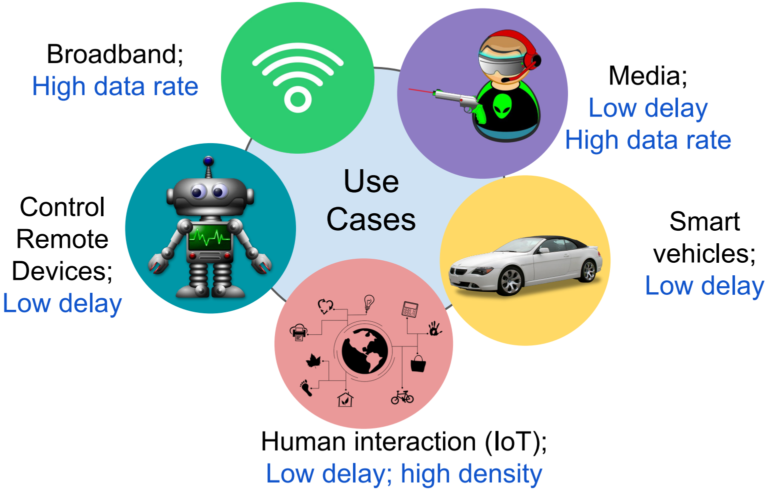
TurboEPC: Leveraging Network Programmability to Accelerate the Mobile Packet Core

Rinku Shah, Vikas Kumar, Mythili Vutukuru, Purushottam Kulkarni

Department of Computer Science & Engineering
Indian Institute of Technology Bombay

ACM SIGCOMM Symposium on SDN Research (SOSR)
March 3, 2020

Requirements of telecom applications



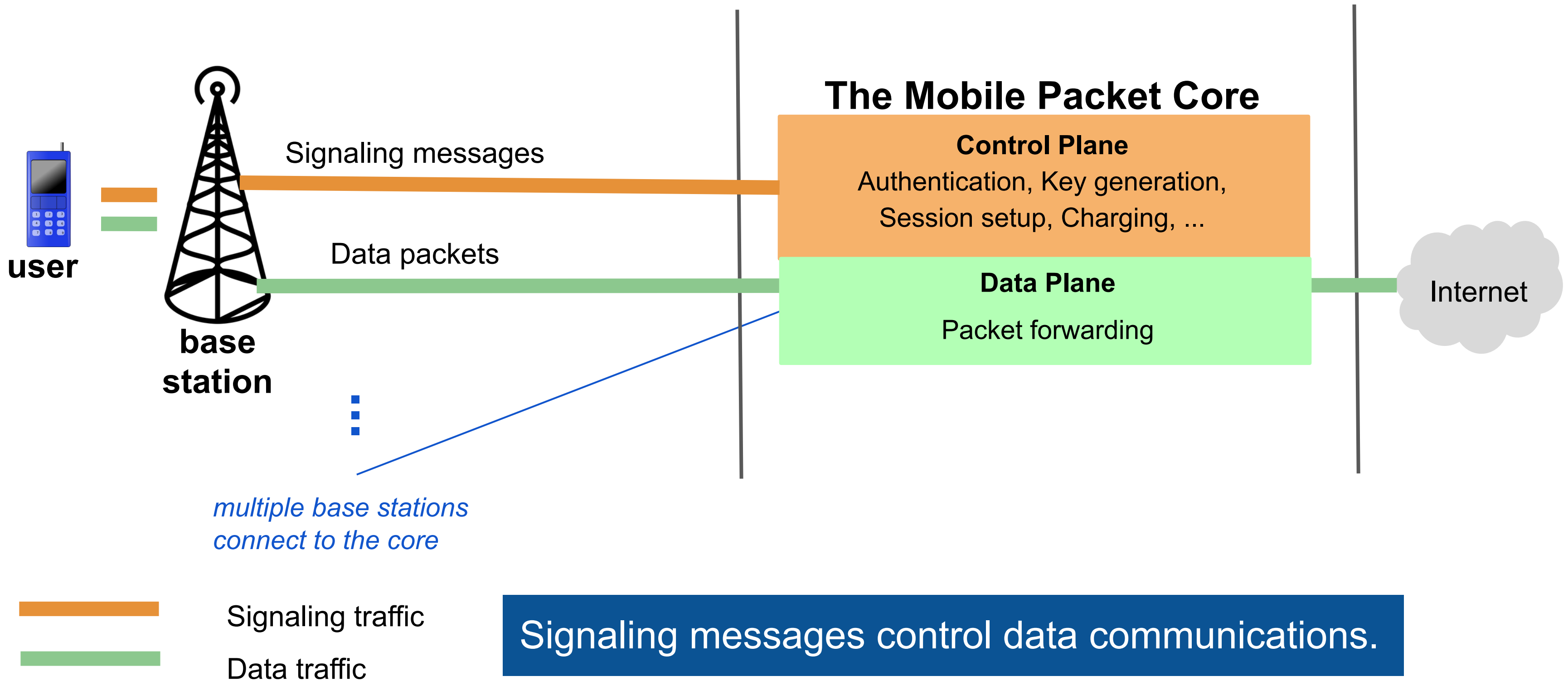
Requirements of recent telecom applications *

- Ultra-low latencies for control/data plane
  - Latency as low as 1ms - 10ms for certain services
- Extremely high data rates
- Large number of mobile nodes per cell

Mobile network providers should ensure required user scalability and performance

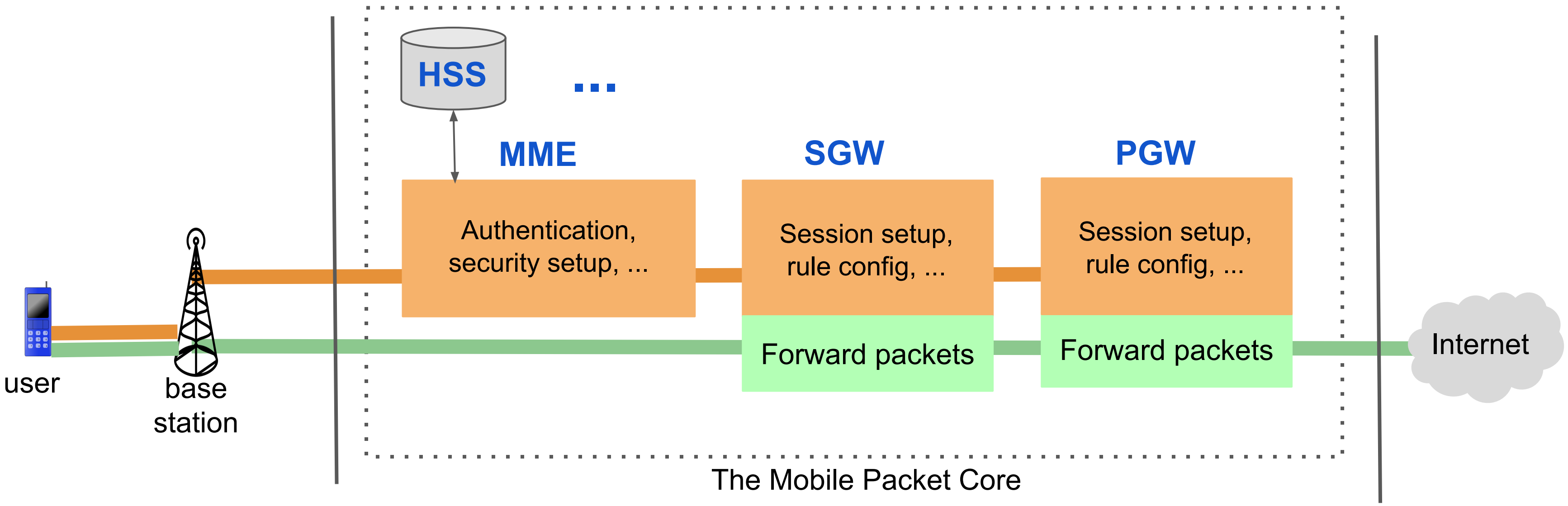
* 5G 3GPP specifications (2017), Qualcomm (2018)

Typical cellular network



Signaling messages control data communications.

