MMLite: A Scalable and Resource Efficient Control Plane for Next Generation Cellular Packet Core

Vasudevan Nagendra

Co-authors:
Arani Bhattacharya, Anshul Gandhi, Samir R. Das

Stony Brook University

Heterogeneity: Devices, Traffic characteristics & SLO requirements

100's of types of devices

Enterprise, Consumer, Industrial

Communication Standards

Challenge: Utilizing resources efficiently while providing SLO requirements

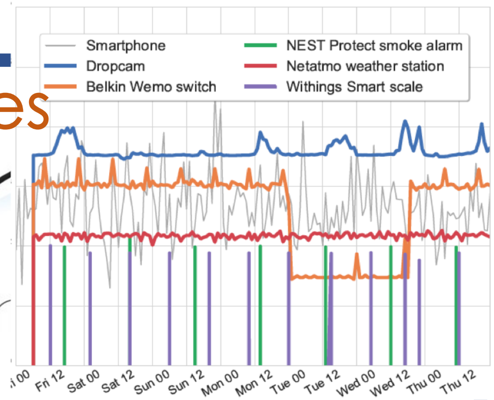
Heterogeneity: Devices, Traffic characteristics & SLO requirements

100's of types of devices

IoT Connection Cost:
Cents, $

Wide range of SLO requirements

Enterprise, Consumer



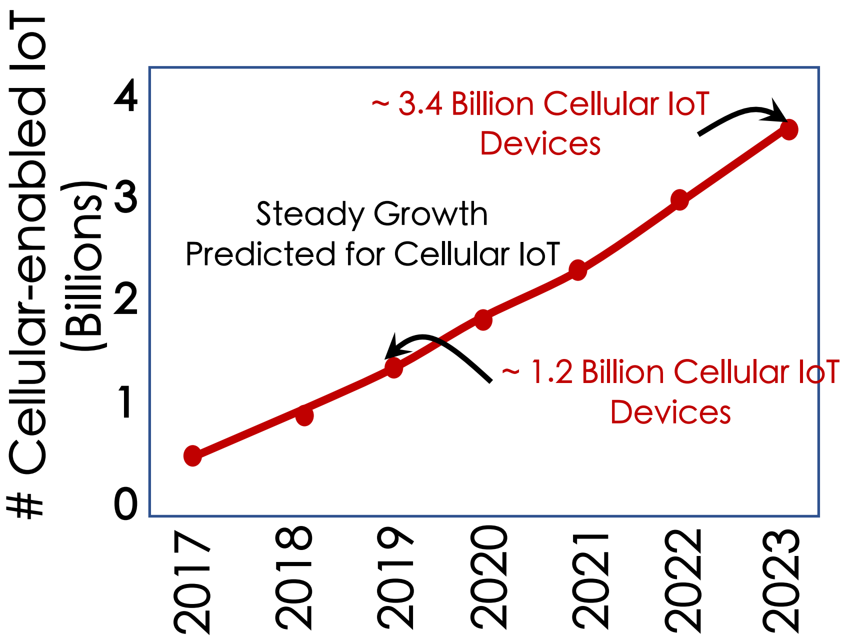
Mobile Vs IoT Traffic:

- Bursty
- Sporadic
- Temporally distributed
- Diurnal
- High Frequent

Communication Standards

Challenge: Utilizing resources efficiently while providing SLO requirements

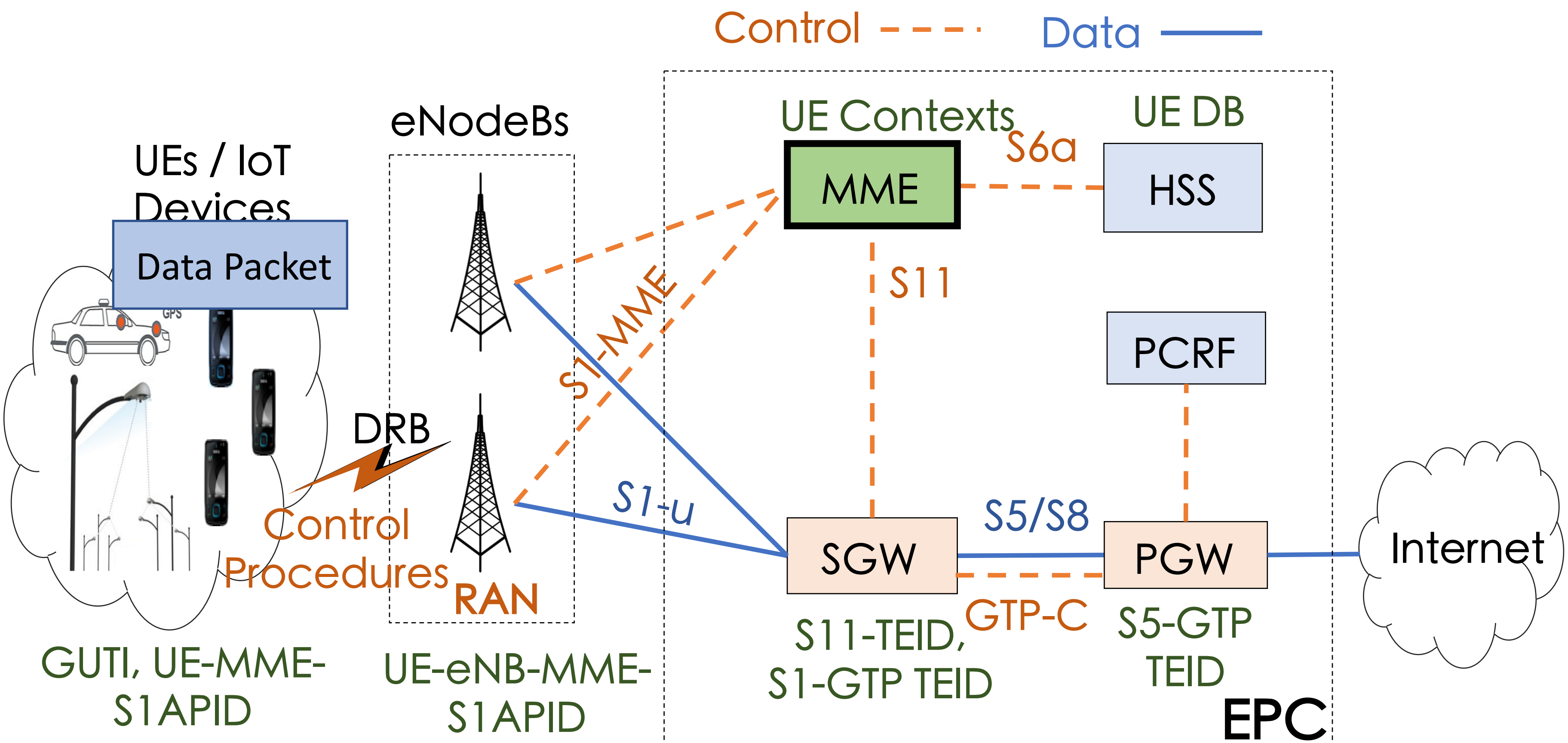
Scale of IoT deployments



Data Courtesy: Ericsson Mobility Report, June 2018 (PDF, 36 pp., no opt-in).

Providing Performance, SLO & Resource Management a challenging in large scale deployments

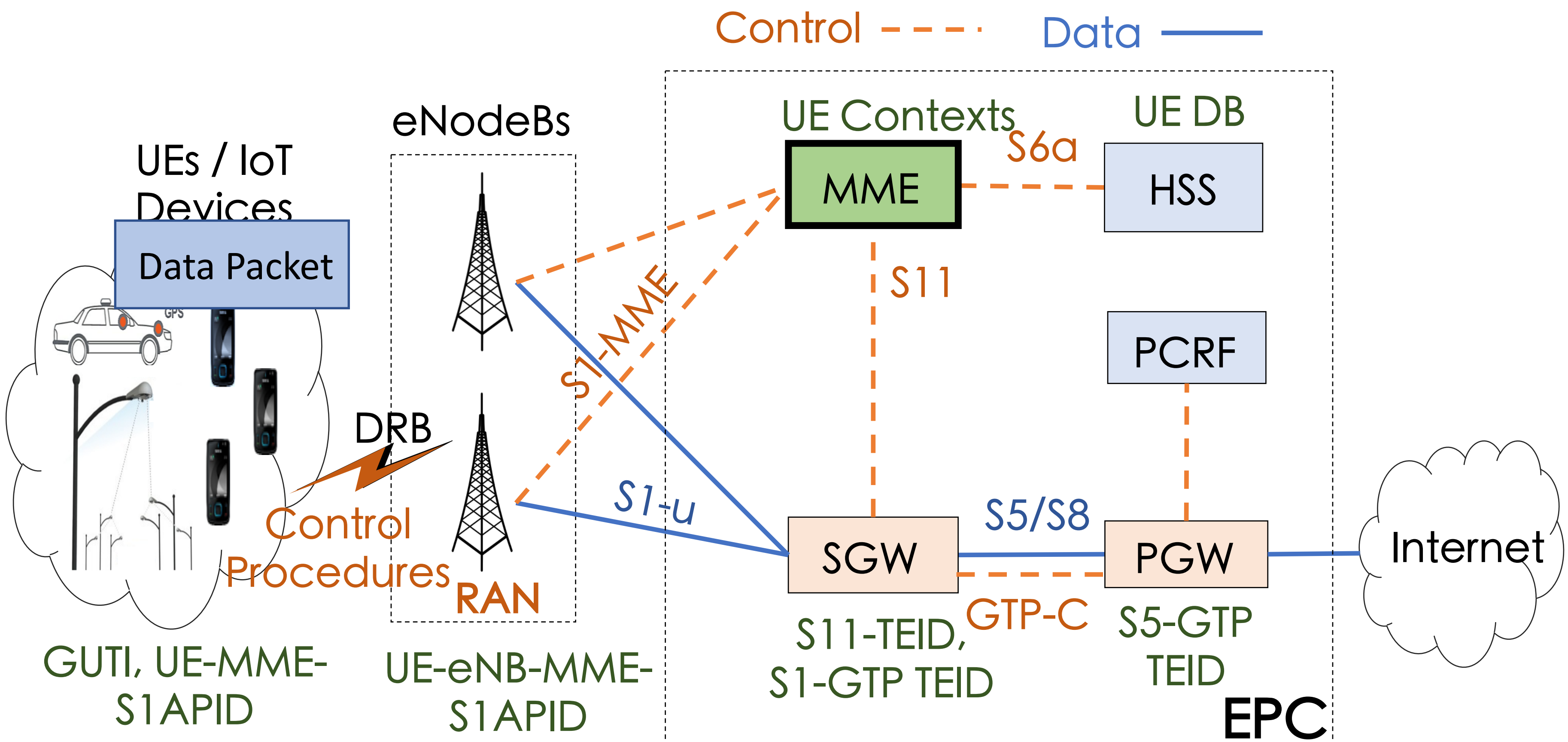
Convoluted Traditional Cellular Core (LTE) with static bindings



Static Bindings & Contextual Information:
Association between UE → eNB → MME

Static Bindings & convoluted cellular architecture makes scaling challenging

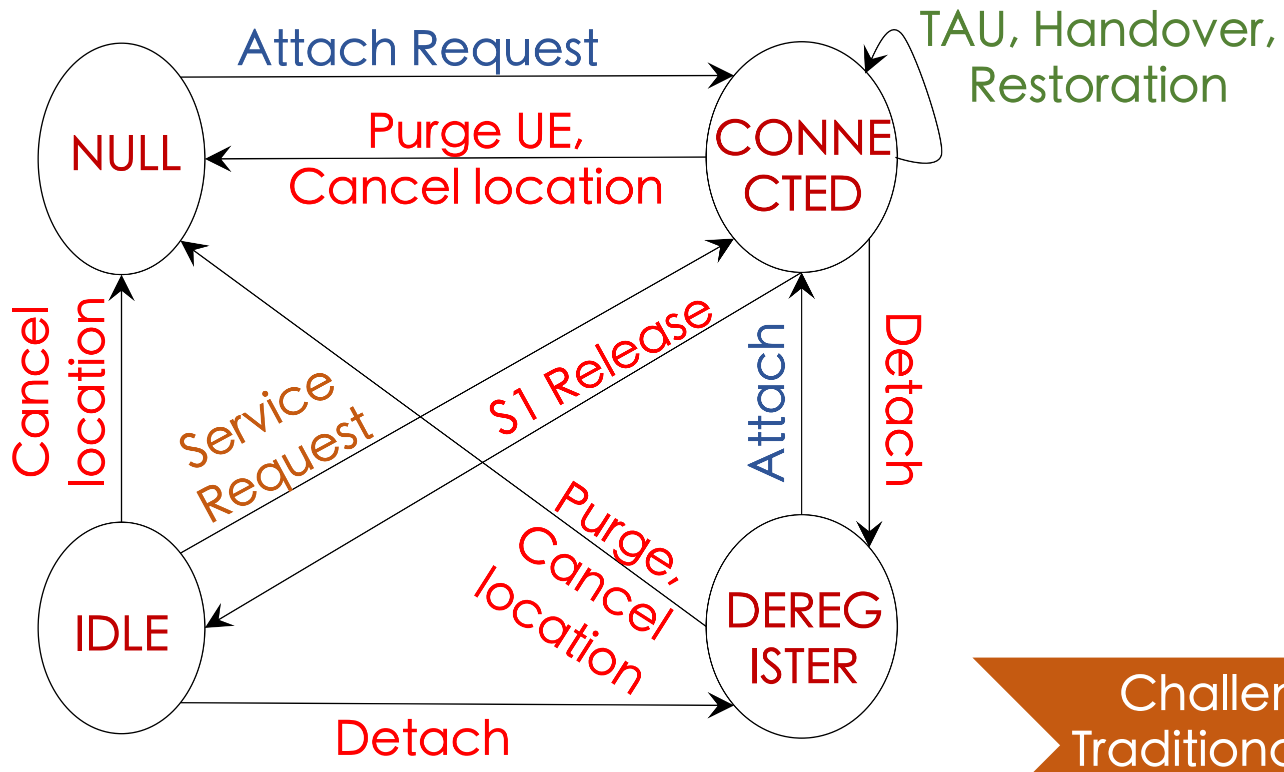
Convoluted Traditional Cellular Core (LTE) with static bindings



