P4 2022 Workshop
May 24-26th

# Accelerating 5G (Mobile Core) Control Plane using P4

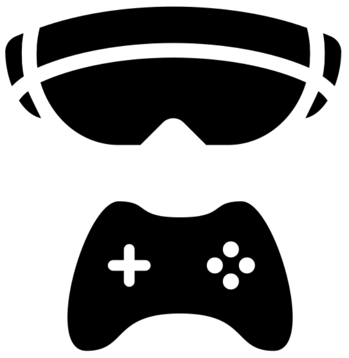
Jingqi Huang*, Jiayi Meng*, Iftekharul Alam, Christian Maciocco, Y. Charlie Hu, Muhammad Shahbaz

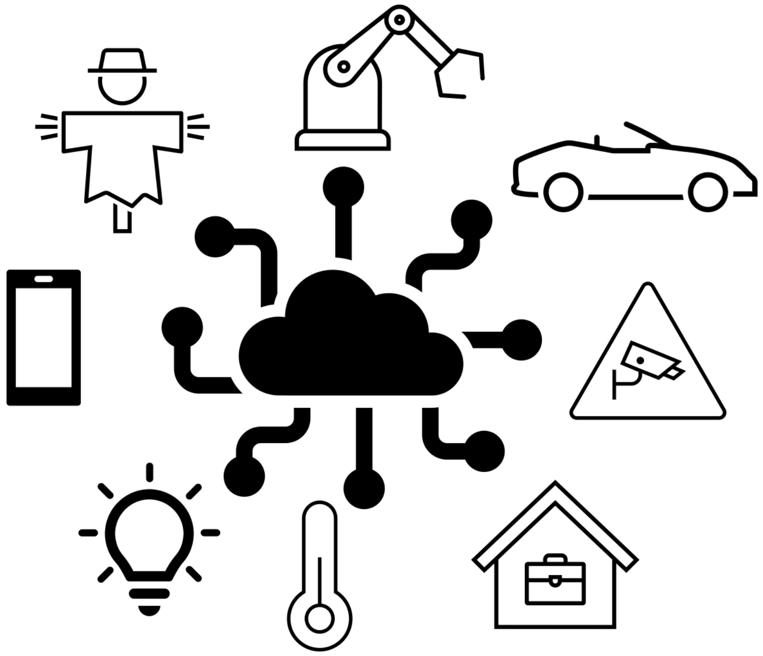
***Purdue University, Intel Labs***

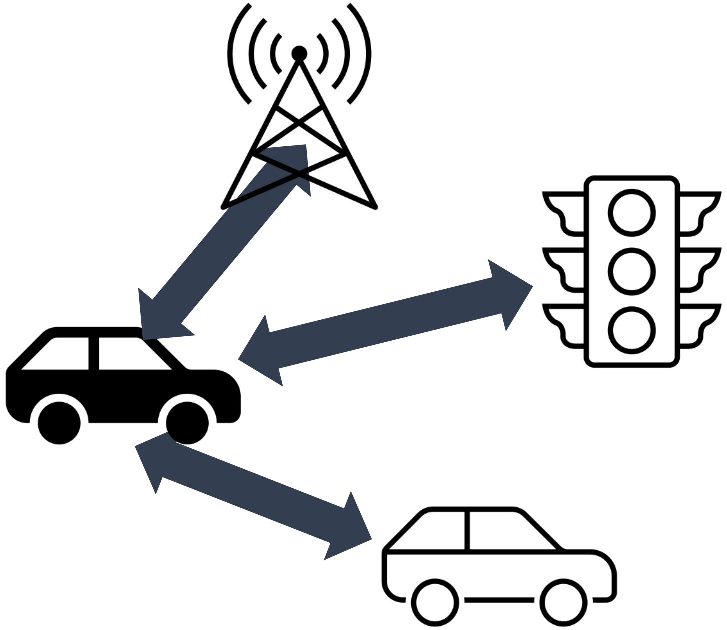
* Co-primary author

# Emerging Applications over Mobile Network

- Trillions of devices are connecting the Internet!

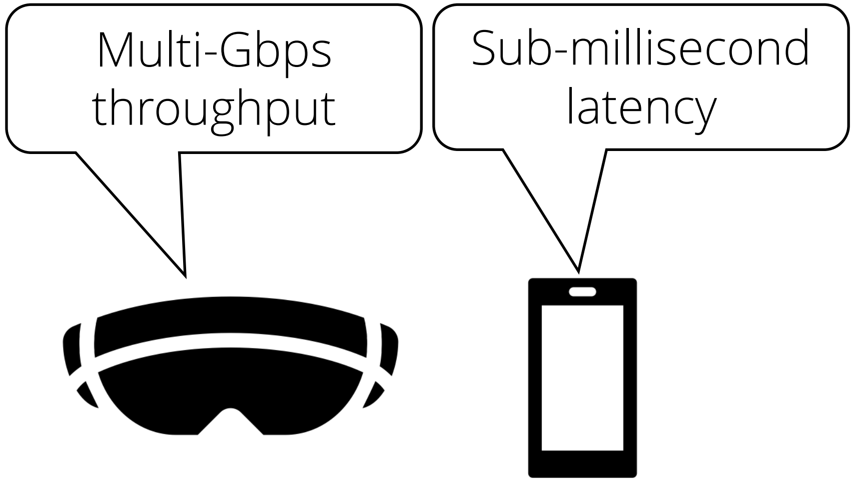


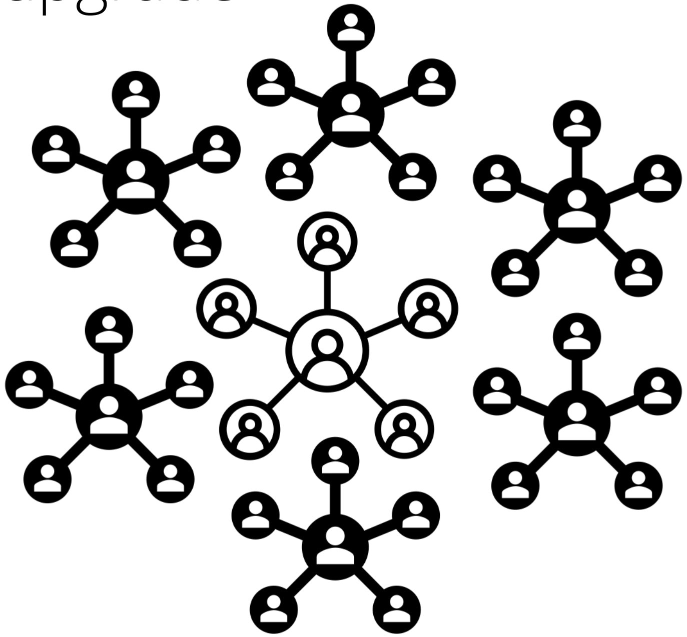


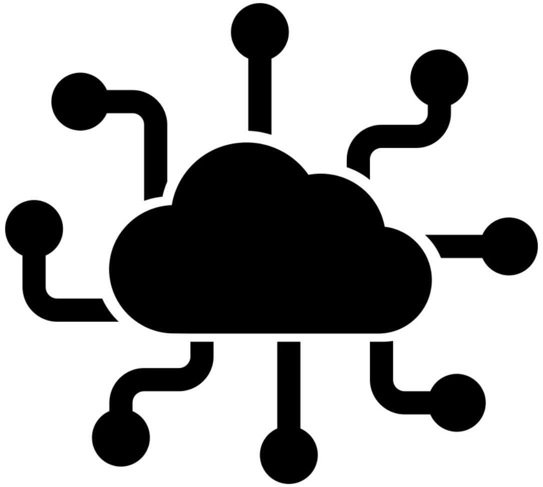


# Requirements of Next-Gen Mobile Cores

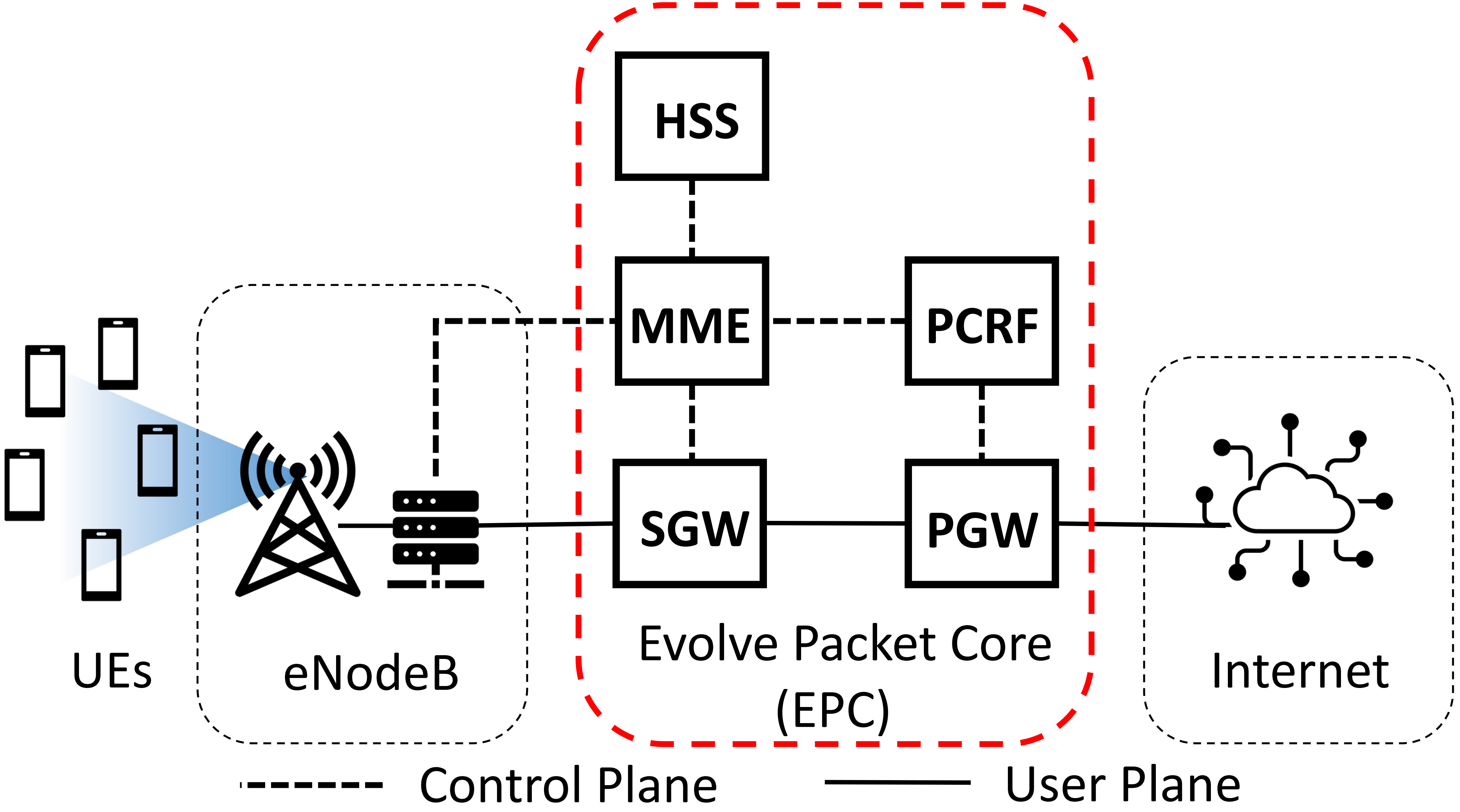
- High-performant
  - Meet both throughput and latency requirements
- Scalable and highly available
  - Support large-scale devices
- Flexible
  - Easy to manage and upgrade



