THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Edge-RT: OS Support for Controlled Latency in the Multi-Tenant, Real-Time Edge

Wenyuan Shao, Bite Ye, Huachuan Wang, Gabriel Parmer and Yuxin Ren
{shaowy, bitye, hcwang, gparmer, ryx}@gwu.edu

Interacting with the Cloud

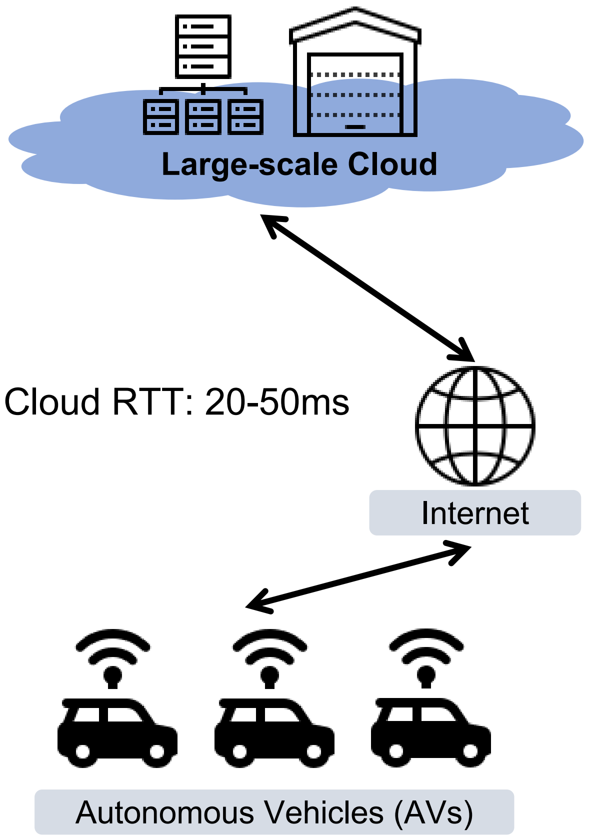
Main purpose of interacting with cloud:

1. Offload computations to the cloud.
2. Aggregate data from multiple clients.

Interacting with the Cloud

Main purpose of interacting with cloud:

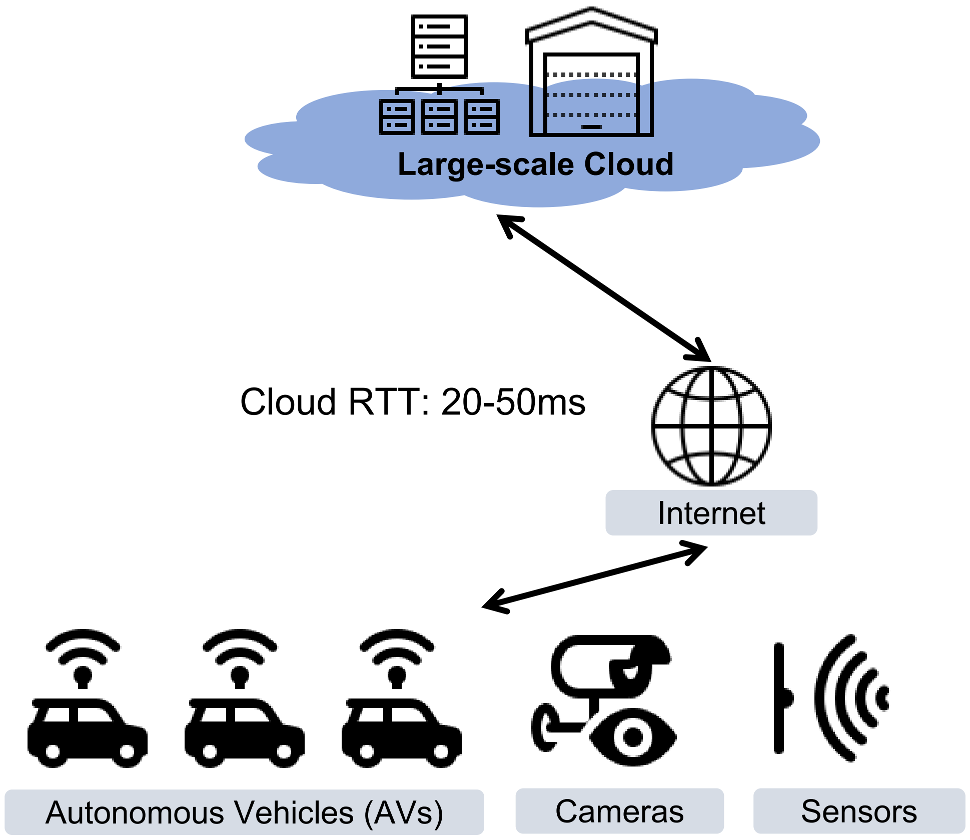
1. **Offload computations to the cloud.**
2. Aggregate data from multiple clients.



Interacting with the Cloud

Main purpose of interacting with cloud:

1. Offload computations to the cloud.
2. **Aggregate data from multiple clients.**

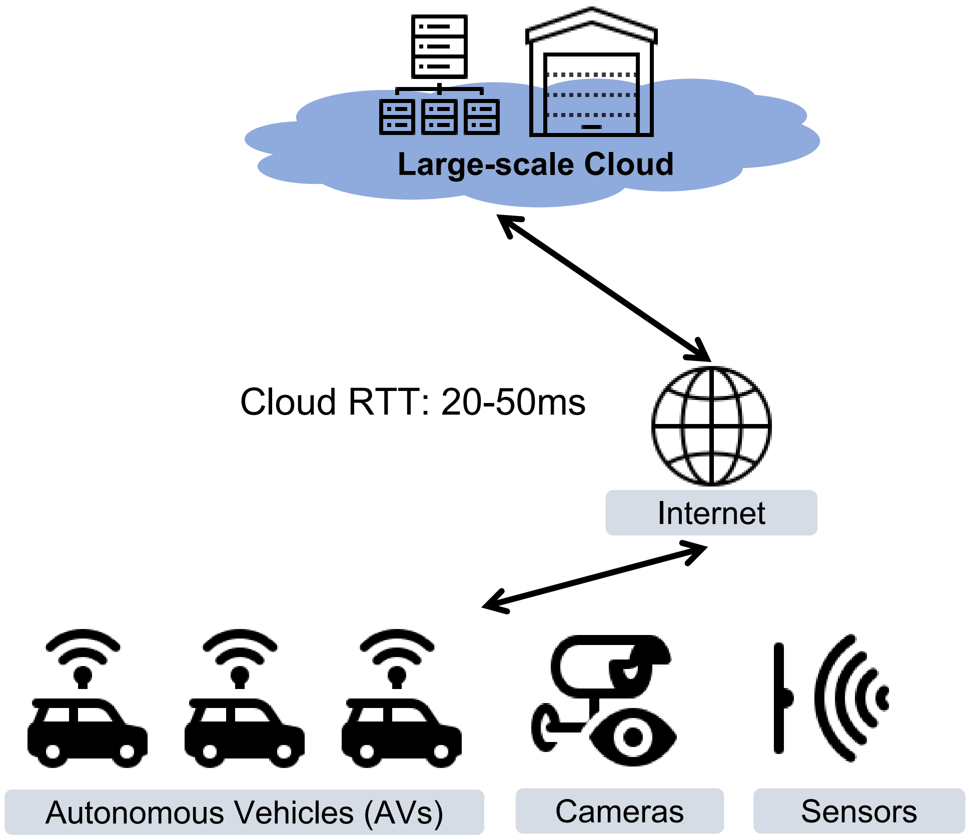


Interacting with the Cloud

Main purpose of interacting with cloud:

1. Offload computations to the cloud.
2. Aggregate data from multiple clients.

Problem:
For both examples, multiple clients require real-time responses from the cloud.



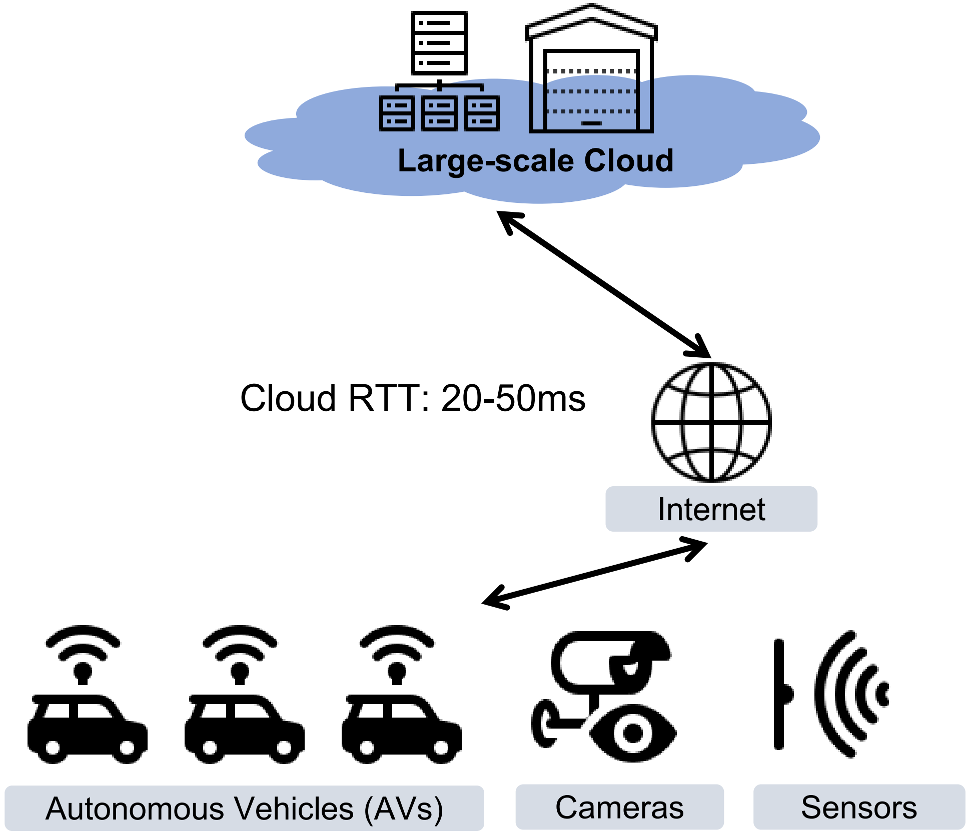
Interacting with the Cloud

Main purpose of interacting with cloud

1. Offload computations to the cloud.
2. Aggregate data from multiple clients.

Problem:
For both examples, multiple clients require real-time responses from the cloud.

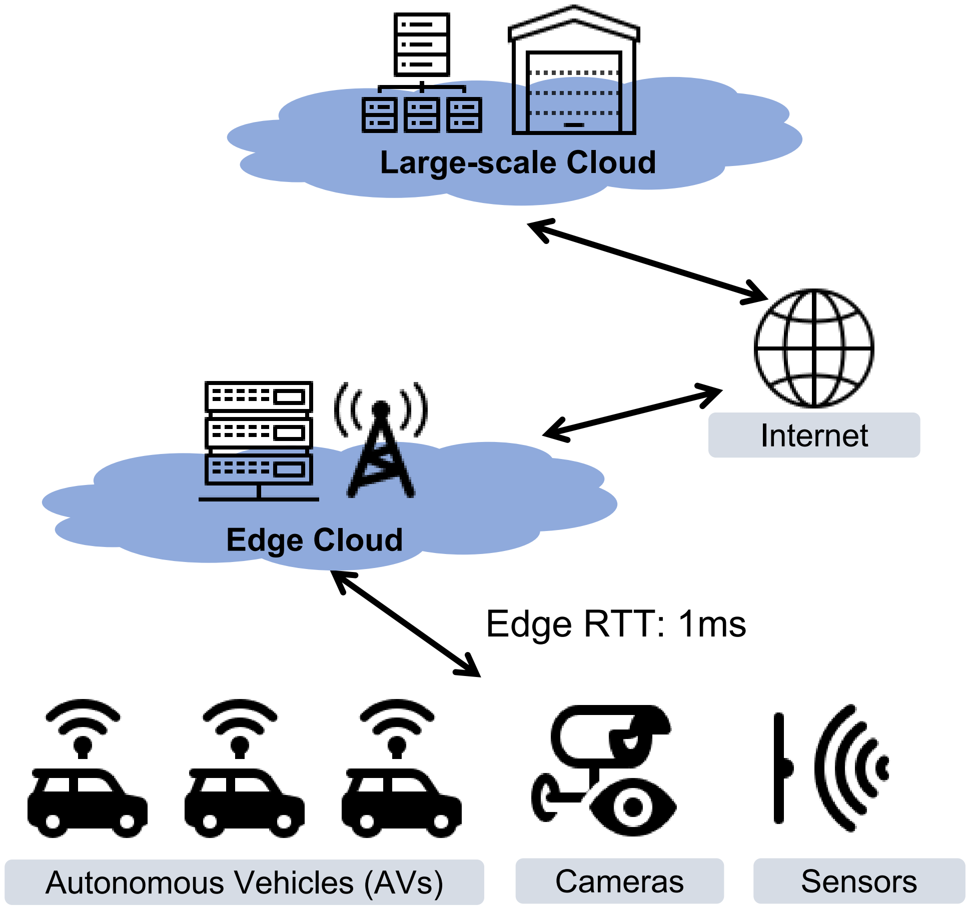
Core Challenges:
1. Huge and unpredictable latency.
2. WAN bandwidth is running out.



Interacting with the Edge

What is Edge cloud?

Why Edge cloud?

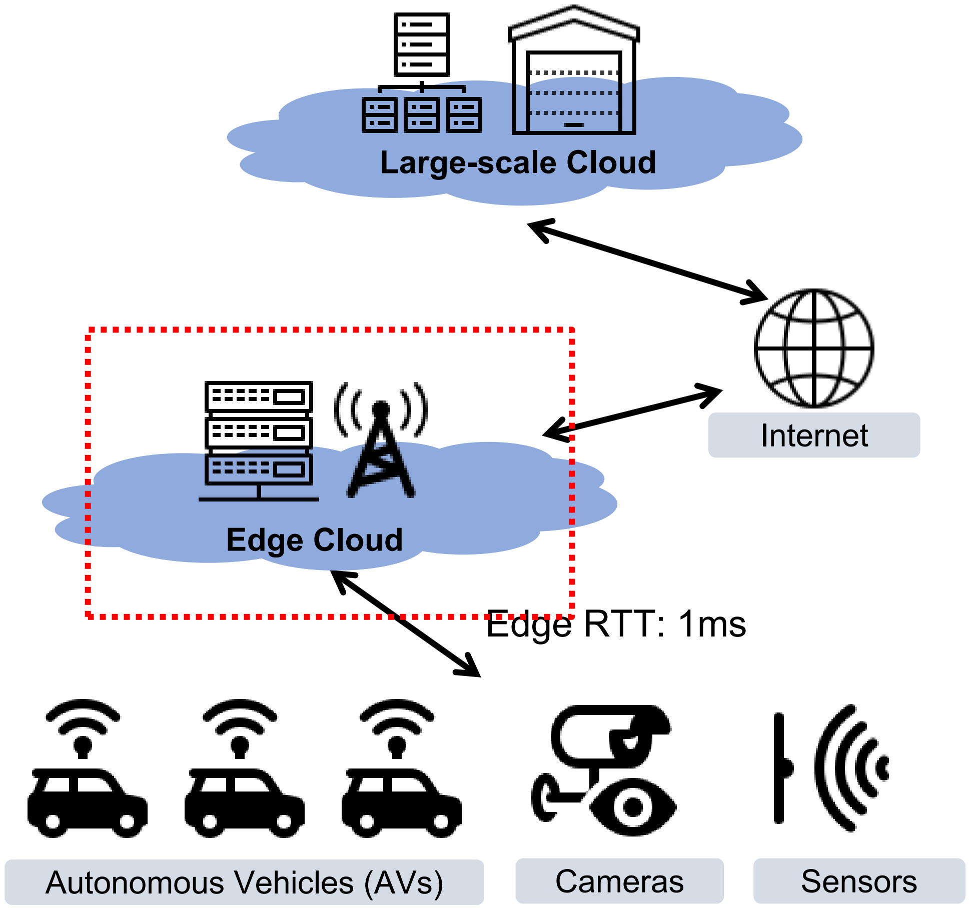


Interacting with Edge

What is Edge cloud?

1. Compute assets co-located with client devices.
2. Scale: few servers up-to a rack of servers.

Why Edge cloud?

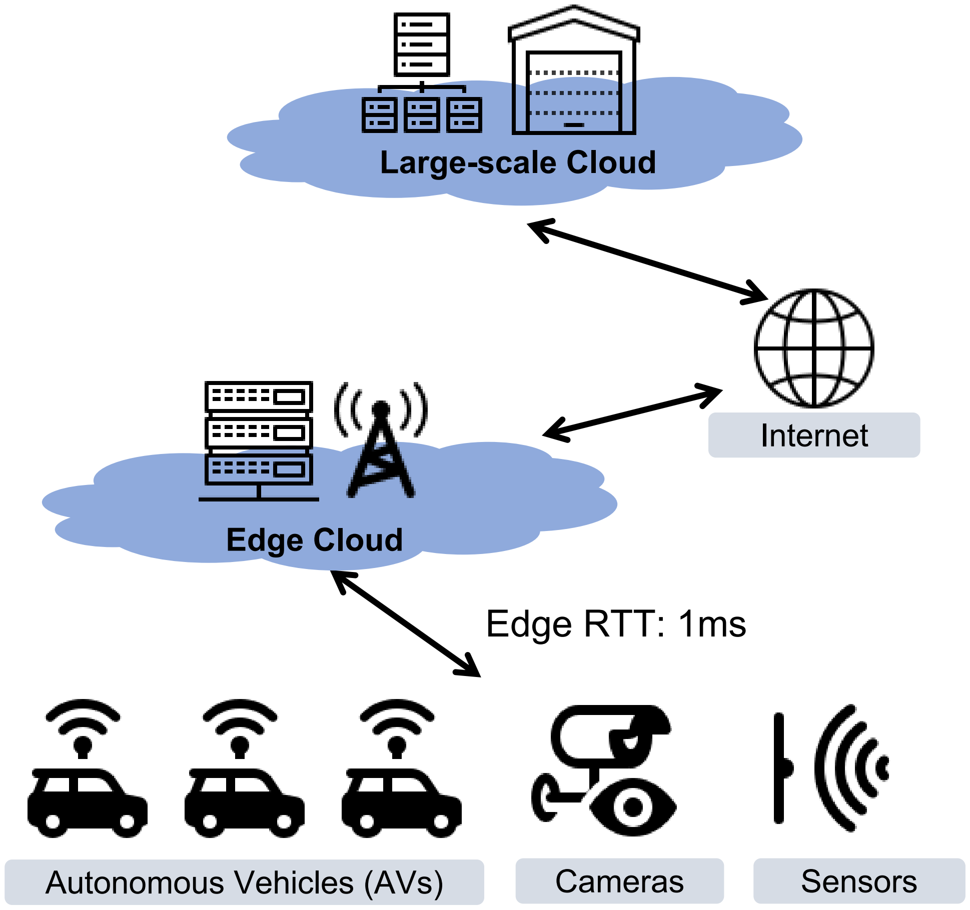


Interacting with Edge

What is Edge cloud?

1. Compute assets co-located with client devices.
2. Scale: few servers up-to a rack of servers.

Why Edge cloud?



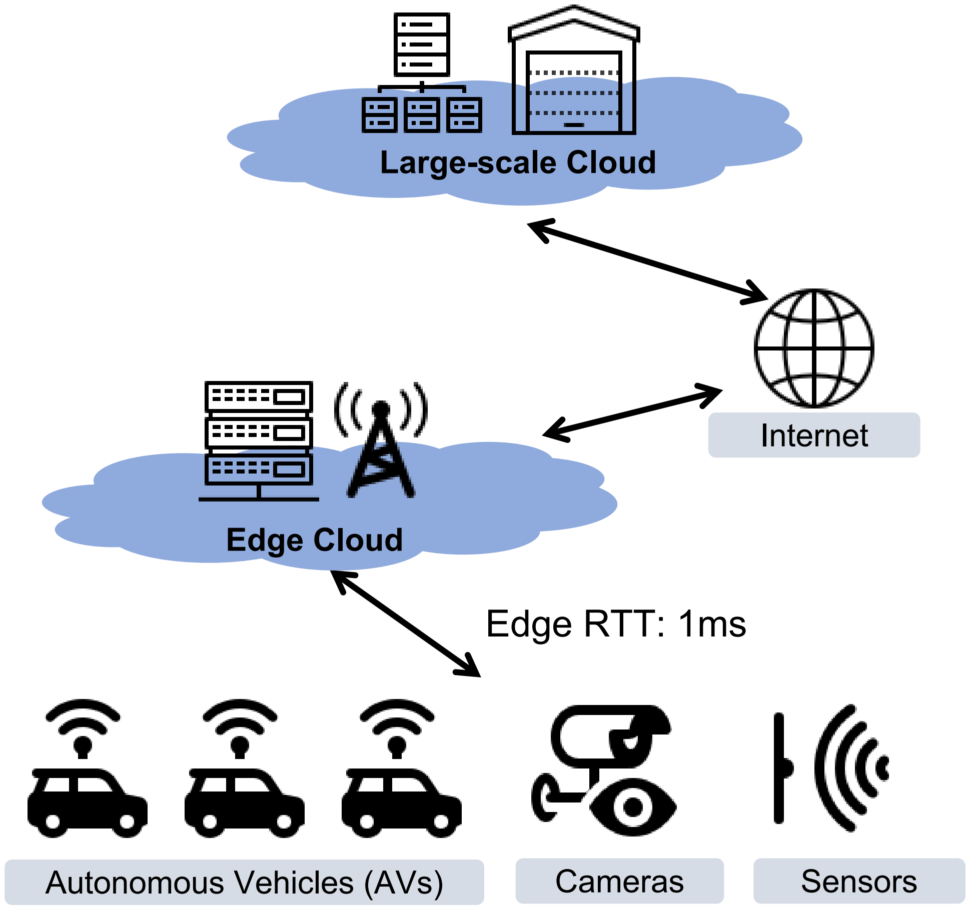
Interacting with the Edge

What is Edge cloud?