Traditional Mobile Packet Core



Not flexible!

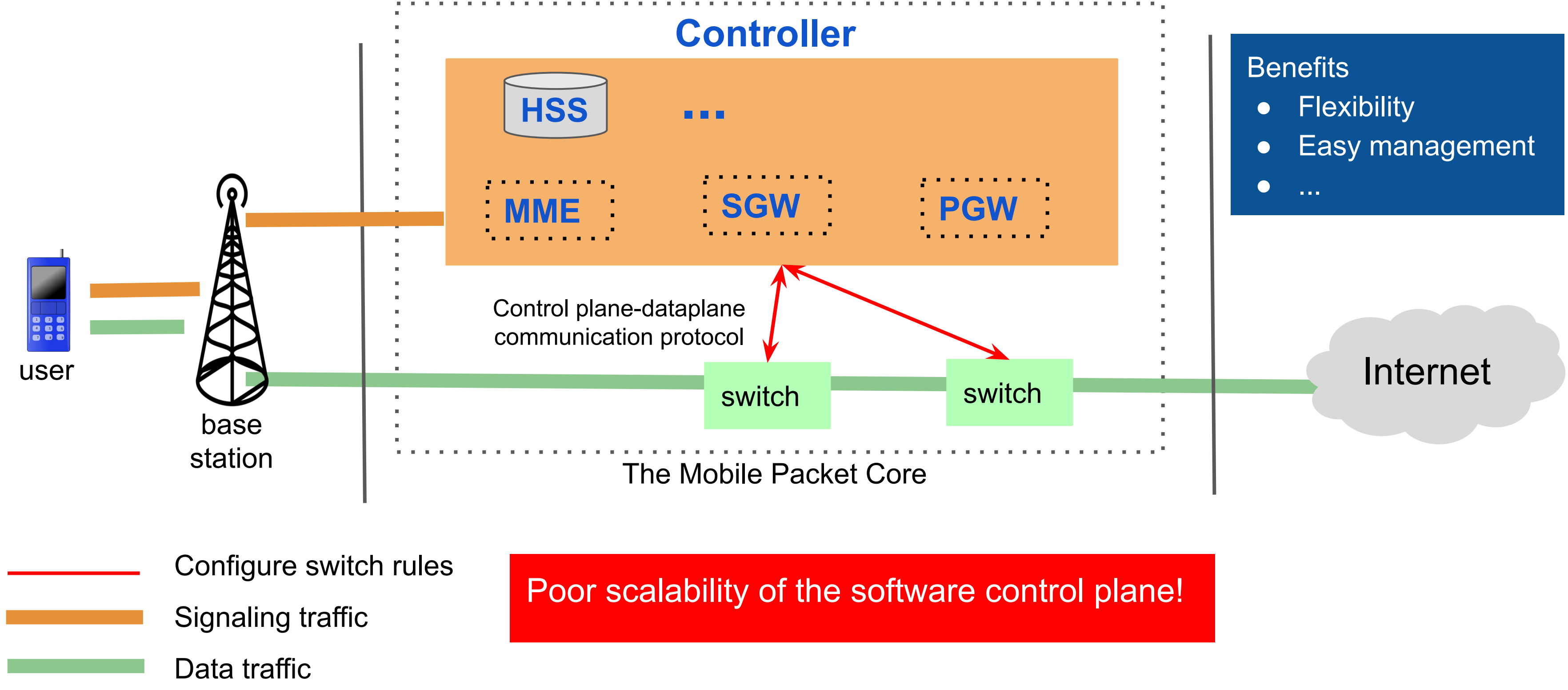
Signaling messages:

- **Attach:** user registers with the network
- **S1 release:** deactivate data channel when idle
- **Service req:** activate data channel when active
- **Handover:** manage network connection when the user changes location
- **Detach:** user is deregistered from the network

MME: Mobility management entity
S/PGW: Service/Packet gateway
HSS: Home subscriber server

CUPS-based Mobile Packet Core

CUPS: Control User Plane Separation



Benefits

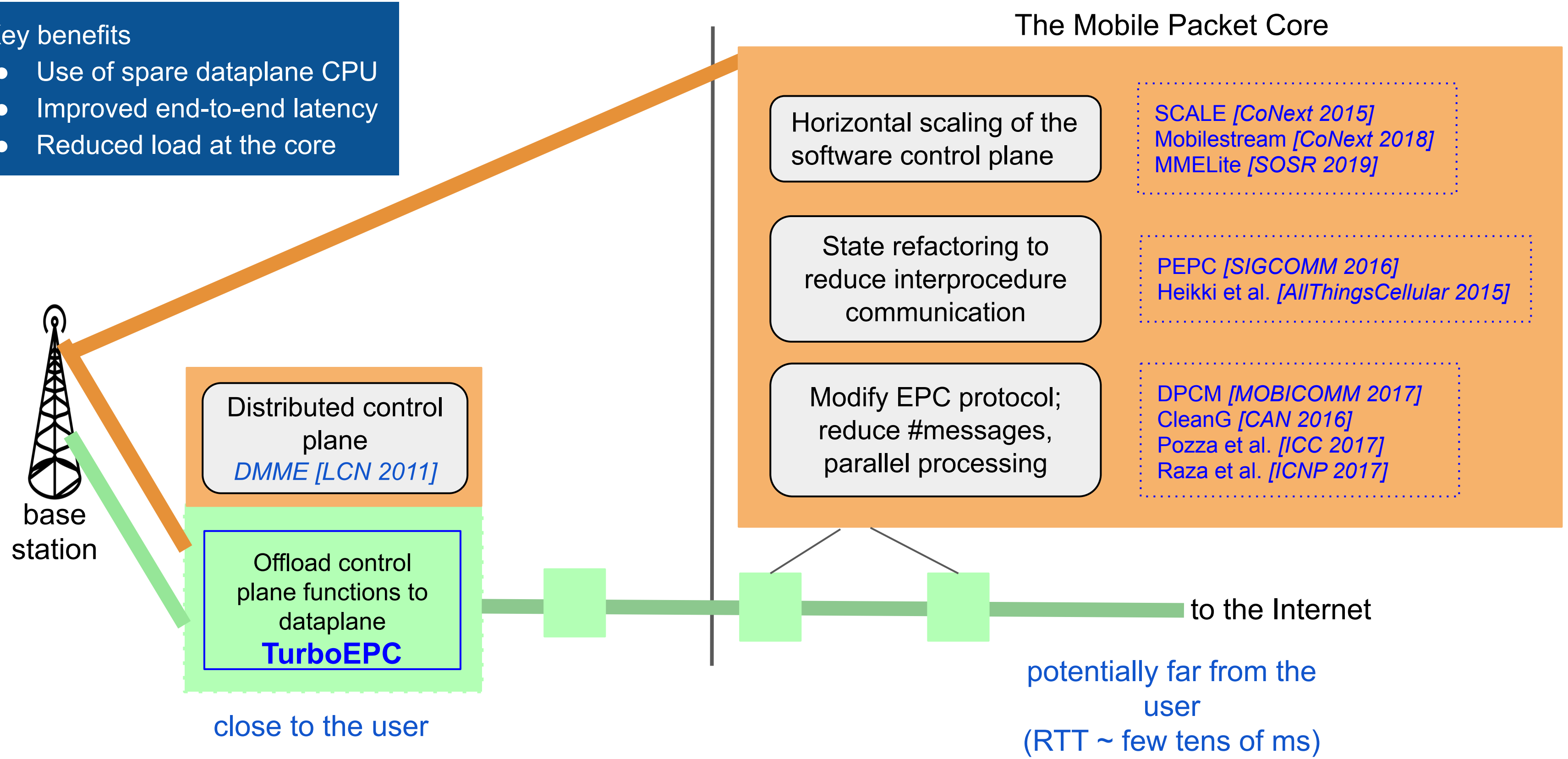
- Flexibility
- Easy management
- ...

