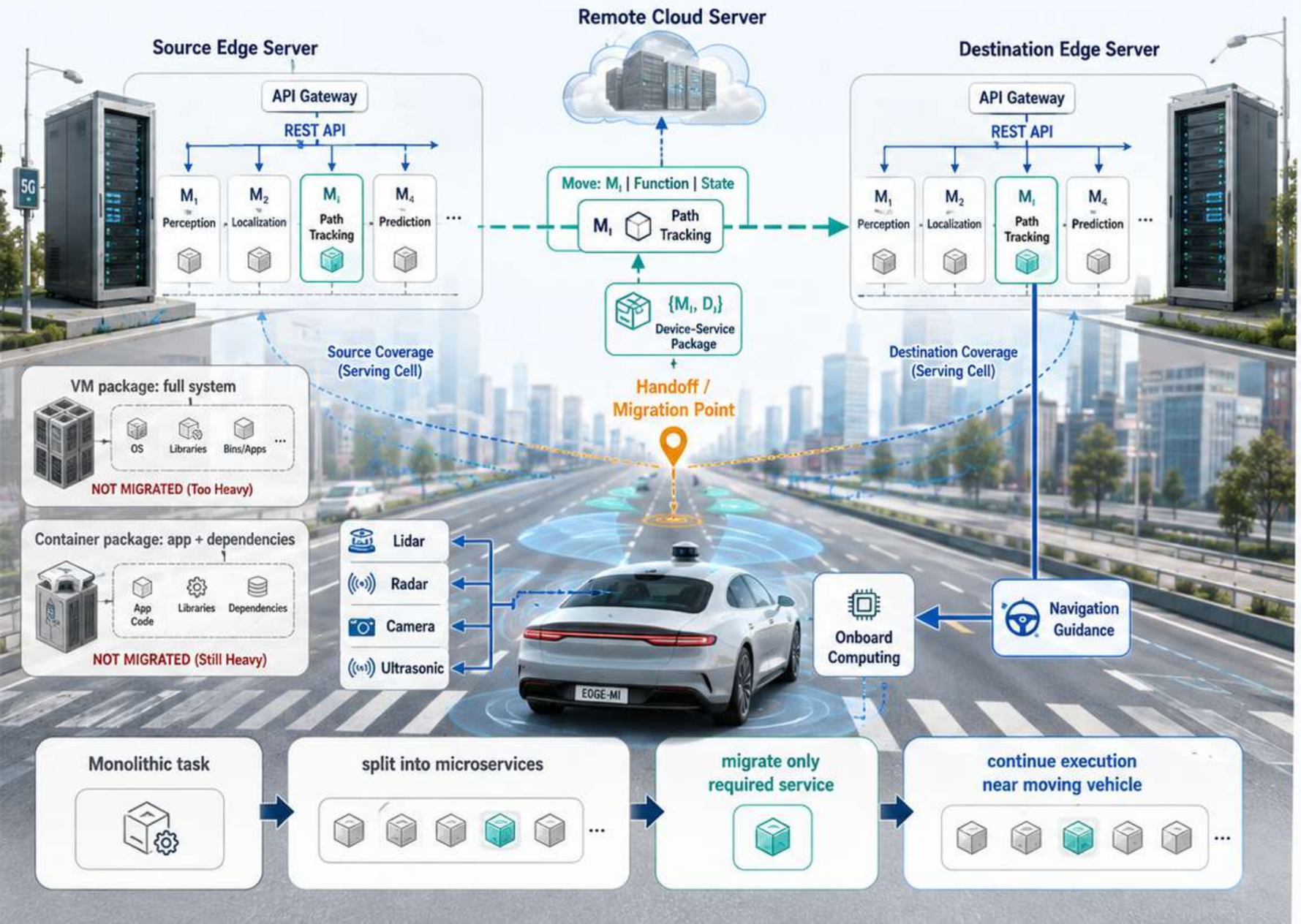
Edge-Mi: Microservice Migration for Mobile Edge Computing

Fine-grained task movement for low-latency autonomous and mobile IoT systems



HIGHLIGHTS OF THE PAPER

01 FINE-GRAINED TASK MIGRATION

Edge-Mi migrates selected microservices and relevant state instead of transferring an entire VM or container.