MME Handles 5X more messages than other Control elements inside core

Static Bindings & convoluted cellular architecture makes scaling challenging

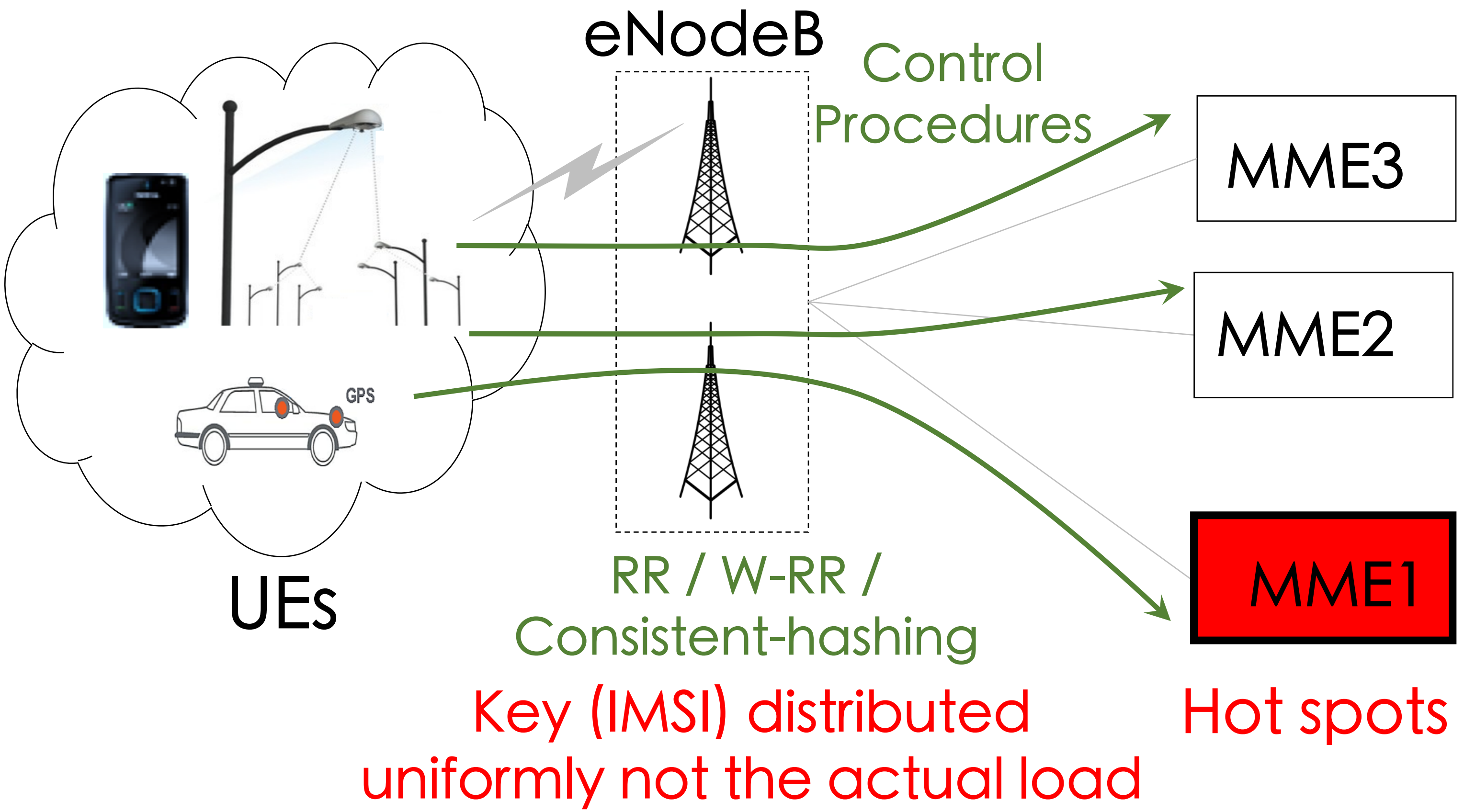
Background: LTE Control plane



Challenges Of Traditional Cellular Core

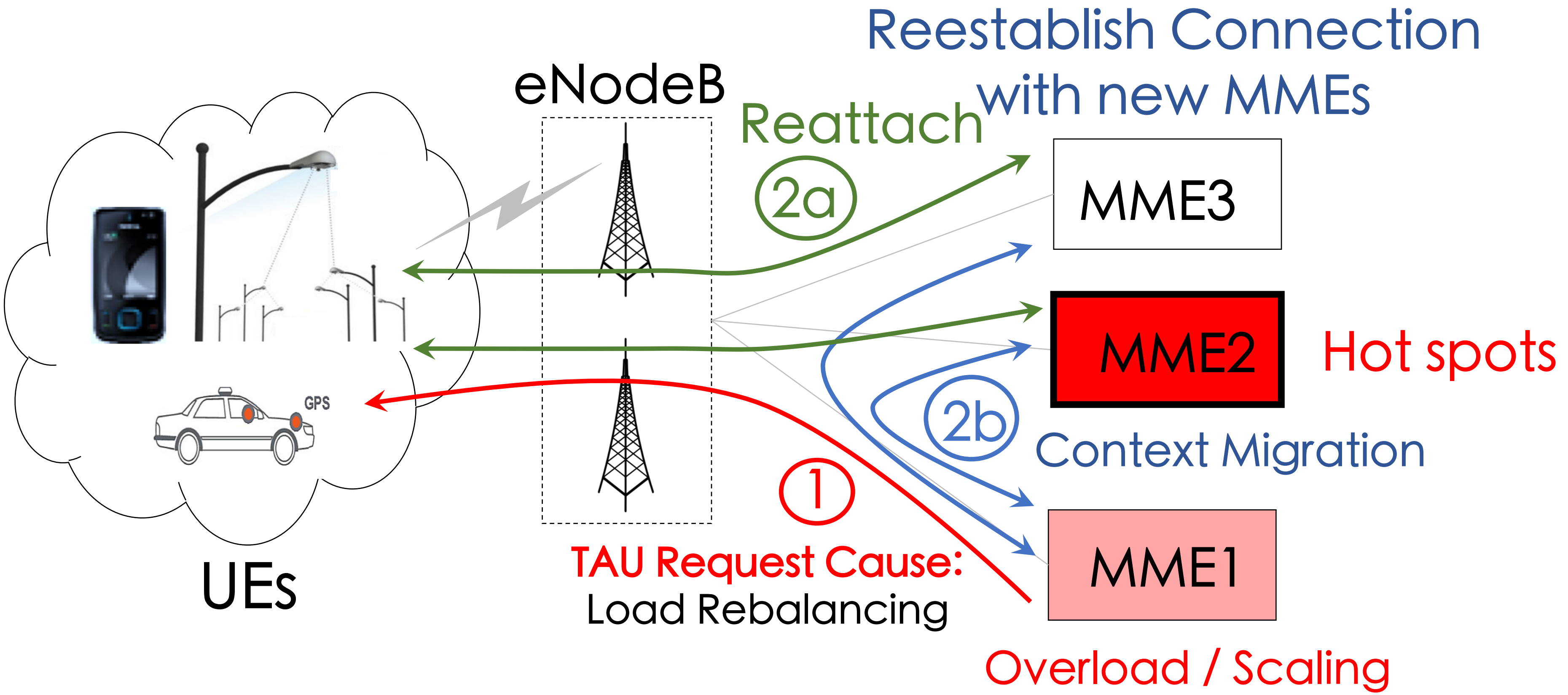
LTE Control Procedures & FSM

Traditional MME Load Redistribution (1)



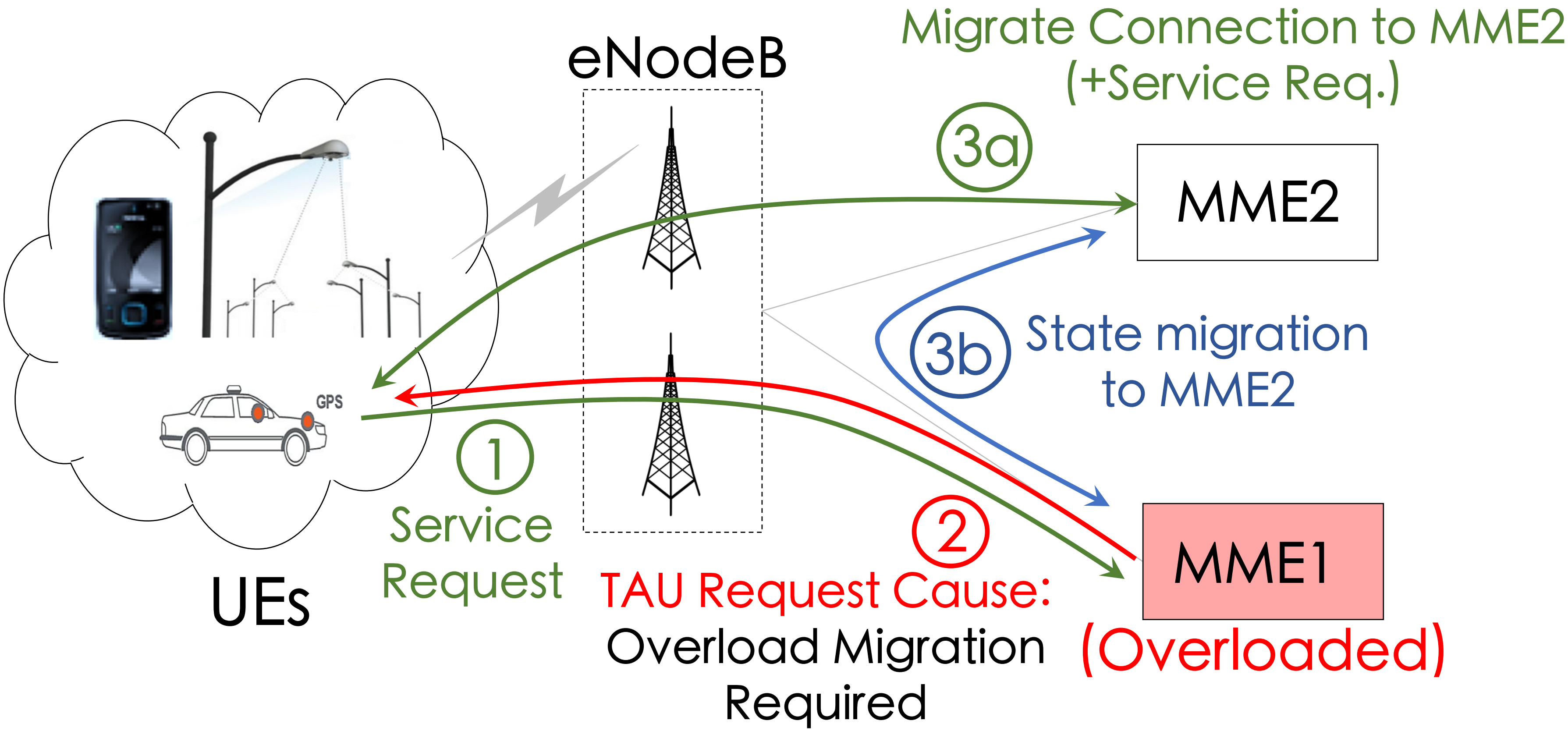
Unequal distribution of load across MMEs leading to hot spots.