Poor scalability of the software control plane!

Solution approaches

Key benefits

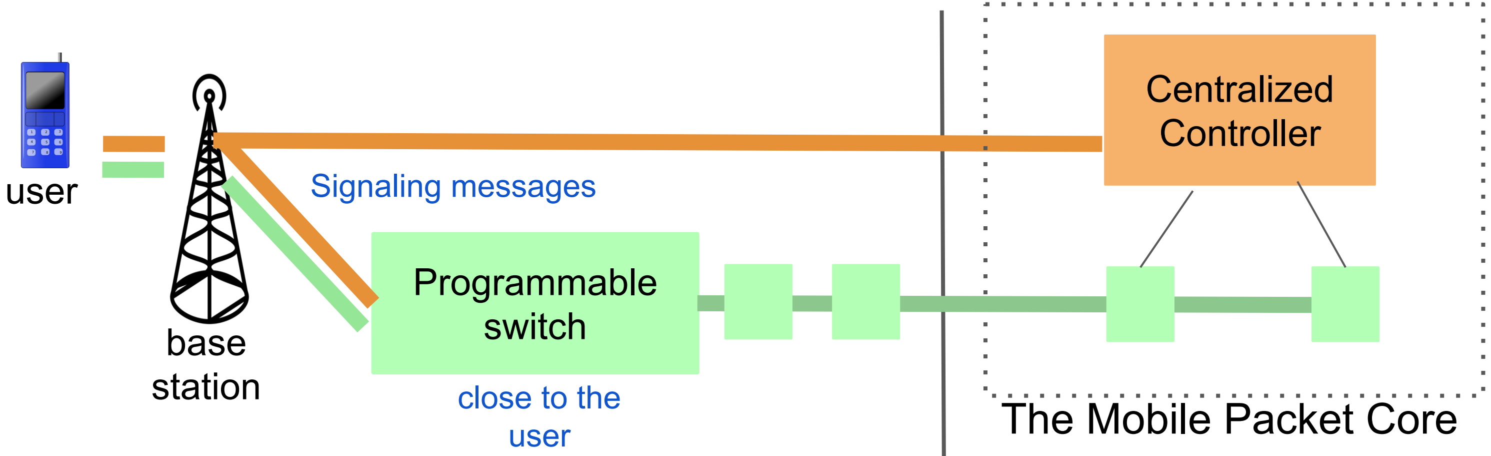
- Use of spare dataplane CPU
- Improved end-to-end latency
- Reduced load at the core



Key idea of TurboEPC

Offload subset of mobile core processing to the edge (close to the user)