02 EDGE-TO-CLOUD MICROSERVICE PLACEMENT

Services are placed on the nearest suitable edge server and move higher in the hierarchy only when resources are insufficient.

03 PROACTIVE MOBILITY SUPPORT

Migration starts before handoff at an established migration point, followed by network reconfiguration.

04 EXPERIMENTAL FINDINGS

The microservice approach shows the lowest latency, energy consumption, network usage and migration count in the presented simulation.

Collision avoidance in autonomous vehicles is used as the main illustrative application.

Move the smallest required service, not the complete computing environment.

Edge-Mi: Edge-based Microservices for Mobility-aware Task Migration Scheme

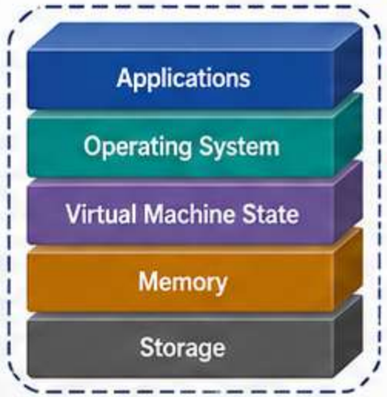
Slide 2

Why Migrate a Microservice Instead of a VM?

Smaller migration units reduce transfer and startup overhead

VIRTUAL MACHINE

VM Migration



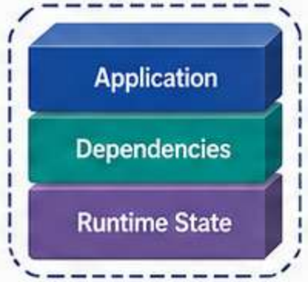
- Largest Package
- Highest Overhead
- Slow Startup
- Strong Isolation

Hypervisor / Full System Environment

Comprehensive but coarse-grained, bandwidth-intensive and slow.

CONTAINER

Container Migration



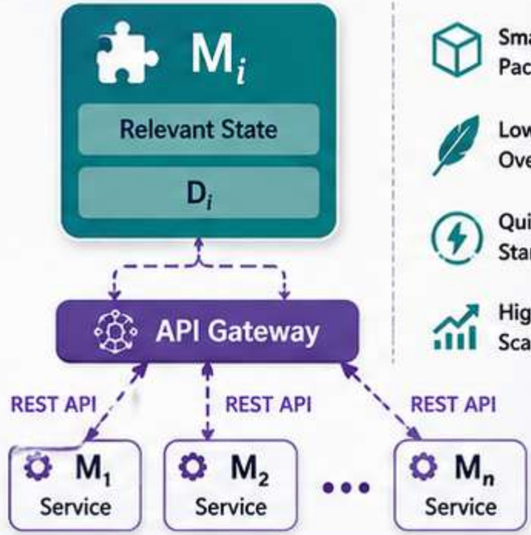
- Medium Package
- Moderate Overhead
- Fast Startup
- High Isolation

Shared Host Operating System

Lighter than a VM, but still carries an application environment and dependencies.

MICROSERVICE

Edge-Mi Microservice Migration



- Smallest Package
- Low Overhead
- Quick Startup
- High Scalability

Move only the required function and its relevant state.

RESEARCH GAP

Previous mobility work largely studies VM and container migration.

The paper addresses microservice-based migration in mobility-aware edge computing.