1. Compute assets co-located with client devices.
2. Scale: few servers up-to a rack of servers.

Why Edge cloud?

Challenges of the Cloud:
1. Huge and unpredictable latency.
2. WAN bandwidth is running out.



THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Interacting with the Edge

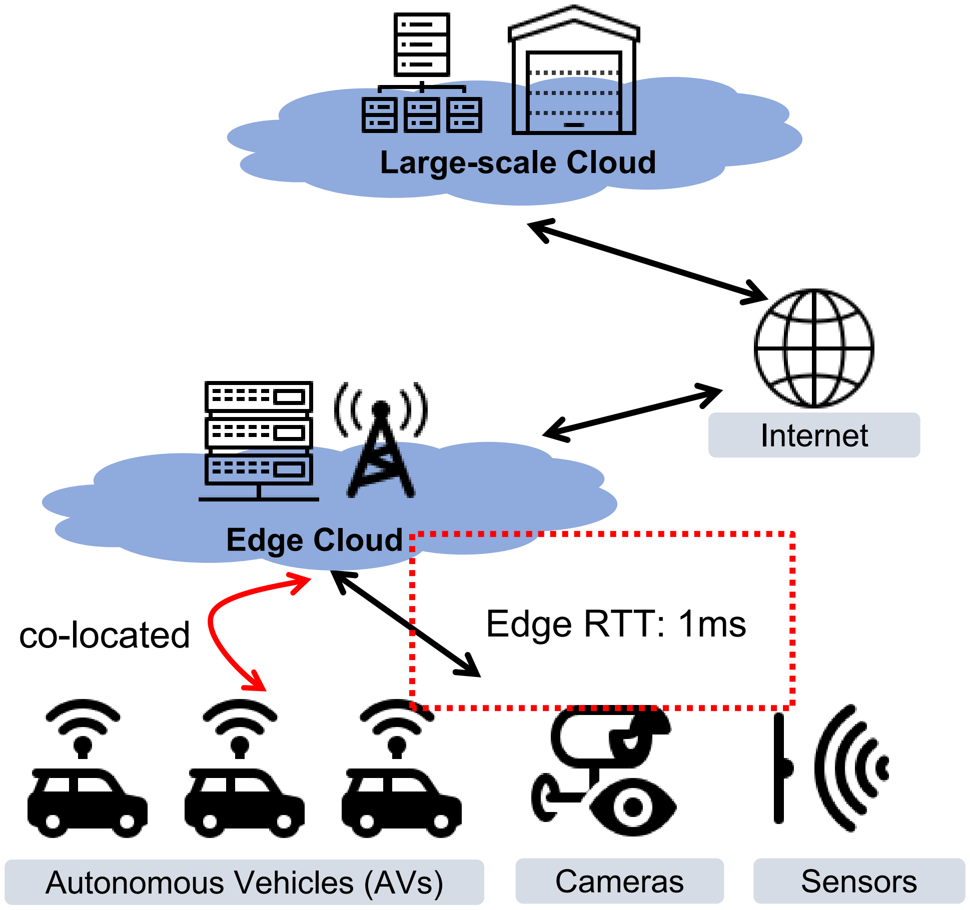
What is Edge cloud?

1. Compute assets co-located with client devices.
2. Scale: few servers up-to a rack of servers.

Why Edge cloud?

Challenges of the Cloud:
1. **Huge and unpredictable latency.**
2. WAN bandwidth is running out.

Edge-Cloud Solutions:
1. **1ms RTT with 5G.**
2. Using local bandwidth.



THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Interacting with the Edge

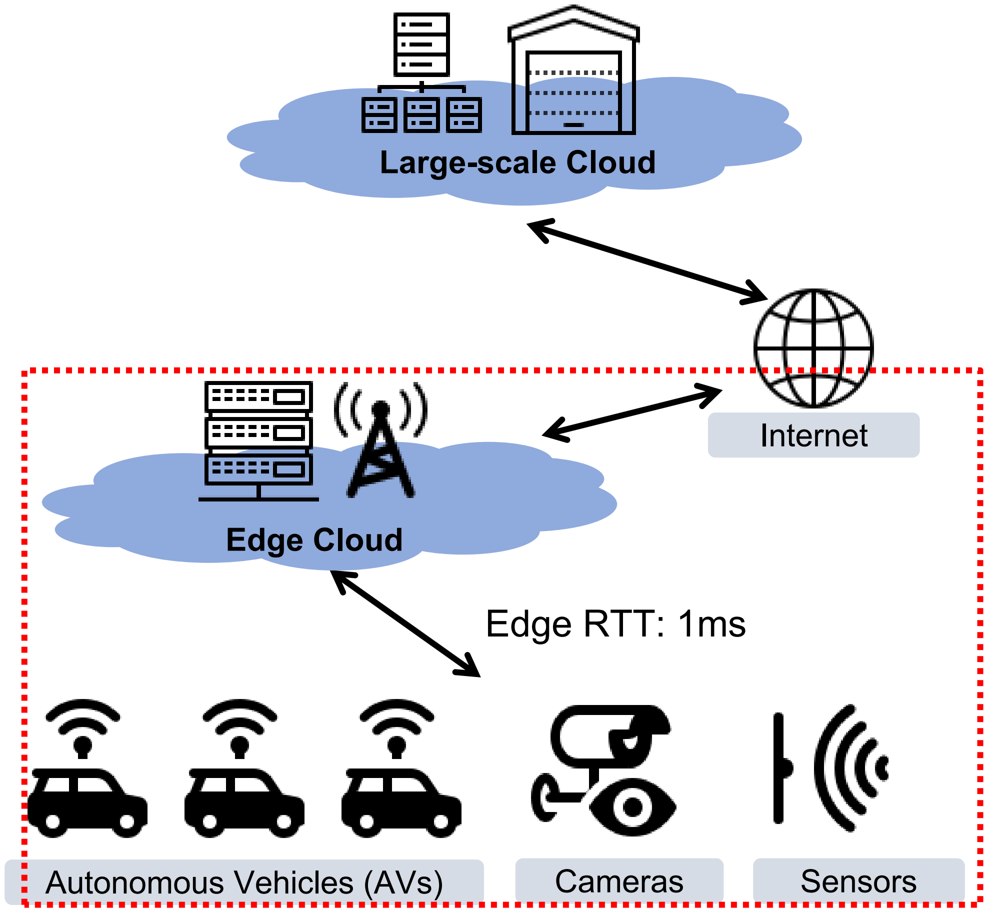
What is Edge cloud?

1. Compute assets co-located with client devices.
2. Scale: few servers up-to a rack of servers.

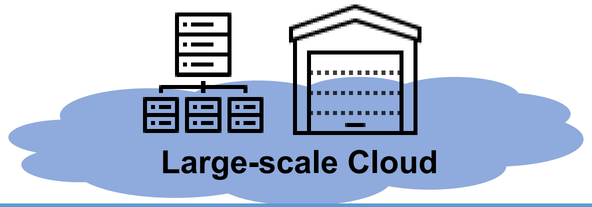
Why Edge cloud?

Challenges of the Cloud:
1. Huge and unpredictable latency.
2. **WAN bandwidth is running out.**

Edge-Cloud Solutions:
1. 1ms RTT with 5G.
2. **Using local bandwidth.**



Interacting with the Edge



What is Edge cloud?

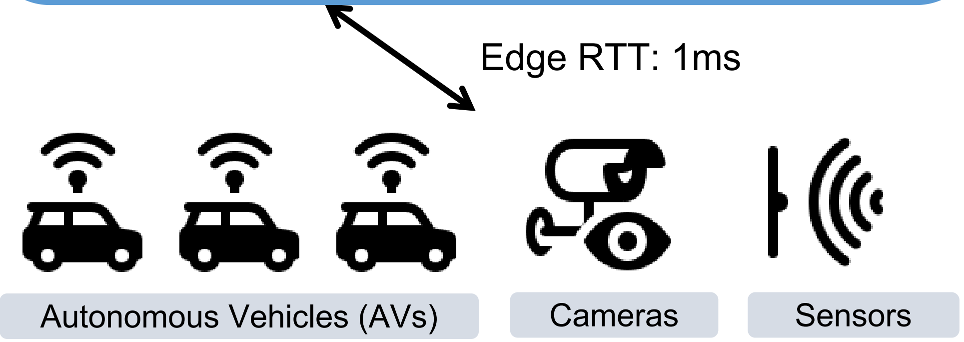
1. Compute assets co-located with client devices.
2. Scale: few servers up-to a rack of servers.

Why Edge cloud?

Challenges of the Cloud:
1. Huge and unpredictable latency.
2. WAN bandwidth is running out.

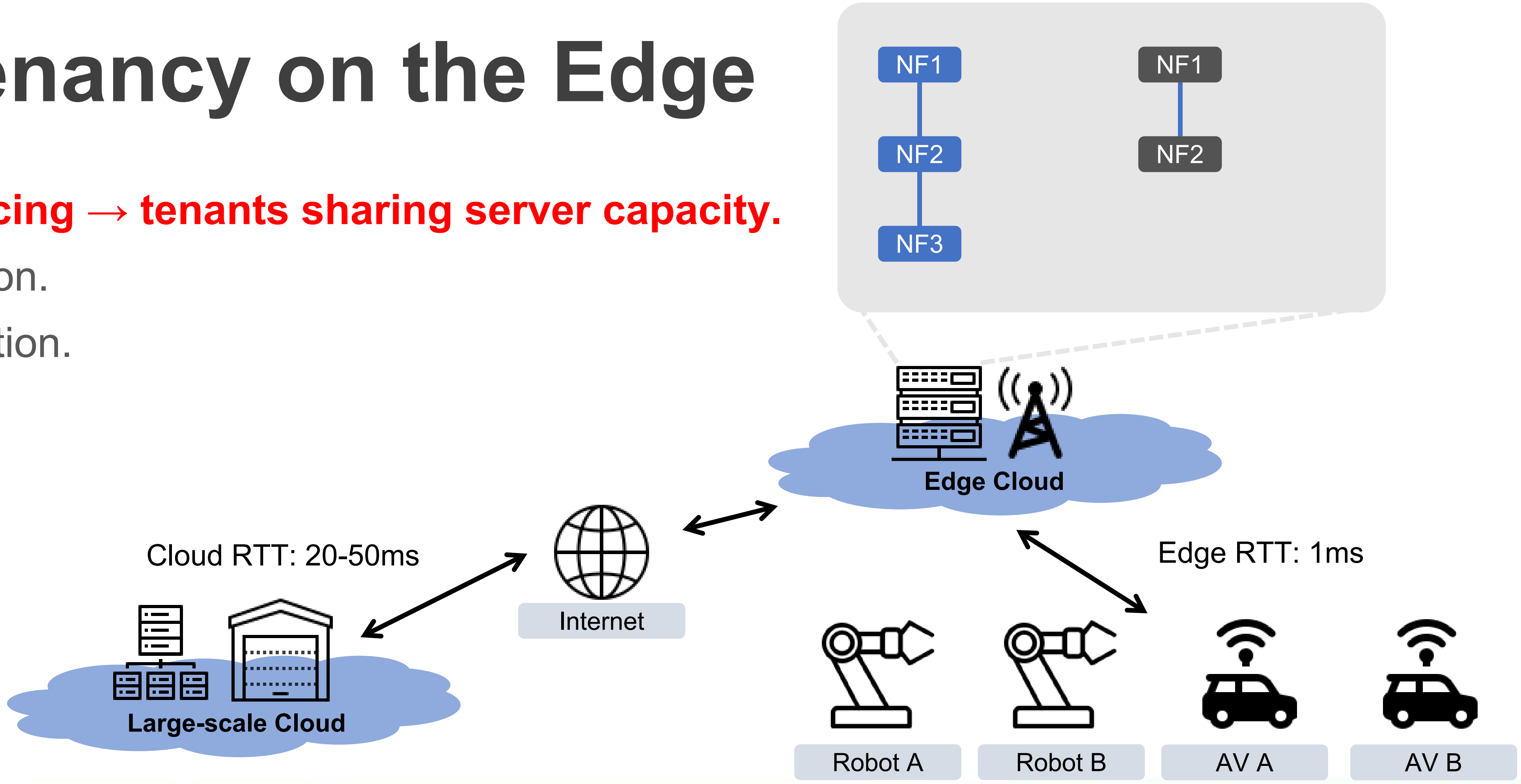
Edge-Cloud Solutions:
1. 1ms RTT with 5G.
2. Using local bandwidth.

Core Challenges of the Edge:
1. **Multi-tenancy.**
2. High density dynamic workload.
3. Deadline-aware.



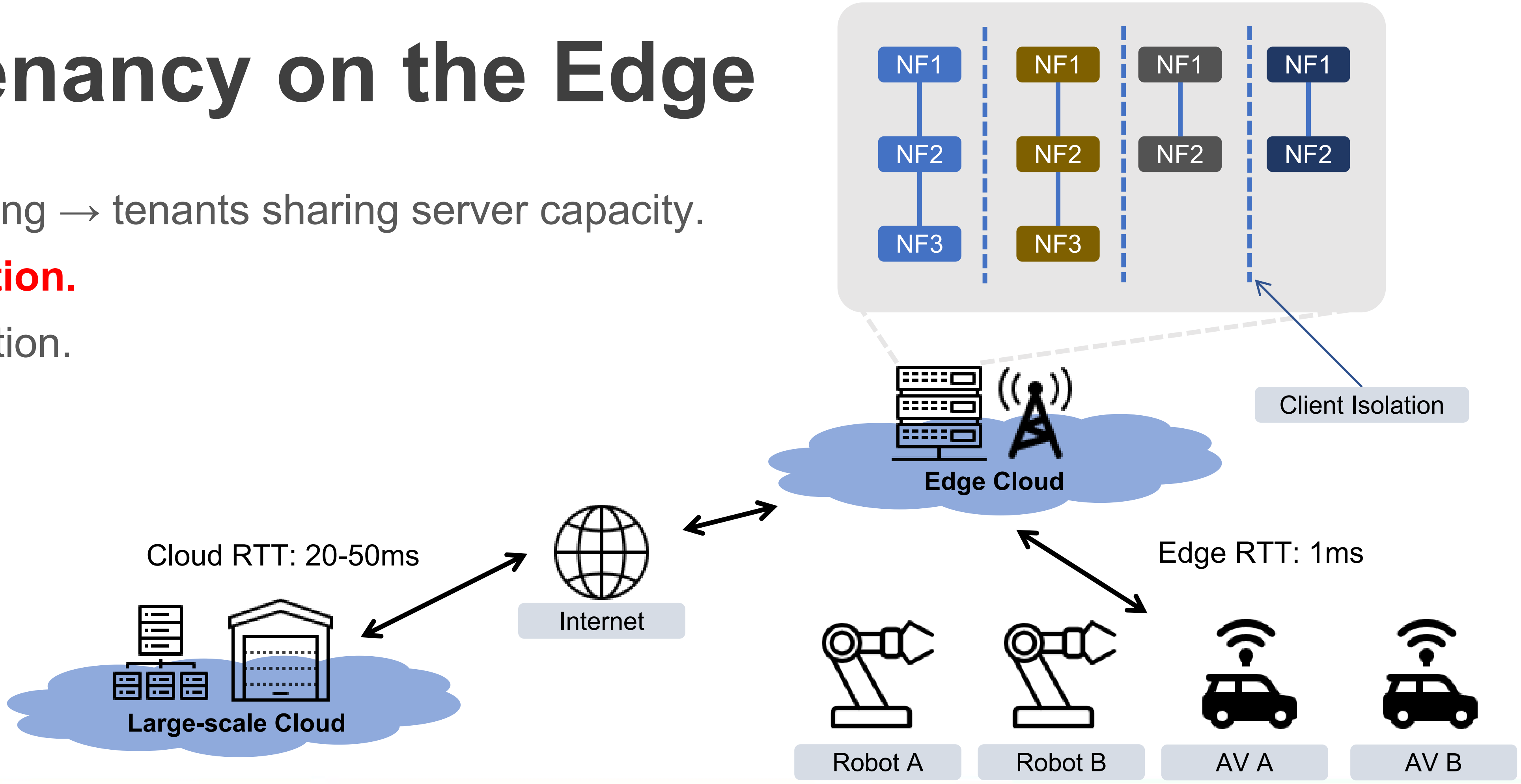
Multi-tenancy on the Edge

- **Network slicing → tenants sharing server capacity.**
- Client isolation.
- Tenant isolation.



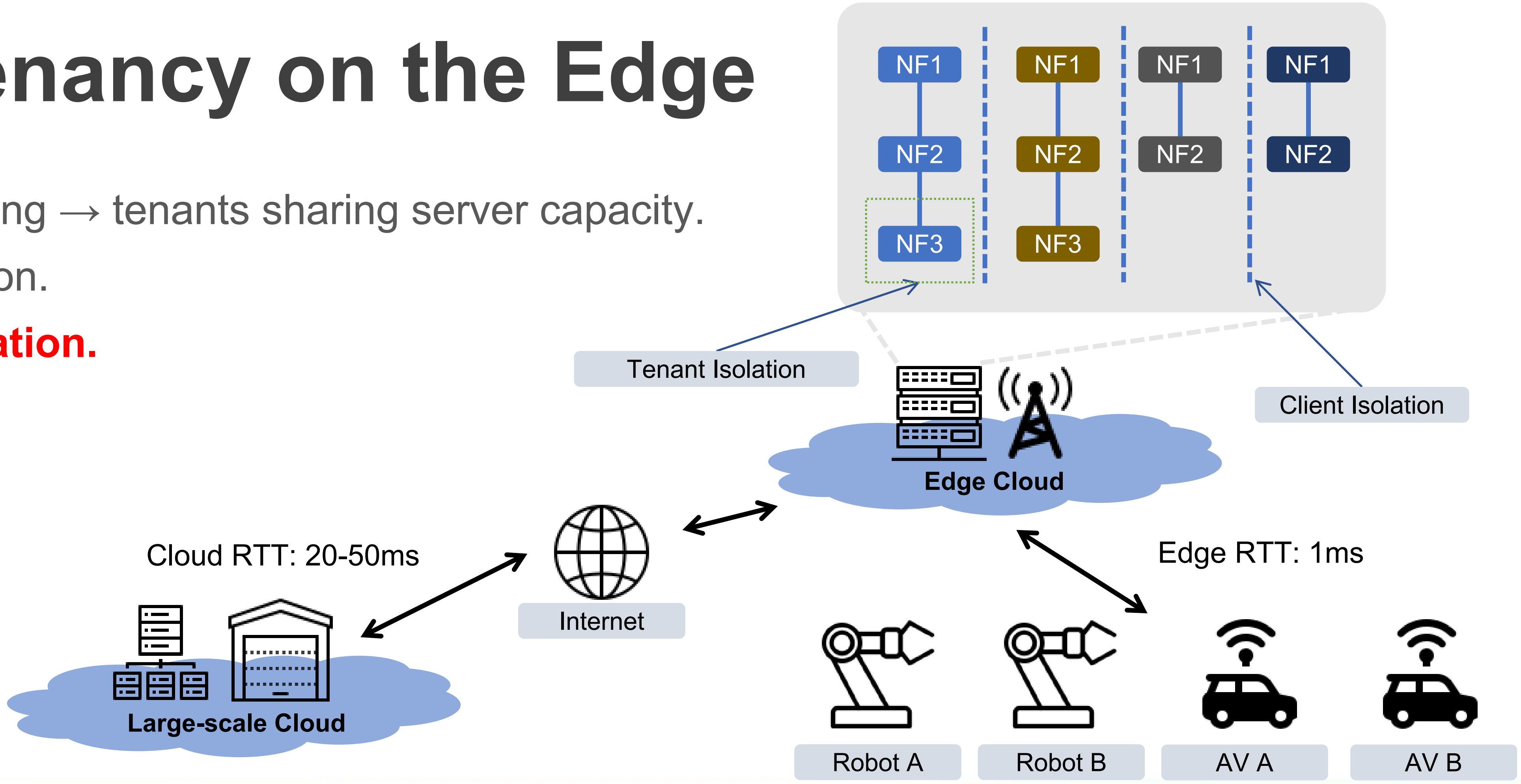
Multi-tenancy on the Edge

- Network slicing → tenants sharing server capacity.
- **Client isolation.**
- Tenant isolation.