at programmable hardware switches

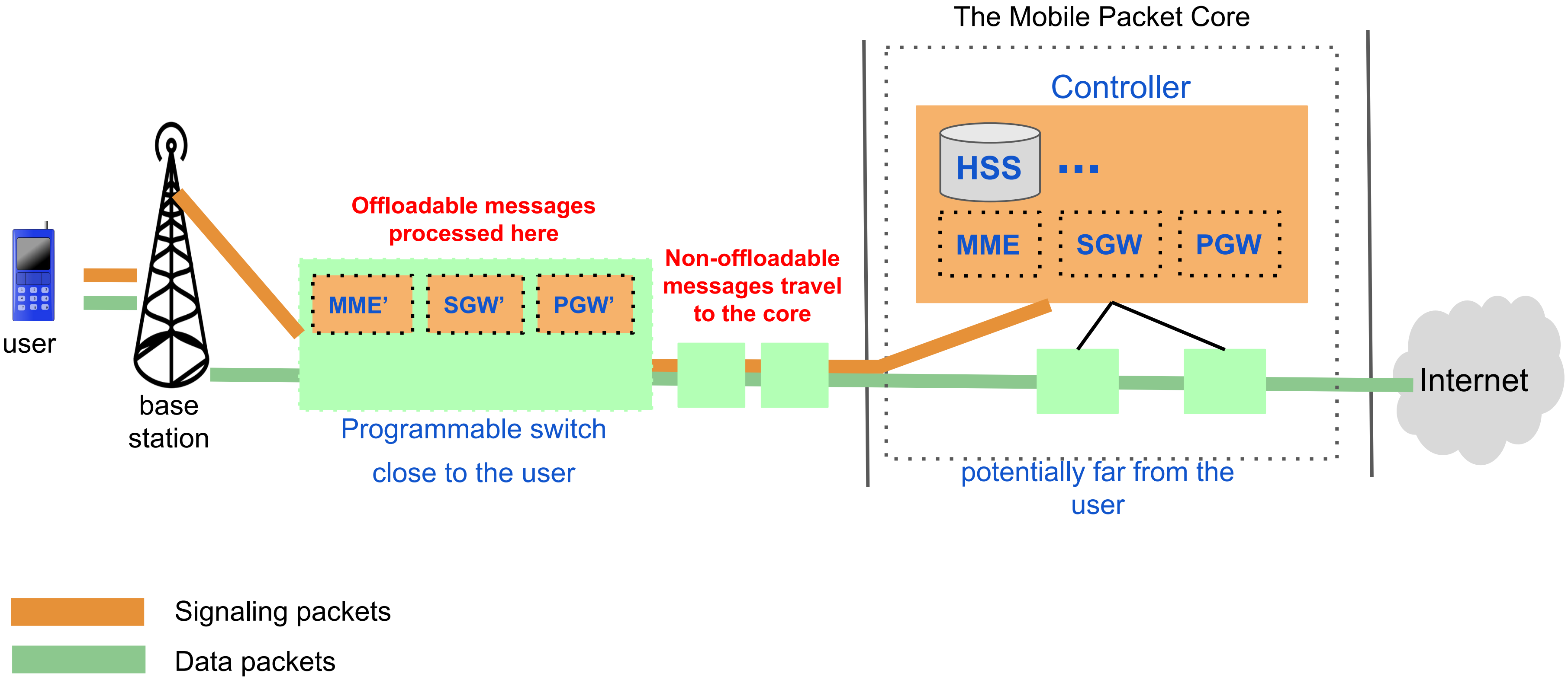


potentially far from the user

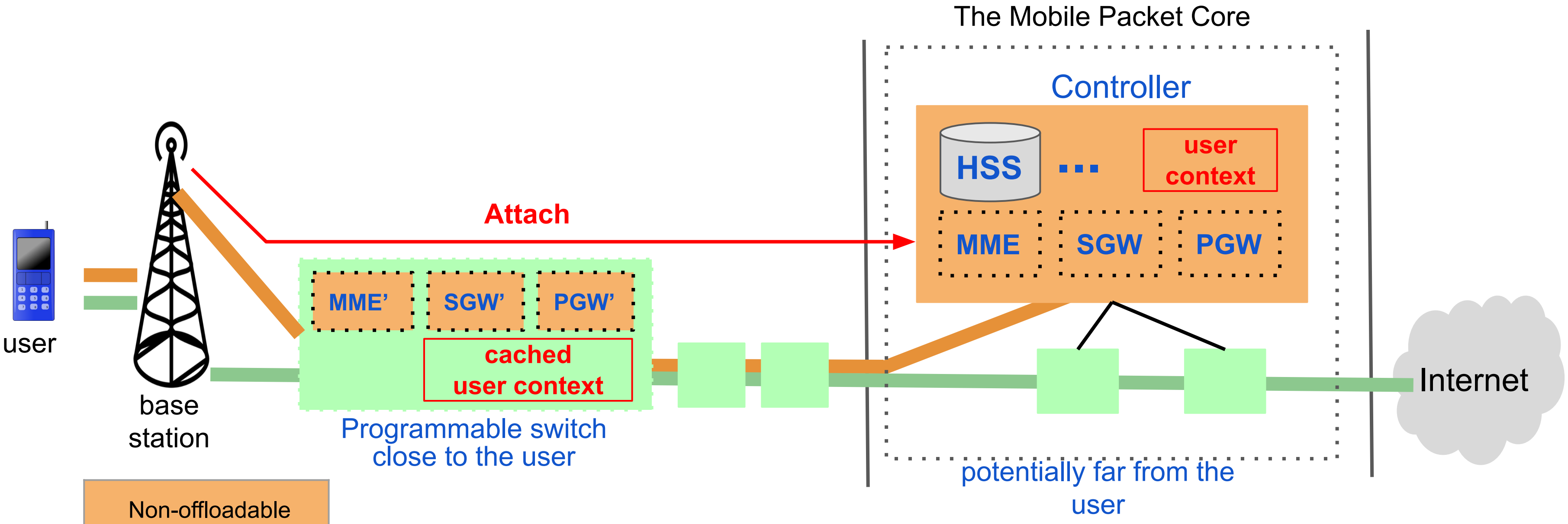
What all can we program at the switch?

- Packet header
- Packet parser
- Packet lifecycle
- ...

TurboEPC architecture



TurboEPC: Non-offloadable signaling message processing

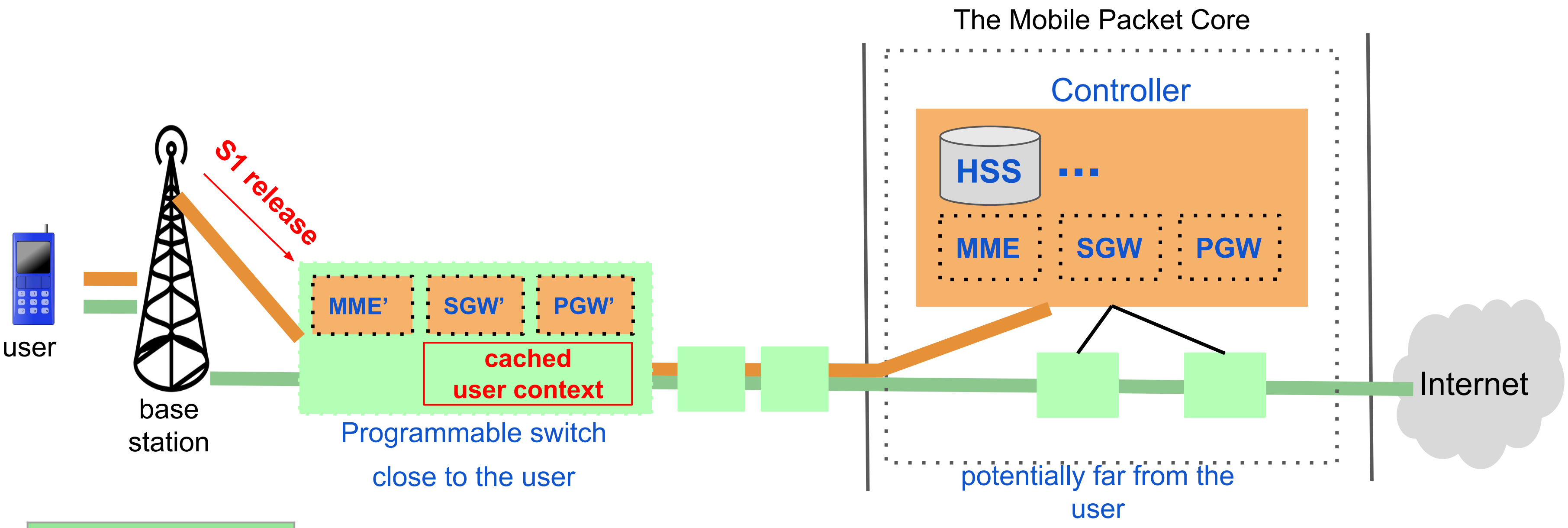


Non-offloadable signaling messages
Attach
Detach
Handover

Attach procedure

- User context is created at the core
- This context is then cached at the edge switch

TurboEPC: Offloadable signaling message processing



Offloadable signaling messages
S1 release
Service req

S1 release / Service request processing

- Cached user context is accessed and modified at the edge switch

Challenge I: Which EPC signaling messages can be offloaded?

EPC state classification

Offloadable state:

- Switch-local or session-wide scope
- Not accessed concurrently from multiple network locations

Examples
User connection state: idle, active
Forwarding state: IP addr & tunnel ID
Temporary subscriber identifiers
User QoS state, charging state
...

Non-offloadable state:

- Global, network-wide scope
- Can be accessed concurrently from multiple network locations

Examples
Security keys generated during the session (HSS)
User registration state; registered or not?
Free pool of IP addr & tunnel identifiers
Permanent subscriber identifiers
...

Challenge I: Which EPC signaling messages can be offloaded?

If all states accessed by the message are Offloadable, message is *Offloadable*.