	Granularity	Speed	Resource Overhead	Scalability	Maintenance
Microservice-based Migration	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
Container-based Migration	● ● ○	● ● ○	● ● ○	● ● ○	● ● ○
VM-based Migration	● ○ ○	● ○ ○	● ○ ○	● ○ ○	● ○ ○

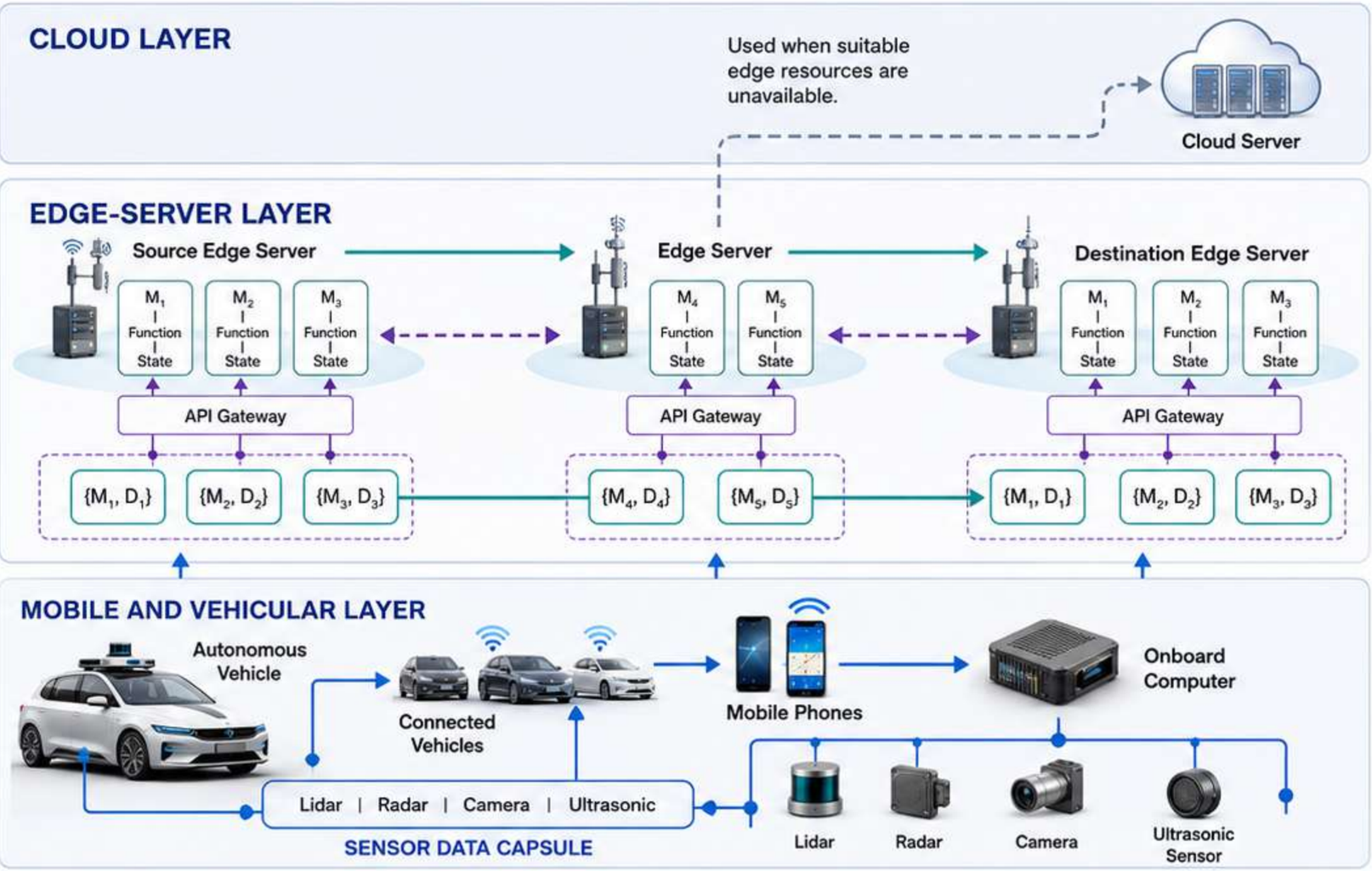
- ● Strongest / Most Flexible
- ● Middle Option
- ● Highest Migration Burden

