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

- **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)
Signaling messages control data communications.
Traditional Mobile Packet Core

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

**Forward packets**
- User registers with the network
- Deactivate data channel when idle
- Activate data channel when active
- Manage network connection when the user changes location
- User is deregistered from the network

**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

**Not flexible!**

MME: Mobility management entity
S/PGW: Service/Packet gateway
HSS: Home subscriber server
CUPS-based Mobile Packet Core

CUPS: Control User Plane Separation

- Configure switch rules
- Signaling traffic
- Data traffic

Internet

Controller

HSS...MME SGW PGW

Control plane-dataplane communication protocol

The Mobile Packet Core

base station

user

- 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

Offload control plane functions to dataplane
TurboEPC
close to the user

Horizontal scaling of the software control plane

State refactoring to reduce interprocedure communication

Modify EPC protocol; reduce #messages, parallel processing

The Mobile Packet Core

potentially far from the user
(RTT ~ few tens of ms)

Distributed control plane
DMME [LCN 2011]

Solution approaches

- TurboEPC
- Offload control plane functions to dataplane
- Distributed control plane
- Use of spare dataplane CPU
- Improved end-to-end latency
- Reduced load at the core

The Mobile Packet Core

- Horizontal scaling of the software control plane
- State refactoring to reduce interprocedure communication
- Modify EPC protocol; reduce #messages, parallel processing

- SCALE [CoNext 2015]
- Mobilestream [CoNext 2018]
- MMELite [SOSR 2019]
- PEPC [SIGCOMM 2016]
- Heikki et al. [AllThingsCellular 2015]
- DPCM [MOBICOMM 2017]
- CleanG [CAN 2016]
- Pozza et al. [ICC 2017]
- Raza et al. [ICNP 2017]

- DPCM [MOBICOMM 2017]
- CleanG [CAN 2016]
- Pozza et al. [ICC 2017]
- Raza et al. [ICNP 2017]
Key idea of TurboEPC

Offload subset of mobile core processing to the edge (close to the user) at programmable hardware switches

What all can we program at the switch?
- Packet header
- Packet parser
- Packet lifecycle
- ...

Programmable switch

Centralized Controller

The Mobile Packet Core

user

base station

Signaling messages
TurboEPC architecture

The Mobile Packet Core

Offloadable messages processed here
Non-offloadable messages travel to the core

Programmable switch close to the user

potentially far from the user

User

base station

Signaling packets
Data packets
TurboEPC: Non-offloadable signaling message processing

The Mobile Packet Core

Controller

HSS

user context

MME

SGW

PGW

Internet

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

Attach

Detach

Handover
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

<table>
<thead>
<tr>
<th>EPC signaling messages</th>
<th>Offloadable state access</th>
<th>Non-offloadable state access</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attach</strong></td>
<td>user connection state, forwarding state, user QoS state, temporary subscriber identifiers, location state</td>
<td>security keys, user registration state, permanent identifiers, IP address pool, tunnel identifier pool</td>
</tr>
<tr>
<td><strong>Detach</strong></td>
<td>user connection state, forwarding state, temporary subscriber identifiers, location state</td>
<td>user registration state, permanent identifiers, IP address pool, tunnel identifier pool</td>
</tr>
<tr>
<td><strong>Service request</strong></td>
<td>user connection state, forwarding state, temporary subscriber identifiers</td>
<td>---</td>
</tr>
<tr>
<td><strong>S1 release</strong></td>
<td>user connection state, forwarding state, temporary subscriber identifiers</td>
<td>---</td>
</tr>
<tr>
<td><strong>Handover</strong></td>
<td>user connection state, forwarding state, user QoS state, temporary subscriber identifiers, location state</td>
<td>security keys, user registration state, permanent identifiers, IP address pool, tunnel identifier pool</td>
</tr>
</tbody>
</table>

**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.

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

---

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

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.

\(^5\) https://www.netronome.com/m/documents/PB_NFP-4000.pdf
Implementation of TurboEPC dataplane switch

- Packet processing pipeline
- Table
  - key
  - pointer
- Register array
- Cached state
- Offloadable messages (written in P4 language)
- Custom packet pipeline + configuration

Compiler
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

- Controller
  - Cache user-i’s state on the switch
  - Attach response (user-i)

- Packet processing pipeline
  - Table:
    - Key
    - Pointer
    - User-i
  - Register array:
    - User-i’s state
  - Cached state

- **S1 release / Service req (user-i)**
- **Attach (user-i)**
TurboEPC software and hardware setup

Intel Xeon E5-2697@2.6GHz
16 core
24GB RAM

Intel Xeon E5-2670@2.3GHz
24 core
64GB RAM

Centralized controller
(ONOS / Python)

Control plane - dataplane communication channel

C++ program
Load generator

bmv2 / Netronome CX 2X10GbE
base station

bmv2 / Netronome CX 2X10GbE
edge switch

bmv2 / Netronome CX 2X10GbE
PGW switch

Sink

Intel Xeon E5-2670@2.3GHz
24 core
64GB RAM
TurboEPC evaluation: CUPS-based EPC vs. TurboEPC
Throughput and latency results

TurboEPC results are for single edge switch (software)

- 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

<table>
<thead>
<tr>
<th>Traffic Mix</th>
<th>Attachment/Release</th>
<th>Service Request</th>
<th>Handover</th>
</tr>
</thead>
<tbody>
<tr>
<td>Att-1</td>
<td>≈ 2%</td>
<td>≈ 90 - 94%</td>
<td>≈ 5%</td>
</tr>
<tr>
<td>Att-5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Att-10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Att-50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO-5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Typical</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Nokia, 2013 ; ITU-INT, 2016*
TurboEPC hardware switch performance

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