Classes of EPC state

EPC signaling messages

	Offloadable state access	Non-offloadable state access
Attach	user connection state, forwarding state, user QoS state, temporary subscriber identifiers, location state	security keys, user registration state, permanent identifiers, IP address pool, tunnel identifier pool
Detach	user connection state, forwarding state, temporary subscriber identifiers, location state	user registration state, permanent identifiers, IP address pool, tunnel identifier pool
Service request	user connection state, forwarding state, temporary subscriber identifiers	---
S1 release	user connection state, forwarding state, temporary subscriber identifiers	---
Handover	user connection state, forwarding state, user QoS state, temporary subscriber identifiers, location state	security keys, user registration state, permanent identifiers, IP address pool, tunnel identifier pool

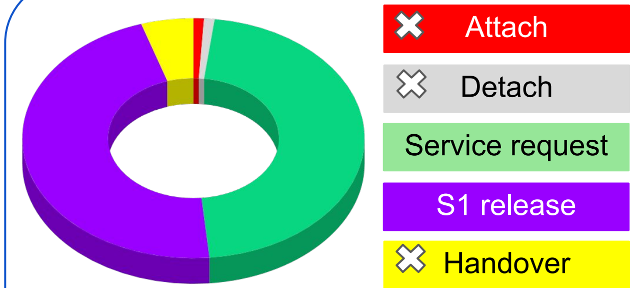
Probable *Offloadable* candidates:
Service request, S1 release

Solution I: Guide to identify offloadable messages

An EPC message is a good candidate for offload if,

- All states accessed are Offloadable.
- It spans significant fraction of total traffic.
- It is possible to implement over programmable switch.
- Offloadable state is not accessed frequently by the non-offloadable message.

Not offloadable



EPC traffic distribution [1, 2]

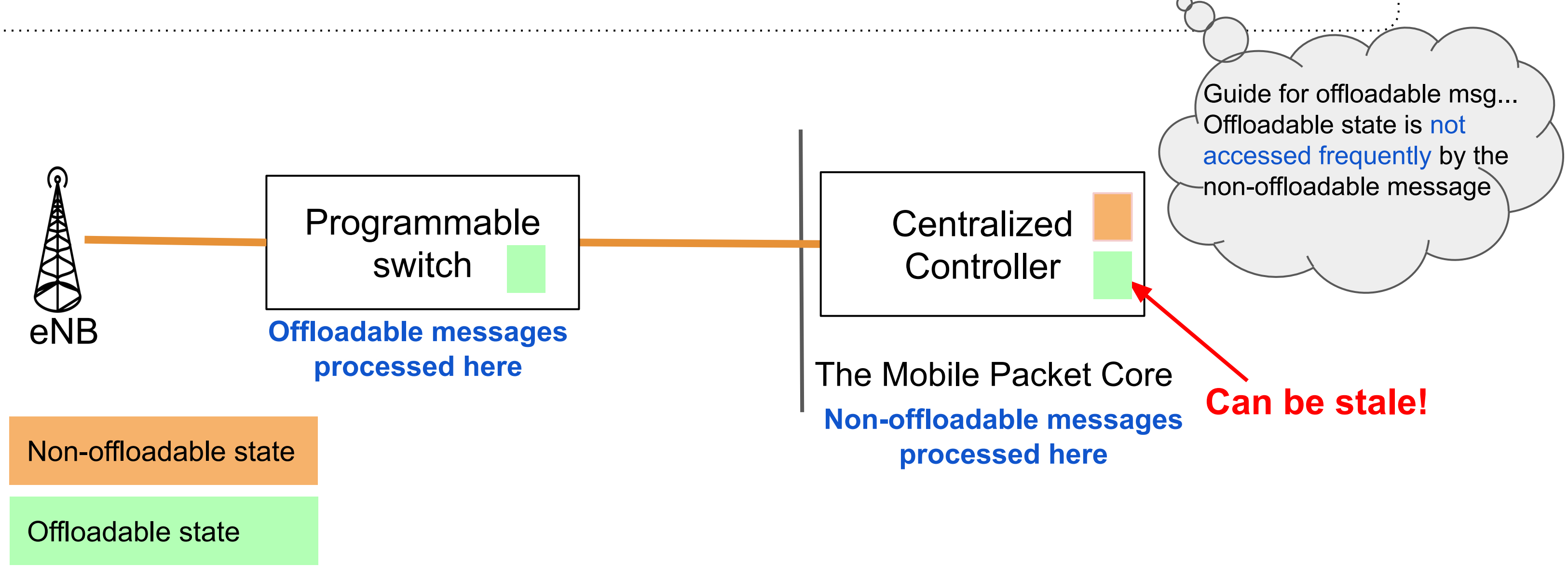
TurboEPC offloads *S1 release* & *Service request* messages to the programmable switch at the edge

Our ideas can be generalized to other systems as well; where these definitions apply.

[1] Managing LTE Core Network Signaling Traffic. Nokia. 2013.
[2] Core network and transmission dimensioning. ITU-INT. 2016.

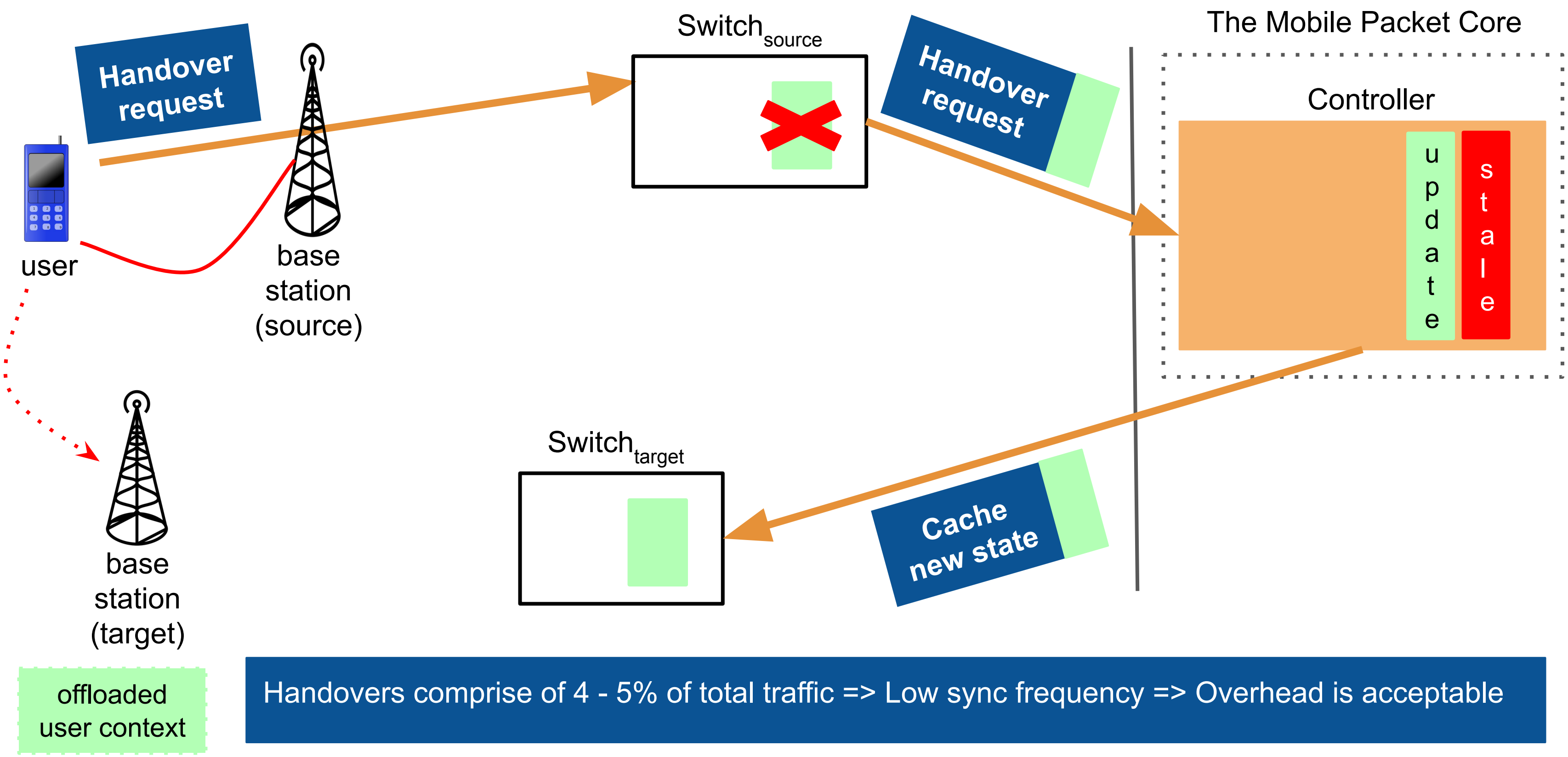
Challenge II: Ensure consistency of offloaded state