Load Redistribution in overload / scaling scenarios (2)



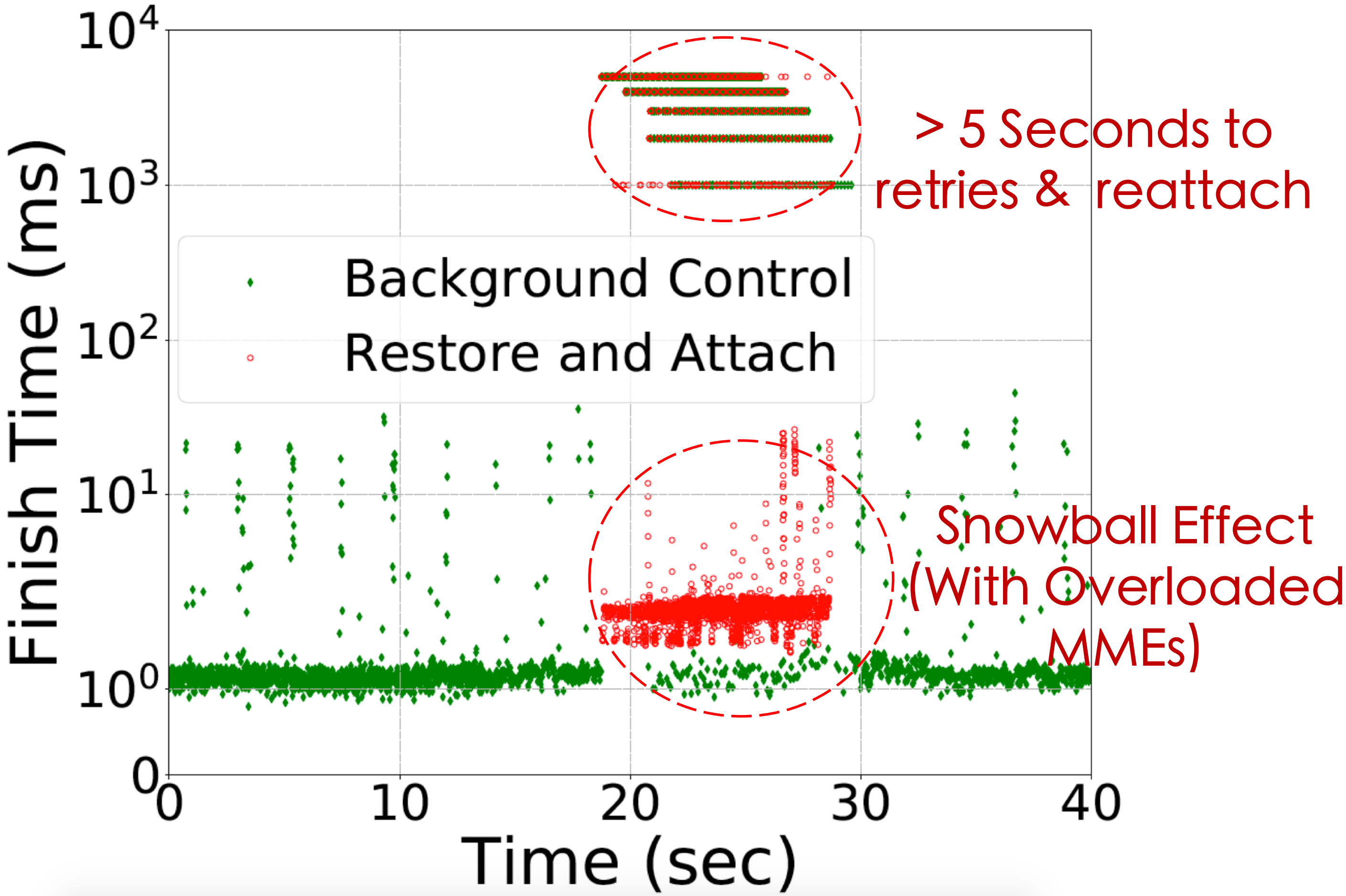
Create Hot spots & SLO violations

Traditional Overload Protection Mechanism



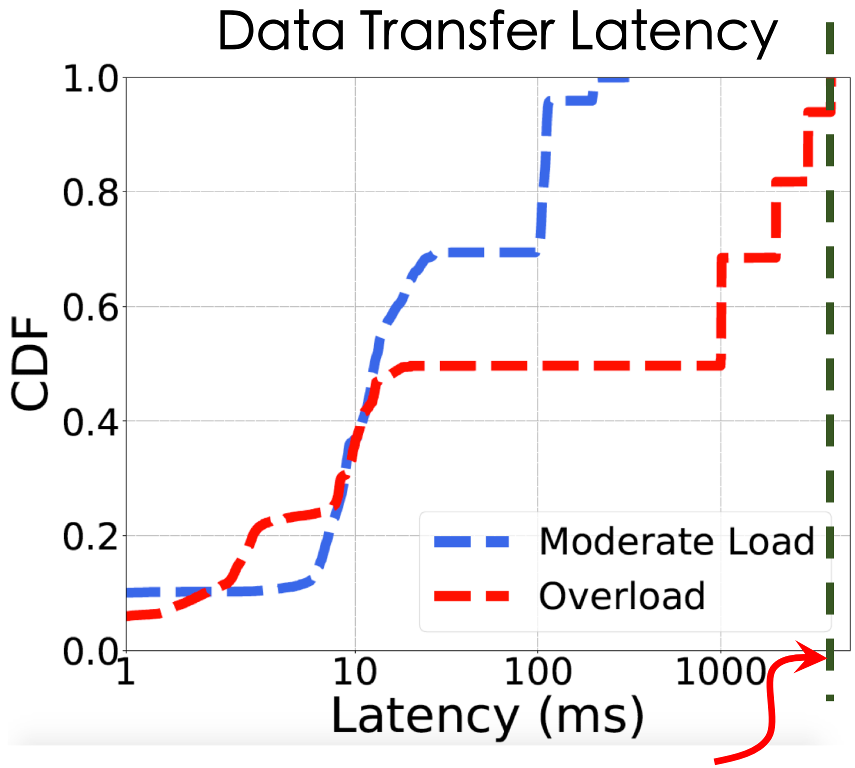
SLO violations to control procedure & data transmission delays

Limitations: Overload & Inefficient load distribution



Control procedure SLO deterioration & Message flooding at MMEs

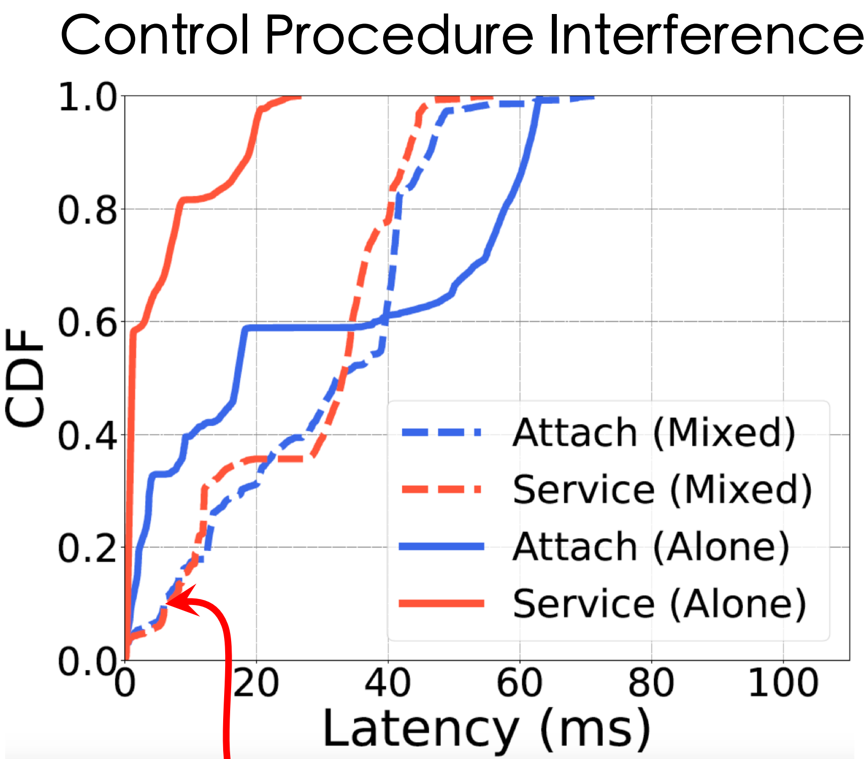
Limitations: Overload & Inefficient load distribution



> 5 Seconds to Initiate Data Transfer

Control procedure completion times impacting data transfer latency

Limitations: Overload Protection & Inefficient load distribution



Interference among control procedures (Impact on Service request latency)

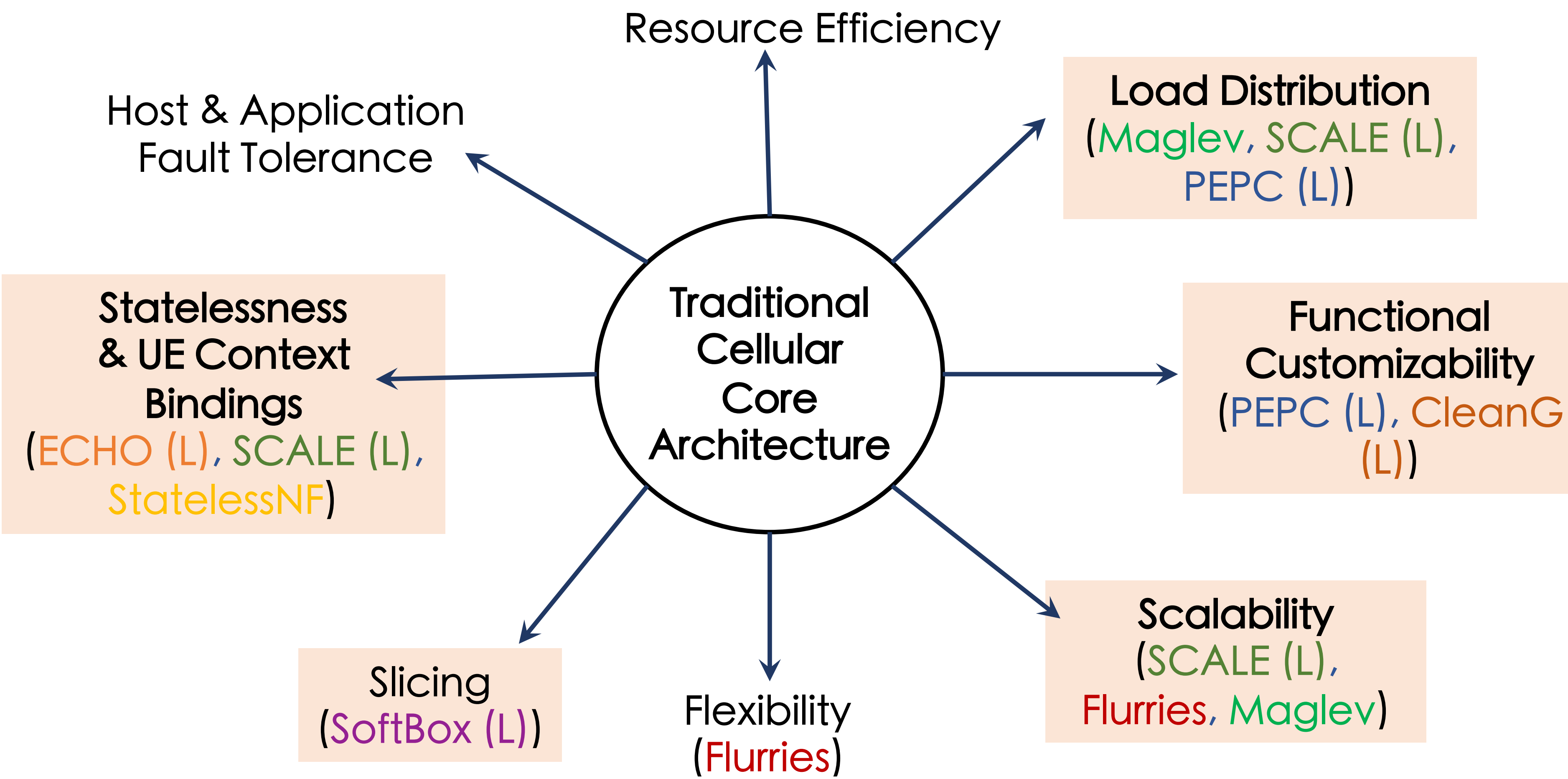
Control procedure completion times impacting data transfer latency

Recap: Limitations of Traditional Cellular Core

- Inefficient Resource Assignment
  - Specific to IoT Traffic characteristics
  - 20 – 30$ connections (Mobile) Vs few cents to few $ connections (IoT)
- Interference & SLO Violations
- Inefficient Scaling.
- Inefficient Load-balancing

How MMLite handles such challenges?