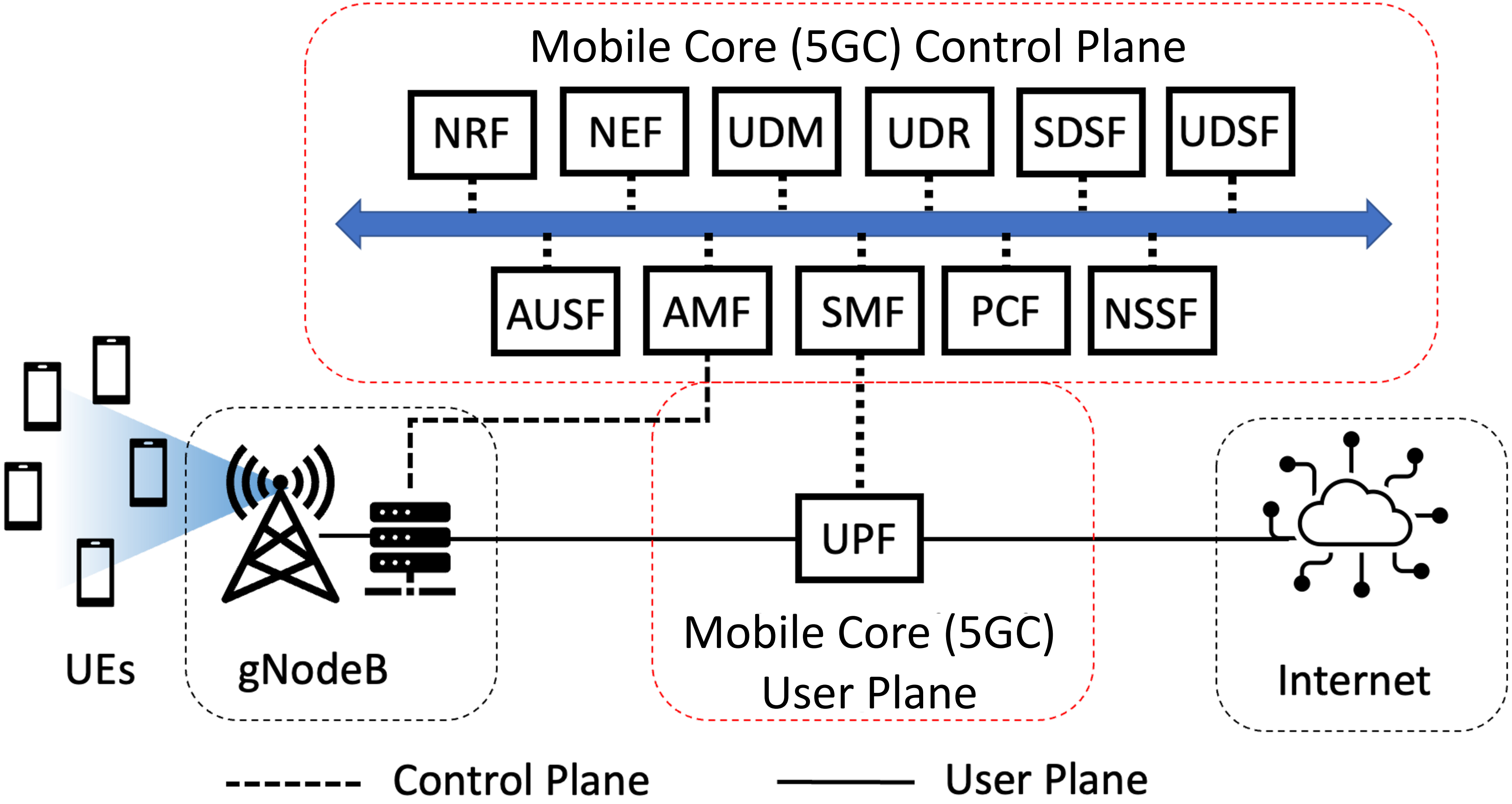




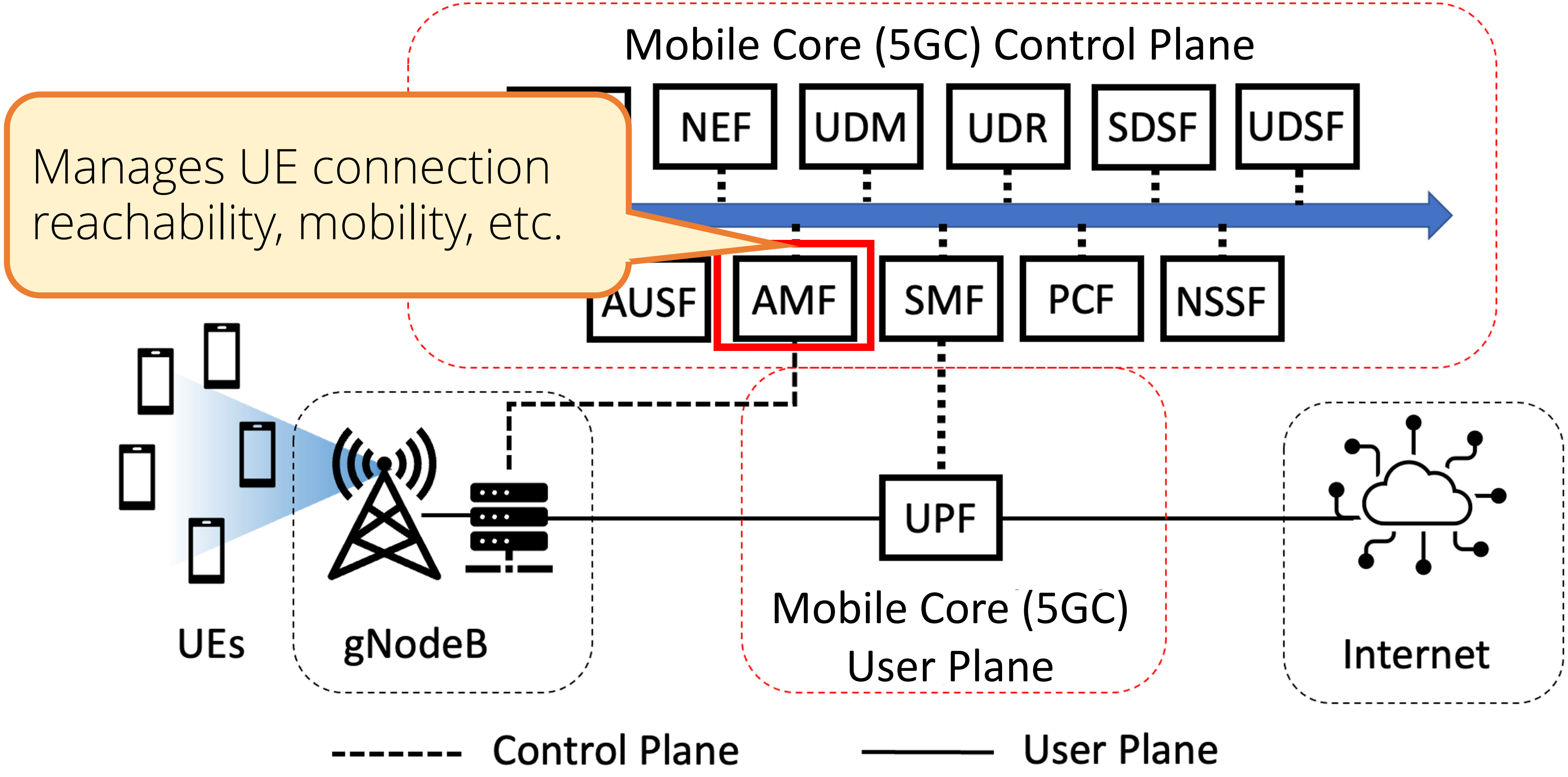
# Vertically Integrated EPC in LTE



# Horizontally Disaggregated 5G Mobile Core: Service-based Architecture



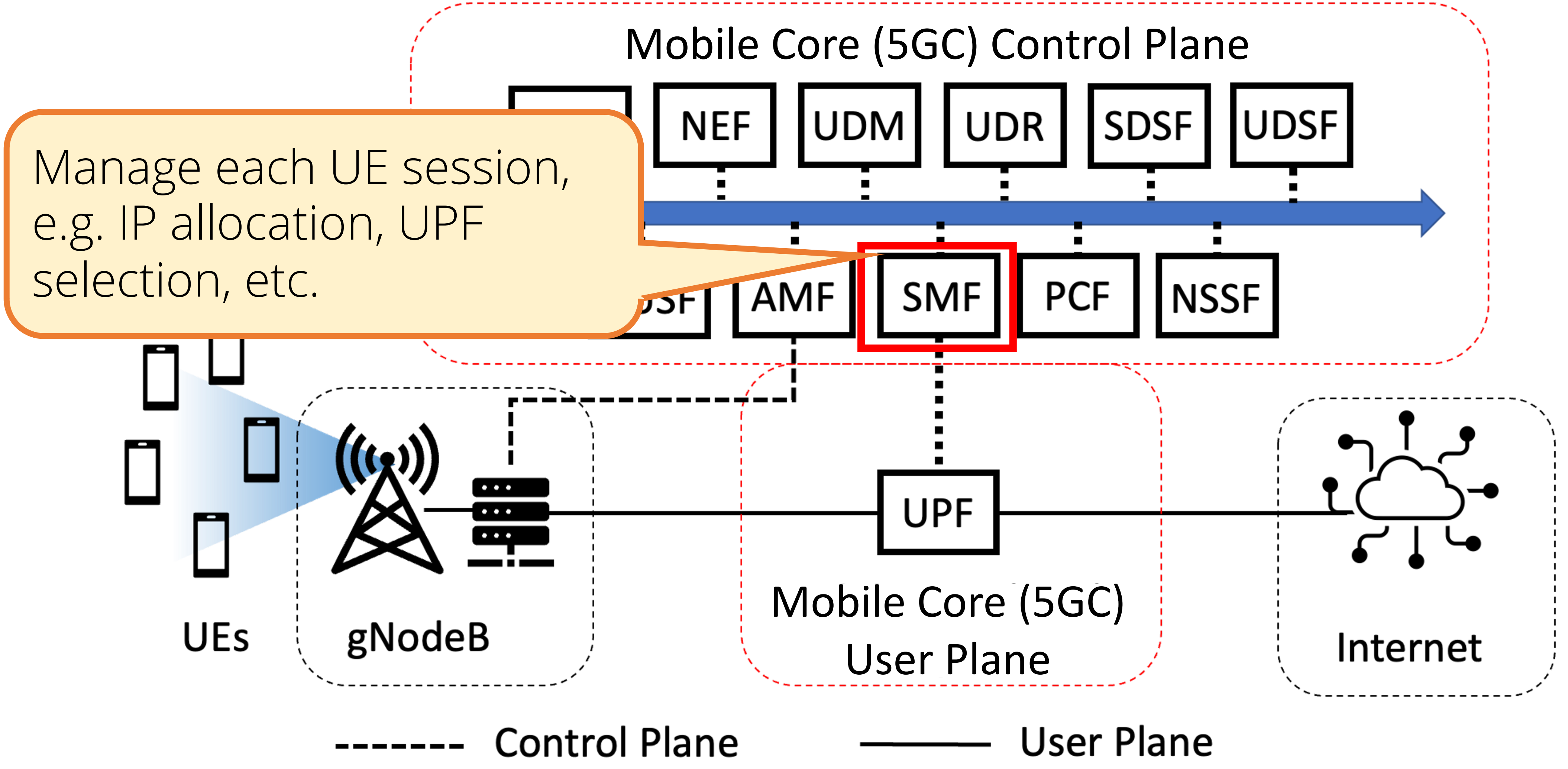