The migration unit determines *how much data, time and energy the system must spend.*

Inside the Edge-Mi Architecture

Place each service near its data source and migrate it when mobility requires

1. Vehicle and sensors generate data.
2. Break the application into independent microservices.
3. Identify candidate edge-to-cloud placement paths.
4. Select a microservice whose predecessors are already placed.
5. Try the nearest suitable edge server.
6. If resources are insufficient, consider another edge server or cloud.
7. Repeat until all services are placed.
8. Create service-discovery information.
9. Monitor the vehicle's movement.
10. Migrate the required service when the vehicle approaches another edge region.



Placement Principle

Keep services as close as possible to their data source.

Dependency Rule

Place a service only after its predecessor services are available.

Migration Unit

$\{M_i, D_i\}$

Required microservice, state and associated task data.

Sensor / Normal Execution Path

Successful Microservice Placement / Migration

Service Discovery / API Links

Resource Insufficiency (Consider Another / Cloud)

Cloud Fallback Path (Upper Hierarchy)

Edge-Mi combines service placement, service discovery and mobility-aware migration.

How Edge-Mi Moves a Collision-Avoidance Service

Proactive migration begins before the vehicle completes its handoff

1 CONTINUOUS SENSING

The onboard system predicts trajectory and evaluates collision risk.

SENSOR DATA CAPSULE
Lidar | Radar | Camera | Ultrasonic

2 EDGE DISCOVERY

Is a nearby edge server available?

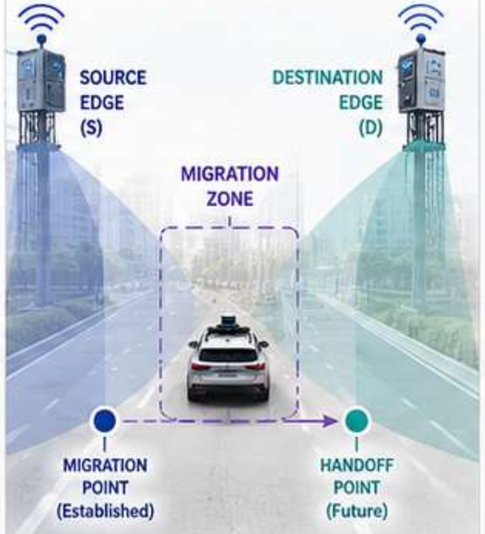
NO → Continue onboard processing

YES → Evaluate migration conditions

SENSOR DATA CAPSULE
Lidar | Radar | Camera | Ultrasonic

3 MIGRATION ZONE

Edge-Mi uses proactive migration, beginning before the handoff starts.



SENSOR DATA CAPSULE
Lidar | Radar | Camera | Ultrasonic

4 PREPARE MIGRATION

COLLISION-AVOIDANCE MICROSERVICE

M_i | Function | State

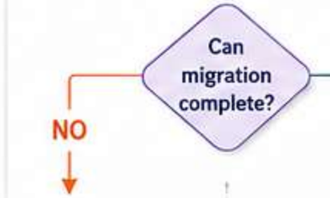
DEVICE–SERVICE PACKAGE

$\{M_i, D_i\}$

1. Package microservice state
2. Package relevant sensor data
3. Check package size
4. Check network speed
5. Create checkpoint
6. Begin data transfer

SENSOR DATA CAPSULE
Lidar | Radar | Camera | Ultrasonic

5 FAILURE-SAFE DECISION



NO → Stop migration → retain microservice at Source Edge

SOURCE EDGE (S)

Collision-Avoidance Microservice (M_i)
● ACTIVE

Service remains active at the source.