What if non-offloadable message needs access to offloaded state?



Naive solution: On every state update, sync state from switch to controller.
Problem: expensive, negates benefit of offload

Solution II: Synchronize state on demand



Handovers comprise of 4 - 5% of total traffic => Low sync frequency => Overhead is acceptable

Challenge III: State offload to memory-constrained hardware

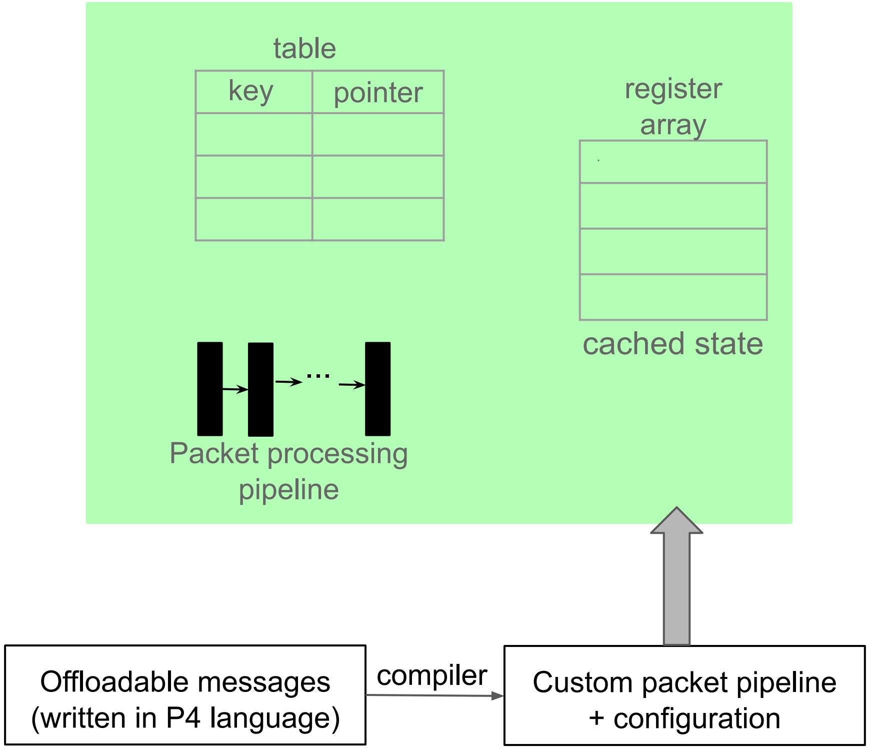
- TurboEPC per-user state size on edge switch ≈ 96 bytes
- Netronome smartNIC[5] capacity ≈ 65K EPC users
- Barefoot Tofino switch[4] capacity ≈ few 100K EPC users
- Typical number of EPC users per core network[1,2,3] ≈ few millions

Offload the state only for a subset of users
AND/OR
Use more than one switch to store offload state

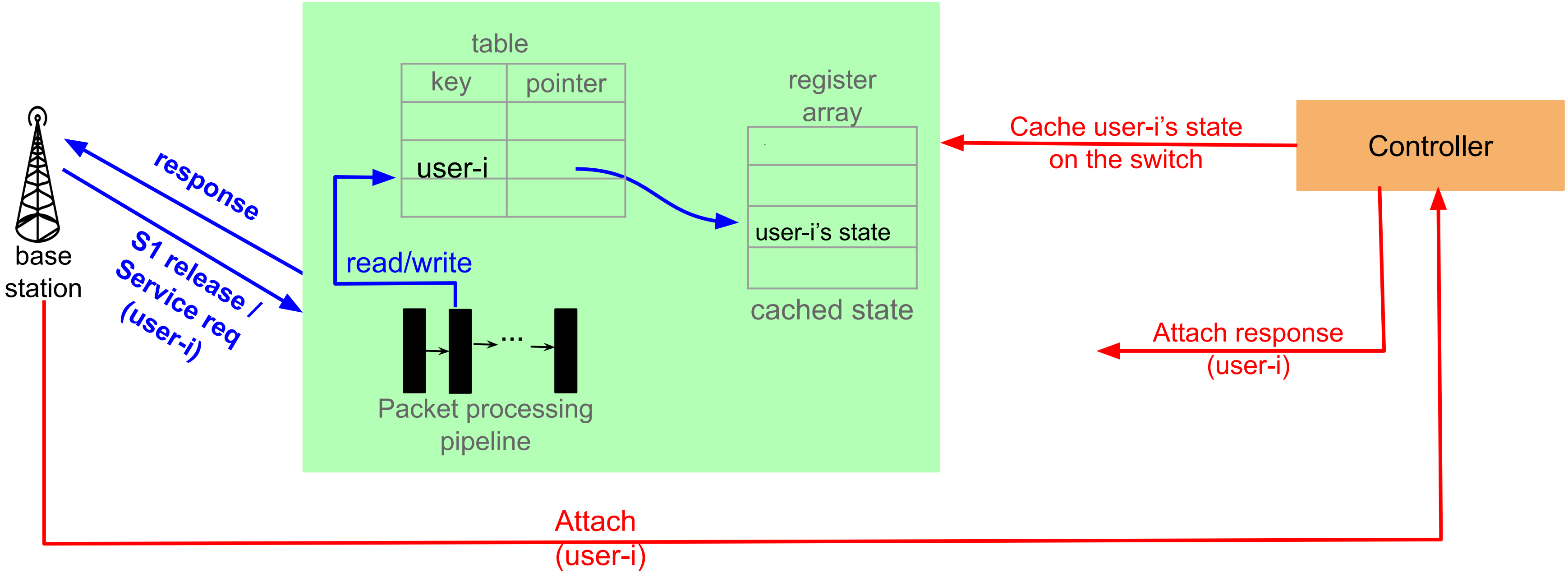
TurboEPC partitions the offload state and stores it over multiple switches.