# Components of 5G Mobile Core: AMF



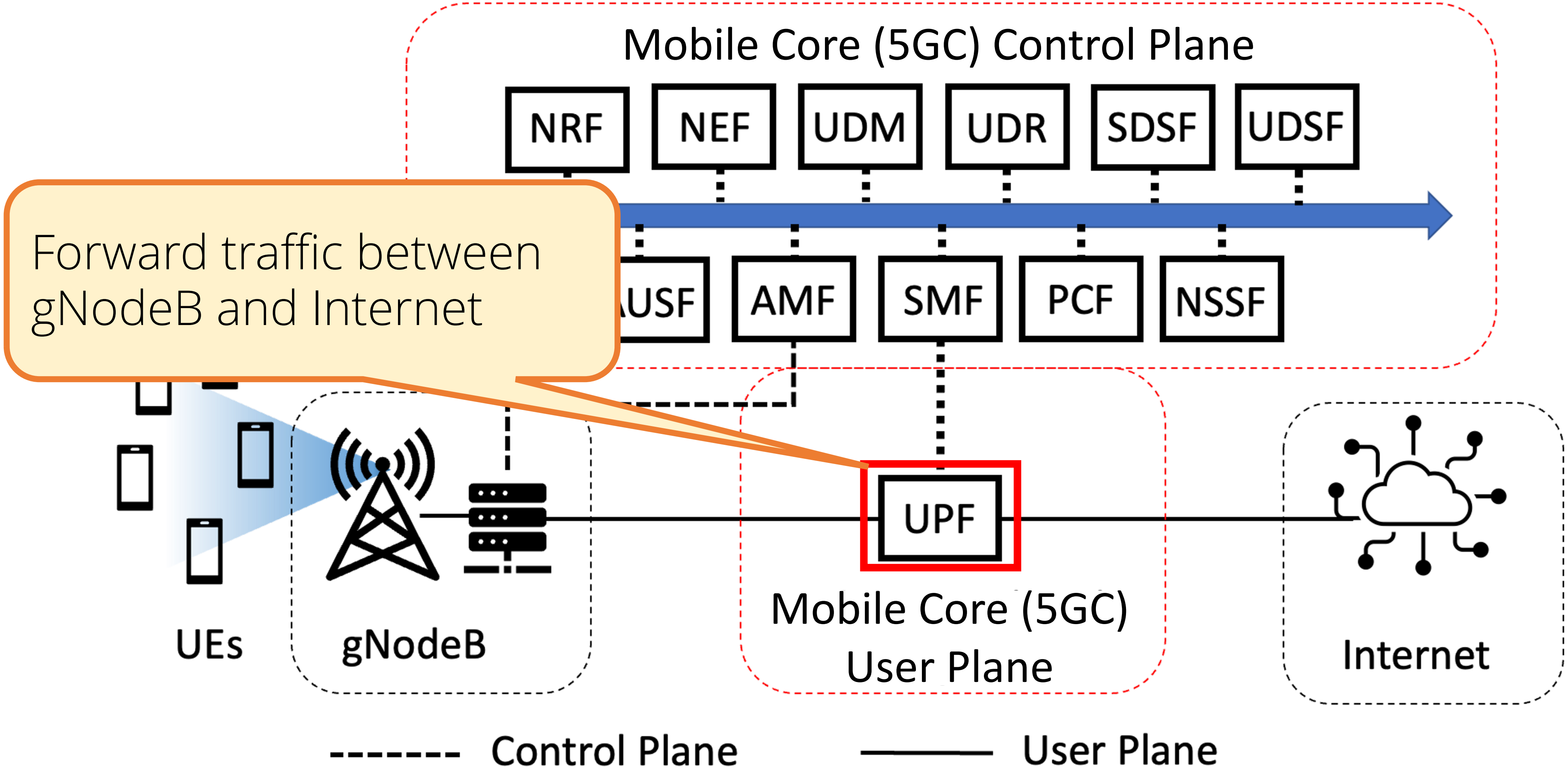
Manages UE connection reachability, mobility, etc.

# Components of 5G Mobile Core: SMF

Manage each UE session, e.g. IP allocation, UPF selection, etc.