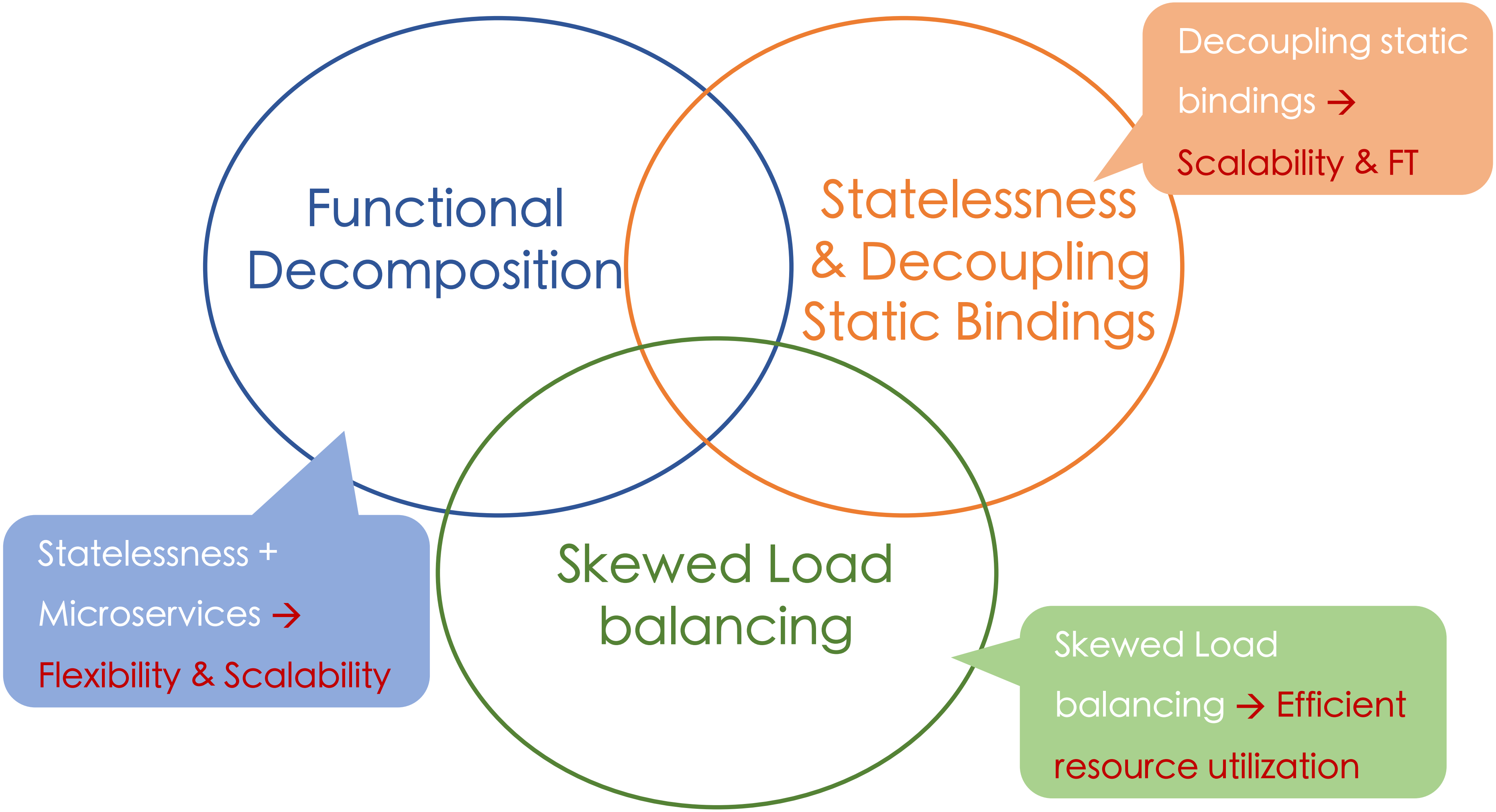
Literature: Along multiple dimensions



(L) Indicates work done in LTE/Cellular space.

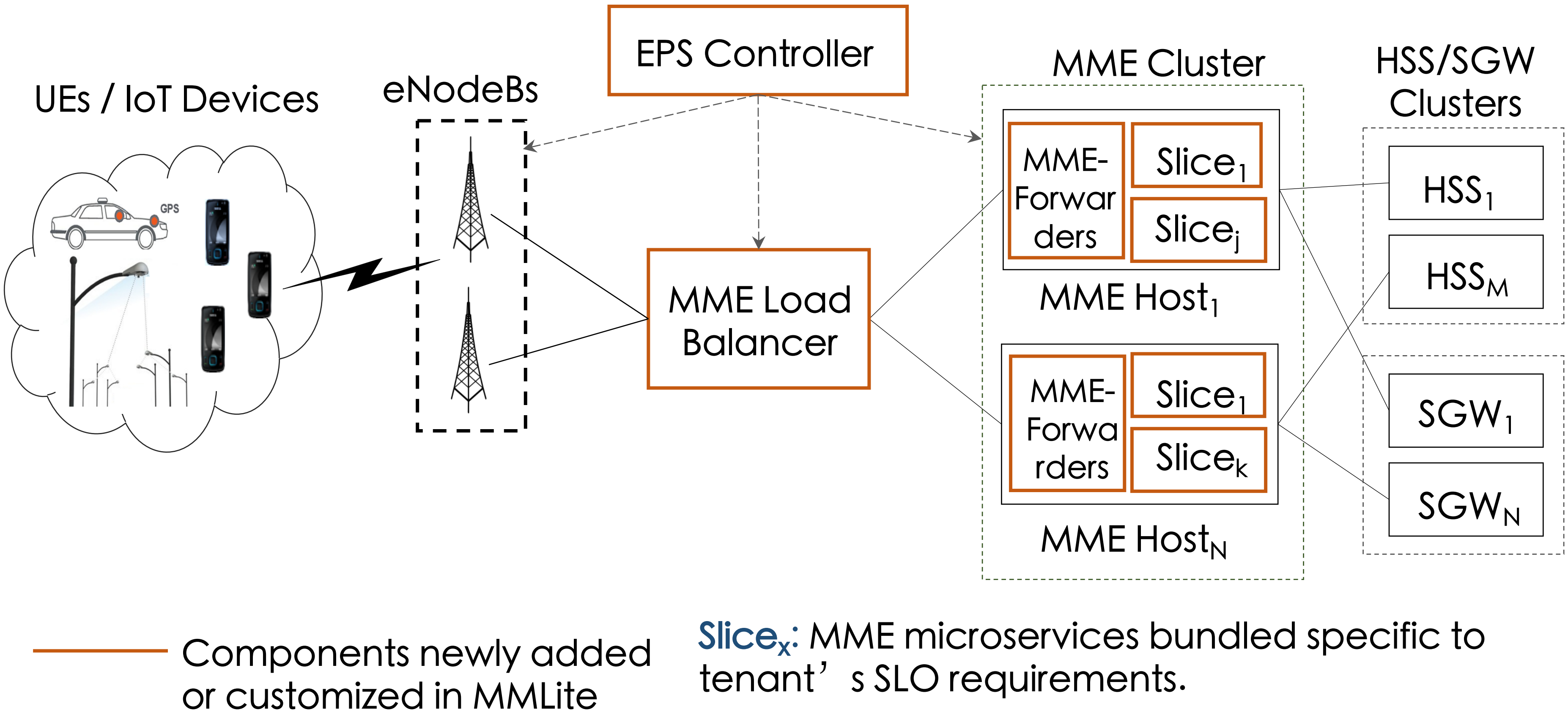
Revisit cellular core architecture for IoT & next generation applications (MMLite)

Our Approach (MMLite)



Scalability & Resource Efficiency with MMLite

Overall MMLite Cellular Core Architecture

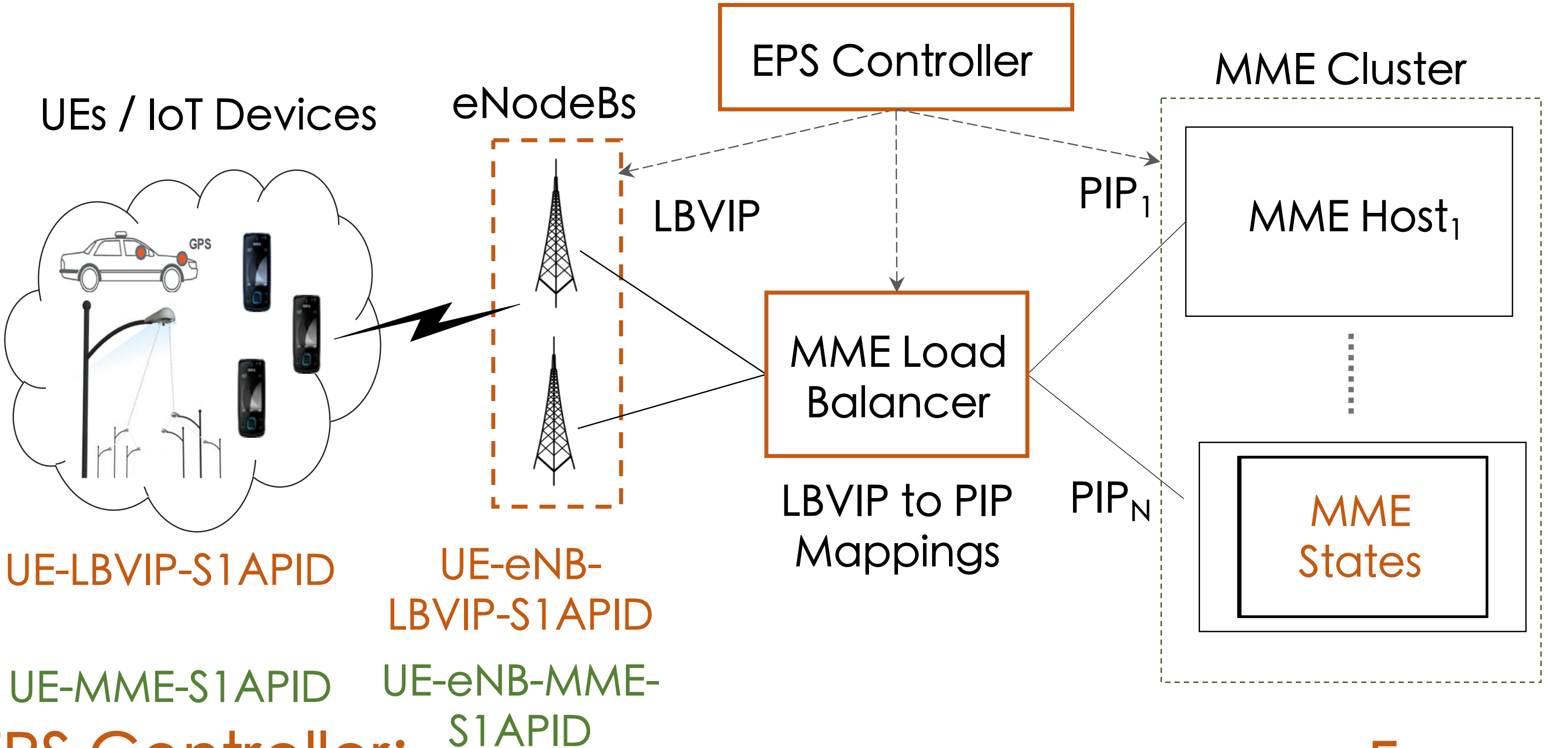


$Slice_x$: MME microservices bundled specific to tenant's SLO requirements.

Microservice: MME specific to each control procedure

Scalability, Flexibility, Fault Tolerance & Resource Efficiency

1. Decoupling Static Bindings & Statelessness for Flexibility & Scaling



EPS Controller:

MME hosts scaling on the basis of:

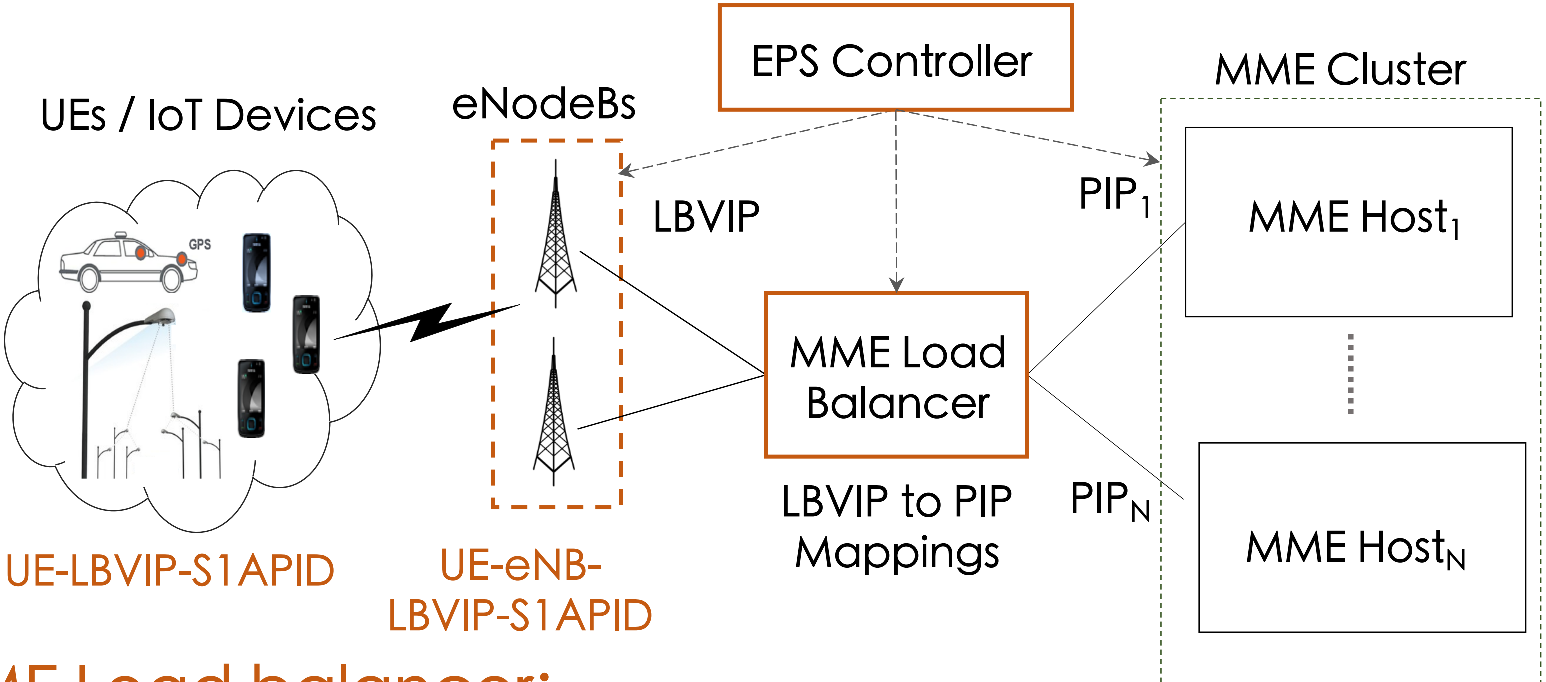
- SLO violations
- Resource requirements