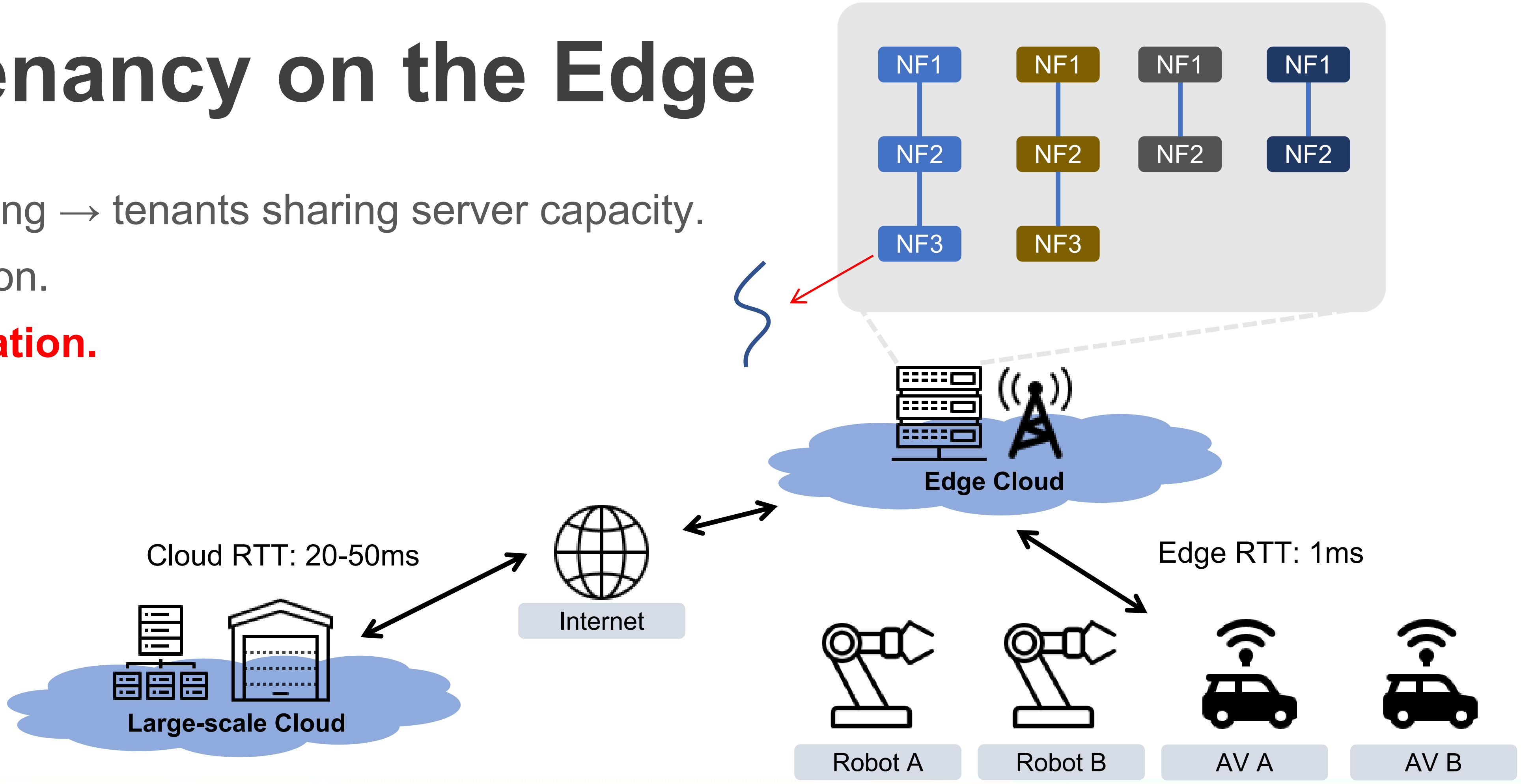
Multi-tenancy on the Edge

- Network slicing → tenants sharing server capacity.
- Client isolation.
- **Tenant isolation.**

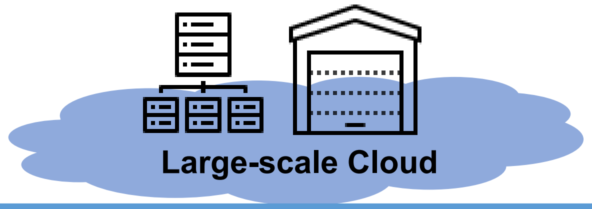


Multi-tenancy on the Edge

- Network slicing → tenants sharing server capacity.
- Client isolation.
- **Tenant isolation.**



Interacting with the Edge



What is Edge cloud?

1. Compute assets co-located with client devices.
2. Scale: few servers up-to a rack of servers.

Why Edge cloud?

Challenges of the Cloud:

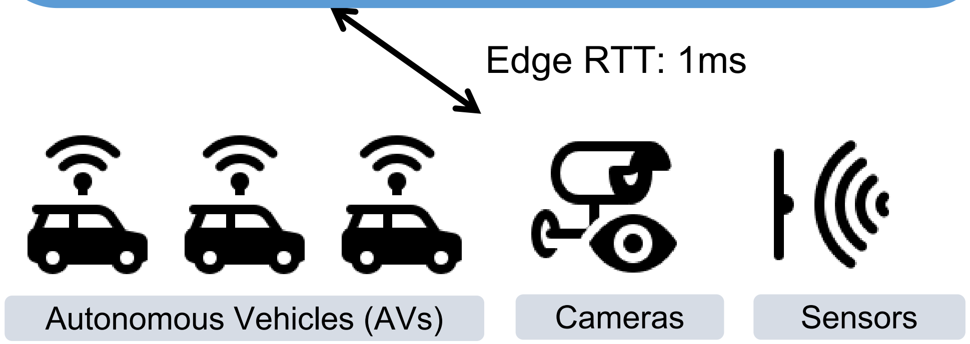
1. Huge and unpredictable latency.
2. WAN bandwidth is running out.

Edge-Cloud Solutions:

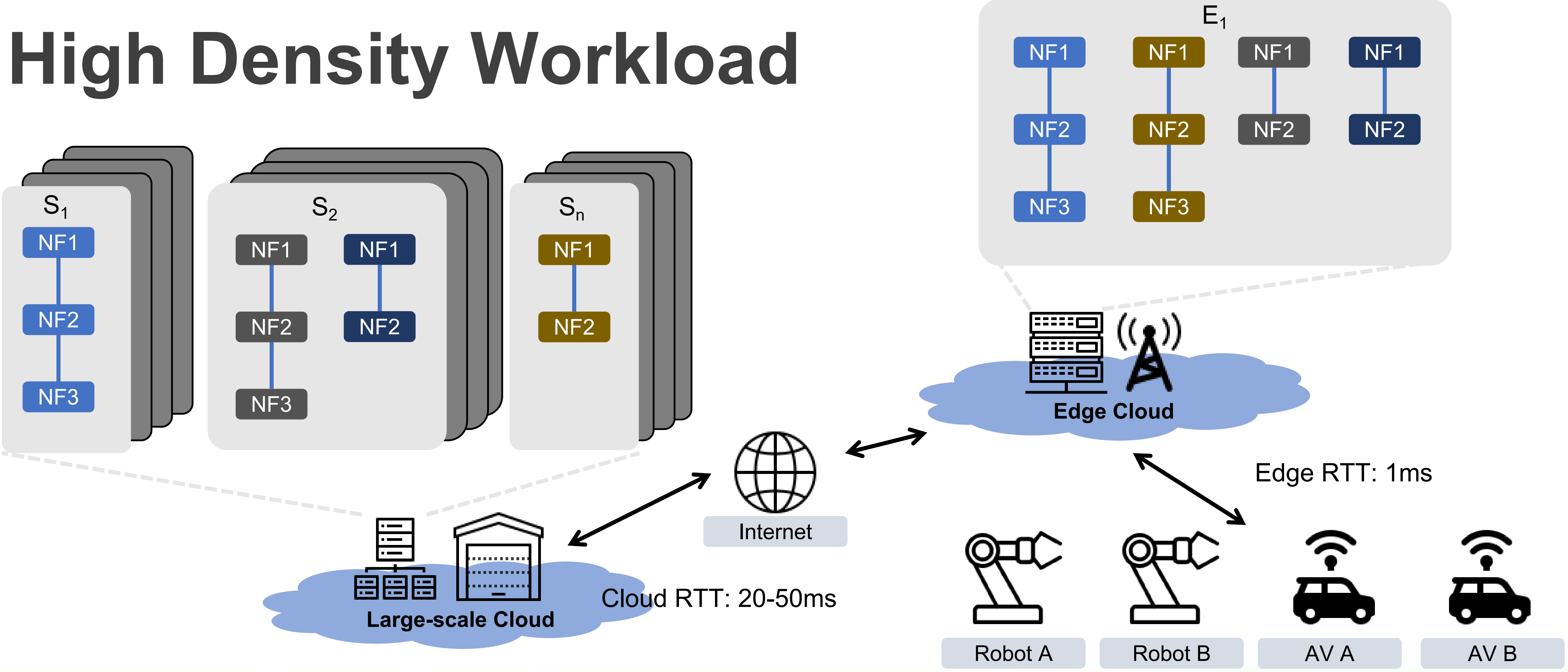
1. 1ms RTT with 5G.
2. Using local bandwidth.

Core Challenges of the Edge:

1. Multi-tenancy.
2. **High density dynamic workload.**
3. Deadline-aware.

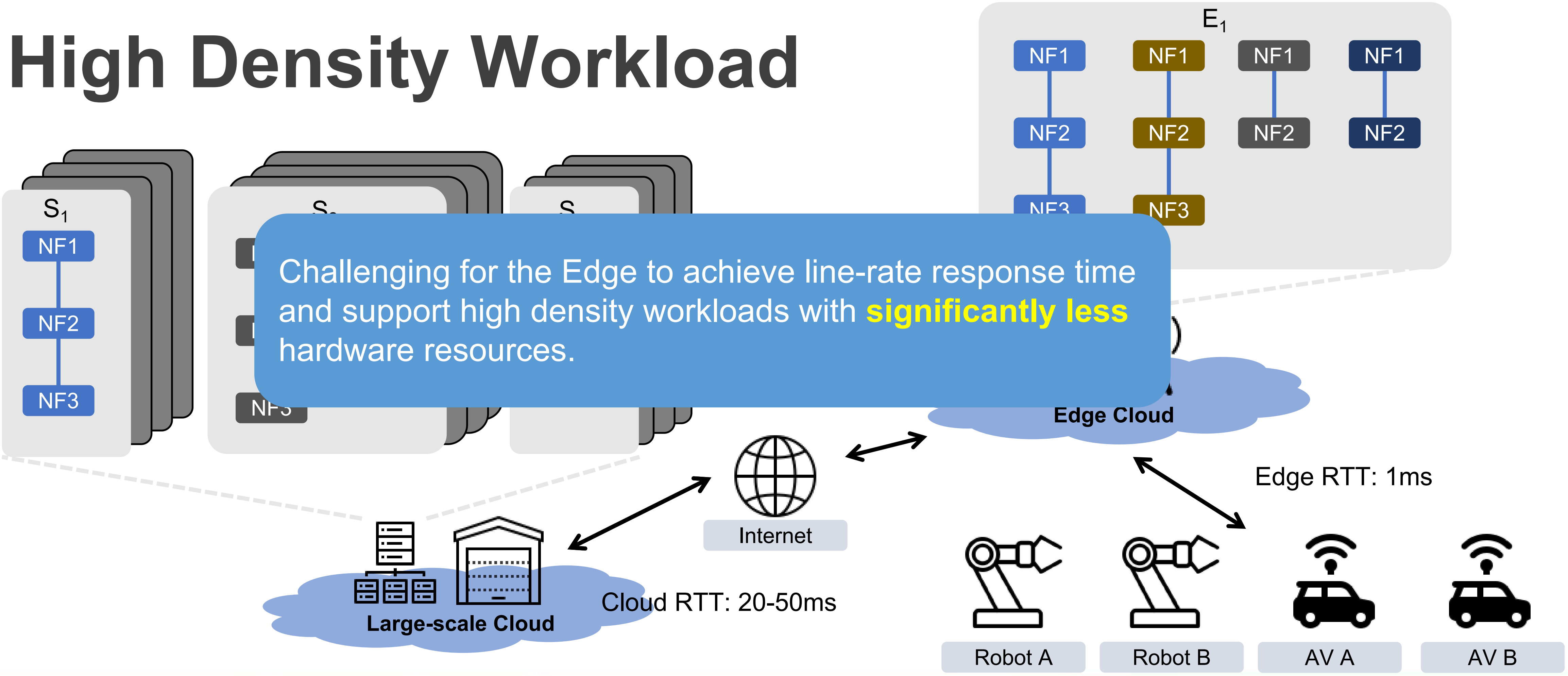


High Density Workload

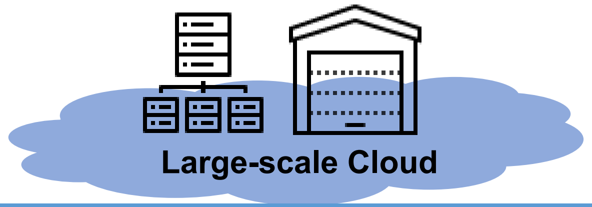


High Density Workload

Challenging for the Edge to achieve line-rate response time and support high density workloads with **significantly less** hardware resources.



Interacting with the Edge



What is Edge cloud?

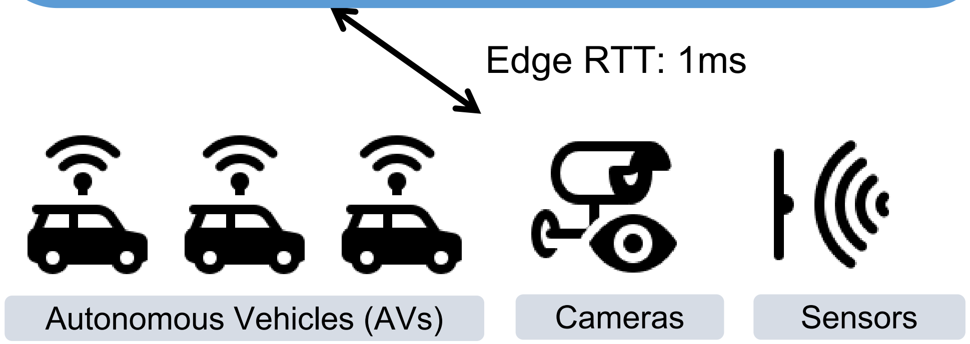
1. Compute assets co-located with client devices.
2. Scale: few servers up-to a rack of servers.

Why Edge cloud?

Challenges of the Cloud:
1. Huge and unpredictable latency.
2. WAN bandwidth is running out.

Edge-Cloud Solutions:
1. 1ms RTT with 5G.
2. Using local bandwidth.

Core Challenges of the Edge:
1. Multi-tenancy.
2. High density dynamic workload.
3. **Deadline-aware.**

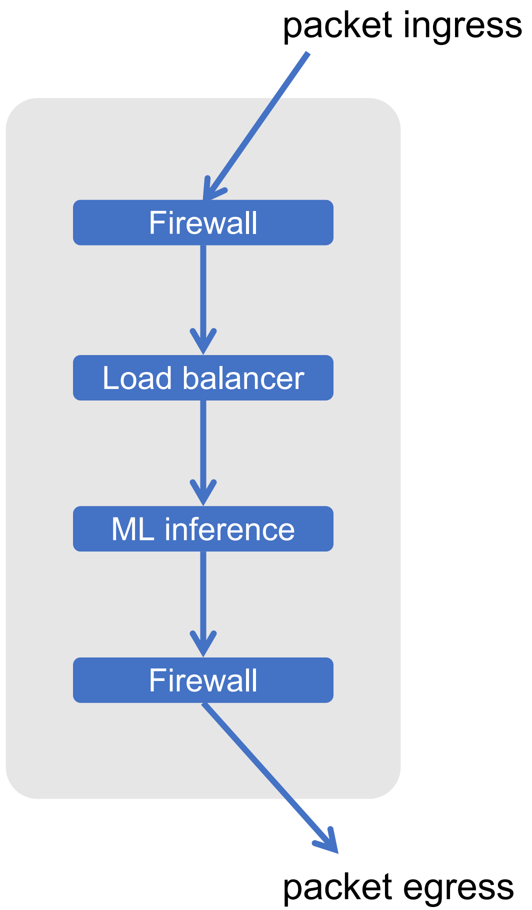


Deadline-aware Scheduling

Why challenging?

1. Chains of computations span cores.

2. Must meet end-to-end deadline of packets.

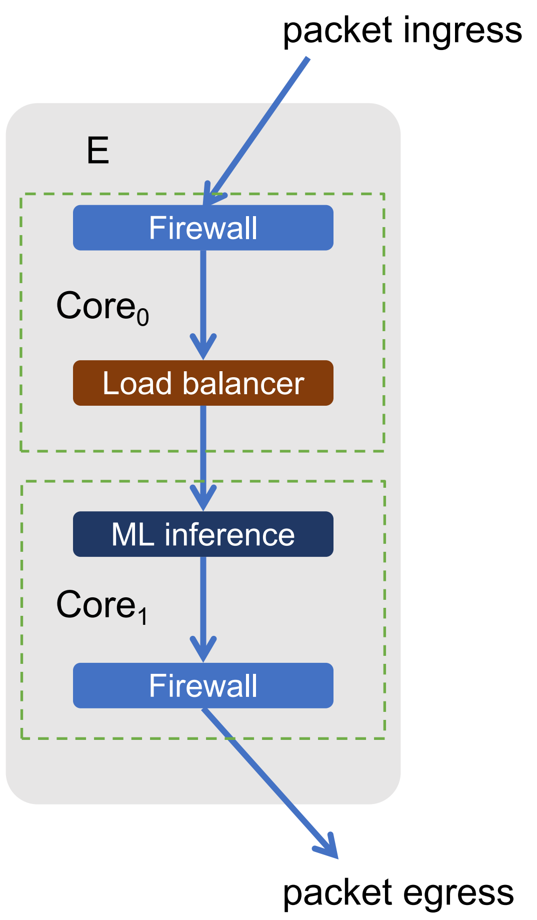


Deadline-aware Scheduling

Why challenging?

1. Chains of computations span cores.

2. Must meet end-to-end deadline of packets.

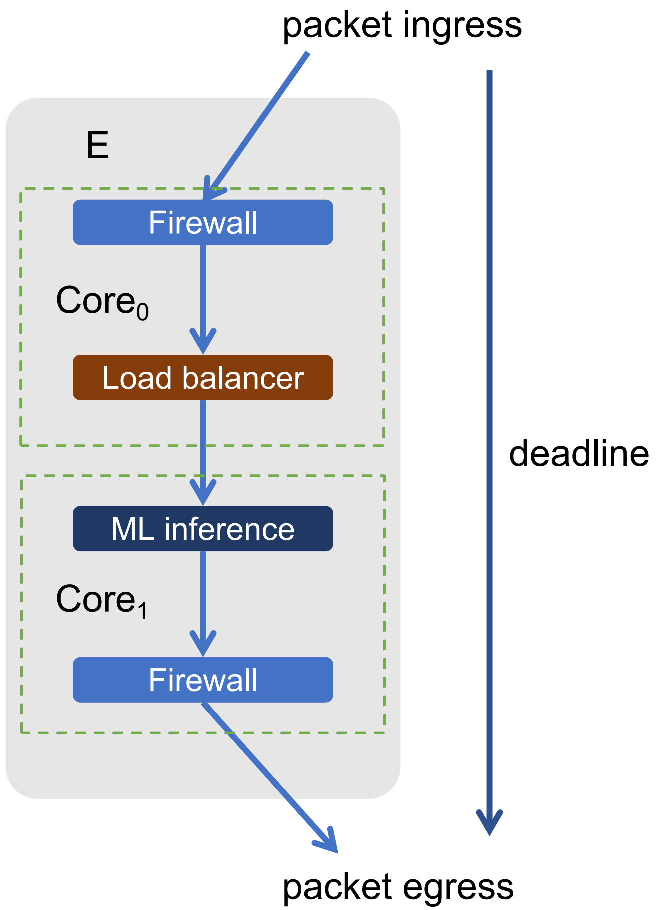


Deadline-aware Scheduling

Why challenging?

1. Chains of computations span cores.

2. Needs to meet end-to-end deadline of packets.



Existing Technologies

1. Optimize throughput by reducing system overhead.
    - Kernel bypass networking.
    - In-kernel Sandbox.
2. Thread-based deadline-aware scheduling.

Existing Technologies

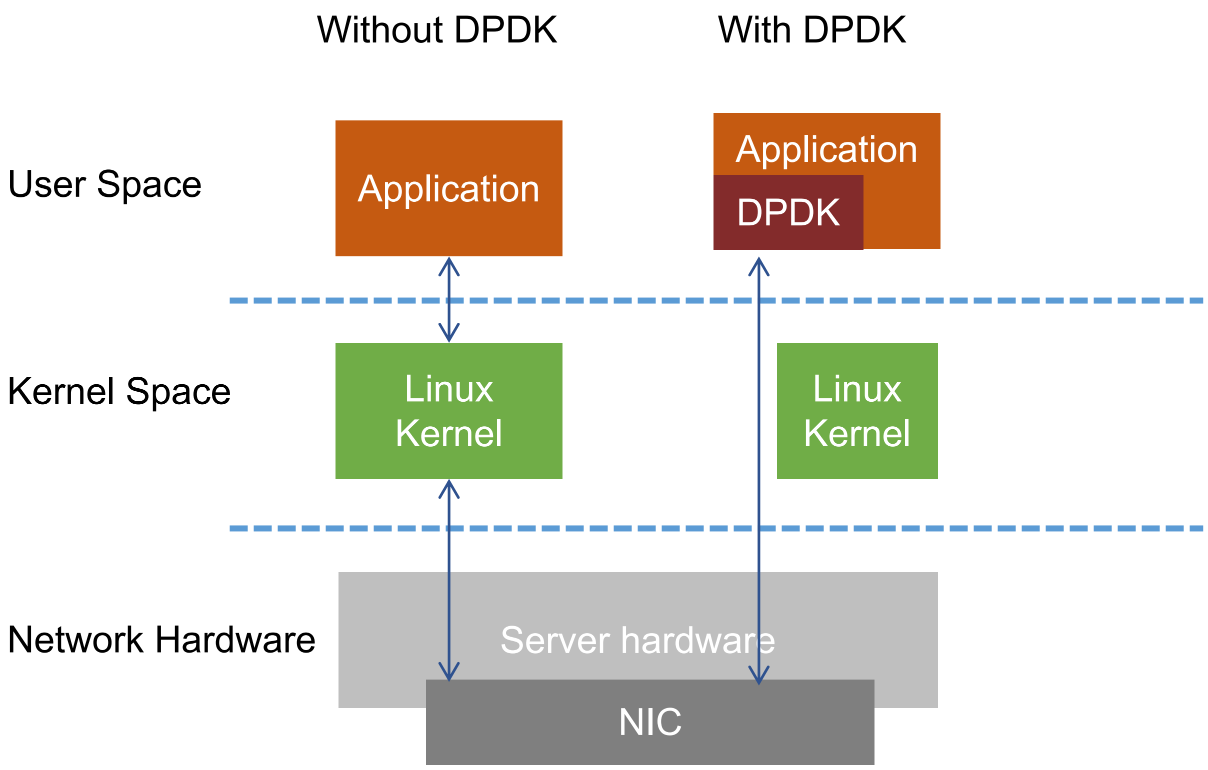
1. Optimize throughput by reducing system overhead.