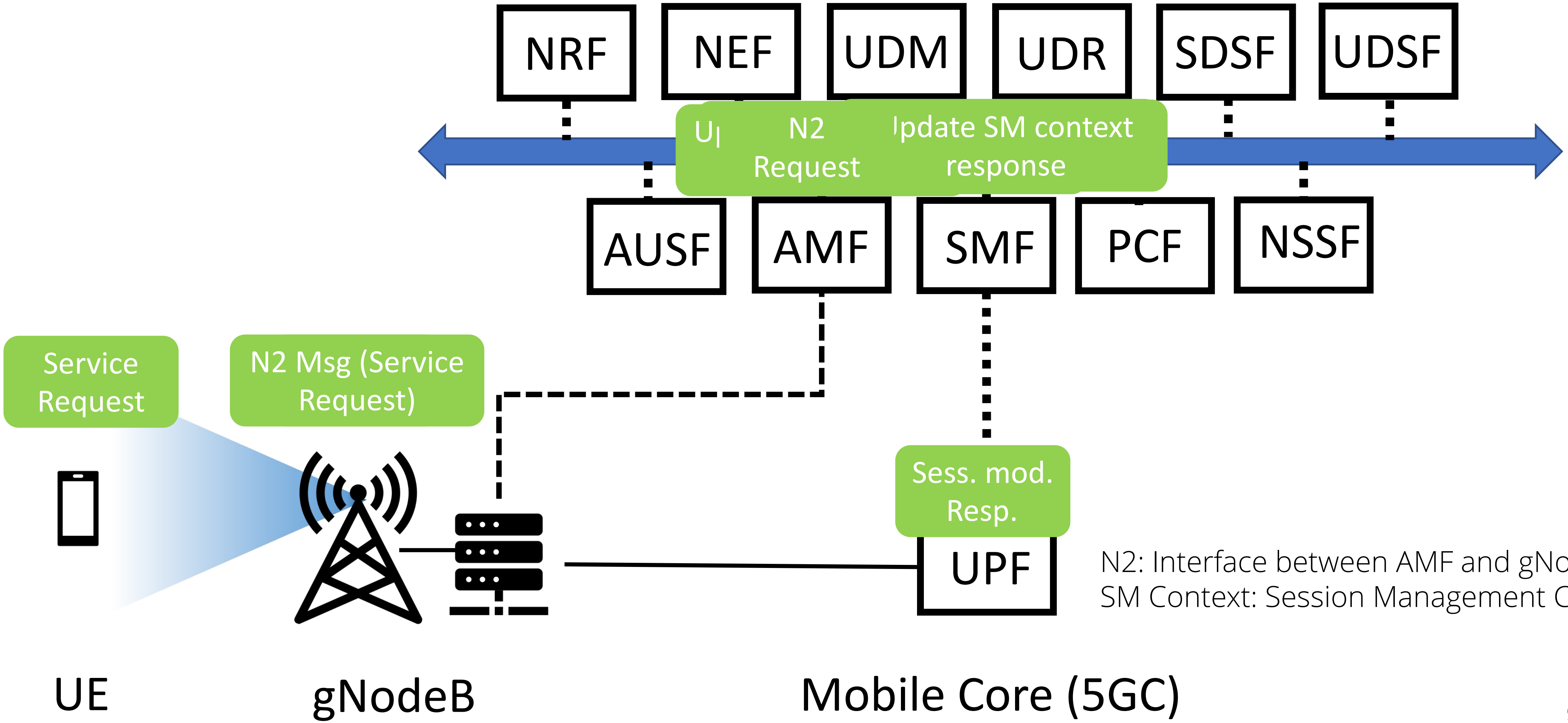


# Components of 5G Mobile Core: UPF



Forward traffic between gNodeB and Internet

# Life Cycle of Service Request

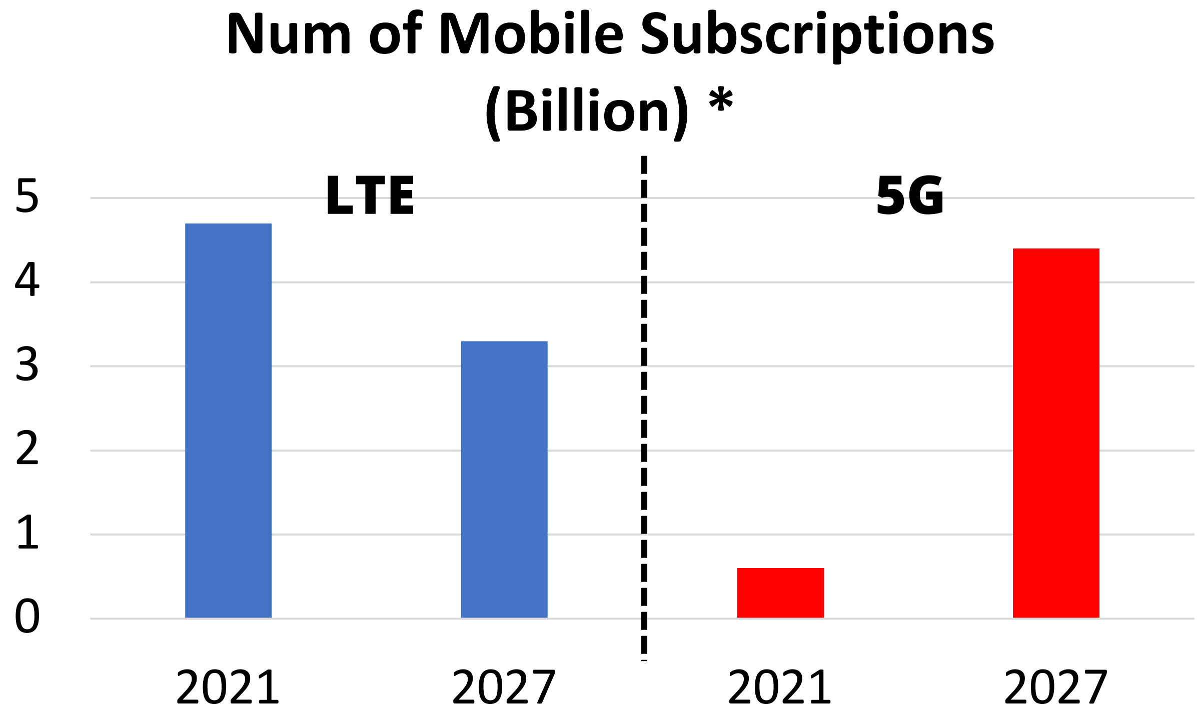


N2: Interface between AMF and gNodeB
SM Context: Session Management Context

# Increasing Control Plane Traffic in 5G Mobile Core

# Increasing Control Plane Traffic in 5G Mobile Core

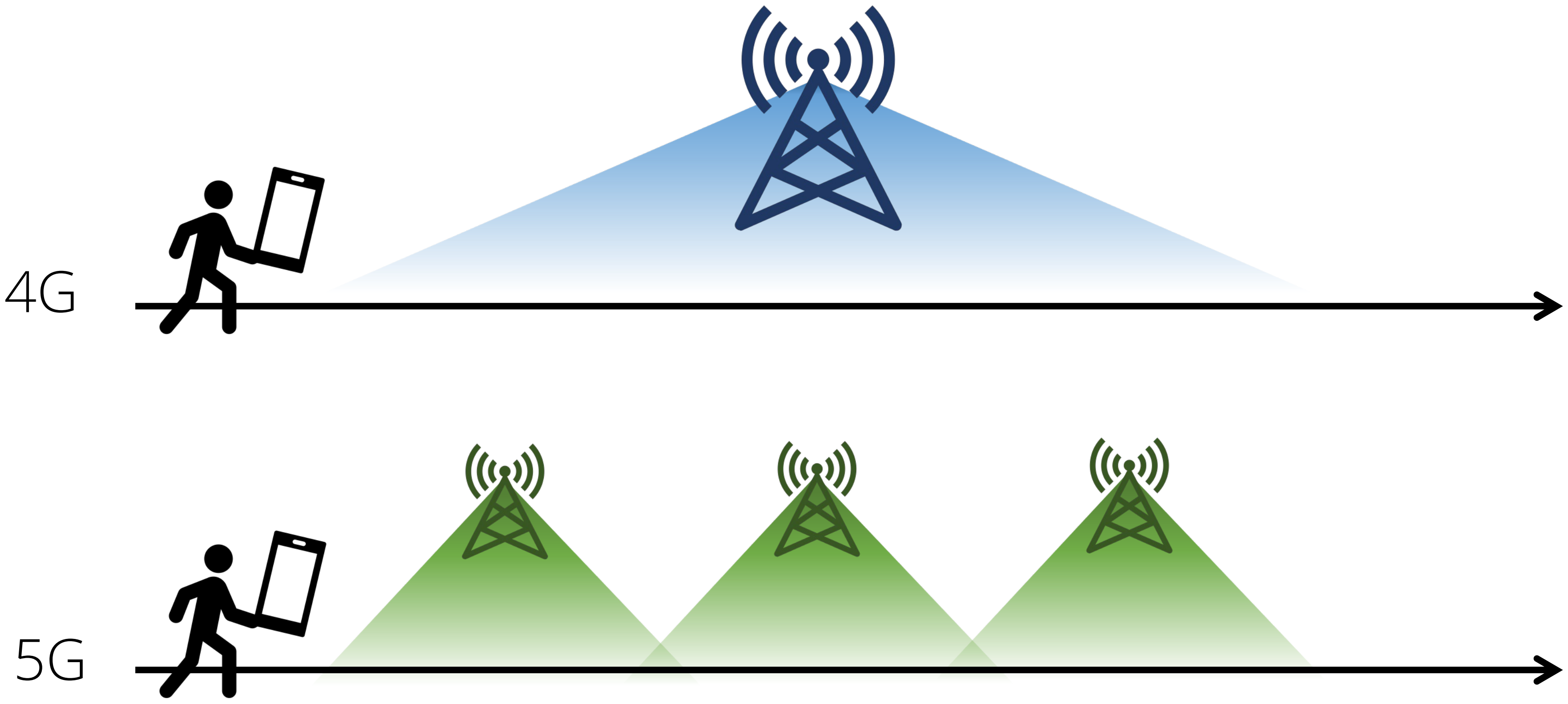
- Increasing UE number



*https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-subscriptions-outlook

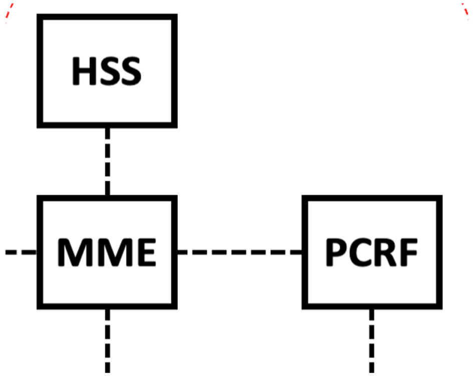
# Increasing Control Plane Traffic in 5G Mobile Core

- Increasing UE number
- Increasing control event frequency

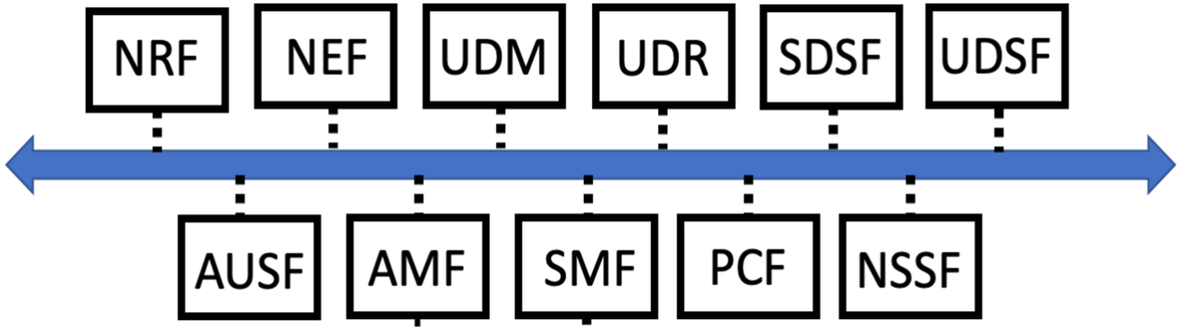


# Increasing Control Plane Traffic in 5G Mobile Core

- Increasing UE number
- Increasing control event frequency
- Increasing transaction number per control event



