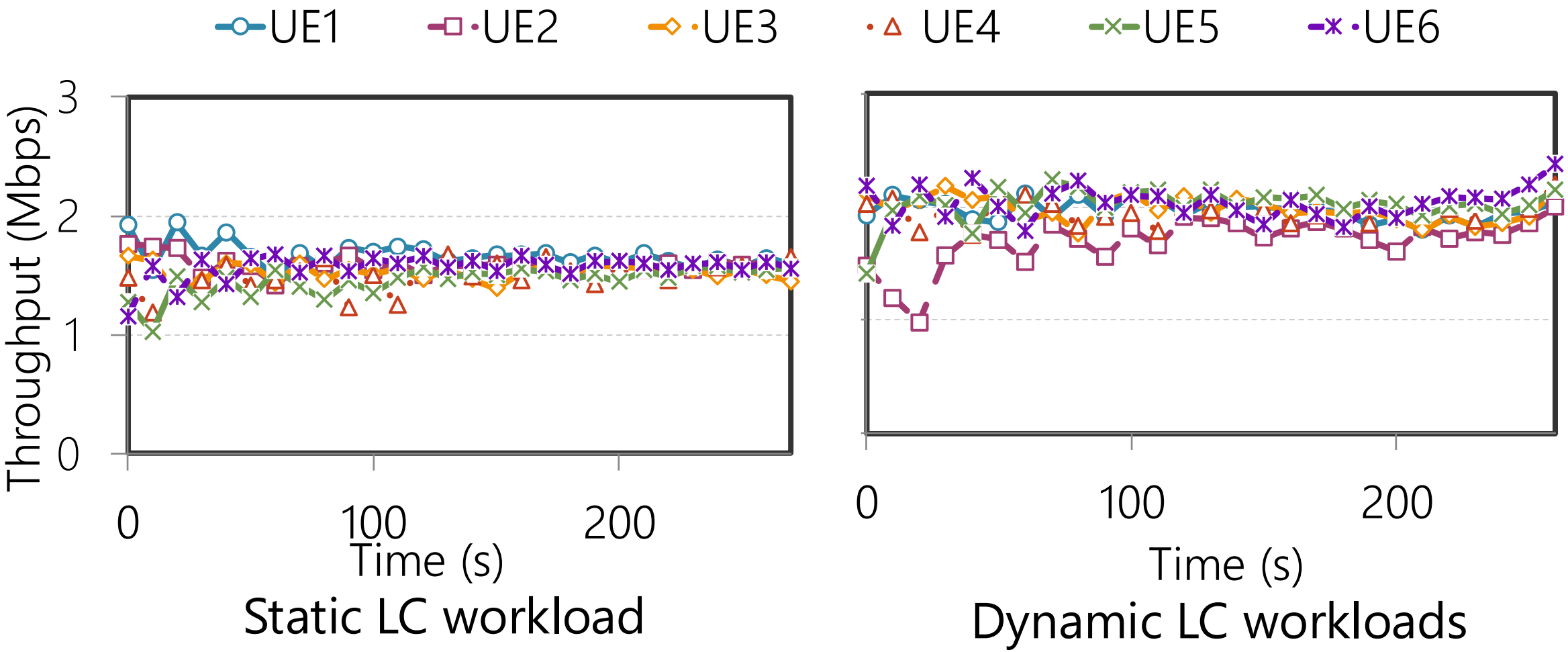
RAN resource manager uses t_{budget}^{RAN}

- RAN traffic consists of latency-critical (LC) and best-effort (BE) requests
- Among LC requests: Prioritize requests with smaller remaining time budget
- Between LC and BE requests: Prioritize LC requests over BE requests

Edge resource manager uses t_{budget}^{edge}

- CPU Scheduling: Allocates an additional core to apps with urgent requests
- GPU scheduling: Increases the priority-level for urgent requests
- Early drop: Discards requests with overly tight deadlines

SMEC does not starve best effort applications



BE applications are not starved for bandwidth and share it fairly