- Kernel bypass networking.
- In-kernel Sandbox.

2. Thread-based deadline-aware scheduling.

Kernel bypass Networking

Data Plane Development Kit (DPDK)

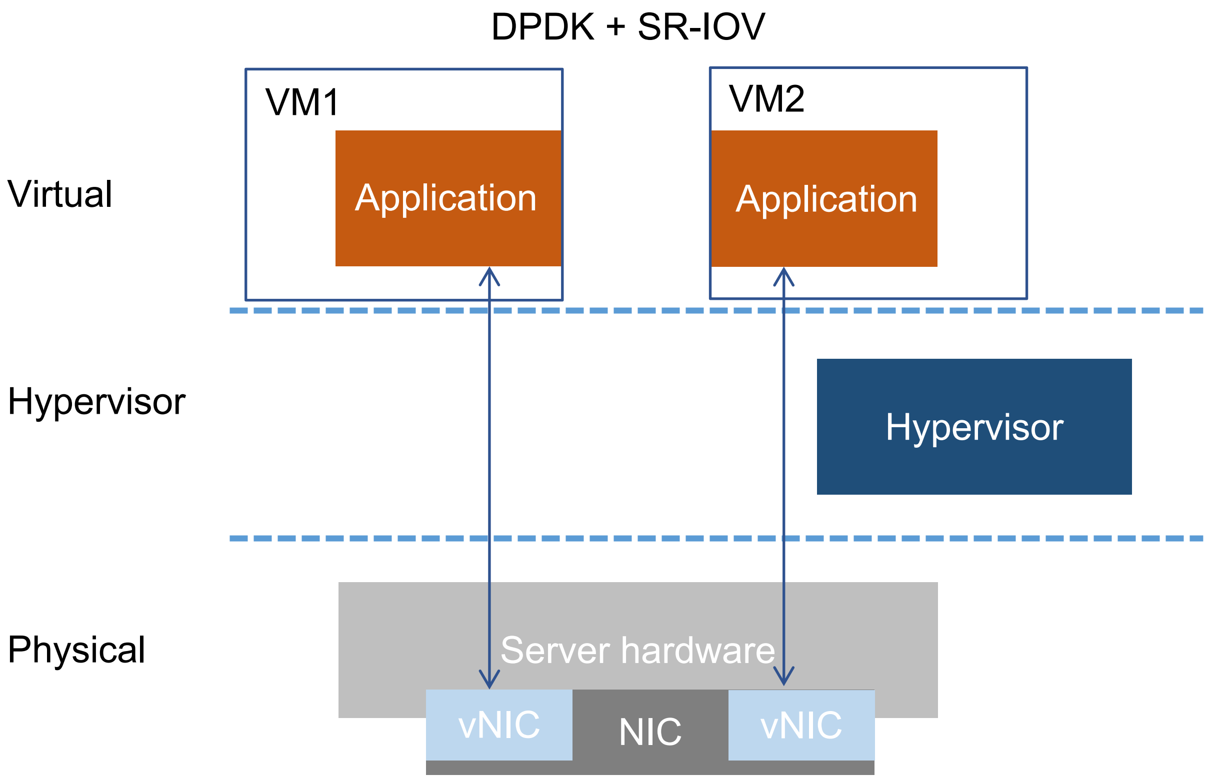


Kernel bypass Networking

Data Plane Development Kit (DPDK)

- Achieving isolation between tenants.
  1. **DPDK + Single Root I/O Virtualization (SR-IOV)**

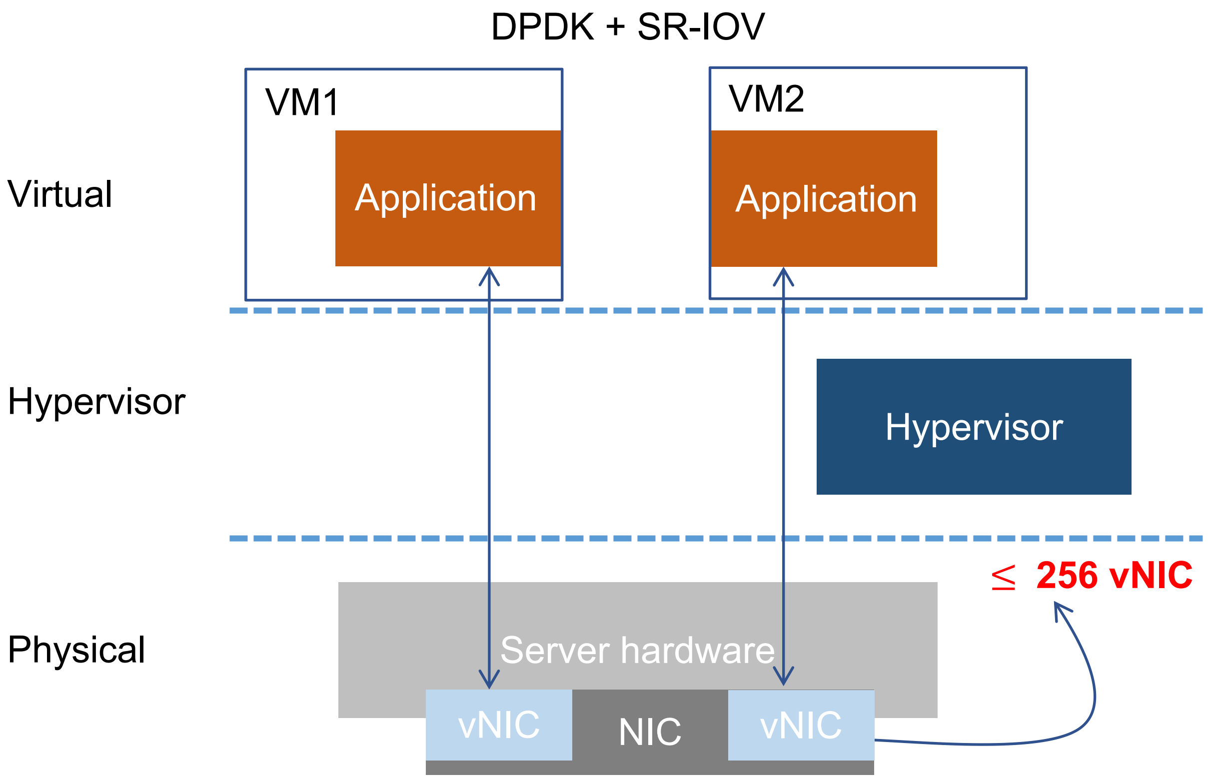
  2. DPDK + Open vSwitch (OVS)



Kernel bypass Networking

Data Plane Development Kit (DPDK)

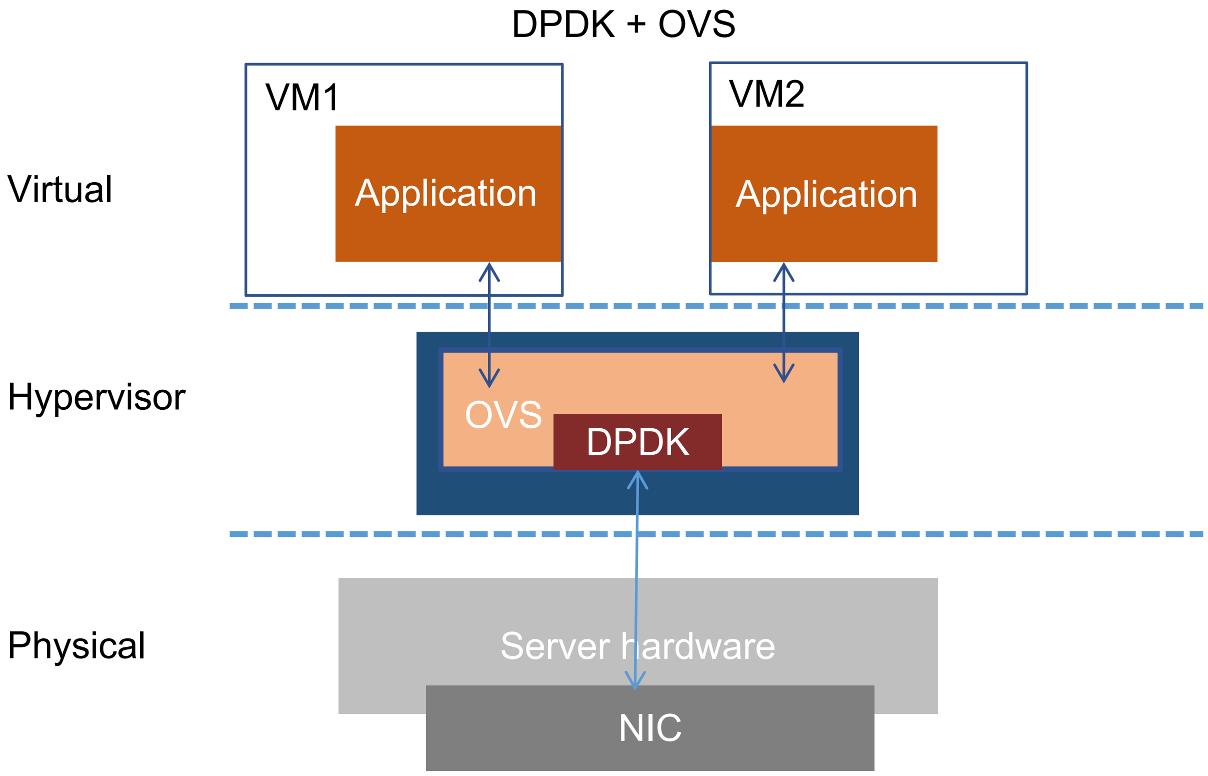
- Achieving isolation between tenants.
  1. **DPDK + Single Root I/O Virtualization (SR-IOV)**
     - Multi-tenancy: ✔
     - Scalability: ✘
     - Deadline-aware: ✘
  2. DPDK + Open vSwitch (OVS)



Kernel bypass Networking

Data Plane Development Kit (DPDK)

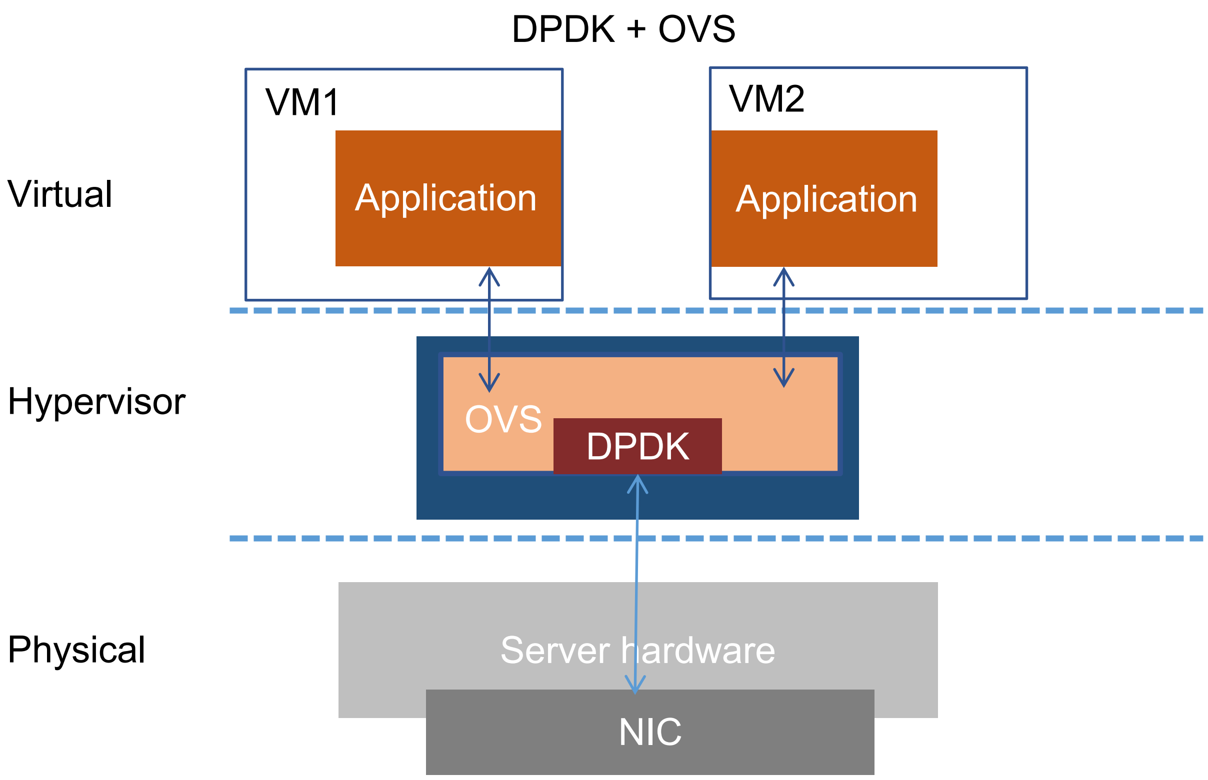
- Achieving isolation between tenants.
  1. DPDK + Single Root I/O Virtualization (SR-IOV)
     - Multi-tenancy: ✓
     - Scalability: ✗
     - Deadline-aware: ✗
  2. **DPDK + Open vSwitch (OVS)**



Kernel bypass Networking

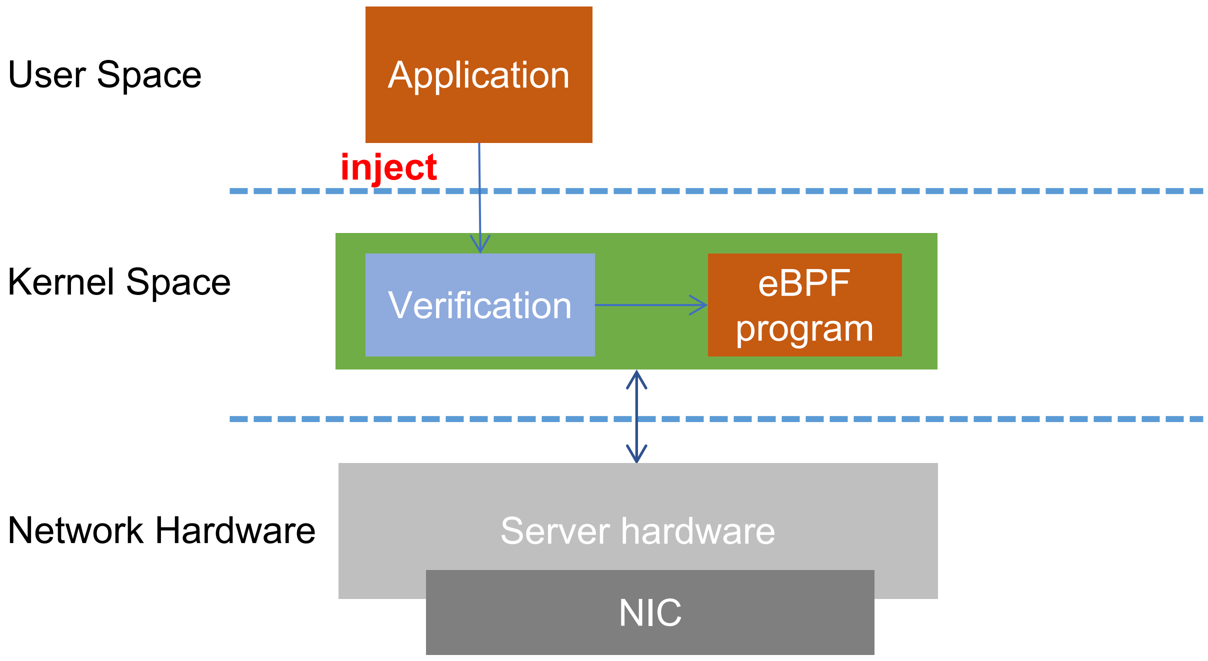
Data Plane Development Kit (DPDK)

- Achieving isolation between tenants.
  1. DPDK + Single Root I/O Virtualization (SR-IOV)
     - Multi-tenancy: ✓
     - Scalability: ✗
     - Deadline-aware: ✗
  2. **DPDK + Open vSwitch (OVS)**
     - Multi-tenancy ✓
     - Scalability: ✗
     - Deadline-aware ✗



In-kernel Sandbox

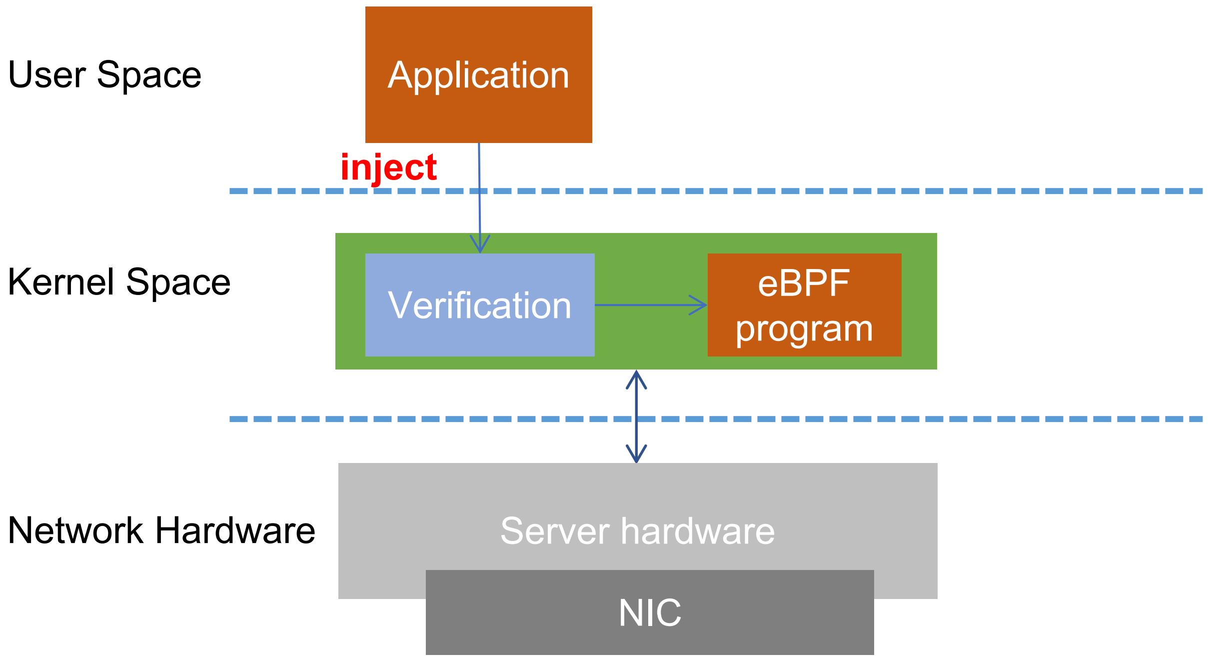
extended Berkeley Packet Filter (eBPF)



In-kernel Sandbox

extended Berkeley Packet Filter (eBPF)

- Evaluation:
  - Multi-tenancy: ❌
  - Deadline-aware: ❌



Existing Technologies

1. Optimize throughput by reducing system overhead.
   - Kernel bypass networking. ❌
   - In-kernel Sandbox. ❌

2. Thread-based deadline-aware scheduling.

Thread-based Scheduling.