YES → Continue transfer → reconfigure network

NETWORK RECONFIGURATION

Traffic paths are updated to the destination edge.

6 DESTINATION EXECUTION

DESTINATION EDGE (D)

Collision-Avoidance Microservice (M_i)
● ACTIVE

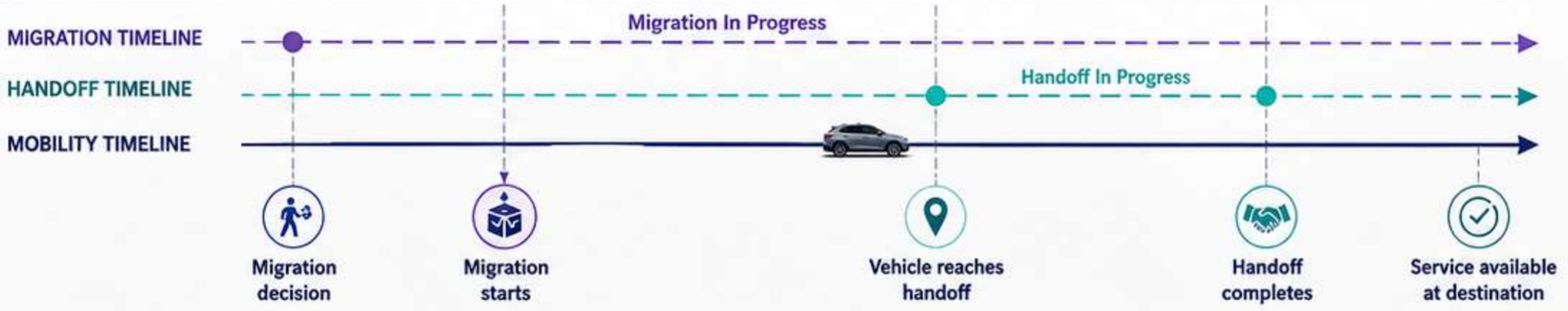
Sensor data received at edge

Collision-avoidance analysis

NAVIGATION GUIDANCE

The vehicle follows a safer trajectory.