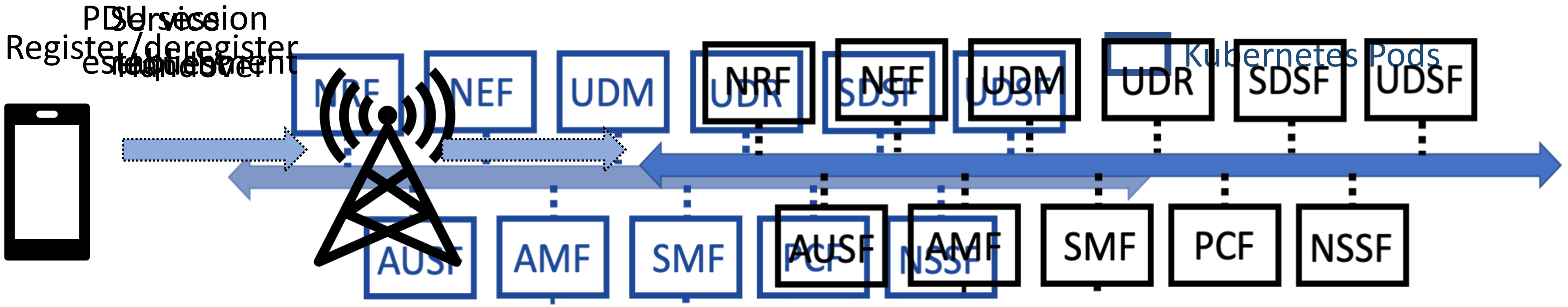
4G Core



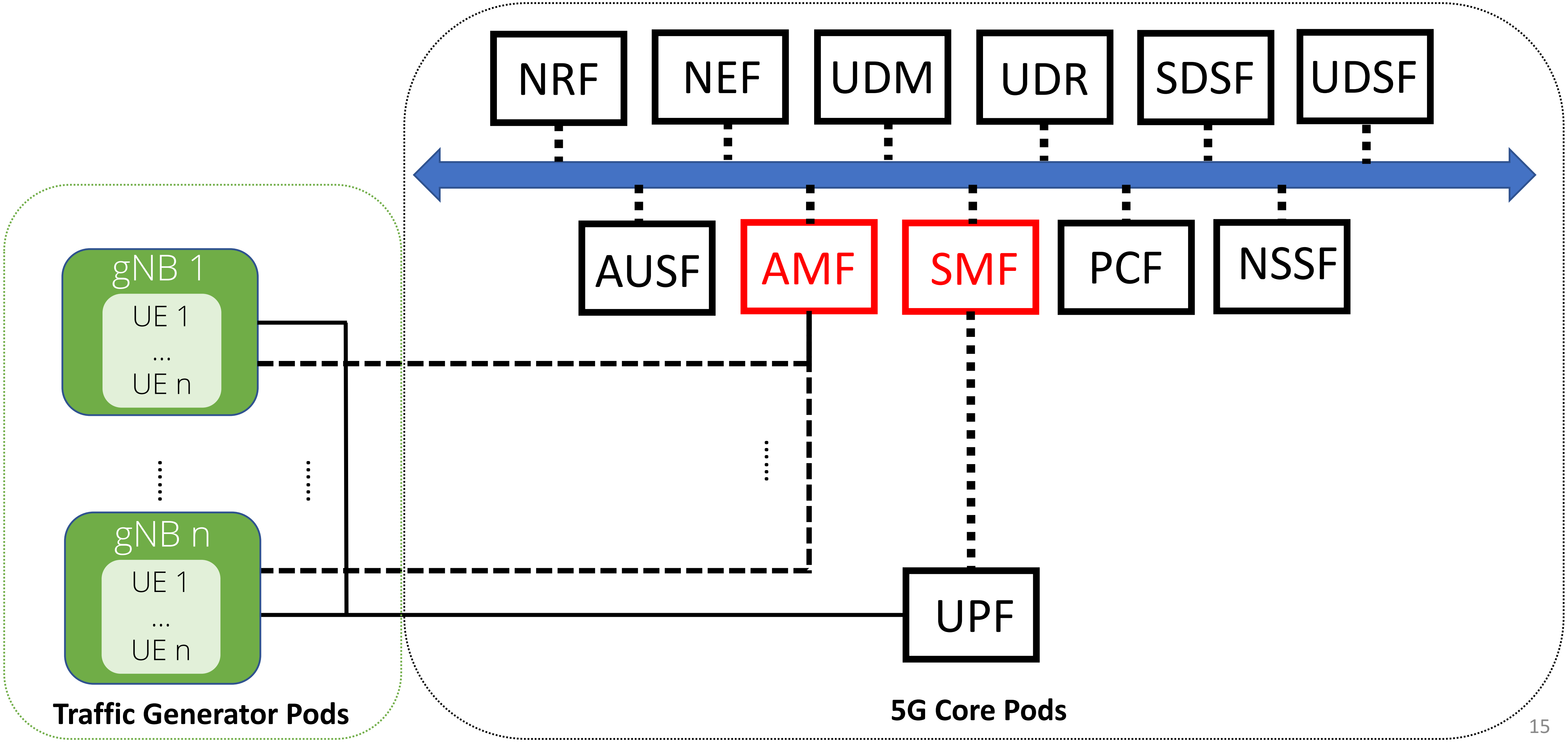
5G Core

# Introducing SD-core from ONF Aether Project

- Powered by Kubernetes
- Initially based and forked from Free5GC in 2020, 3GPP release 15, with features and functionality enhanced
  - Support 5G standalone mode
  - Support major 5G control events



# Experimental (Kubernetes) Setup

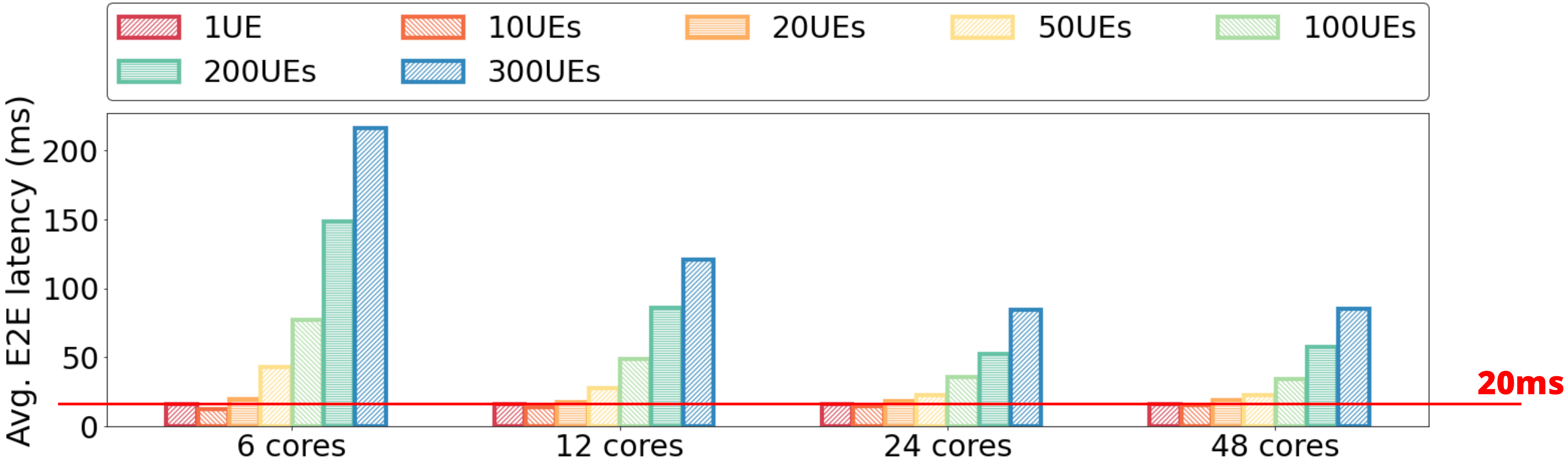


# Experiment Methodology

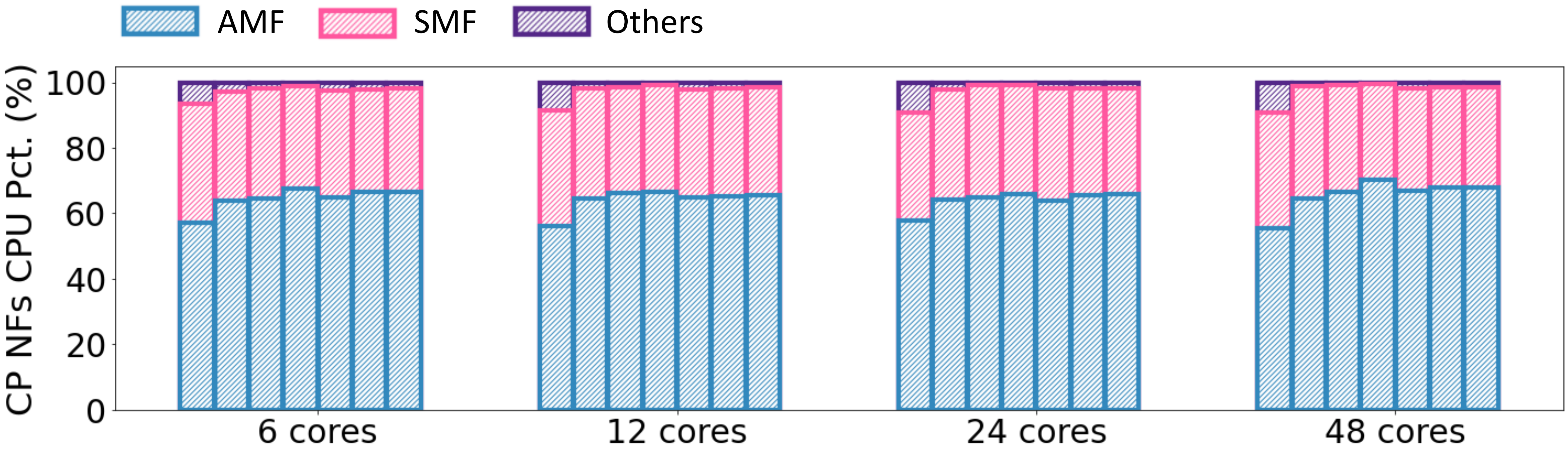
- Assign different number of cores to control-plane network functions
- Initiate different number of UEs in traffic generator, connected to control-plane network functions simultaneously
- Every UE triggers service requests in a back-to-back way
- Collect stats during the experiments
  - Latency, CPU utilization, etc