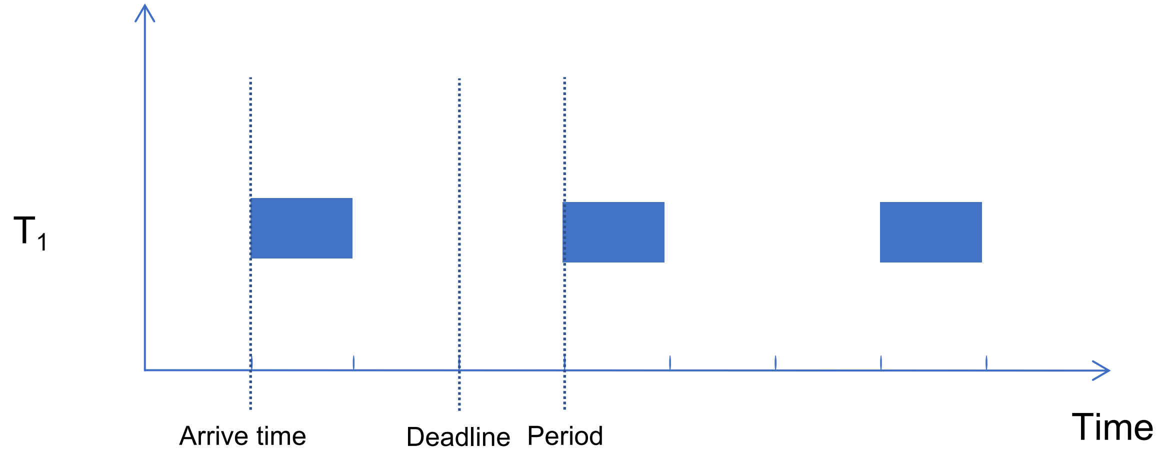
SCHED_DEADLINE

Multi-tenancy: ✓

Scalability: ✗

Dynamic workload: ✗

Periodic tasks with known execution time

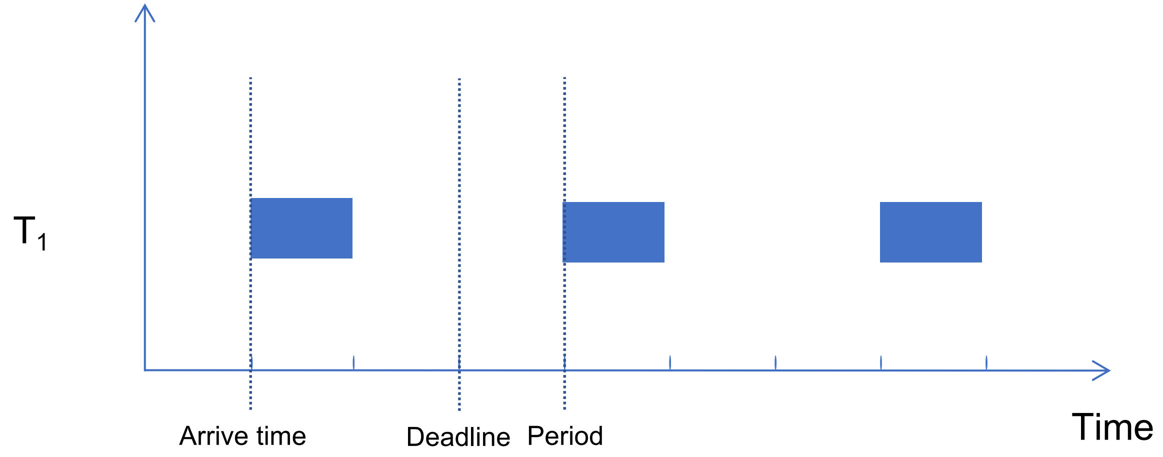


Thread-based Scheduling.

SCHED_DEADLINE

- Multi-tenancy: ✓
- Scalability: ✗
- Dynamic workload: ✗

Periodic tasks with known execution time ✓

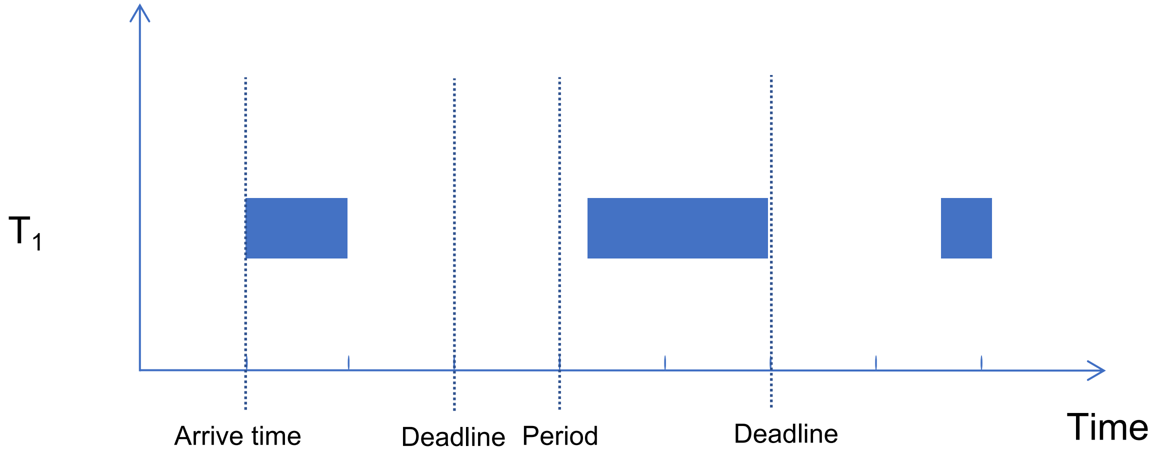


Thread-based Scheduling.

SCHED_DEADLINE

- Multi-tenancy: ✓
- Scalability: ✗
- Dynamic workload: ✗

Aperiodic tasks with unknown execution time ✗

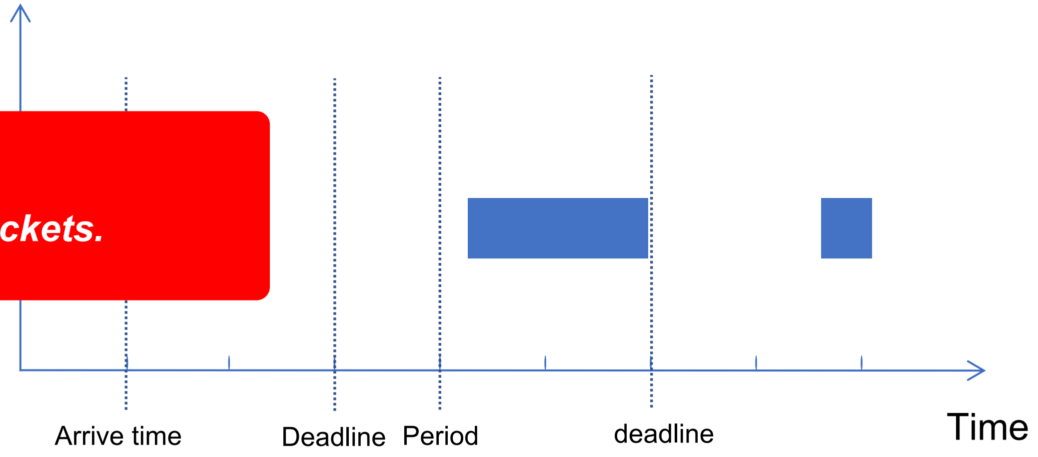


Thread-based Scheduling.

SCHED_DEADLINE

- Multi-tenancy: ✓
- Scalability: ✗
- Dynamic workload:

Aperiodic tasks with unknown execution time ✗



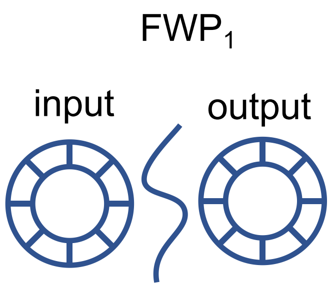
Core Problem:
Schedule ***tasks*** instead of ***packets.***

Edge-RT

1. Background:
   - Built upon EdgeOS:
     1. Light-weighted isolation abstraction: feather weight process (FWP),
     2. DPDK-based fast networking,
     3. Fast memory movement between FWPs.
2. End-to-end deadline packet scheduling.

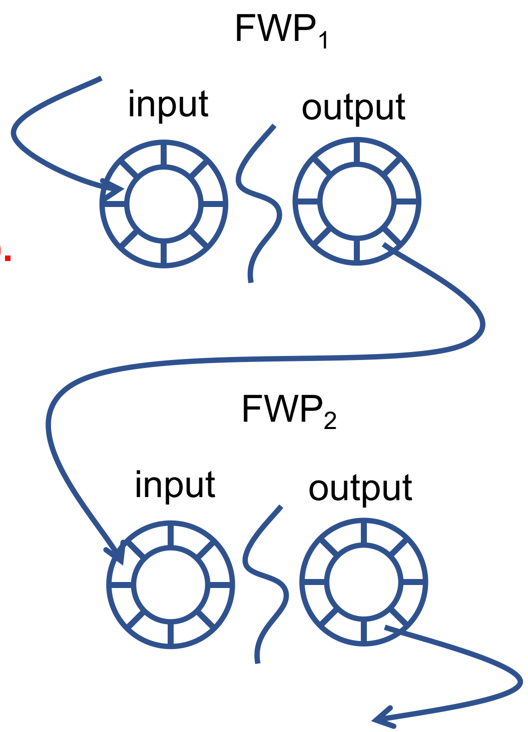
Edge-RT

1. Background:
   - Built upon EdgeOS:
     1. **Light-weighted isolation abstraction: feather weight process (FWP).**
     2. DPDK-based fast networking.
     3. Fast memory movement between FWPs.
2. End-to-end deadline packet scheduling.



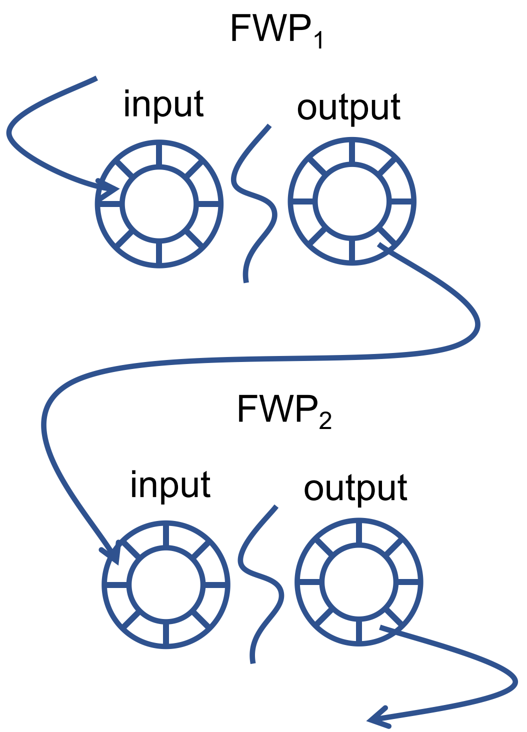
Edge-RT

1. Background:
   - Built upon EdgeOS:
     1. **Light-weighted isolation abstraction: feather weight process (FWP).**
     2. DPDK-based fast networking.
     3. Fast memory movement between FWPs.
2. End-to-end deadline packet scheduling.



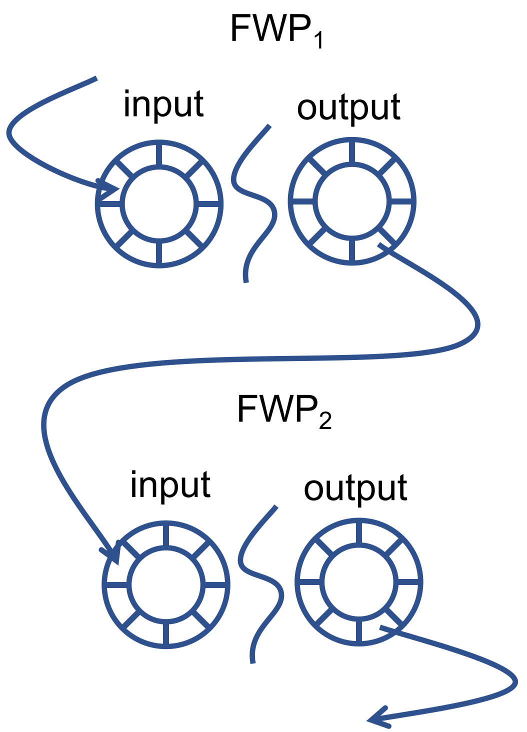
Edge-RT

1. Background:
    - Built upon EdgeOS:
        1. Light-weighted isolation abstraction: feather weight process (FWP).
        2. **DPDK-based fast networking.**
        3. Fast memory movement between FWPs.
2. End-to-end deadline packet scheduling.



Edge-RT

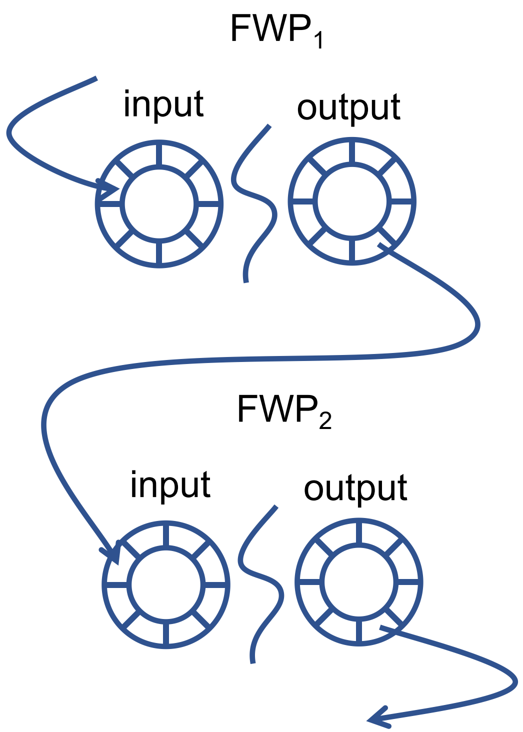
1. Background:
    - Built upon EdgeOS:
        1. Light-weighted isolation abstraction: feather weight process (FWP).
        2. DPDK-based fast networking.
        3. **Fast memory movement between FWPs.**
2. End-to-end deadline packet scheduling.



Edge-RT

1. Background:
   - Built upon EdgeOS:
     1. Light-weighted isolation abstraction: feather weight process (FWP).
     2. DPDK-based fast networking.
     3. Fast memory movement between FWPs.

2. Goal: End-to-end deadline scheduling of packets.

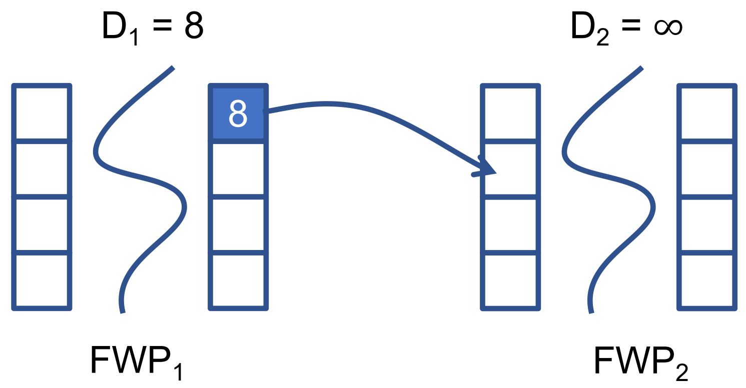


End-to-end Packet Scheduling

1. Deadline inheritance.

- Thread inherit deadline from the packet.

2. Schedule chains of computations.

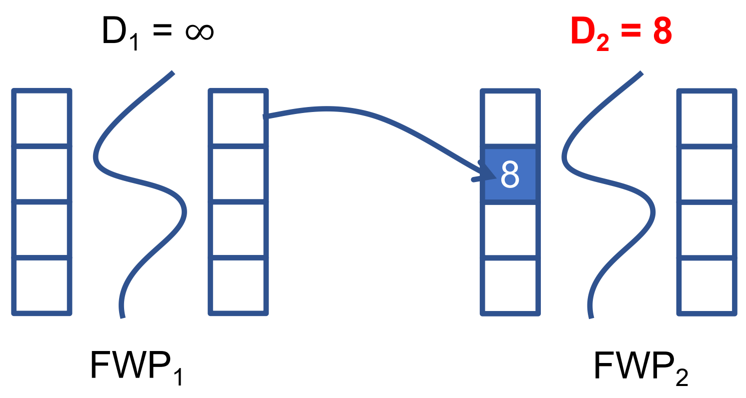


End-to-end Packet Scheduling

1. Deadline inheritance.

- Thread inherit deadline from the packet.

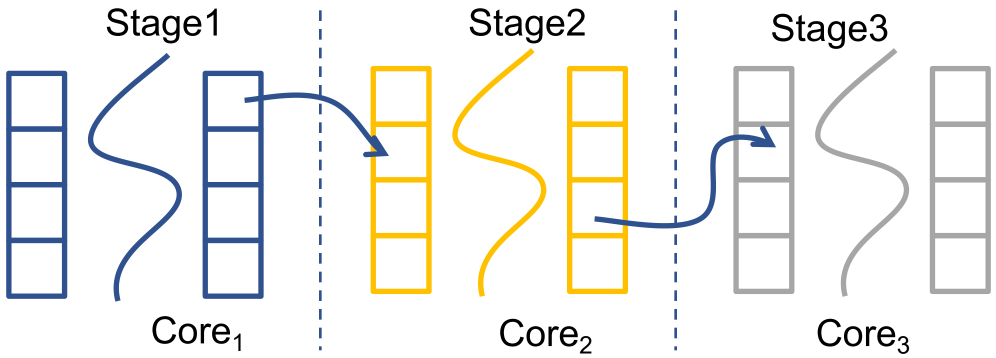
2. Schedule chains of computations.



End-to-end Packet Scheduling

1. Deadline inheritance.
   - Thread inherit deadline from the packet.

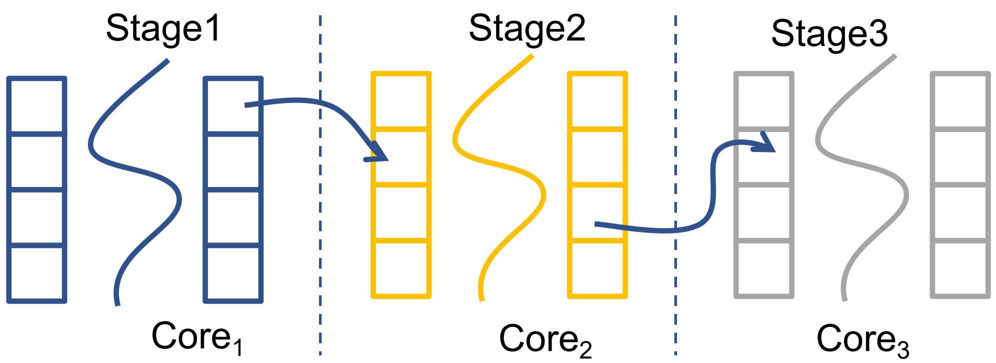
2. Schedule chains of computations.