[1] https://telecom.economictimes.indiatimes.com/news/reliance-jio-gujarat-andhra-pradesh-top-circles-total-user-base-crosses-24-million-mark/55032251, Sep 2016.
[2] https://telecom.economictimes.indiatimes.com/news/total-mobile-subscribers-base-grows-to-981-65-million/62549385, Jan 2018.
[3] https://www.itu.int/en/ITU-D/Regional-Presence/AsiaPacific/SiteAssets/Pages/Events/2016/Aug-WBB-Iran/Wirelessbroadband/corenetworkdimensioning.pdf, Aug 2016.
[4] https://noviflow.com/wp-content/uploads/NoviWare-Tofino-Datasheet.pdf
[5] https://www.netronome.com/m/documents/PB_NFP-4000.pdf

Implementation of TurboEPC dataplane switch



Implementation of TurboEPC dataplane switch



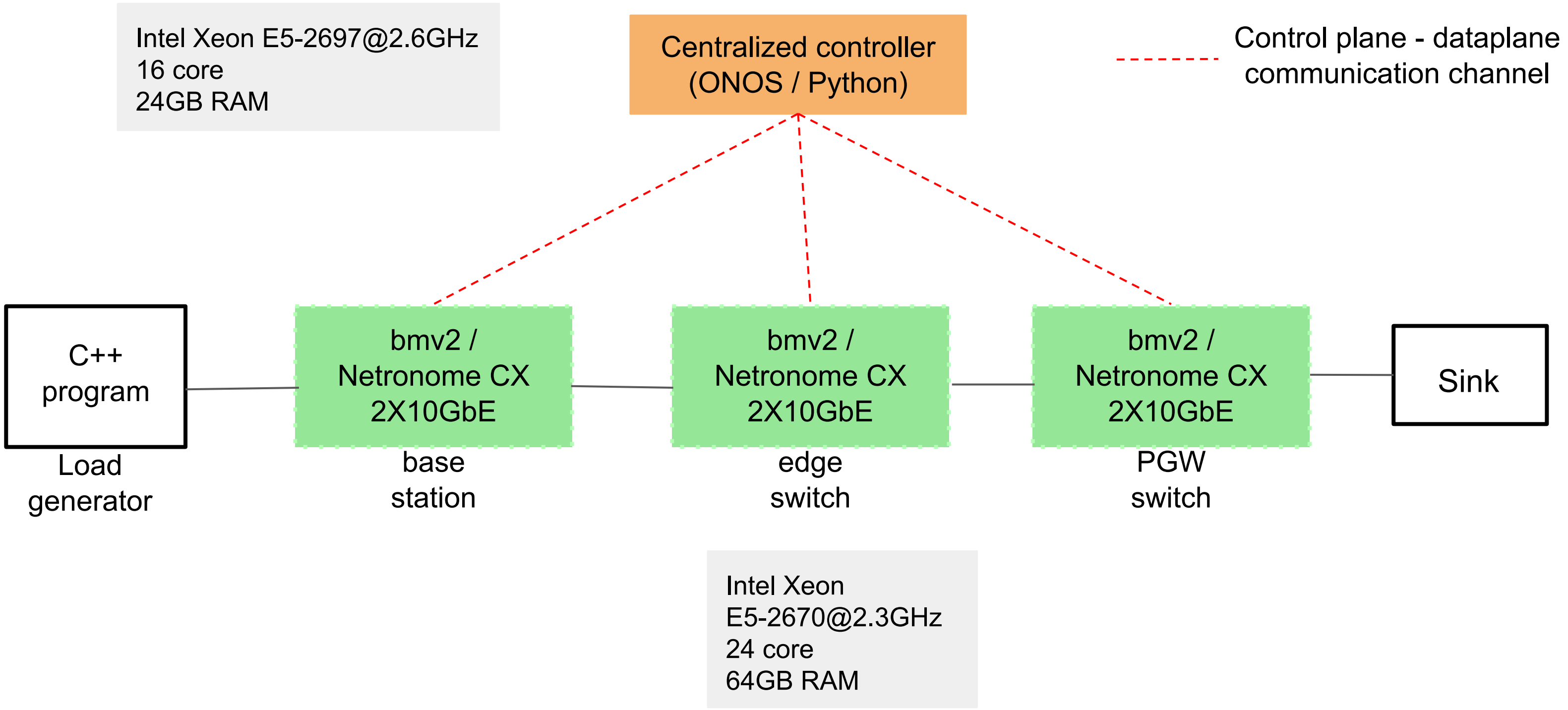
P4 tables

- Match/action support
- Updated via control plane

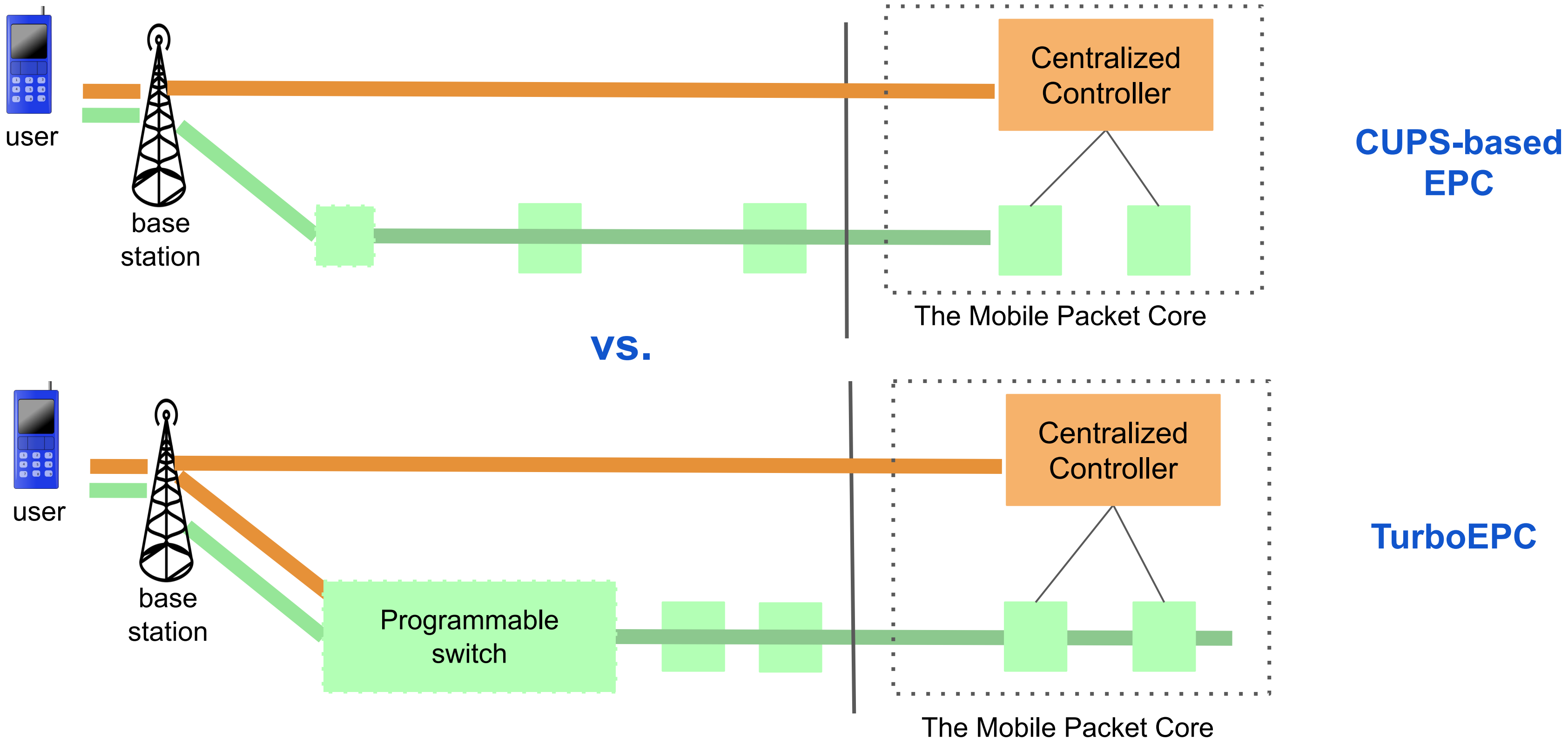
P4 register array

- No match/action support; accessed using index
- Can be updated within dataplane

TurboEPC software and hardware setup

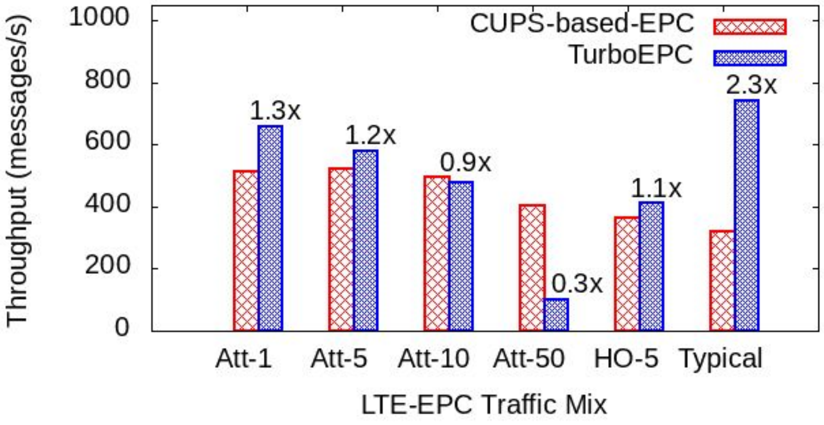


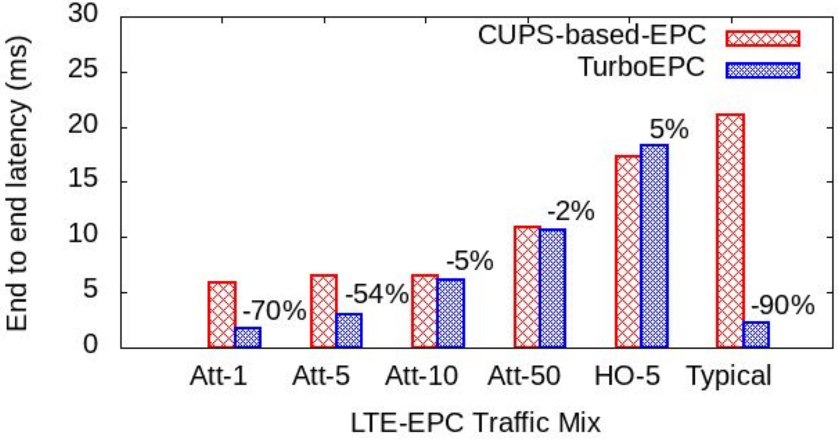
TurboEPC evaluation: CUPS-based EPC vs. TurboEPC



Throughput and latency results

TurboEPC results are for single edge switch (software)