# Current System Scales Poorly with Increasing UEs

- Service request end-to-end latency
  - ETSI recommends service request end-to-end latency to be 10−20 ms

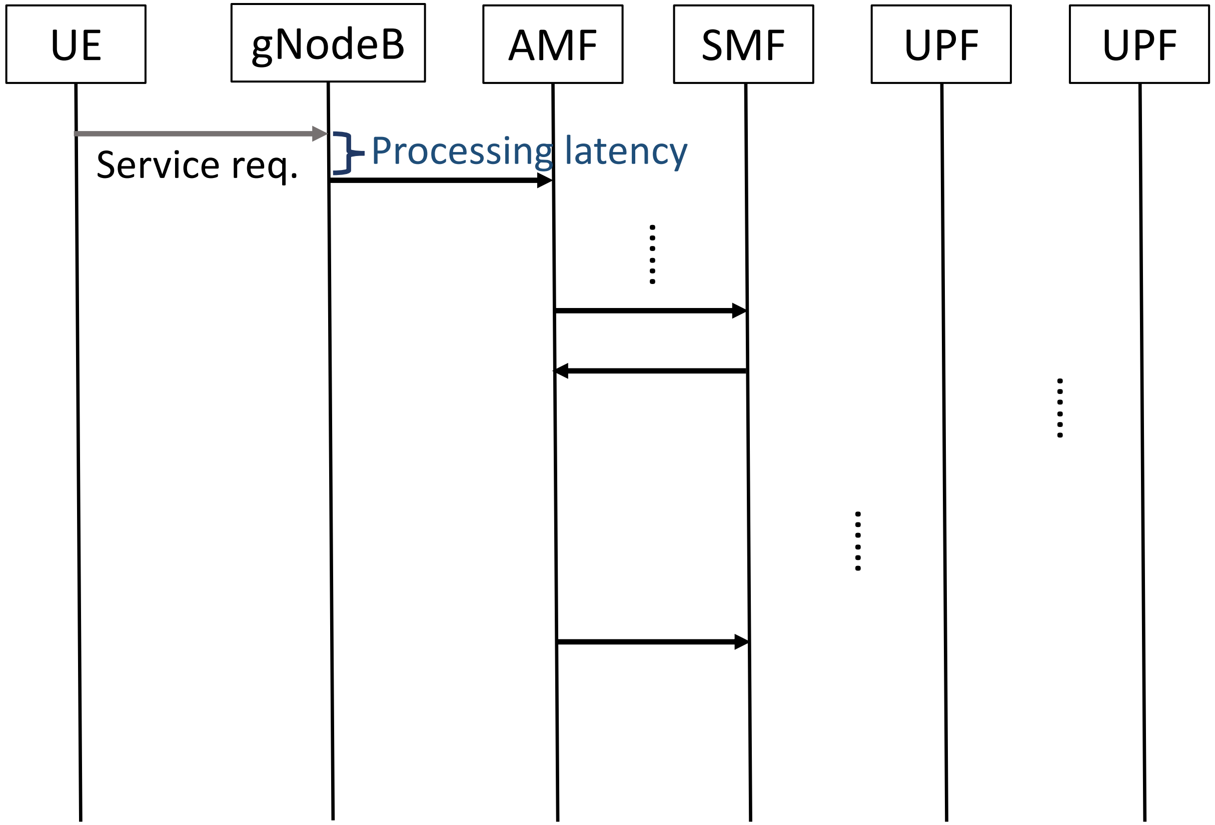


# AMF and SMF are Hot Spots



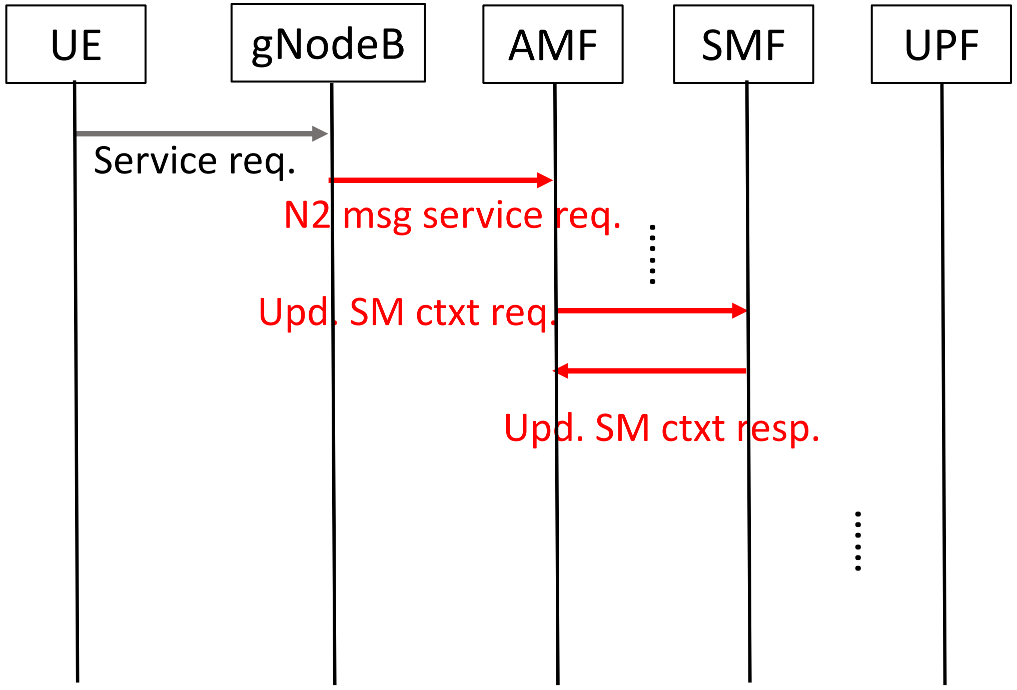
# Identifying Bottleneck Transactions

- Checking processing latency of service request



# Identifying Bottleneck Transactions

- Checking processing latency of service request
- Bottleneck transactions:
  - N2 msg service request*, update SM context request and response



UE

gNodeB

AMF

SMF

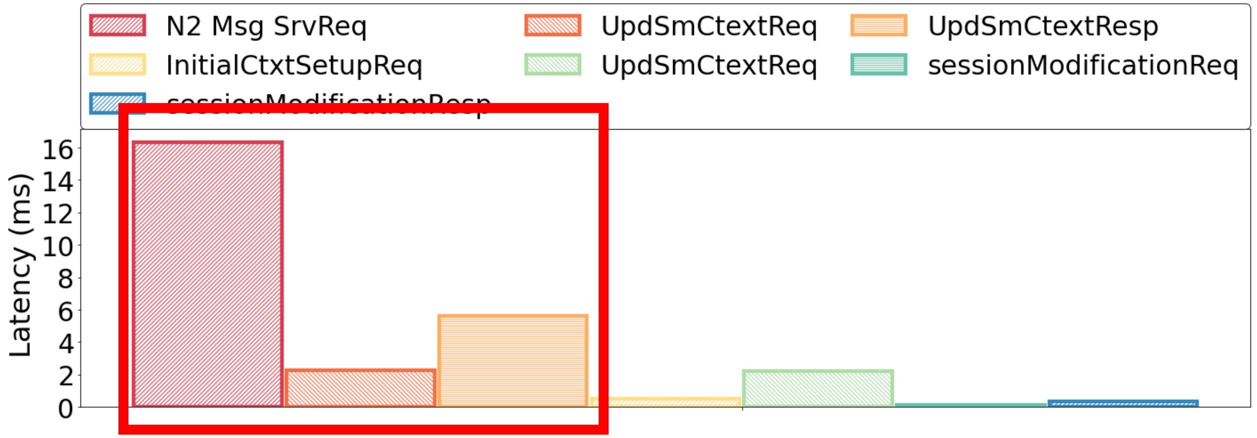
UPF

Service req.

N2 msg service req.

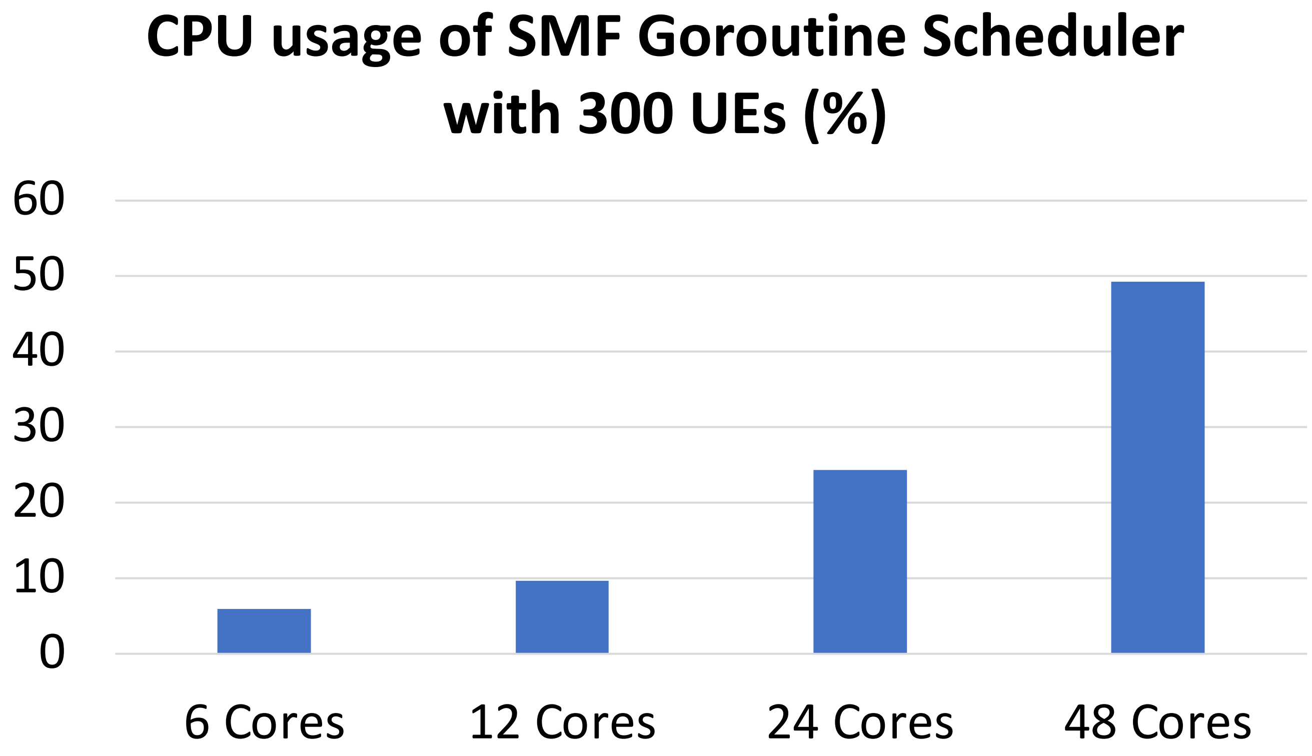
Upd. SM ctxt req.