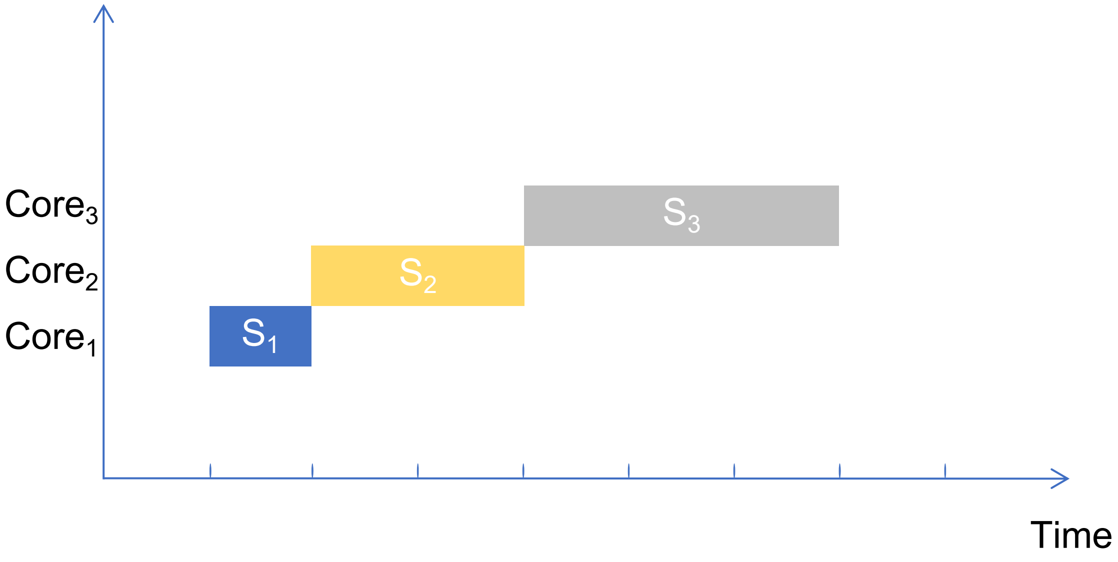


End-to-end Packet Scheduling

1. Deadline inheritance.
    - Thread inherit deadline from the packet.

2. Schedule chains of computations.

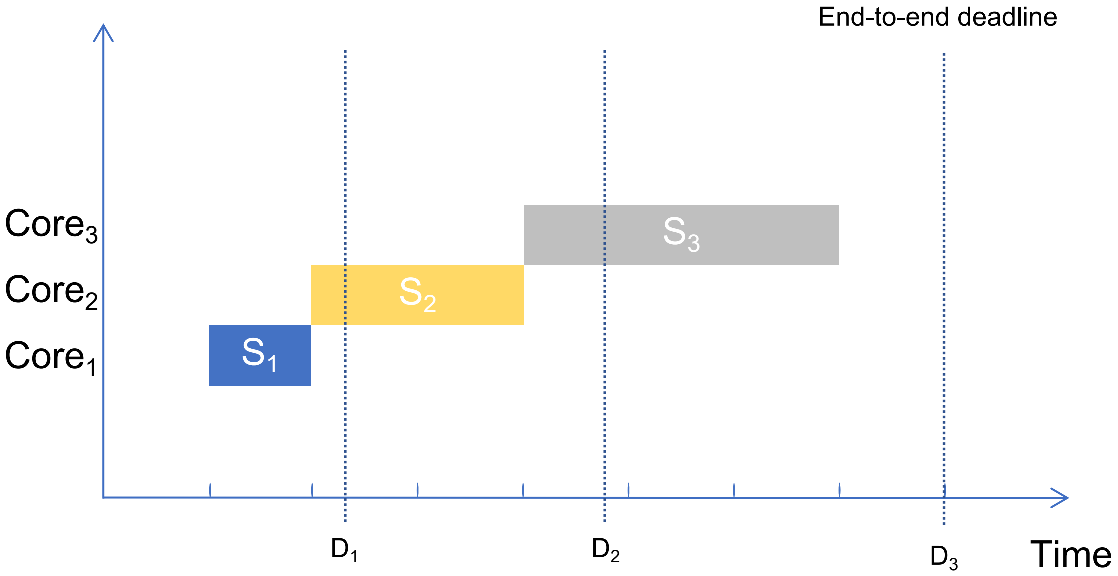


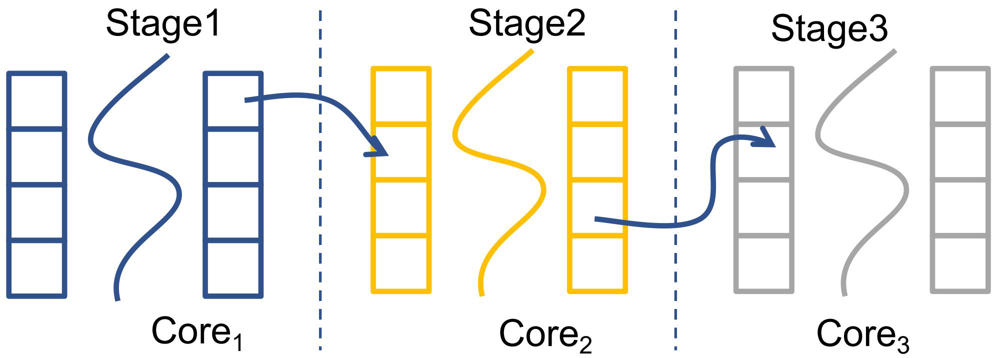


End-to-end Packet Scheduling

1. Deadline inheritance.
   - Thread inherit deadline from the packet.

2. Schedule chains of computations.

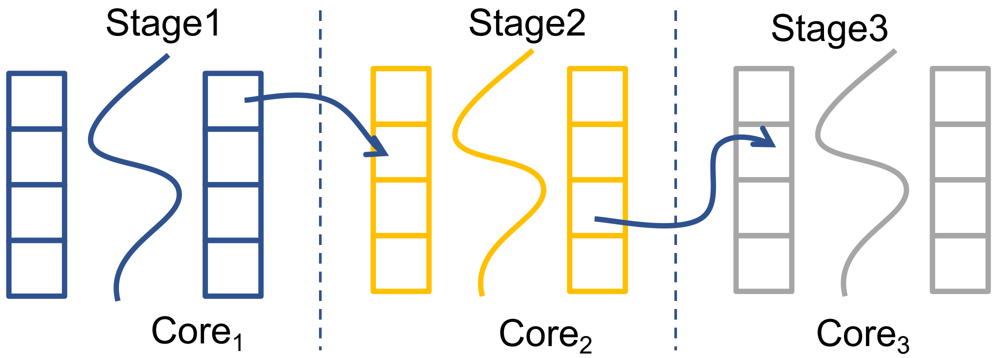


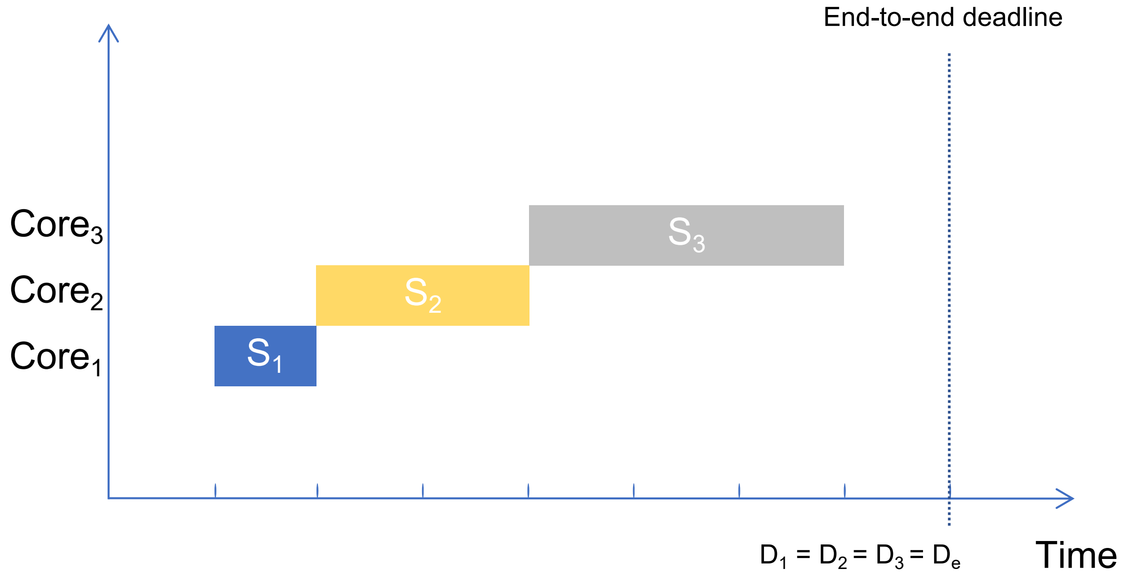


End-to-end Packet Scheduling

1. Deadline inheritance.
    - Thread inherit deadline from the packet.

2. Schedule chains of computations.
- Same deadline for all stages.
- Makes no assumption based on WCET.

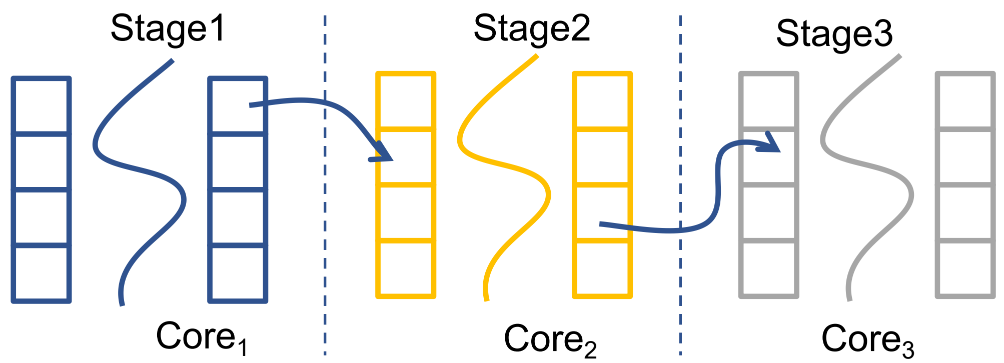


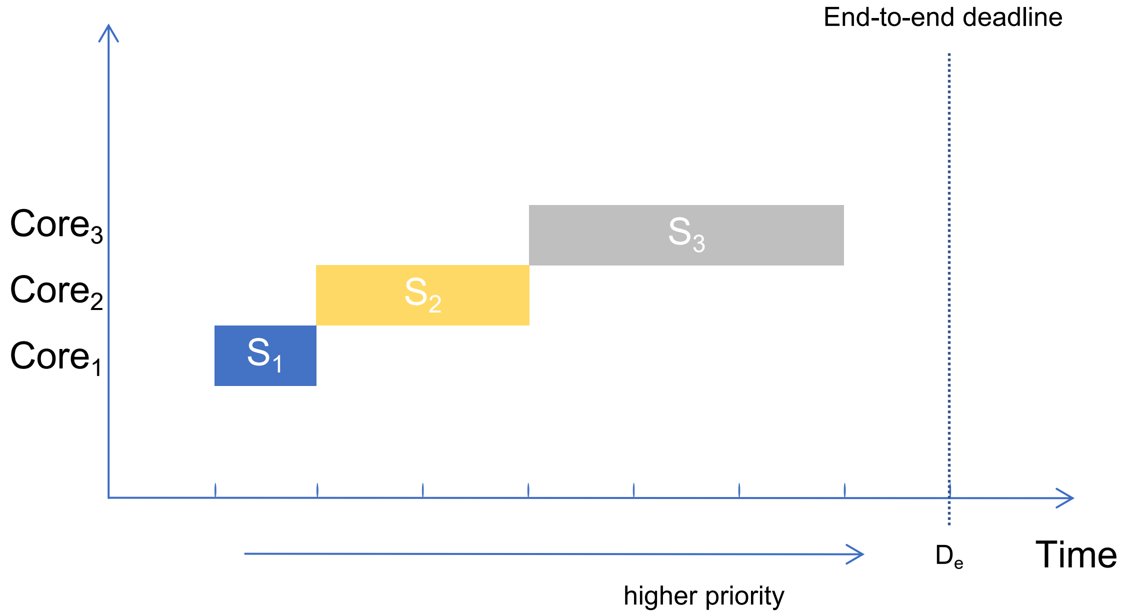


End-to-end Packet Scheduling

1. Deadline inheritance.
    - Thread inherit deadline from the packet.

2. Schedule chains of computations.
    - Same deadline for all stages.
    - Makes no assumption based on WCET.





Remaining Challenges

1. Frequent thread activation and deactivation.

2. Frequent (inter-core) event notification.

3. EDF policy overheads for frequent activation.

Optimization

1. Frequent thread activation and deactivation.

2. Frequent (inter-core) event notification.

3. EDF policy overheads for frequent activation.

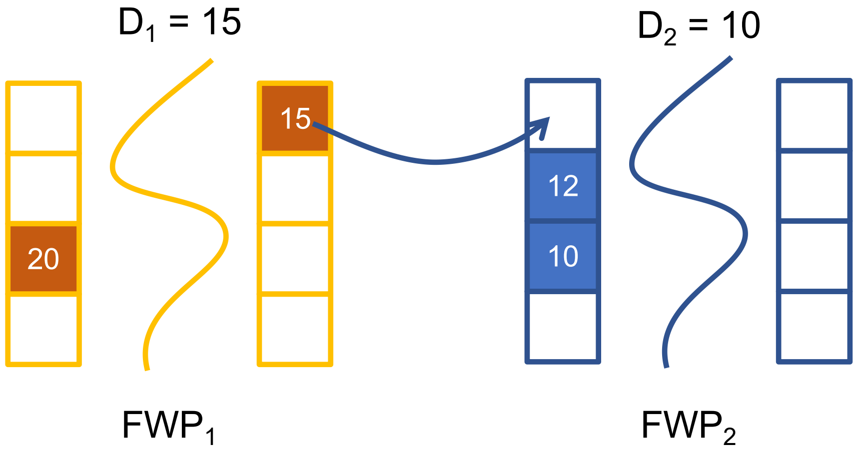
1. **Deadline-aware batching.**

2. Periodic Event Notification.

3. Constant-Time EDF.

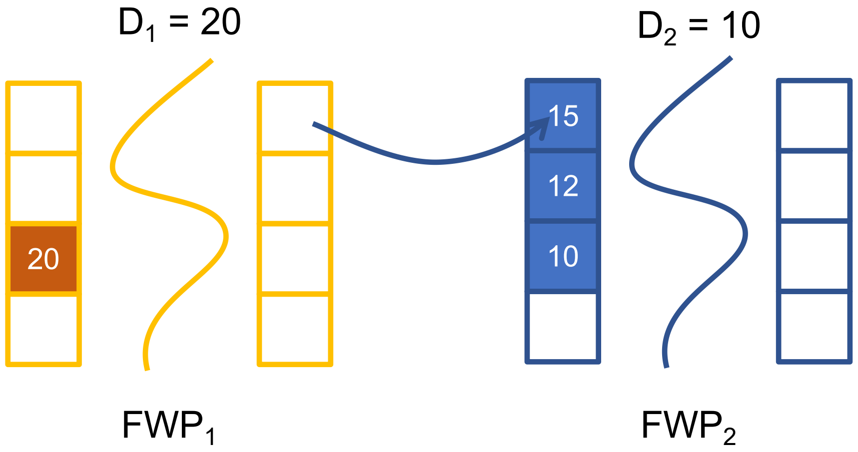
Deadline-aware batching

1. **Deadline inheritance of multiple packets.**
2. Priority inversions (Δ_{batch}).
3. Batching with controlled size.



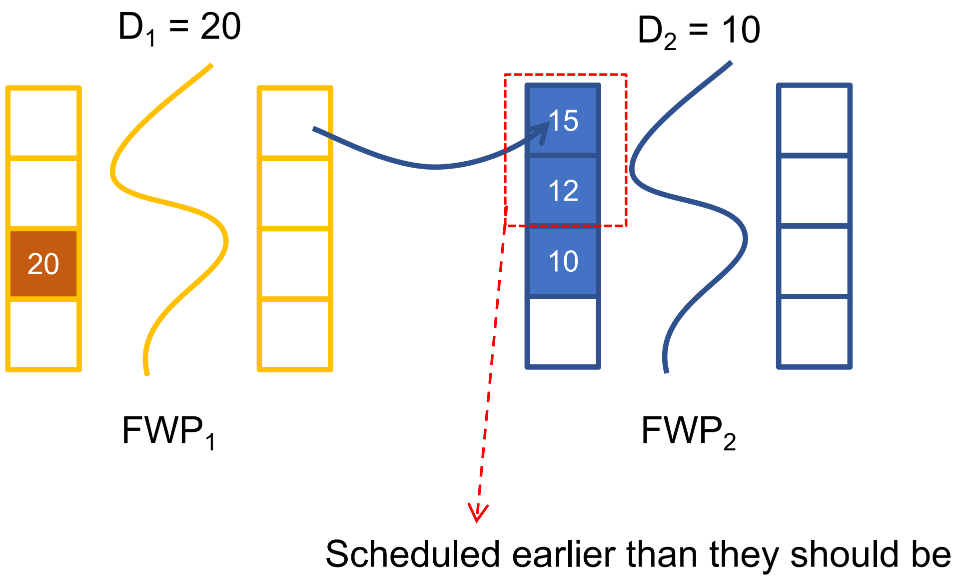
Deadline-aware batching

1. **Deadline inheritance of multiple packets.**
2. Priority inversions (Δ_{batch}).
3. Batching with controlled size.



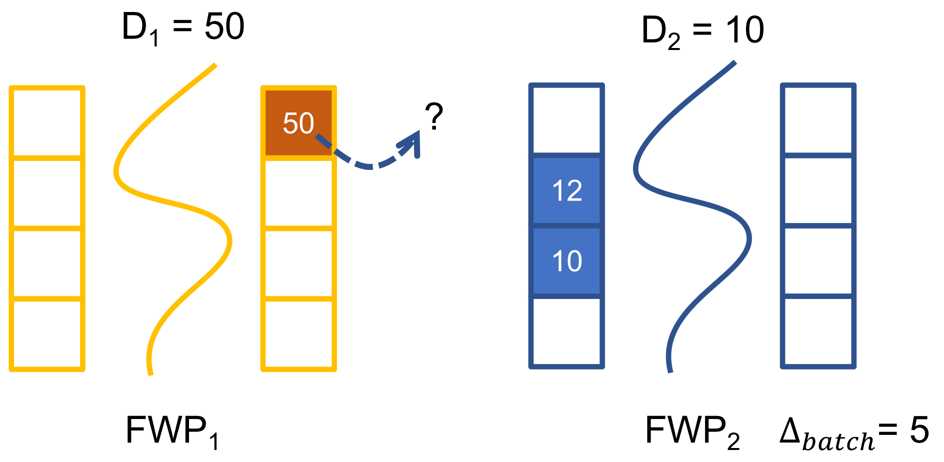
Deadline-aware batching

1. Deadline inheritance of multiple packets.
2. **Priority inversions (Δ_{batch}).**
3. Batching with controlled size.



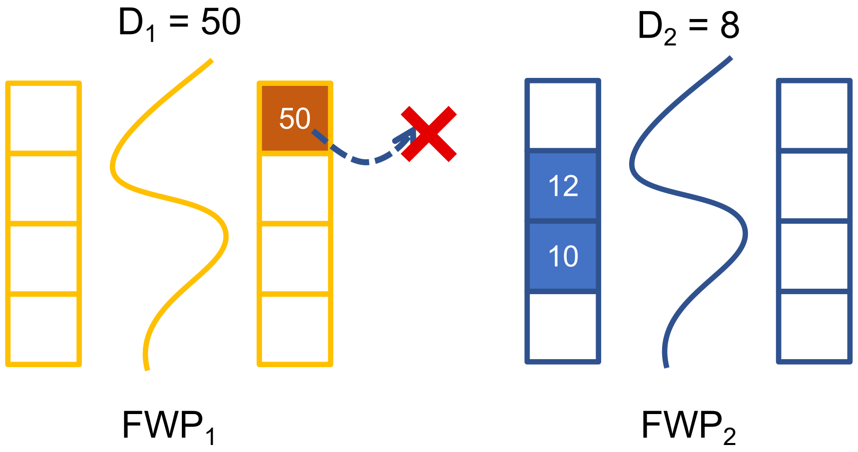
Deadline-aware batching

1. Deadline inheritance of multiple packets.
2. Priority inversions (Δ_{batch}).
3. **Batching with controlled size.**



Deadline-aware batching

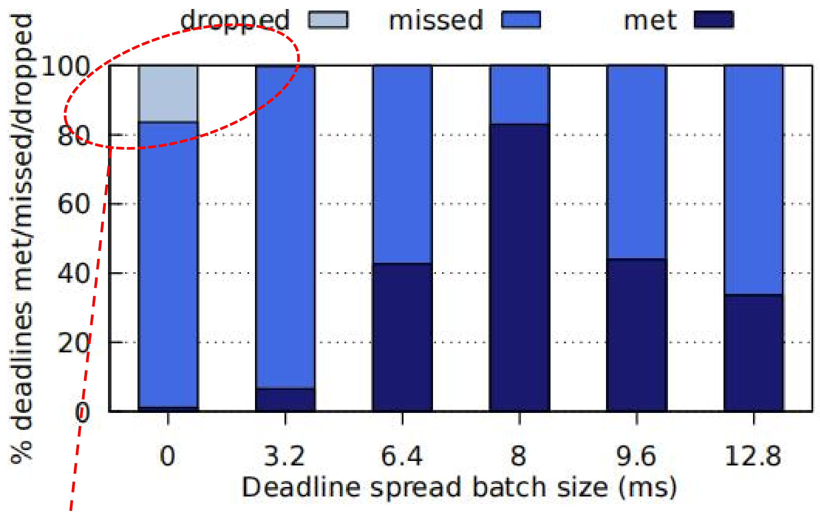
1. Deadline inheritance of multiple packets.
2. Priority inversions (Δ_{batch}).
3. **Batching with controlled size.**



$(P_i, d_i), d \in [d_i - \Delta_{batch}, d_i]$

Deadline-aware batching

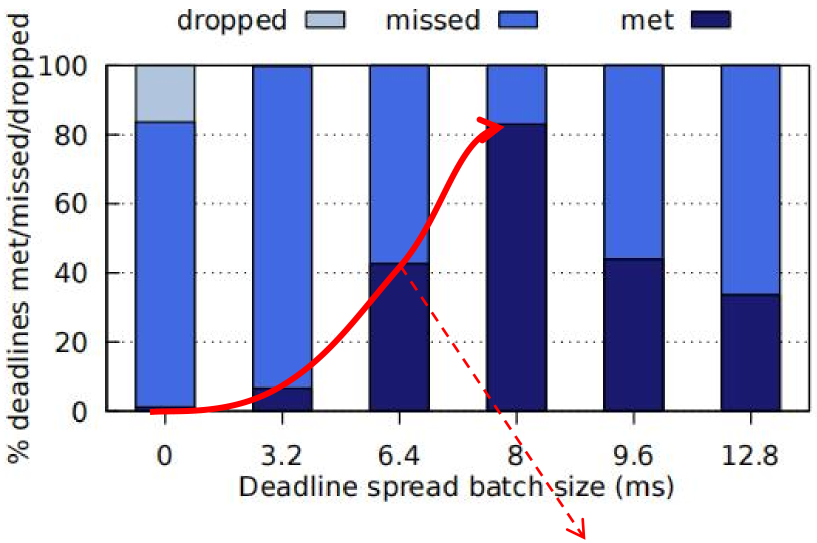
1. Deadline inheritance of multiple packets.
2. Priority inversions (Δ_{batch}).
3. **Batching with controlled size.**



Caused by System overhead

Deadline-aware batching

1. Deadline inheritance of multiple packets.
2. Priority inversions (Δ_{batch}).
3. **Batching with controlled size.**

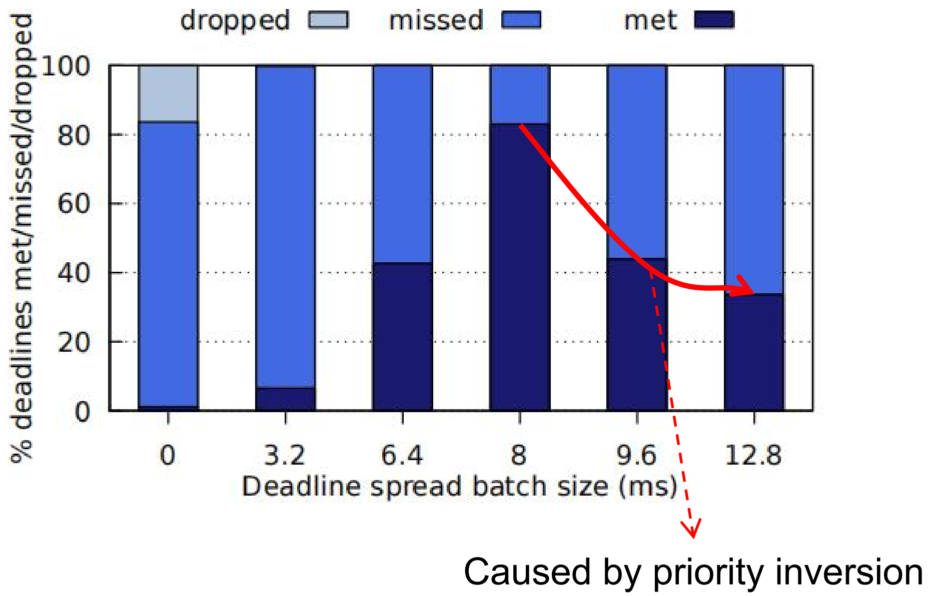


Batching reduces system overhead

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Deadline-aware batching

1. Deadline inheritance of multiple packets.
2. Priority inversions (Δ_{batch}).
3. **Batching with controlled size.**



Optimization

1. Frequent thread activation and deactivation.
2. Frequent (inter-core) event notification.
3. EDF policy overheads for frequent activation.