For Statelessness

States stored in shared memory datastore & constantly migrated to replicas

Achieve Flexibility, Scaling & Enhances Fault Tolerant

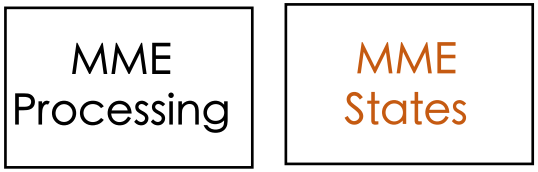
1. Decoupling Static Bindings & Statelessness for Flexibility & Scaling



MME Load balancer:

Helps in Efficient Distribution to MME Host

- MME resources
- SLO requirements
- Slice requirements



For Statelessness

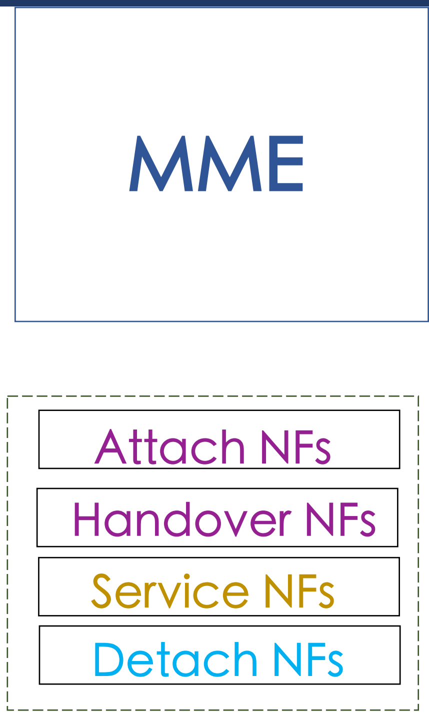
States stored in shared memory datastore & constantly migrated to replicas

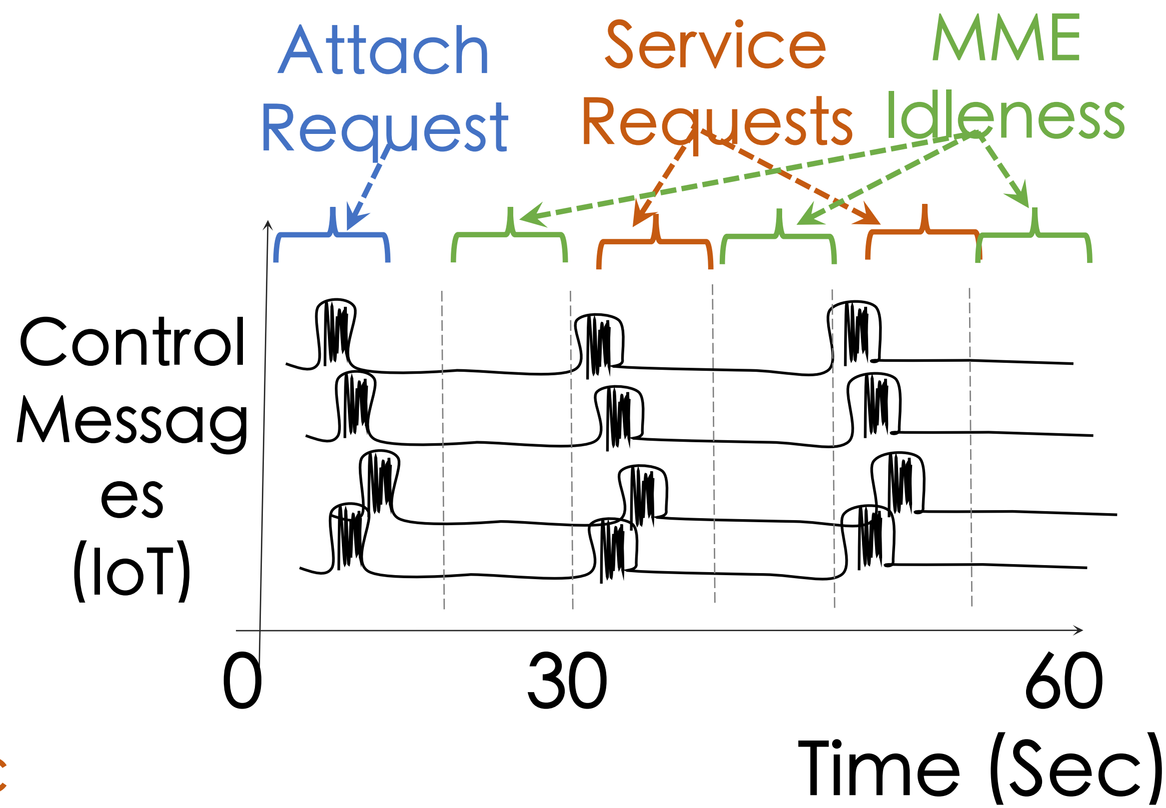
Achieve Flexibility, Scaling & Enhances Fault Tolerant

State Migration Techniques

- Cold Migration
  - Upon completion control procedure
- Hot Migration
  - With each control message
- MMLite Uses both the techniques.

2. Functional Decomposition: MME as microservices



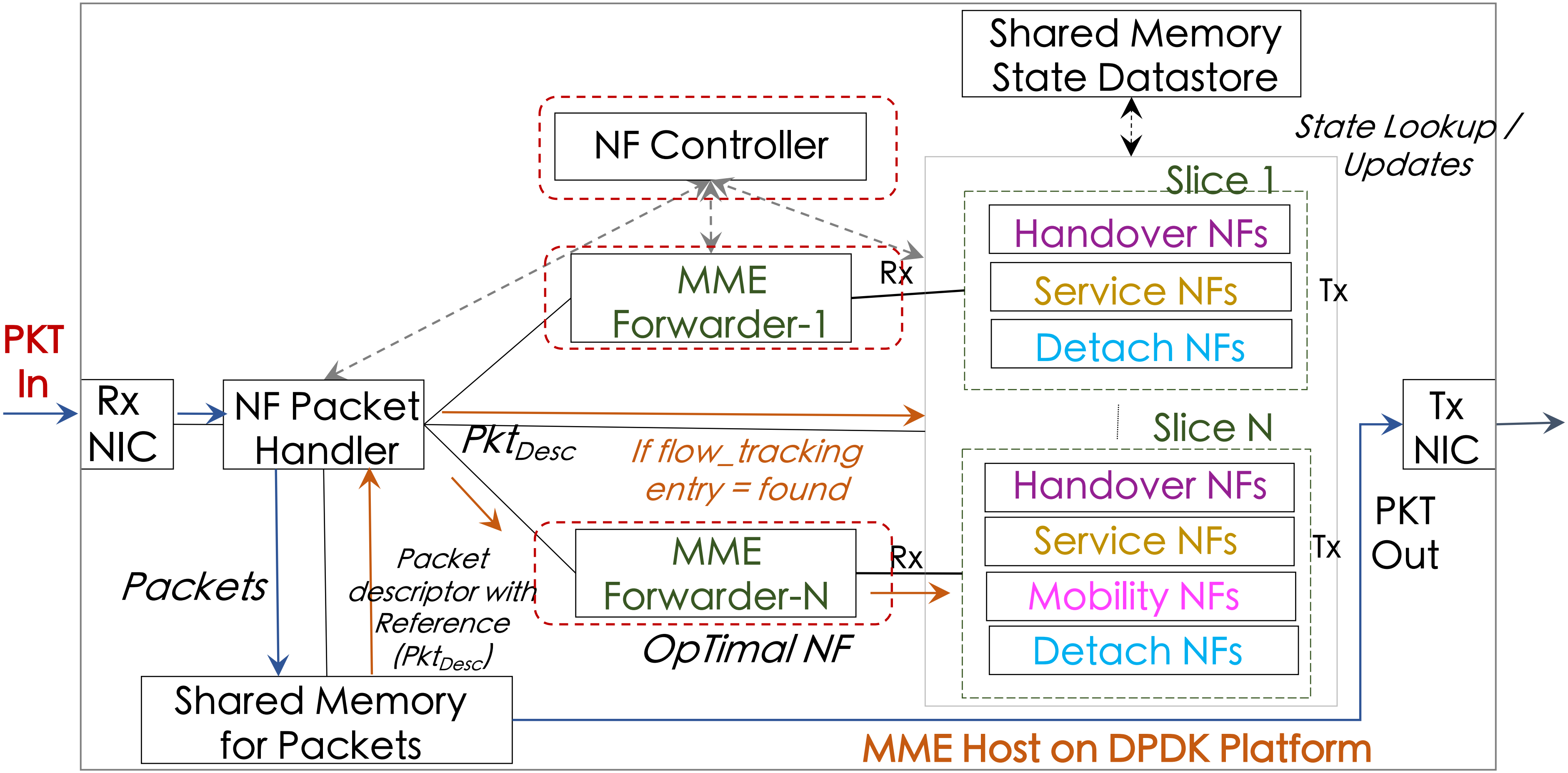


Decomposed into Procedure-specific Microservices

(Vertical Decomposition)