Upd. SM ctxt resp.



# Impact of the Scheduler

- Scheduler uses more CPU when increasing number of cores



# So, in Summary …

- Current system scales poorly with increasing UEs.
- AMF and SMF are hot spots.
- Bottleneck transactions:
  - N2 Message with service request
  - Update SM context request
  - Update SM context response
- Scheduler uses more CPU when the number of cores increases.

# Accelerating 5G Core

- Disaggregating 5G control-plane into microservices

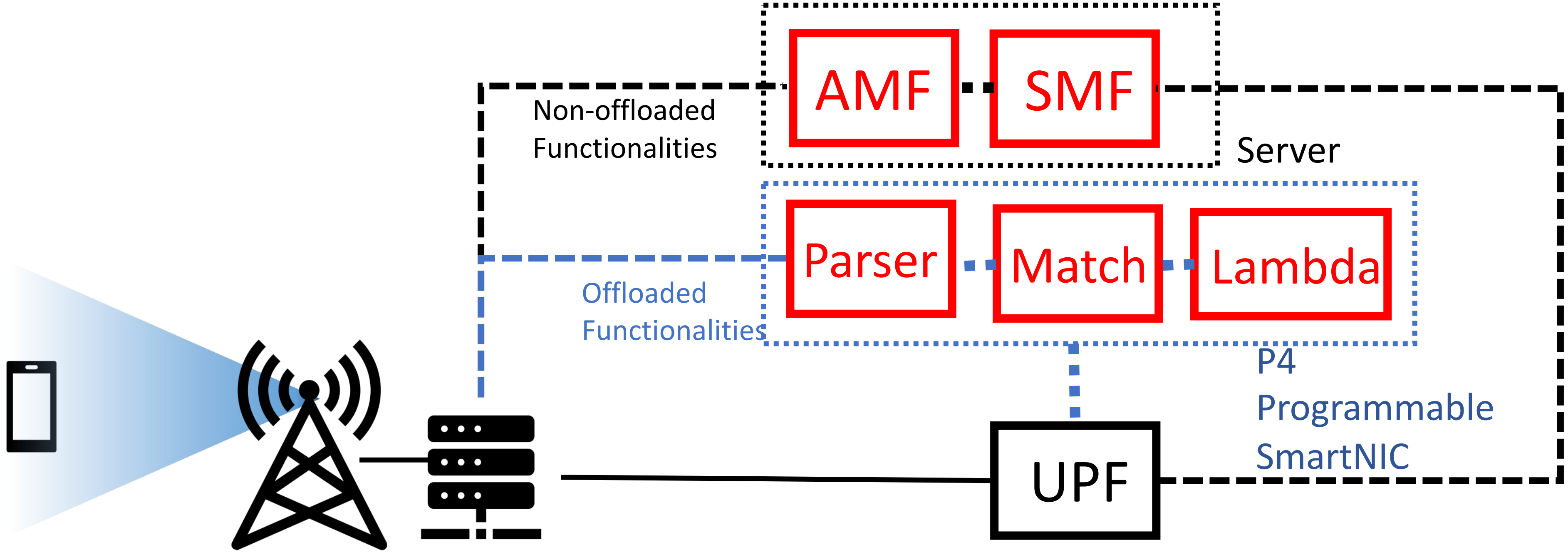
- Accelerating these services using P4 programmable data planes (e.g., SmartNICs)

- Challenge: P4 Match+Action tables (MATs) are too restricted
  - Can only do simple single cycle action

- We need something more flexible and stateful to offload 5G core network functions

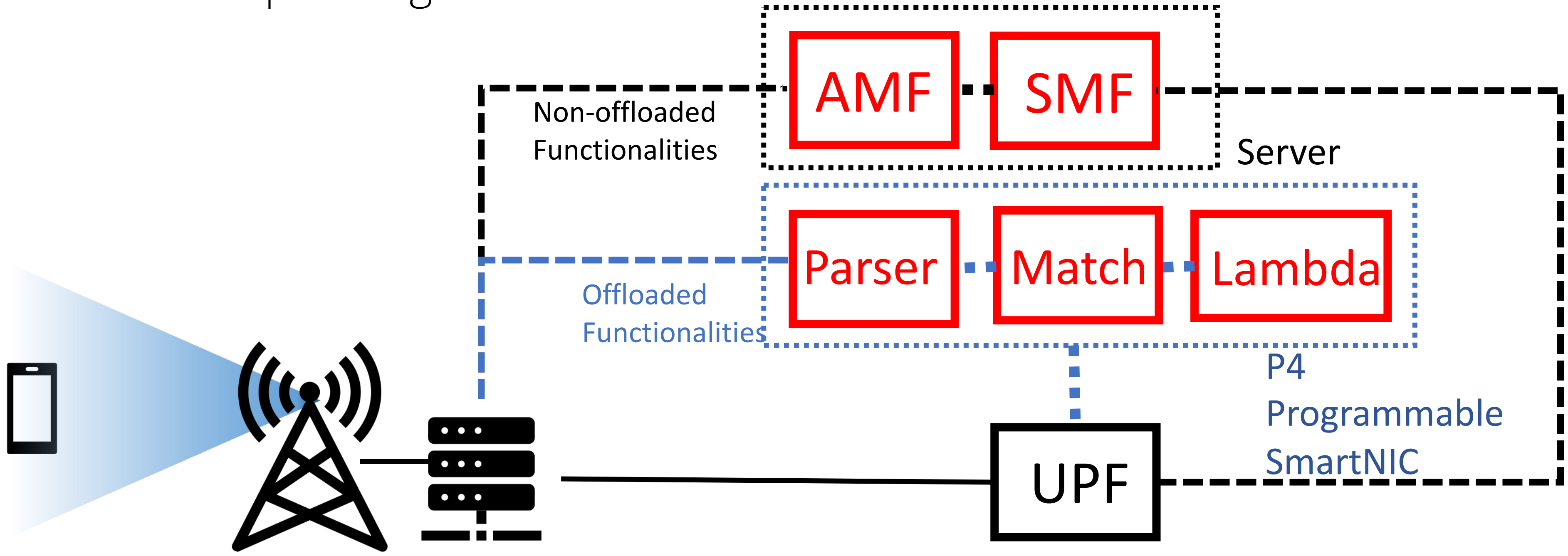
# Accelerating 5G Core using P4

- Solution: 5G core control plane functions using Match+Lambda
  - Extending P4 MAT to Match+Lambda abstraction



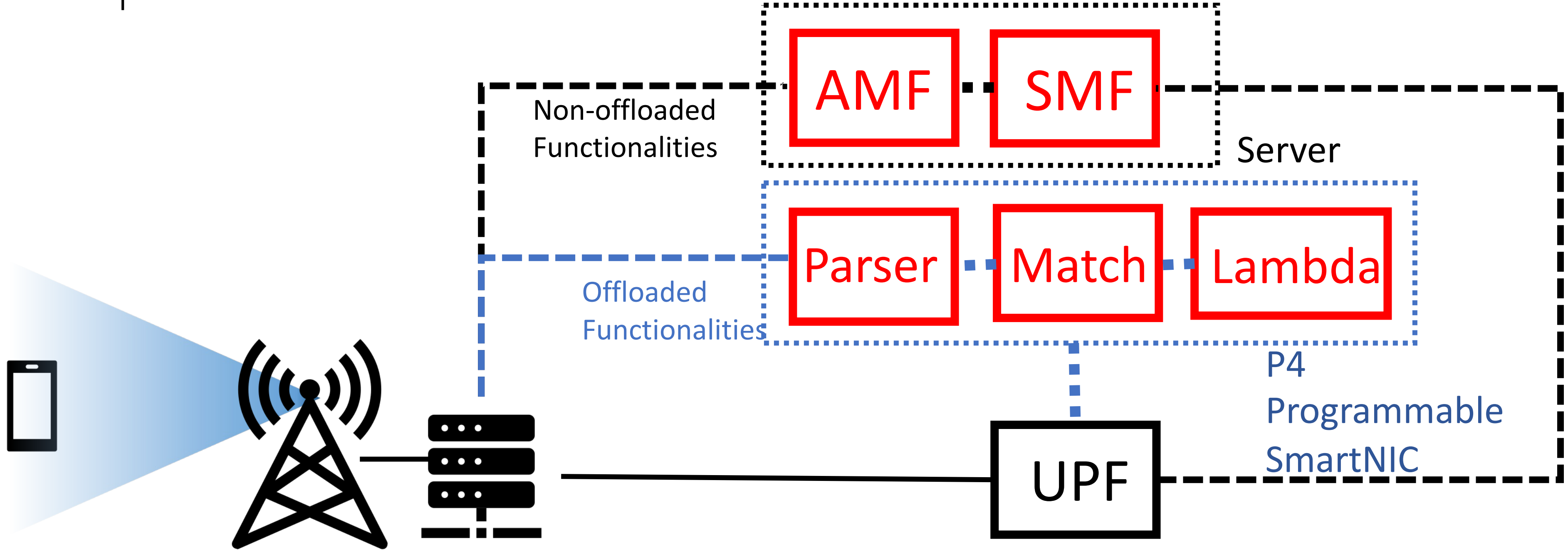
# Towards High Performance, Scalability and Flexibility

- Match+lambda abstraction
  - Design Components
    - Abstract machine model
    - Programming lambdas
    - Expressing match