1. Deadline-aware batching.
2. **Periodic Event Notification.**
3. Constant-Time EDF.

Optimization

1. Frequent thread activation and deactivation.
2. Frequent (inter-core) event notification.
3. EDF policy overheads for frequent activation.

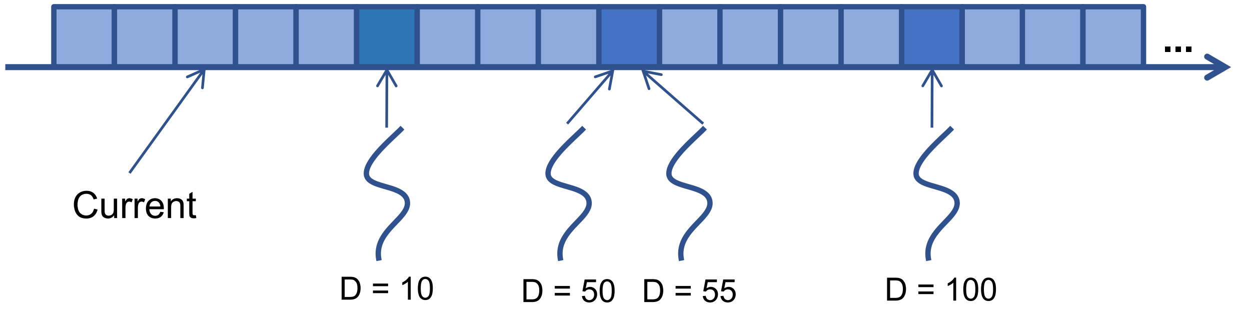
1. Deadline-aware batching.
2. Periodic Event Notification.
3. **Constant-Time EDF.**

Constant-time EDF Scheduling (CT-EDF)

1. **Quantize time into fixed quanta.**
2. Using array to track each quanta.
3. Priority inversion (Δ_{window}).

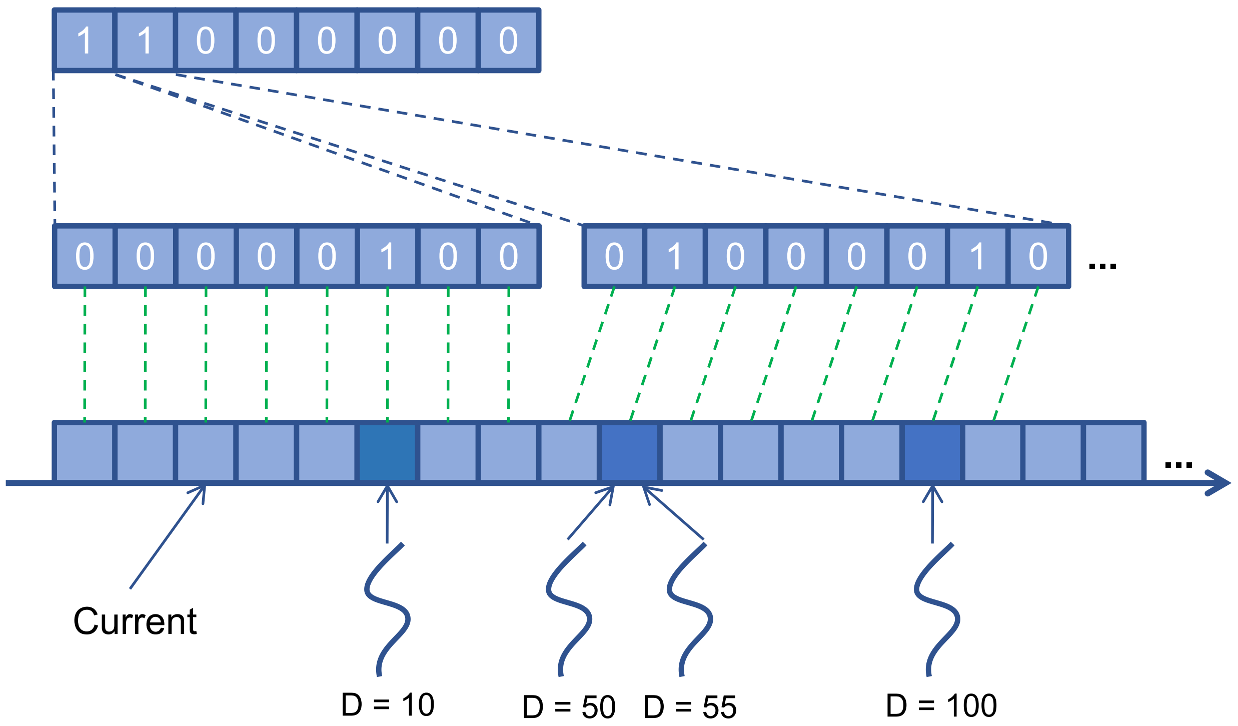
Constant-time EDF Scheduling (CT-EDF)

1. **Quantize time into fixed quanta.**
2. Using array to track each quanta.
3. Priority inversion (Δ_{window}).



Constant-time EDF Scheduling (CT-EDF)

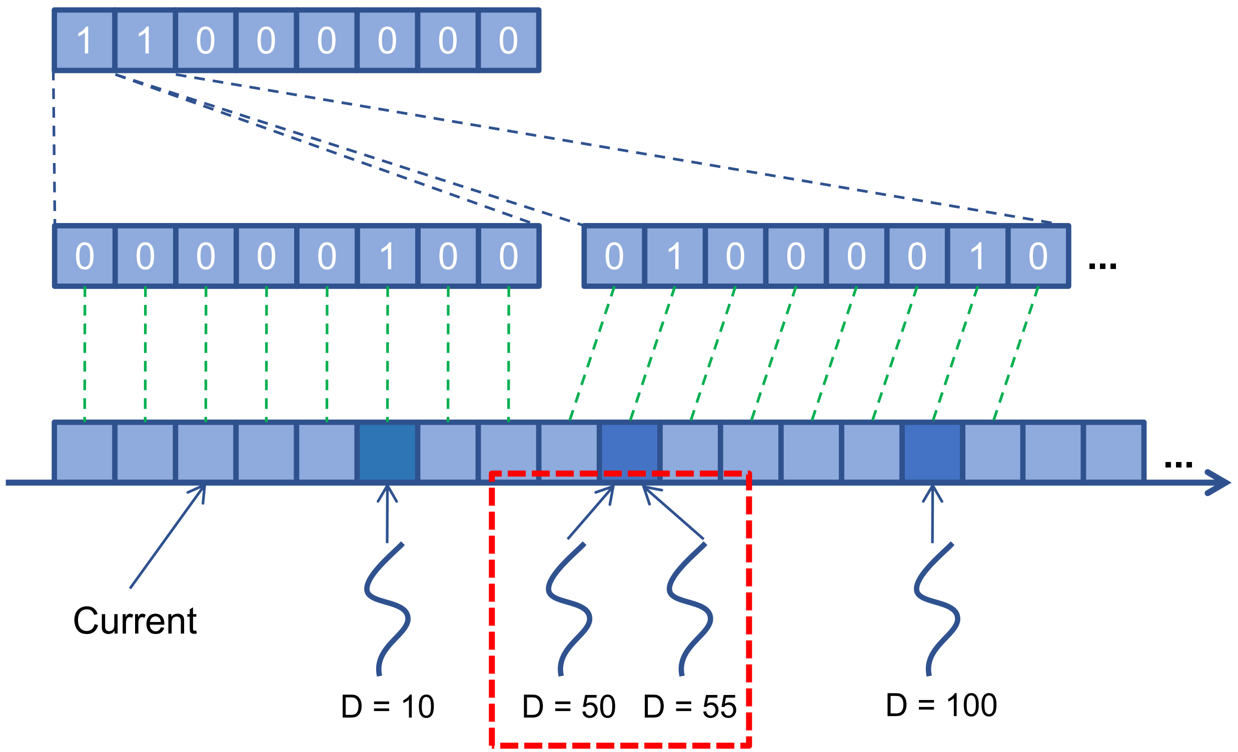
1. Quantize time into fixed quanta.
2. **Using array to track each quanta.**
3. Priority inversion (Δ_{window}).



Constant-time EDF Scheduling (CT-EDF)

1. Quantize time into fixed quanta.
2. Using array to track each quanta.
3. **Priority inversion (Δ_{window}).**

$$(P_i, d_i), d \in [d_i - \Delta_{window}, d_i]$$



Priority Inversions

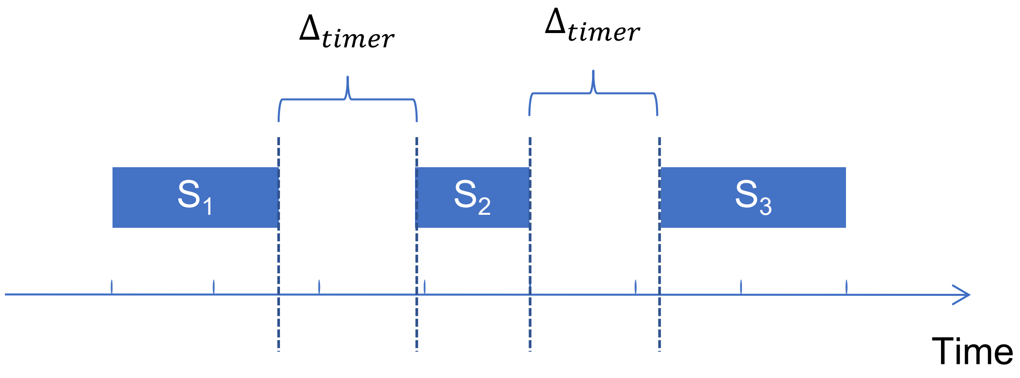
Edge-RT achieves line-rate throughput by creating ***bounded*** deadline inversions:

1. Batching (Δ_{batch}).
2. Periodic event notification (Δ_{timer}).
3. CT-EDF (Δ_{window}).

Priority Inversions

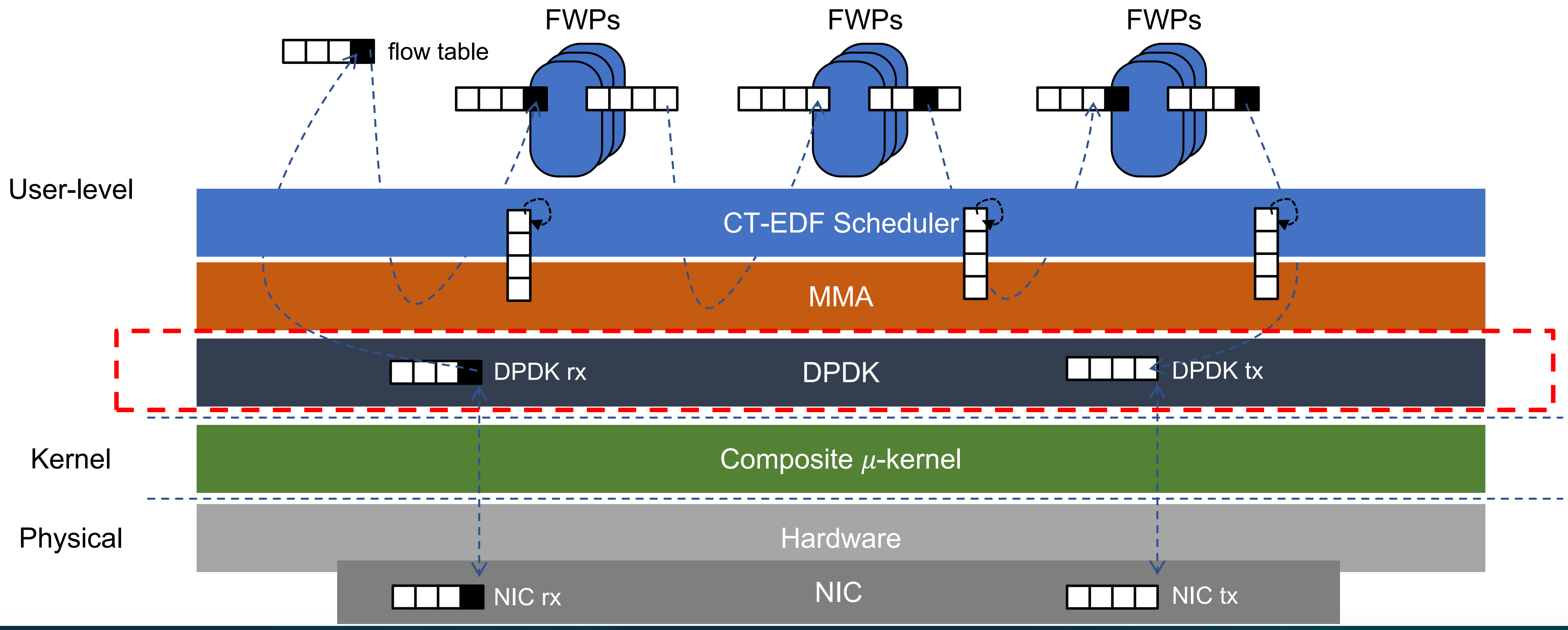
Edge-RT achieves line-rate throughput by creating ***bounded*** deadline inversions:

1. Batching (Δ_{batch}).
2. Periodic event notification (Δ_{timer}).
3. CT-EDF (Δ_{window}).

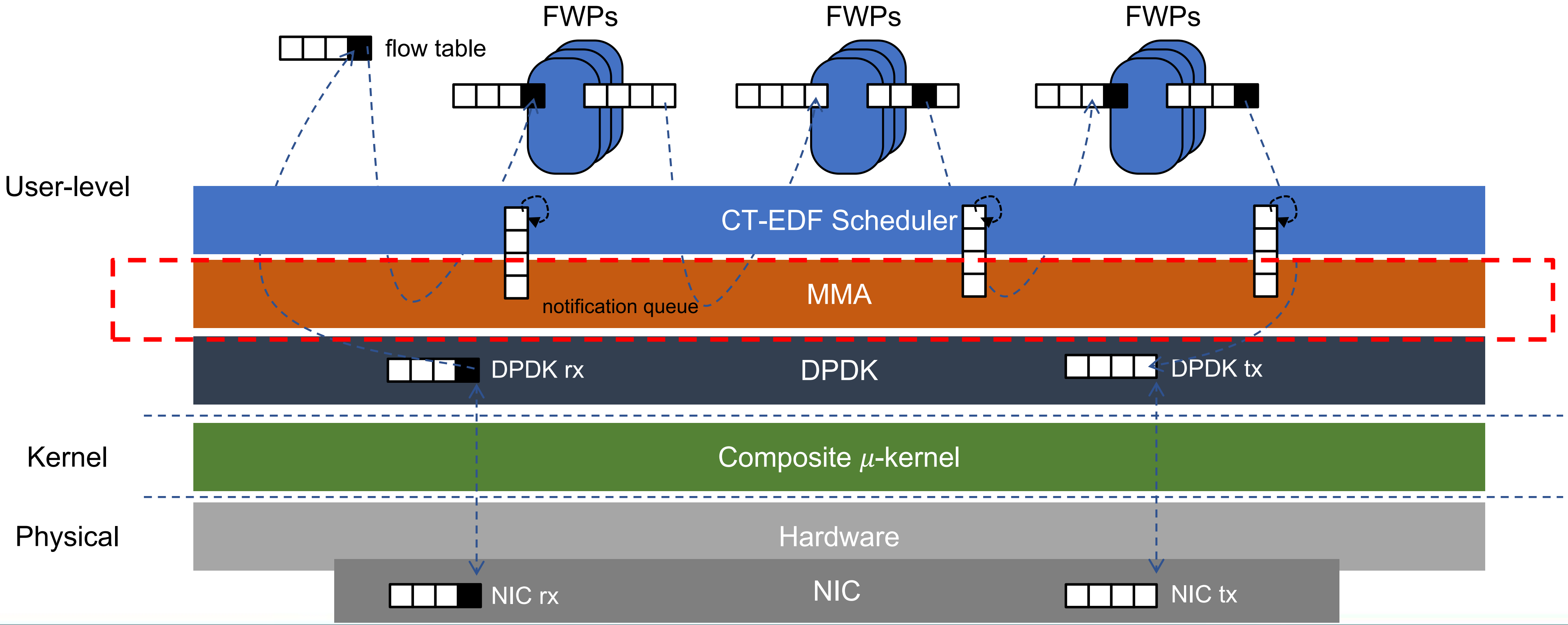


$$d \in [d_i - max(\Delta_{batch}, \Delta_{window}), d_i]$$

System Architecture

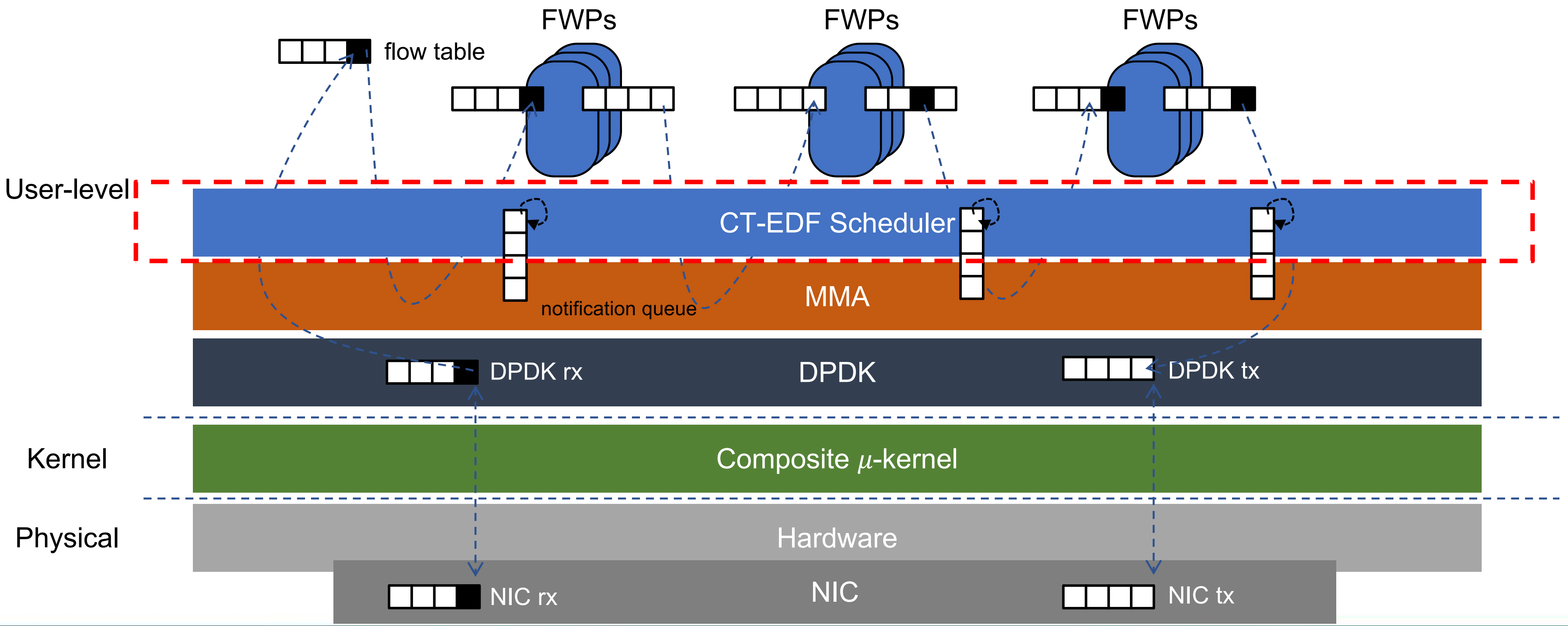


System Architecture



THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

System Architecture



Evaluation

Experiment setup:

- Power Edge R740 servers.
- Two socket Intel(R) Xeon(R) Platinum 8160 CPUs @2.10GHz each with 24 cores.
- Intel X710 for10GbE NIC.
- Compare Linux, EdgeOS and Edge-RT

Evaluation

Workload description:

1. Bimodal workloads.
2. Light computation $WCET = 40\mu s, deadline = 10ms$, (Kalman filtering)
3. Heavy computation $WCET = 5ms, deadline = 500ms$, (ML inference)
4. EdgeRT $\Delta_{batch} = 8ms, \Delta_{timer} = 250\mu s$.
5. 480 clients/chains, chain length 4, 1920 FWPs in total.