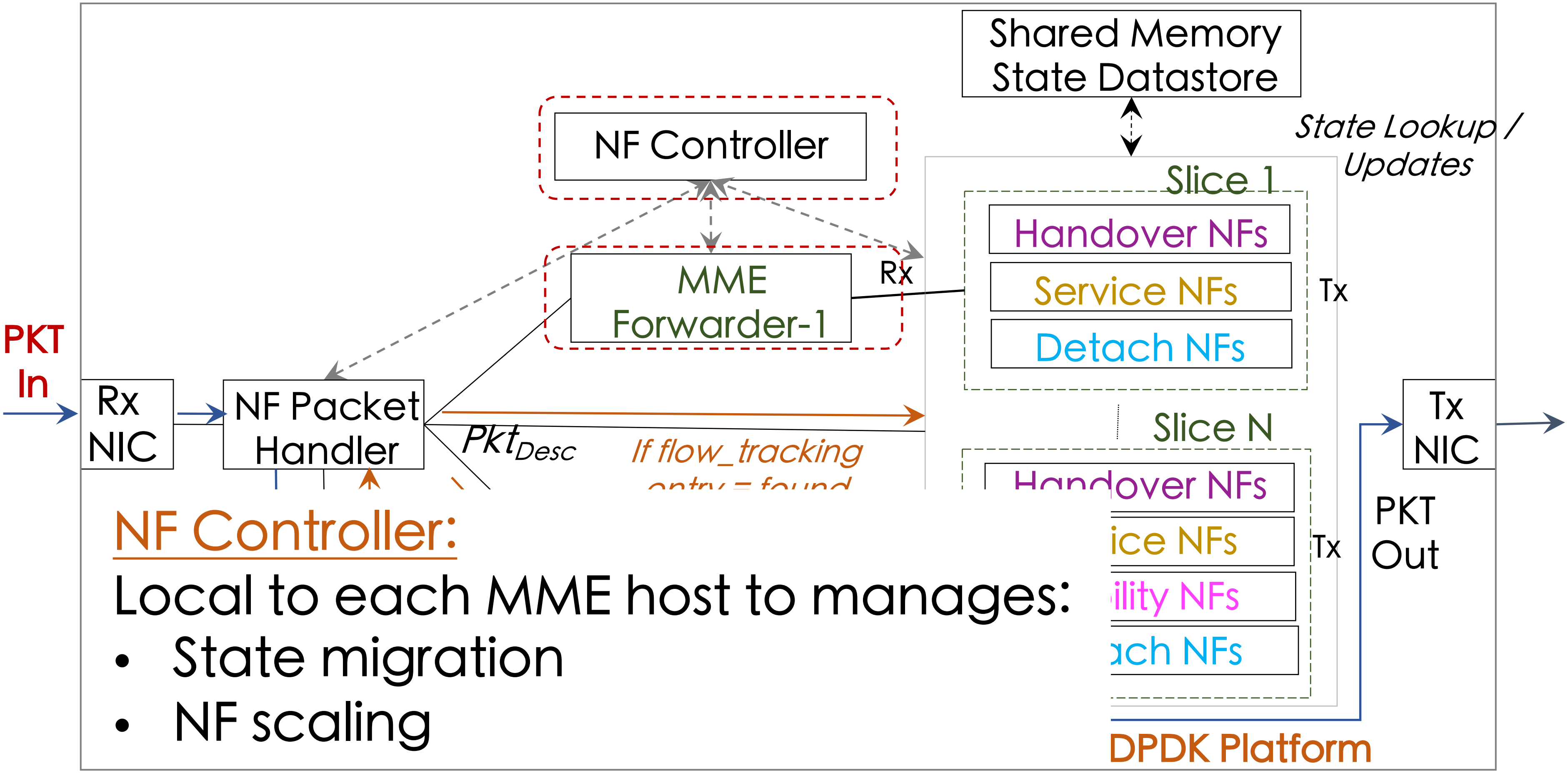
Flexibility & Scaling

2. Functional Decomposition: MME as microservices



Flexibility, Scaling & Fault Tolerant

2. Functional Decomposition: MME as microservices



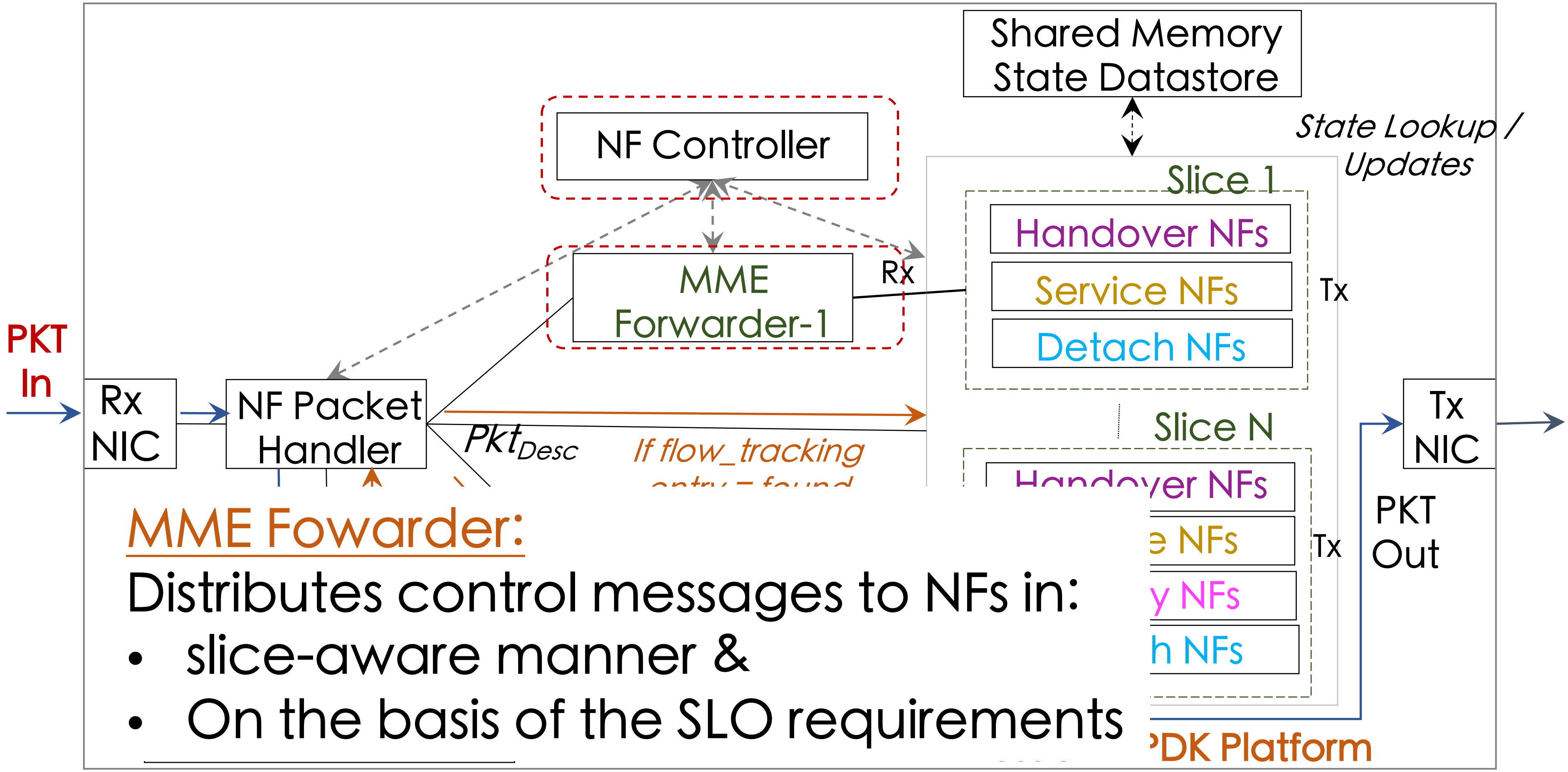
NF Controller:

Local to each MME host to manages:

- State migration
- NF scaling

Flexibility, Scaling & Fault Tolerant

2. Functional Decomposition: MME as microservices



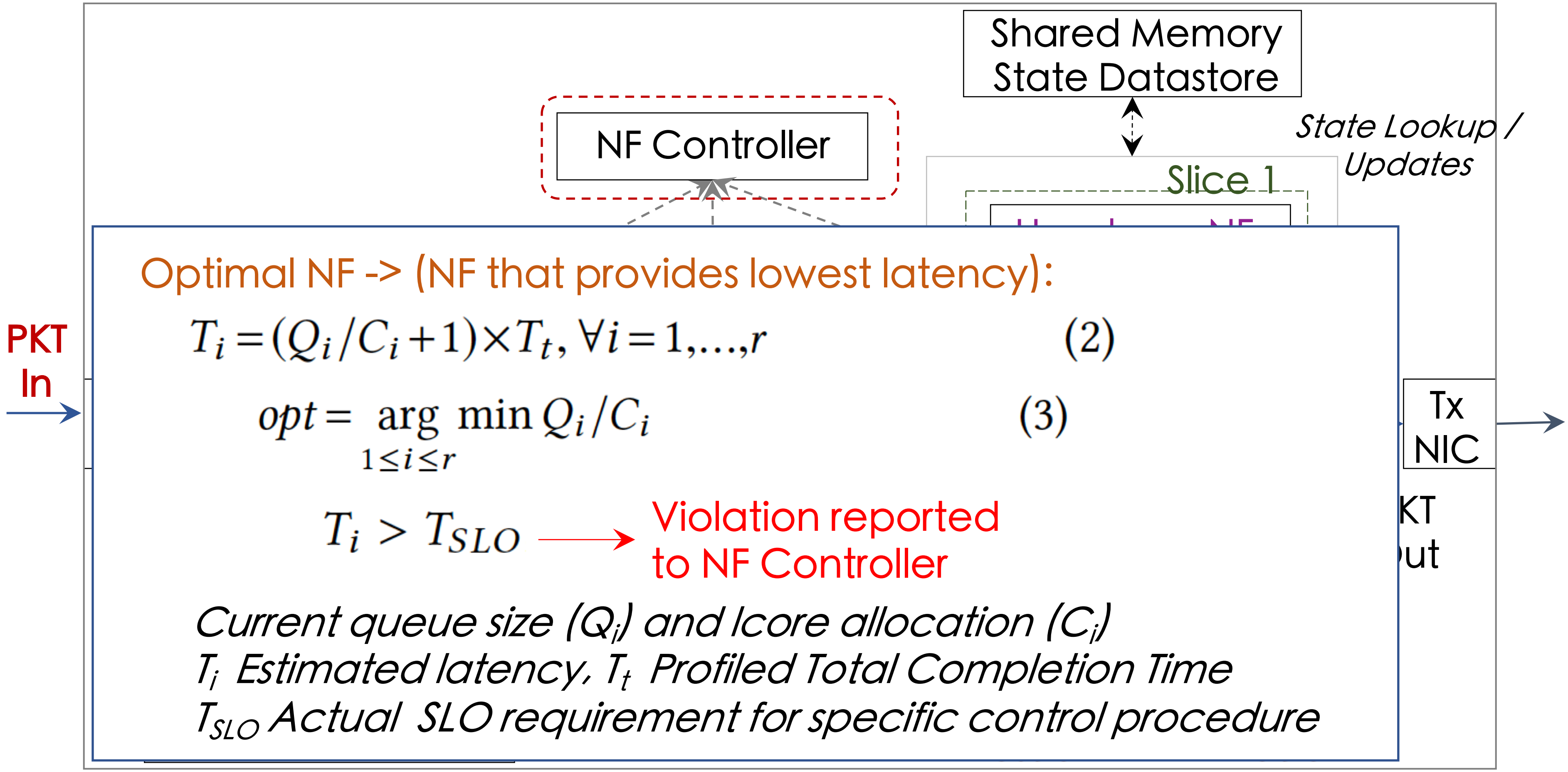
MME Fowarder:

Distributes control messages to NFs in:

- slice-aware manner &
- On the basis of the SLO requirements

Flexibility, Scaling & Fault Tolerant

2. Functional Decomposition: MME as microservices



Optimal NF -> (NF that provides lowest latency):

$$T_i = (Q_i / C_i + 1) \times T_t, \forall i = 1, \ldots, r \qquad (2)$$

$$opt = \underset{1 \leq i \leq r}{\arg\min}\, Q_i / C_i \qquad (3)$$

$T_i > T_{SLO}$ → Violation reported to NF Controller

Current queue size (Q_i) and lcore allocation (C_i)
T_i Estimated latency, T_t Profiled Total Completion Time
T_{SLO} Actual SLO requirement for specific control procedure

Flexibility, Scaling & Fault Tolerant

2. Functional Decomposition: MME as microservices

Open Problem:

Identifying what is the right decomposition for other EPC entities (middleboxes used in EPC)?

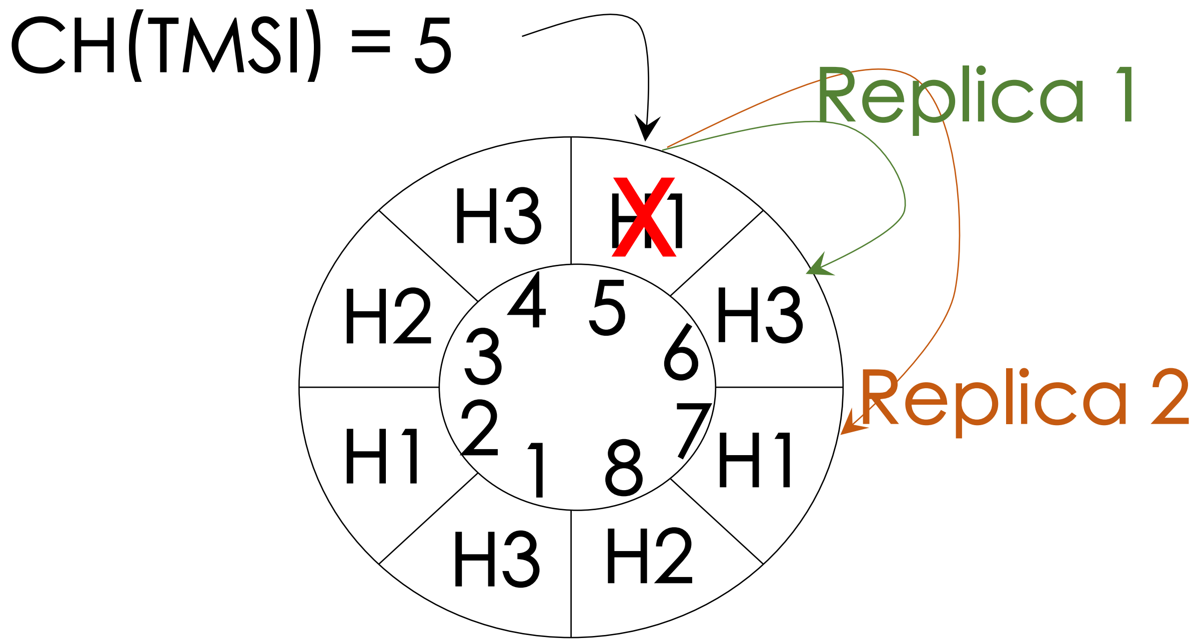
- Reduced code footprint and states
- Optimizing for different requirements
  - SLO, Flexibility & resources

(V

Flexibility & Scaling

3. Slice-aware Skewed Consistent Hashing for Efficient Load Distribution

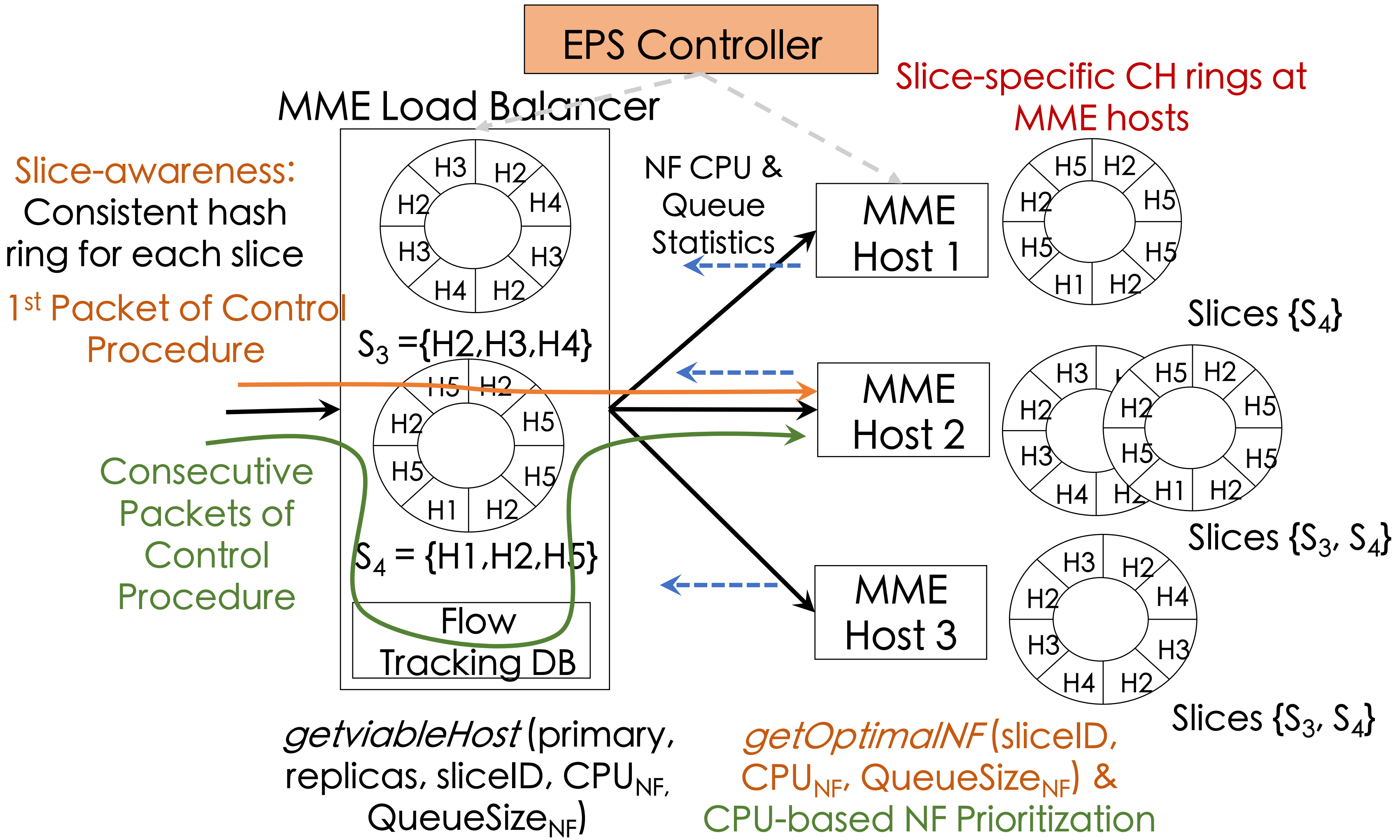
Consistent Hashing



Primary MME Host

Resource Efficient Traffic distribution

3. Slice-aware Skewed Consistent Hashing for Efficient Load Distribution



Resource Efficient Traffic distribution with SK-CH

(Slice-awareness + Viable Host + Optimal NF)