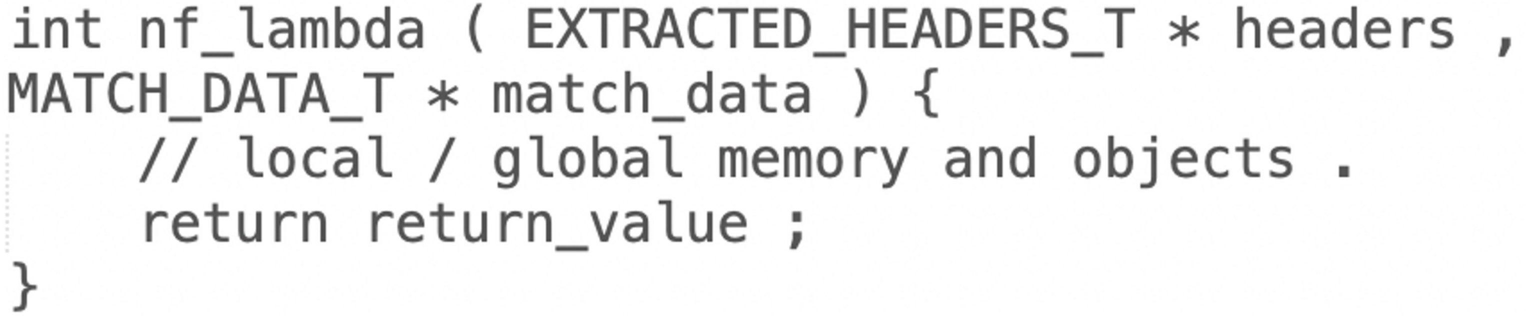
# Proposing Match+Lambda Abstraction (1)

- Abstract machine model
  - Lambdas do not share states
  - Match stage serves as a scheduler
  - A parser handles packet-header operations and lambdas operate on the parsed headers



# Proposing Match+Lambda Abstraction (2)

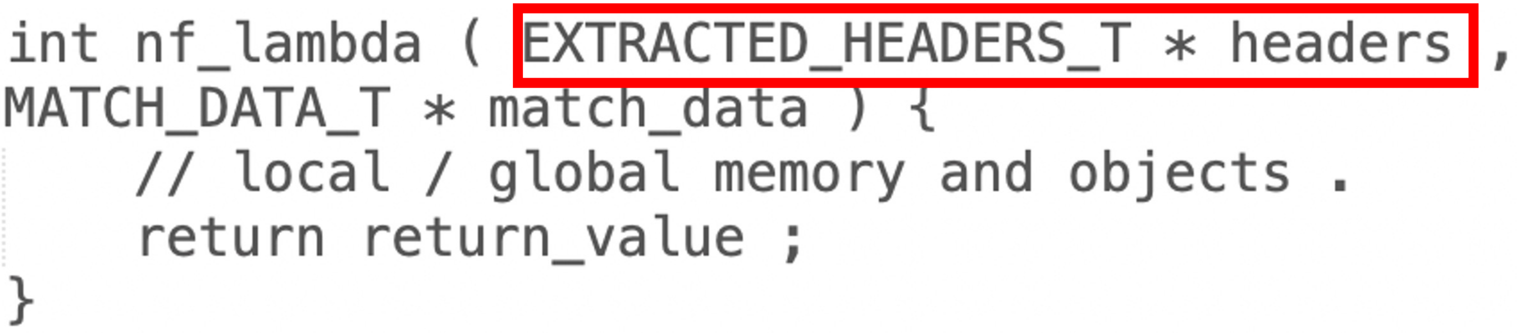
- Programming lambdas
    - Cellular operators will provide one or more lambdas for their network functions
        - Such as Update SM context, Release SM context...
    - Signature of each lambda has two predefined arguments: headers and match_data
    - The lambdas will operate directly on headers and match_data without parsing packets



```
int nf_lambda ( EXTRACTED_HEADERS_T * headers ,
MATCH_DATA_T * match_data ) {
    // local / global memory and objects .
    return return_value ;
}
```

# Proposing Match+Lambda Abstraction (2)

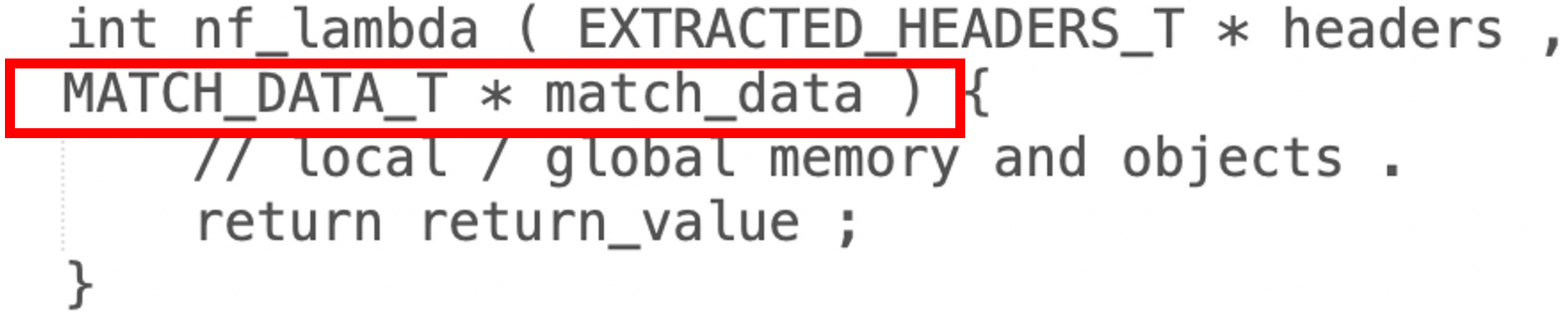
- Programming lambdas
  - Cellular operators will provide one or more lambdas for their network functions
    - Such as Update SM context, Release SM context...
  - Signature of each lambda has two predefined arguments: headers and match_data
  - The lambdas will operate directly on headers and match_data without parsing packets



```
int nf_lambda ( EXTRACTED_HEADERS_T * headers ,
MATCH_DATA_T * match_data ) {
    // local / global memory and objects .
    return return_value ;
}
```

# Proposing Match+Lambda Abstraction (2)

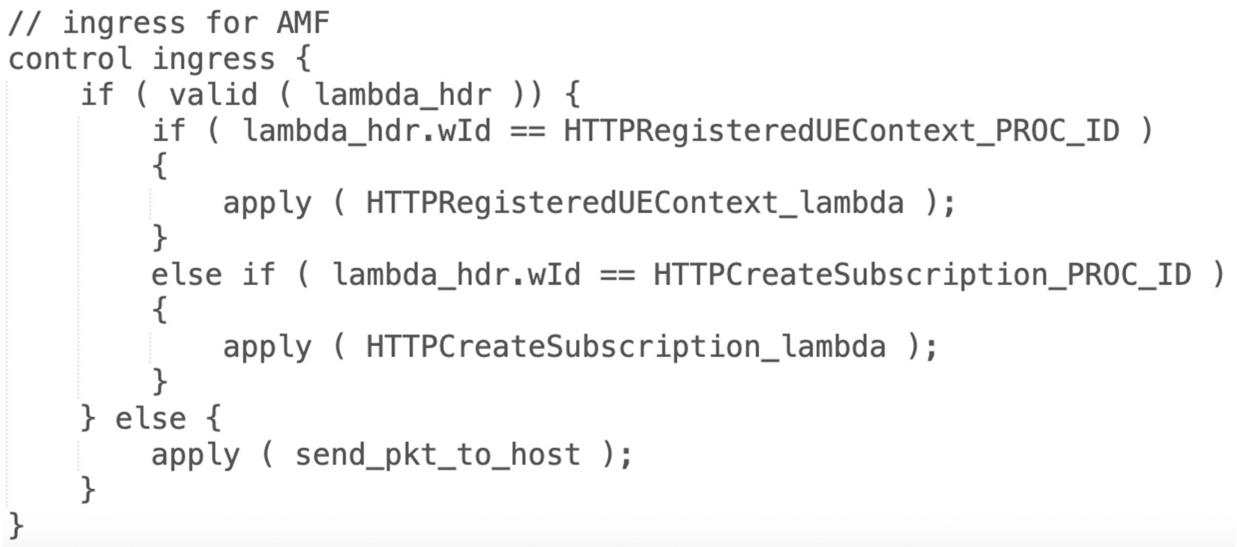
- Programming lambdas
  - Cellular operators will provide one or more lambdas for their network functions
    - Such as Update SM context, Release SM context...
  - Signature of each lambda has two predefined arguments: headers and match_data
  - The lambdas will operate directly on headers and match_data without parsing packets



```
int nf_lambda ( EXTRACTED_HEADERS_T ∗ headers ,
MATCH_DATA_T ∗ match_data ) {
    // local / global memory and objects .
    return return_value ;
}
```

# Proposing Match+Lambda Abstraction (3)

- Expressing match
  - Match lambda ID of the packet with corresponding lambda
  - Workload manager pairs the lambdas and match stage into a single Match+Lambda program, and prepend it with a cellular-specific P4 packet-parsing logic
  - Workload manager compiles and transforms this program into a format that a programmable hardware can execute



```
// ingress for AMF
control ingress {
    if ( valid ( lambda_hdr )) {
        if ( lambda_hdr.wId == HTTPRegisteredUEContext_PROC_ID )
        {
            apply ( HTTPRegisteredUEContext_lambda );
        }
        else if ( lambda_hdr.wId == HTTPCreateSubscription_PROC_ID )
        {
            apply ( HTTPCreateSubscription_lambda );
        }
    } else {
        apply ( send_pkt_to_host );
    }
}
```

# Conclusion

- Next generation mobile core is hosting more emerging applications and control plane traffic.

- Mobile core is rearchitecting for scalability and flexibility.

- 5G core control plane characteristics:
  - Existing core implementation fails meet the latency requirement.
  - Bottleneck network functions and operations.

- Proposing a serverless programmable hardware design and Match+Lambda abstraction to accelerate 5G core.

# Thank You