MIGRATION TIMELINE



MIGRATION STRATEGIES

EDGE-MI

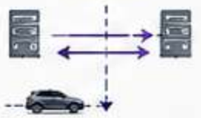
REACTIVE

Starts after handoff

PROACTIVE

Starts before handoff

HYBRID

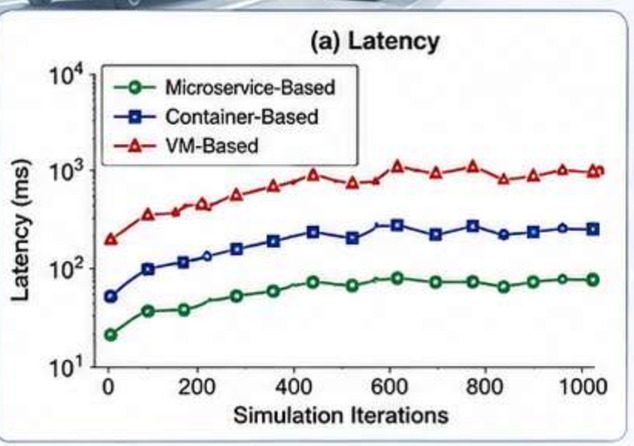


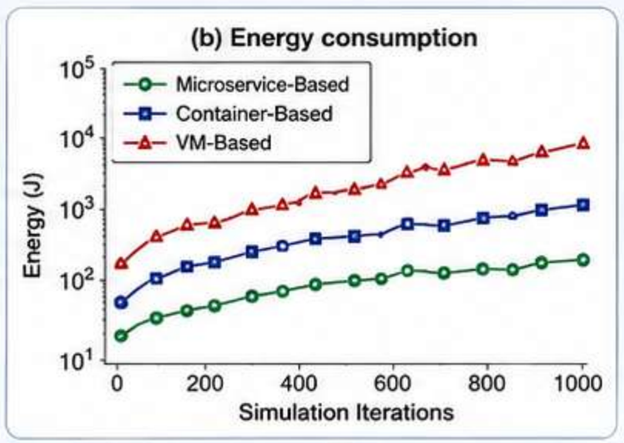
Migration and handoff overlap

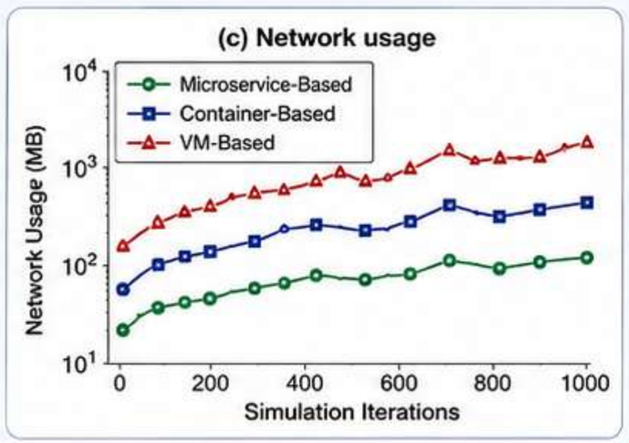
Move the collision-avoidance service before the vehicle leaves the current edge region.

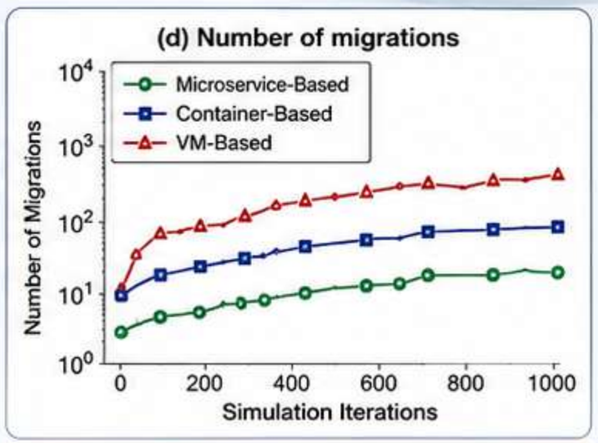
What the Edge-Mi Evaluation Shows

Microservice migration compared with container and VM migration









RESULT AREA 1 — Task Latency

Fine-grained migration moves less state and shortens transfer and startup time.

Microservice-based migration shows the lowest latency.

RESULT AREA 2 — Energy Consumption

Smaller migration payloads reduce communication and processing activity.

Microservice-based migration uses the least energy.

RESULT AREA 3 — Network Usage

Only the selected service and its relevant state are transferred.

Microservice-based migration creates the lowest network load.

RESULT AREA 4 — Migration Count

Granular service placement reduces unnecessary movement of complete environments.

Microservice-based migration requires the fewest migrations.

Microservice → best-performing curves in all four presented metrics

Container → intermediate performance

VM → highest overhead in the plots

EVALUATION DETAILS (SOURCE-SUPPORTED)

- Python-based simulation
- Microservice-Based
- Container-Based
- VM-Based
- Latency
- Energy
- Network Usage
- Number of Migrations
- Graphs plotted over simulation iterations

Limitations: Performance may degrade when edge servers lack sufficient resources, microservices have strong interdependencies, or edge-server fallecay.

Fine-grained services → Smaller migration package → Proactive transfer → Lower communication and computation overhead → Improved mobile-edge execution

FINAL RESEARCH CONCLUSION

Edge-Mi uses independently deployable microservices to make mobility-aware task migration more flexible, scalable and efficient than VM- and container-centred approaches.

Split the application. Move only the required service. Continue computation near the moving device.