3. Slice-aware Skewed Consistent Hashing for Efficient Load Distribution

EPS Controller

MME Load Balancer

Slice-specific CH rings at MME hosts

Slice-awareness:

H3 H2

NF CPU &

H5 H2

Viable MME host -> (viable hosts as those that contain at least one NF i for which $T_i \leqslant T_{SLO}$ & Highest CPU Utilization):

$$W_i = (q_i \cdot T_t)/c_i \ \forall i = 1, \ldots, m$$

Counter Intuitive (But helps scaling down)

$$T_i = W_i + T_t = \left(\frac{q_i}{c_i} + 1\right) \times T_t.$$

$$T_i > T_{SLO}$$ → Violation reported to EPS Controller

T_t is the time for a procedure when run in isolation,

q_i moving average of queue sizes for Nf_i and CPU assigned (c_i),

T_i Estimated latency, W_i Waiting time for the procedure

T_{SLO} Actual SLO requirement for specific control procedure

{S_4}

{S_3, S_4}

S_3, S_4}

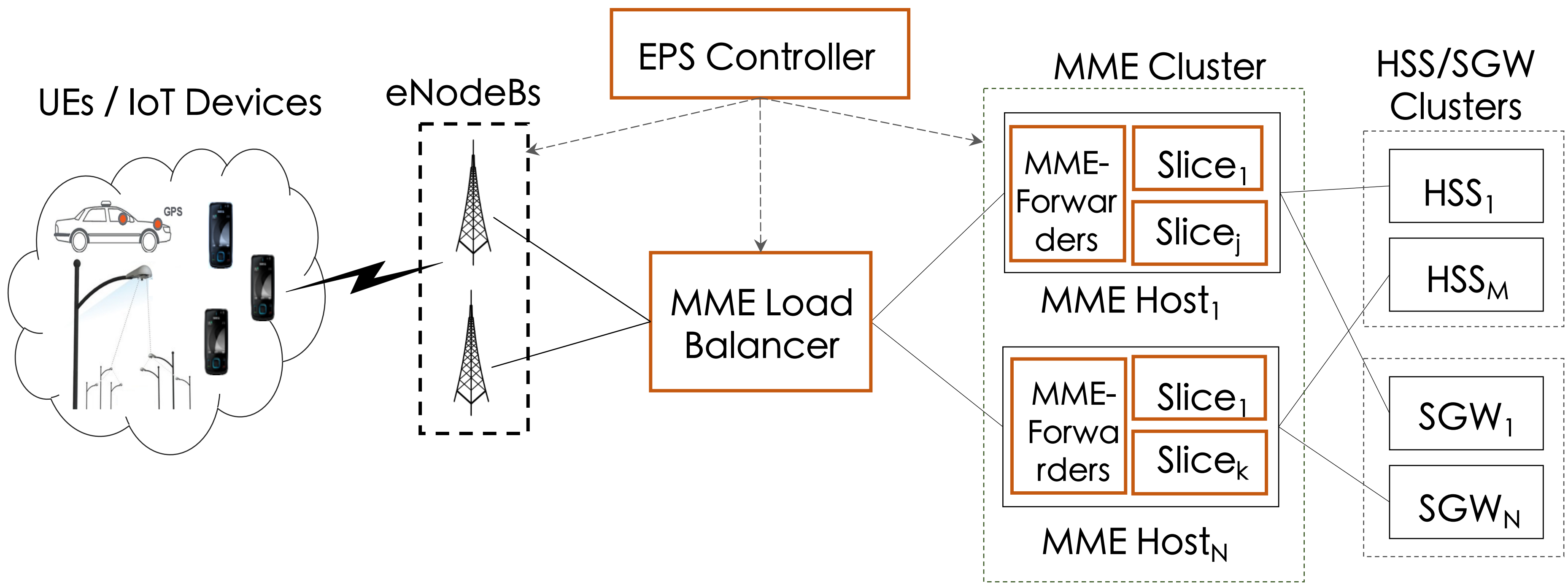
replicas, sliceID, CPU_{NF}, $QueueSize_{NF}$)

CPU_{NF}, $QueueSize_{NF}$) &

CPU-based NF Prioritization

Resource Efficient Traffic distribution with SK-CH (Slice-awareness + Viable Host + Optimal NF)

Overall MMLite Cellular Core Architecture



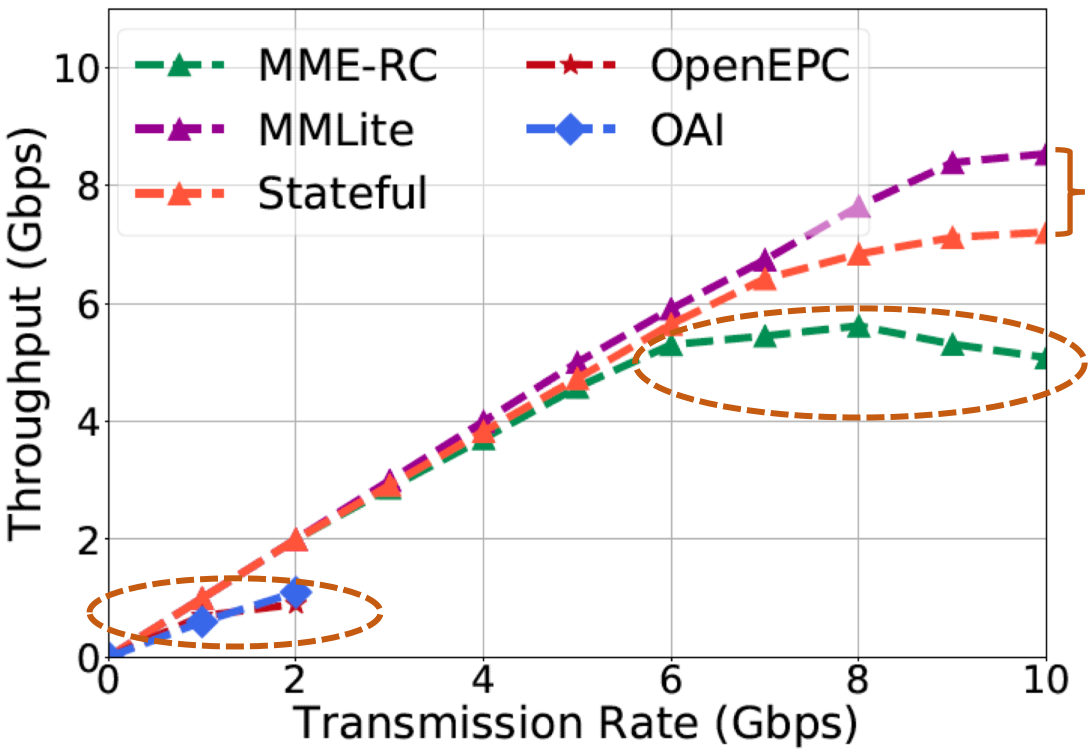
Components newly added or customized in MMLite

Microservice: MME specific to each control procedure

Slice_x: MME microservices bundled specific to tenant's SLO requirements.

Scalability, Flexibility, Fault Tolerance & Resource Efficiency

Throughput Comparison for Different MME prototypes

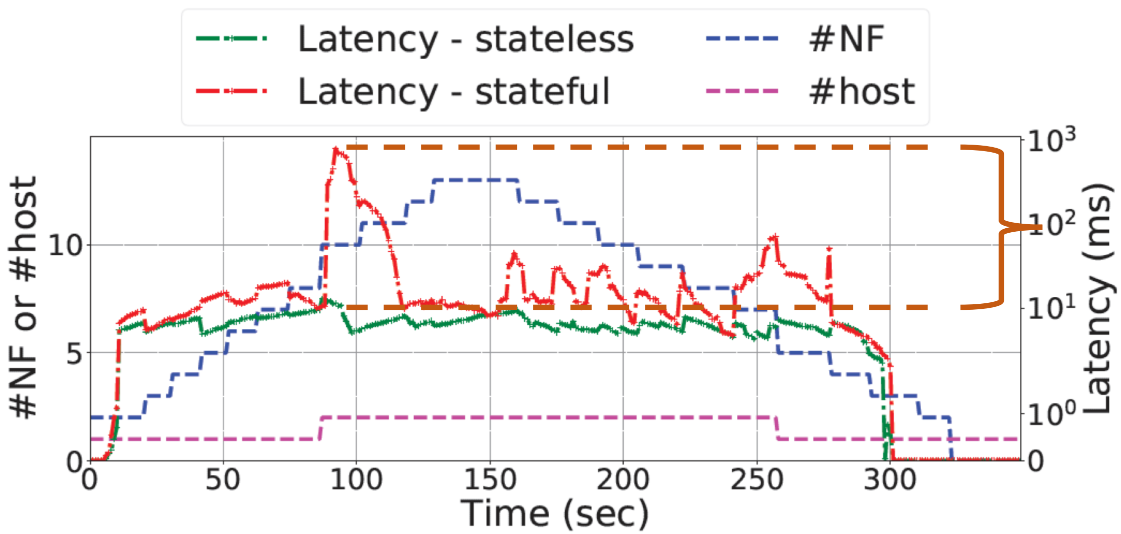


Baseline Comparison:

- MMLite 16% Better Performance than Stateful MME at peak load
- While throughput of MME-RC drops to 5.1Gbps.
- Opensource versions could not scale

% Increase in throughput at peak load

Evaluation: Scaling

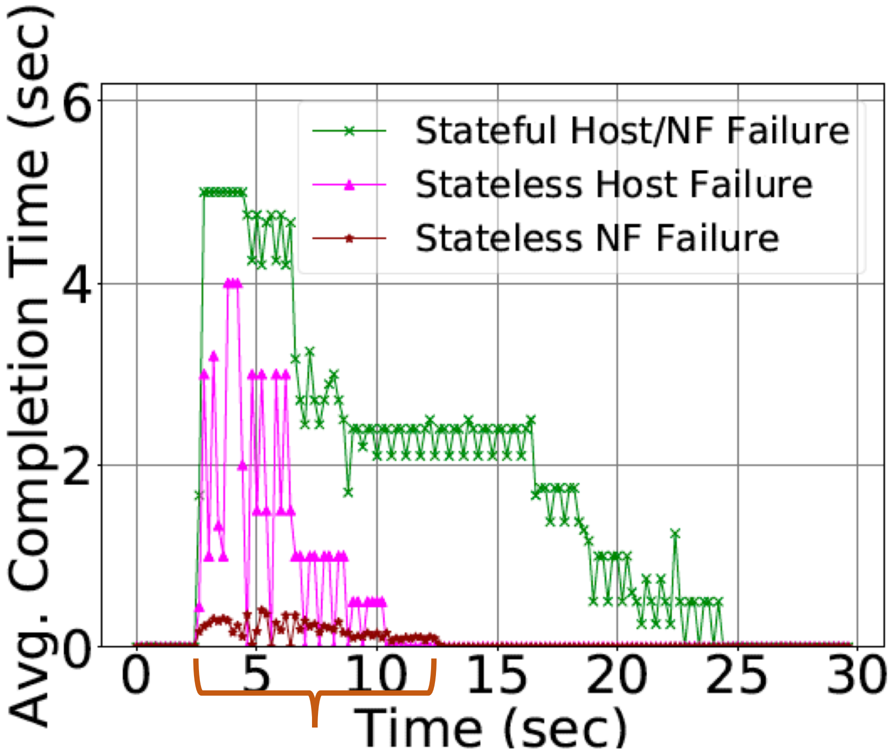


Scaling with MMLite:

- MMLite performs superior during State migration & Load-rebalancing.
  - Especially in scaling
- Up to 50 to 100 X lesser latency during scaling