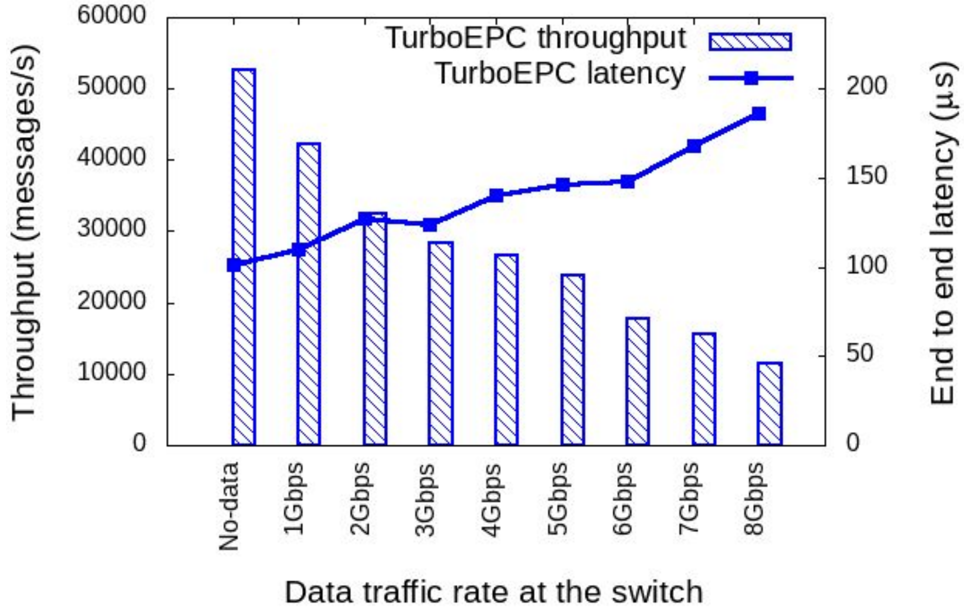
Typical traffic *

Attach/Detach	≈ 2%
S1 release Service request	≈ 90 - 94%
Handover	≈ 5%

* Nokia, 2013 ; ITU-INT, 2016

- Typical traffic: throughput improvement = 2.3x, latency reduction = 90%
- For Att-1 traffic-mix (4 TurboEPC edge switches),
  TurboEPC throughput = 5x of CUPS-based EPC (only 20% core CPU used)
- With high non-offloadable component in the traffic mix,
  TurboEPC performance degrades, for example, Att-50

TurboEPC hardware switch performance



Impact of Data traffic interference

Impact of data traffic interference

- Throughput drops from 52K to 12K
- Latency increases from 100 µs to 180 µs
- Even at linerate, performance much better than CUPS-based EPC

TurboEPC (hardware) vs. CUPS-based EPC

- 22x - 102x throughput improvement
- 97% - 98% latency reduction

Summary

- Key idea: revisit boundary between control and data planes in mobile packet core
  - Process a subset of signaling messages in programmable edge switches
- Improves signaling throughput, reduces processing latency
- Idea extends to the future 5G core

TurboEPC's source code
https://github.com/networkedsystemsIITB/turboepc

Contact: rinku@cse.iitb.ac.in