Utilization Sensitivity

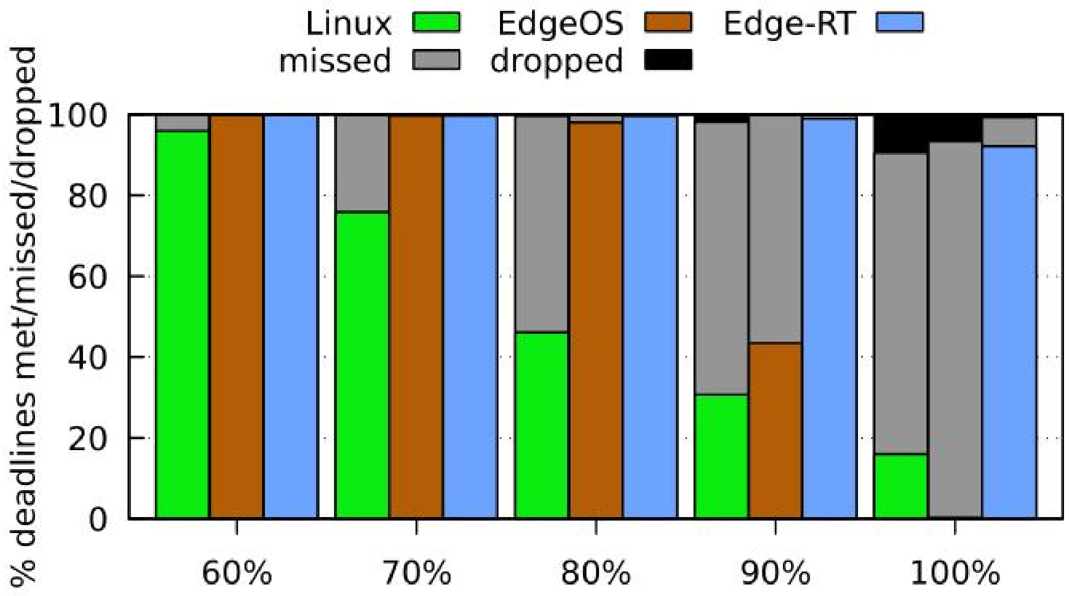


Fig.1. The behavior of **light** tasks with increasing utilization

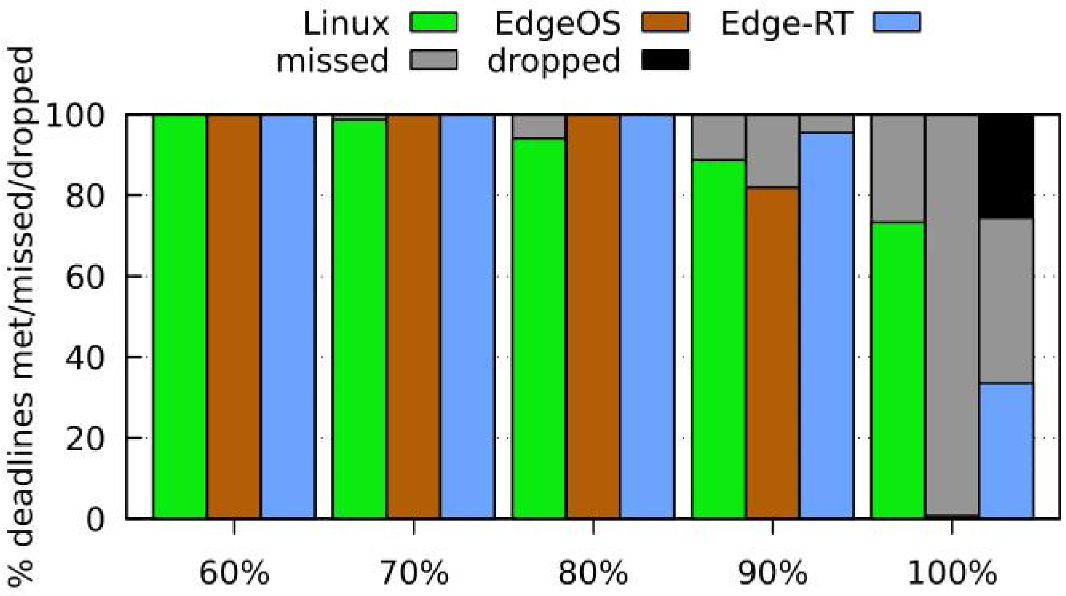


Fig.2. The behavior of **heavy** tasks with increasing utilization

Utilization Sensitivity

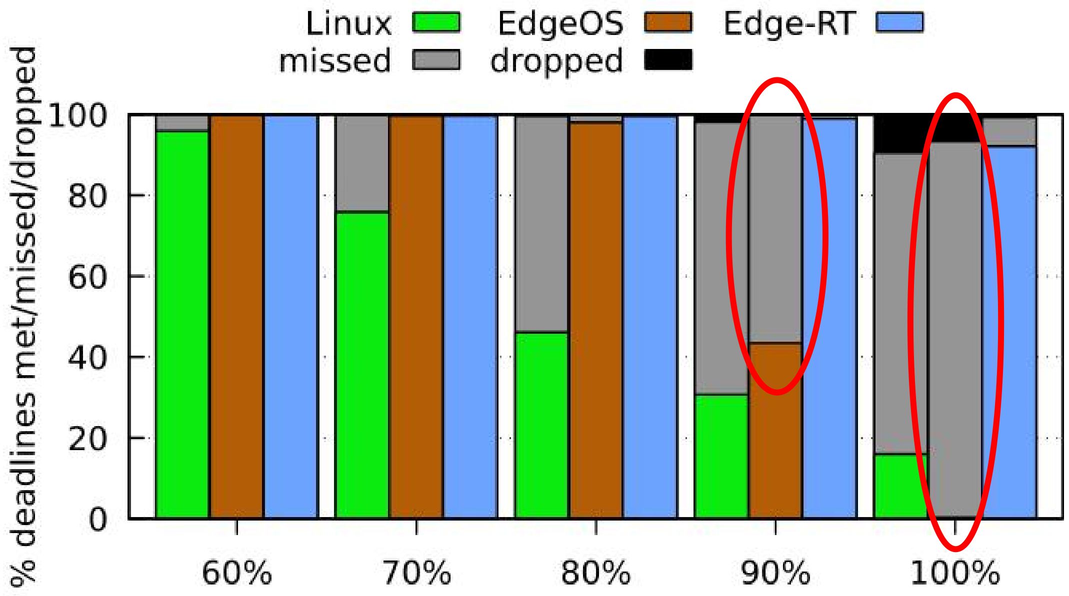


Fig.1. The behavior of **light** tasks with increasing utilization

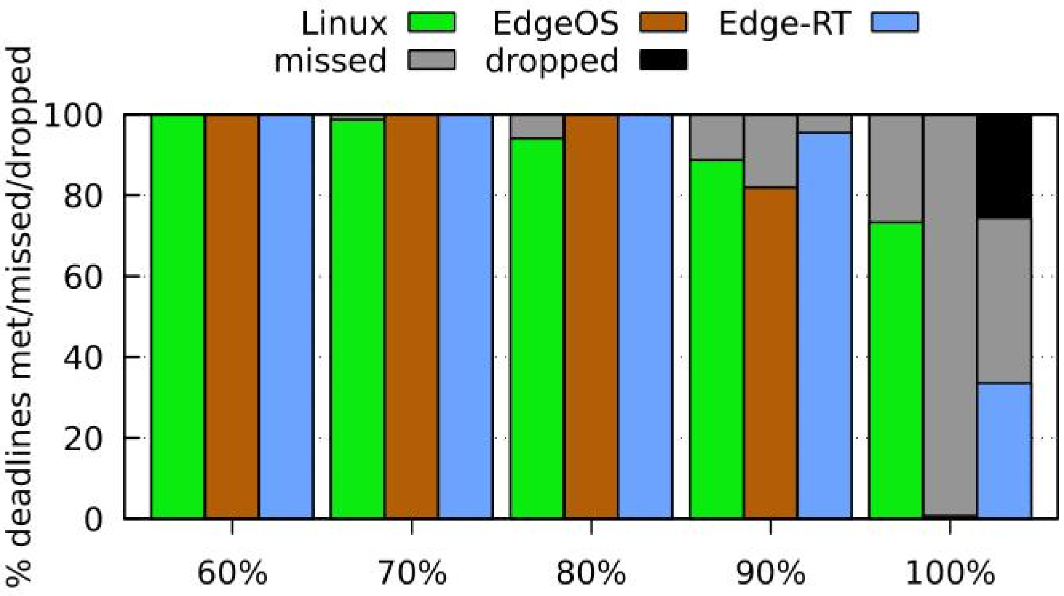


Fig.2. The behavior of **heavy** tasks with increasing utilization

Utilization Sensitivity

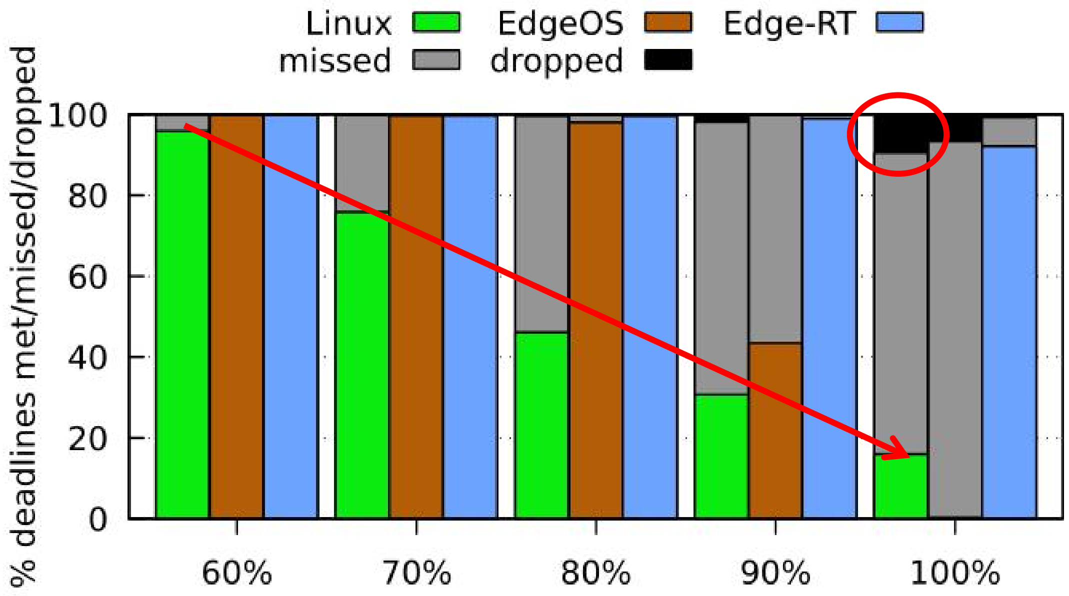


Fig.1. The behavior of **light** tasks with increasing utilization

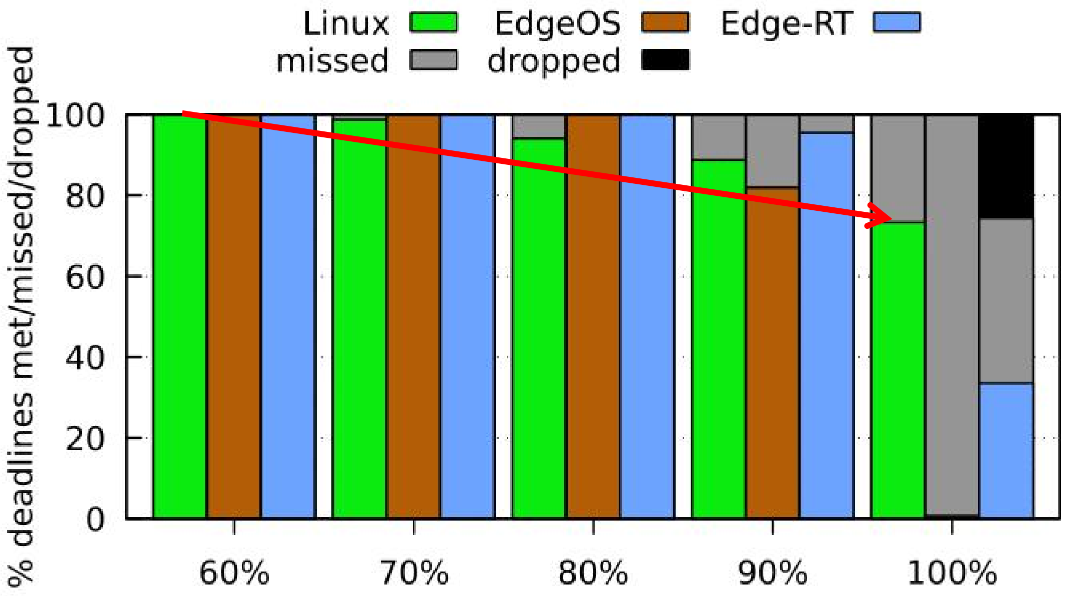


Fig.2. The behavior of **heavy** tasks with increasing utilization

Utilization Sensitivity

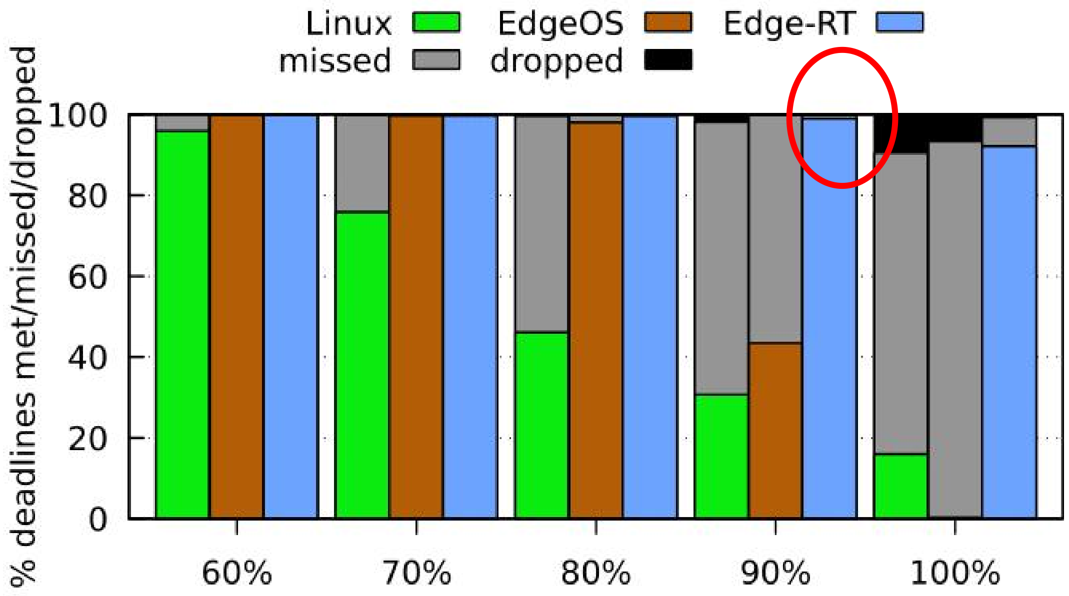


Fig.1. The behavior of **light** tasks with increasing utilization

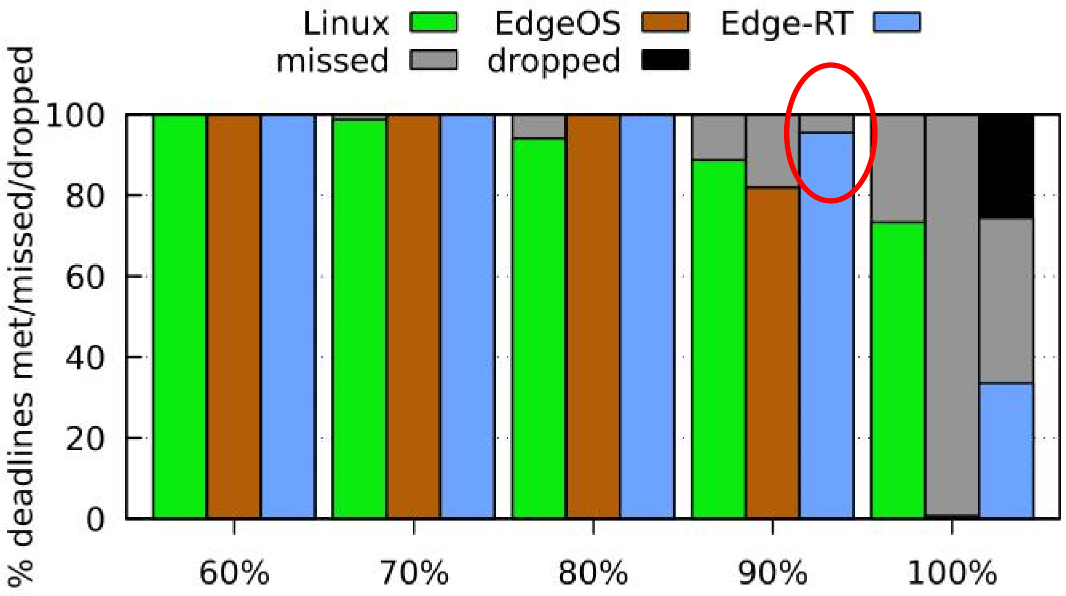


Fig.2. The behavior of **heavy** tasks with increasing utilization

Utilization Sensitivity

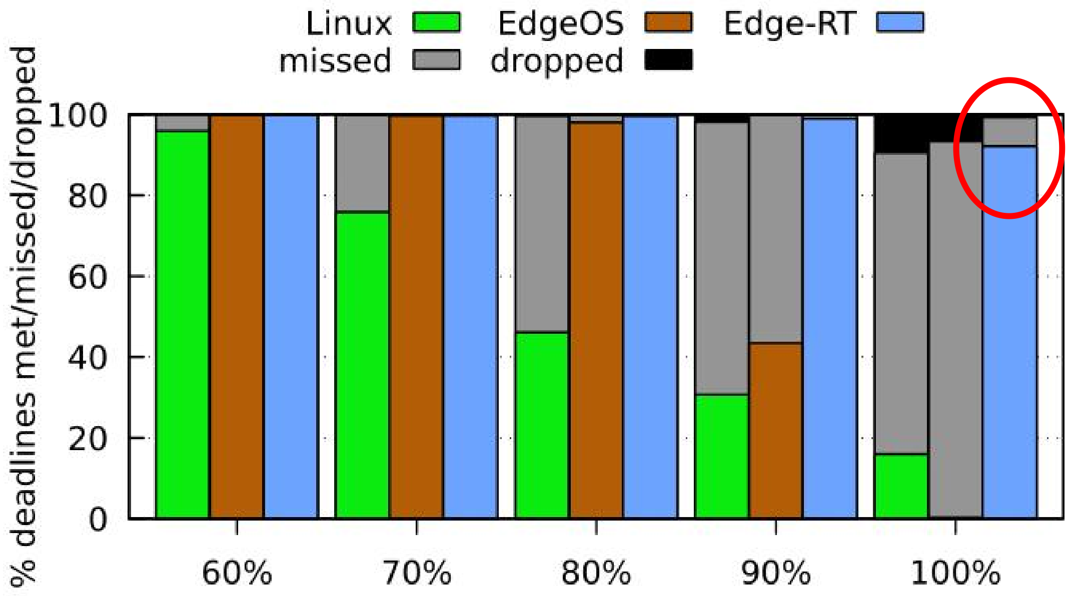


Fig.1. The behavior of **light** tasks with increasing utilization

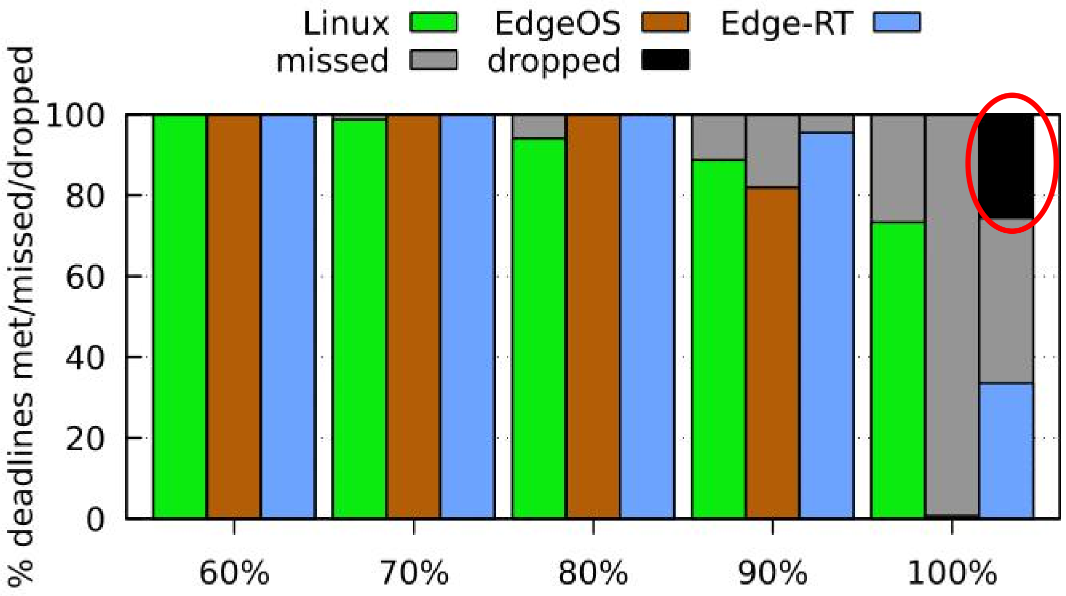


Fig.2. The behavior of **heavy** tasks with increasing utilization

Conclusion

Edge-RT provides a solution for the **multi-tenant**, **dense**, **latency-sensitive** edge cloud.

- Multi-tenant: Strong FWP-based isolation.
- Density: throughput-centric implementation.
- Deadlines:
    - FWP inheritance of packet deadlines,
    - Bounded deadline inversions,
    - End-to-end packet deadline scheduling.

Conclusion

Edge-RT provides a solution for the **multi-tenant**, **dense**, **latency-sensitive** edge cloud.

- Multi-tenant: Strong FWP-based isolation.
- Density: t
- Deadline
  - FWP inheritance of packet deadlines,
  - Bounded deadline inversions,
  - End-to-end packet deadline scheduling.

Edge-RT: strong foundation for the real-time edge

Questions and Comments?

? || /* */

Existing Technologies

- A summary of edge-cloud configurations

Edge Configurations	Deadline-aware	Preemptivity	Client Isolation	Computation Chain	Dynamic Workloads	Scalability
CFS	not deadline-aware	preemptive	process-based	per-client chain	supported	> 2000
DPDK + OVS/SR-IOV	not deadline-aware	non-preemptive	process-based	no chain	supported	~ 256
SCHED_DEADLINE	per-thread	preemptive	process-based	no chain	not supported	< 1000
eBPF + XDP	not deadline-aware	non-preemptive	no isolation	no chain	not supported	-
EdgeOS	not deadline-aware	preemptive	FWP-based	per-client chain	supported	> 2000
Edge-RT	per-packet	preemptive	FWP-based	per-client chain	supported	> 2000