50 – 100 X Lesser latency compared to Stateful

Evaluation: Fault Tolerance

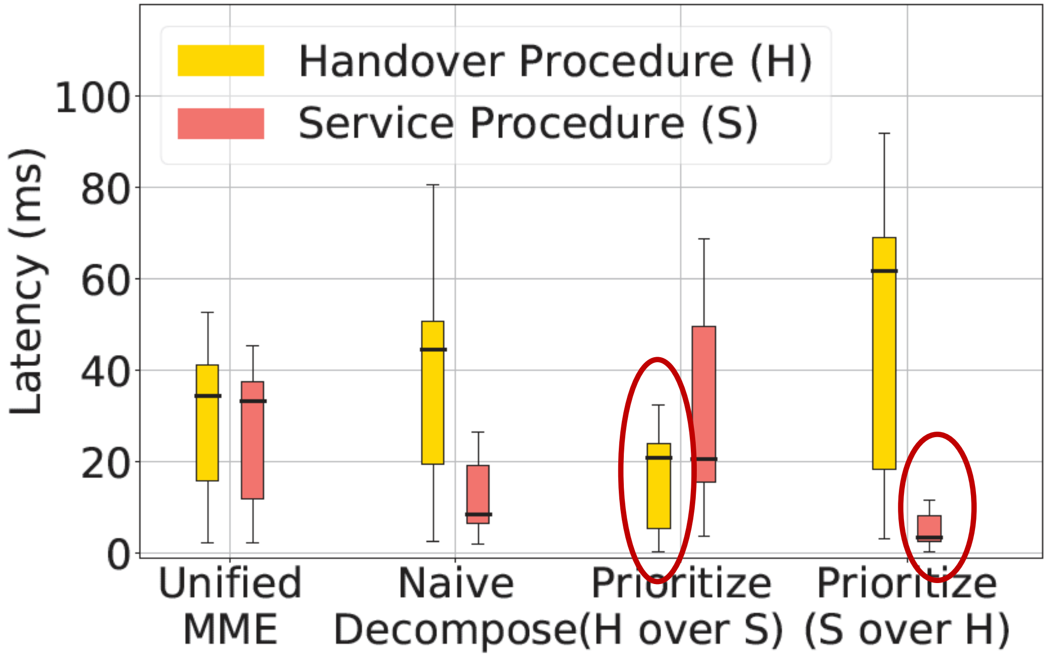


Fault Tolerance:

- MMLite is more responsive after failures.
- The average latency of the control procedure is < 0.5sec with NF failure.
- Up to 2.5sec with host failure.

Highly responsive during failures

Evaluation: Decomposition

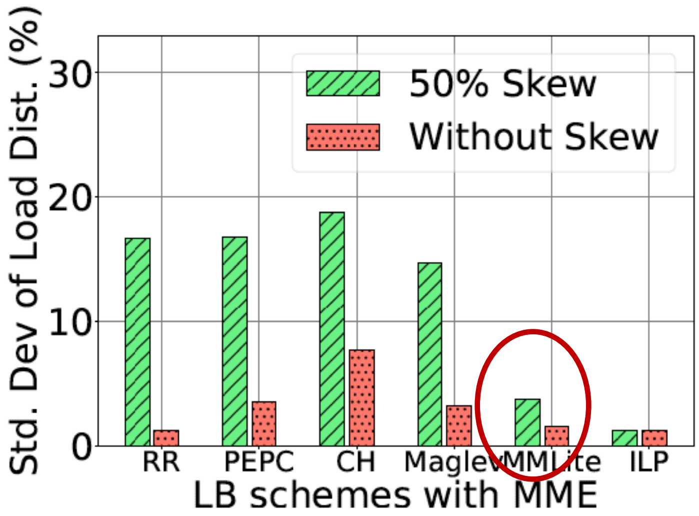


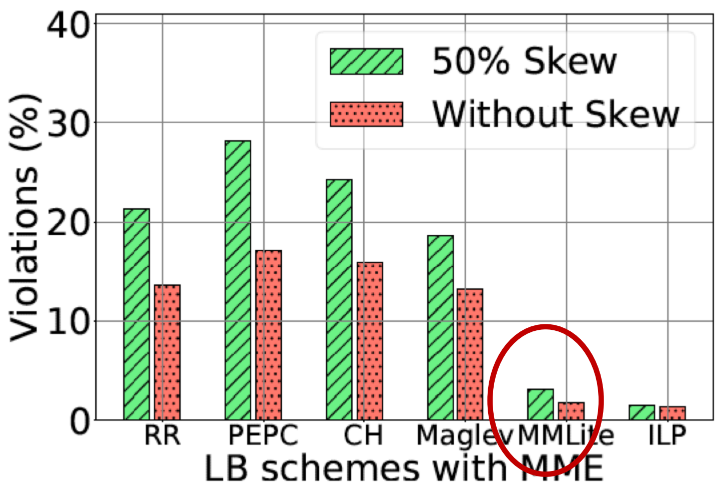
MME Decomposition:

- MMLite with enhanced flexibility
- Efficient Prioritization of control procedures & SLO management

% Increase in resource utilization

Evaluation: Load Balancing



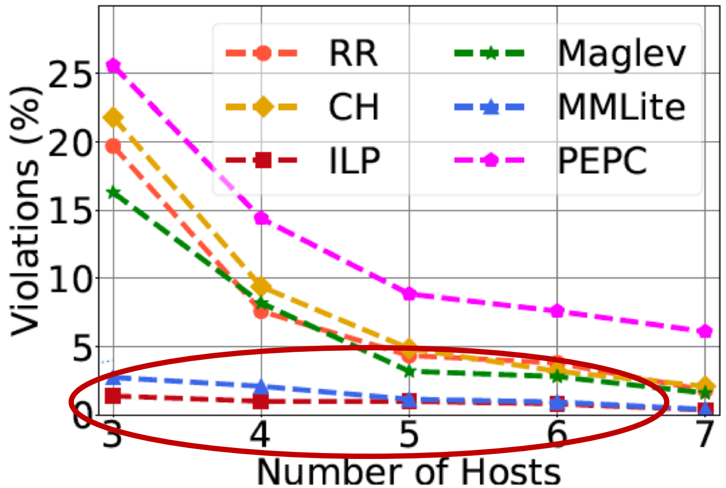


Load Balancing Load Vs SLO:

- MMLite results in about < 4% standard deviation in load distribution.
- Only about 3-4% SLO violations.
  - 3-7× lower compared to other schemes

% Increase in resource utilization

MMLite Load Balancer Performance

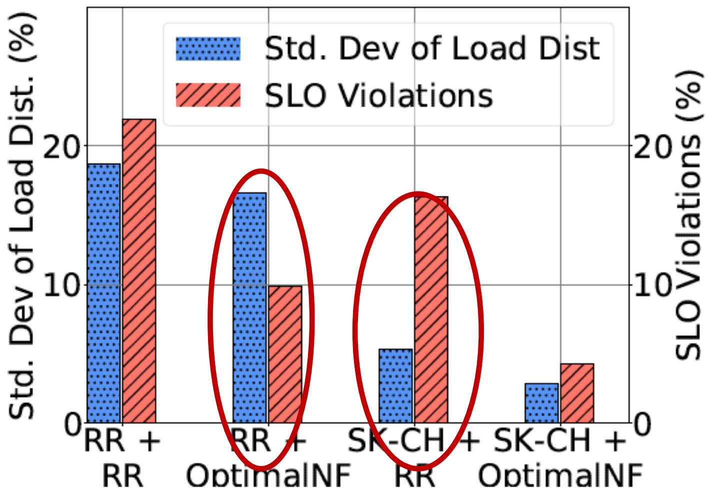


LB Resource Optimization:

- ILP outperforms MMLite
- But, ILP's optimal decisions calculated offline (based on collected workload traces).
- ILP takes on the order of seconds to converge.
- Infeasible in practice (as the ILP is run for each arriving procedure)

% Increase in resource utilization

MMLite Load Balancer Performance



Mix & Match of LB Techniques:

RR with Optimal NF:

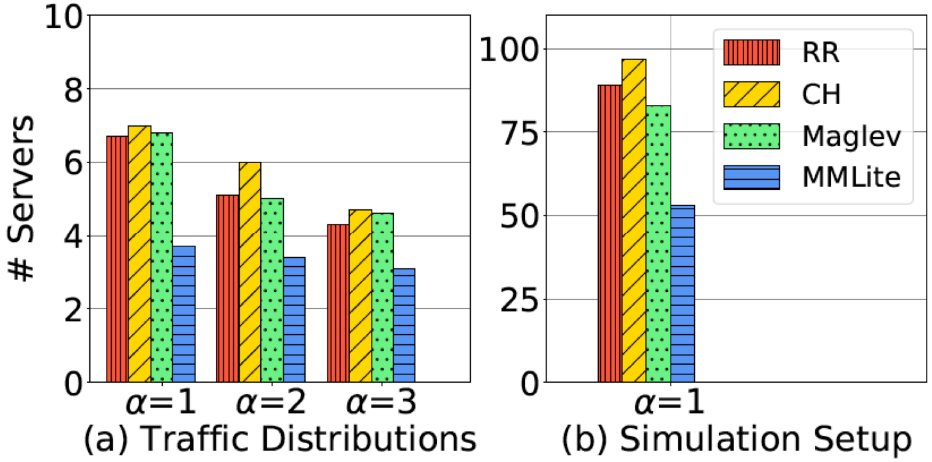
- Performed better in SLO management

SK-CH when used with RR:

- Better Load distribution

% Increase in resource utilization

Performance Benefits



Resource Optimization Efficiency:

For the requirement of ⩽ 5% SLO violations under the Pareto traffic distribution:

- 34 – 47% reduction in resource requirements for all traffic traces.

% Increase in resource utilization

Recap: Benefits with MMLite

- Statelessness & Removes Static Bindings
  - Scaling: Up to 50 to 100 X lesser latency
  - Average latency < 0.5sec with NF failure
- Decomposition & Slicing
  - Flexibility & Isolation in managing latency
- Migration & Skewed Load balancing
  - < 4% standard deviation in load distribution.
  - Only about 3-4% SLO violations.
    - 3-7× lower compared to other schemes

Source Code Available

Initial Version 0.1 of MMLite Emulator source code available

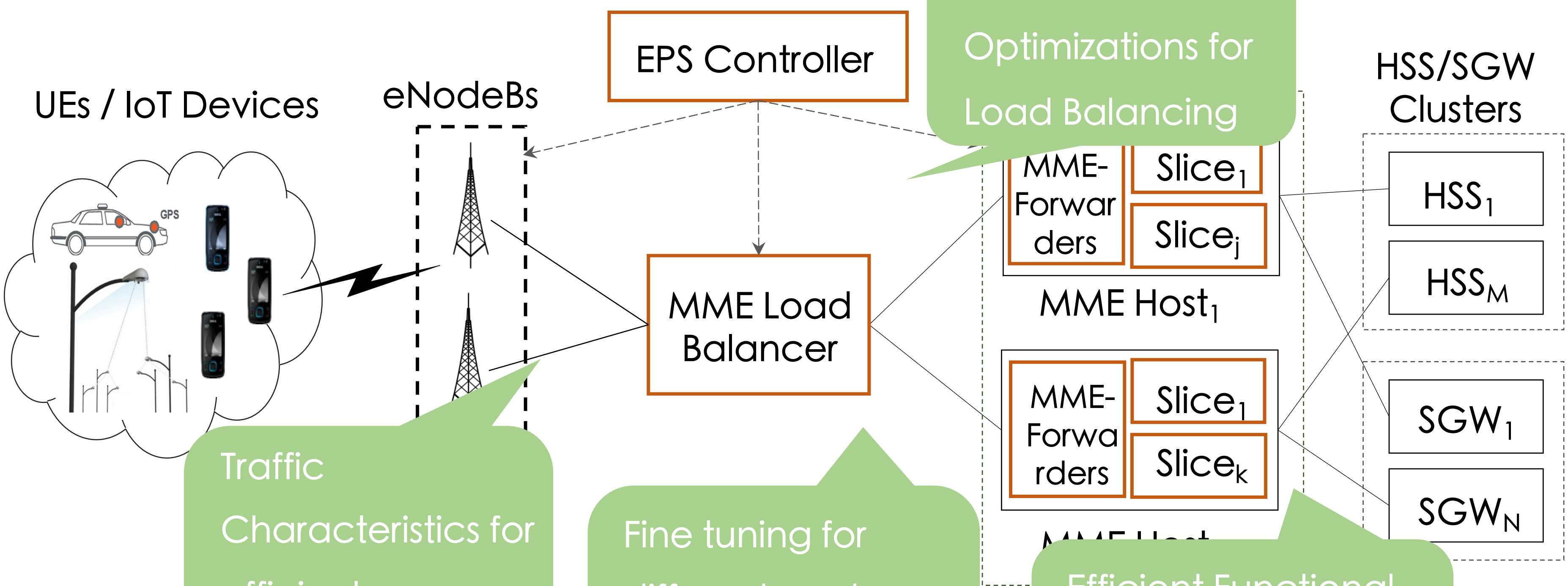
LTE UE

https://github.com/vasu018/LTE-UE

MME + LB + NF Forwarder (Core)

https://github.com/vasu018/MMLite-MME-LB-FW-Microservices

Limitations & Open Challenges



Local Vs Global Optimizations for Load Balancing

Traffic Characteristics for efficient resource utilization

Fine tuning for different goals (Scaling, SLO, resources)

Efficient Functional decompose for resource optimization

Scalability, Flexibility, Fault Tolerance & Resource Efficiency

Email:
vnagendra@cs.stonybrook.edu

Home Page:
https://www3.cs.stonybrook.edu/~vnagendra/