MobileStream - A New Scalable, Programmable, Evolvable and Reliable Mobile Control Plane Platform

Junguk Cho, Ryan Stutsman, and Jacobus Van der Merwe
LTE/EPC Mobile Network

- eNBs
- MME
- SGW
- PGW
- HSS

Radio Access Network
Evolved Packet Core (EPC) Network

S1AP
S11
S6

Control Plane
Data Plane

Internet
LTE/EPC Mobile Network

Radio Access Network

Evolved Packet Core (EPC) Network

Control Plane

Data Plane
Emerging Control Plane Concerns for 5G Network

- New services in 5G network
  - MME: Control as well as data plane for small IoT data
  - C-SGN: Cellular IoT Serving Gateway Node
  - MEC: Mobile Edge Computing
Emerging Control Plane Concerns for 5G Network

- New services in 5G network
  - MME: Control as well as data plane for small IoT data
  - C-SGN: Cellular IoT Serving Gateway Node
  - MEC: Mobile Edge Computing

Programmable & evolvable
Emerging Control Plane Concerns for 5G Network

- New services in 5G network
  - MME: Control as well as data plane for small IoT data
  - C-SGN: Cellular IoT Serving Gateway Node
  - MEC: Mobile Edge Computing

**Programmable & evolvable**

- Control signal storm
  - Mainly from ~26 Billion IoT by 2020
Emerging Control Plane Concerns for 5G Network

● New services in 5G network
  ○ MME: Control as well as data plane for small IoT data
  ○ C-SGN: Cellular IoT Serving Gateway Node
  ○ MEC: Mobile Edge Computing

Programmable & evolvable

● Control signal storm
  ○ Mainly from ~26 Billion IoT by 2020

Scalable
Emerging Control Plane Concerns for 5G Network

- New services in 5G network
  - MME: Control as well as data plane for small IoT data
  - C-SGN: Cellular IoT Serving Gateway Node
  - MEC: Mobile Edge Computing

  **Programmable & evolvable**

- Control signal storm
  - Mainly from ~26 Billion IoT by 2020

  **Scalable**

- Software-based EPC components running on cloud
  - Can fail anytime
Emerging Control Plane Concerns for 5G Network

● New services in 5G network
  ○ MME: Control as well as data plane for small IoT data
  ○ C-SGN: Cellular IoT Serving Gateway Node
  ○ MEC: Mobile Edge Computing

  **Programmable & evolvable**

● Control signal storm
  ○ Mainly from ~26 Billion IoT by 2020

  **Scalable**

● Software-based EPC components running on cloud
  ○ Can fail anytime

  **Reliable**
Our Approach

- Basic building blocks
- Assemble what you want
Our Approach

- Basic building blocks
- Assemble what you want

Programmable & Evolvable
Our Approach

- Basic building blocks
- Assemble what you want
  
  **Programmable & Evolvable**

- Run it on realtime stream frameworks
Our Approach

- Basic building blocks
- Assemble what you want
  
  Programmable & Evolvable

- Run it on realtime stream frameworks
  
  Scalable & Reliable
Basic Building Blocks: Decompose Control Functions

- eNB
- MME

**Basic Building Blocks**
- SCTP
- S1AP
- NAS
Basic Building Blocks: Decompose Control Functions

- Basic function of control plane
  - User context management
  - Security functions
    - Generate security keys
    - Integrity check for NAS messages
    - de/encrypt NAS messages
Basic Building Blocks: Decompose Control Functions

- **Others**
  - S/PGW control plane
  - Protocol header decoder
  - Your own blocks
Assemble What You Want: MME supporting NBloT

**Basic Building Blocks**

- SCTP
- S1AP
- NAS
- UEContext Manager
- AUC
- SECURITY
- S/PGW-C
- HEADER DECODER
- Your own blocks
Assemble What You Want: MME supporting NBloT
Assemble What You Want: MME supporting NBloT

MME supporting NBloT
Run it on Realtime Stream Framework

MME supporting NBloT

Realtime stream framework on cloud
MobileStream: A New Control Plane Platform

Key-value storages

Get/set UE contexts

Control Plane Applications

S/P-GWs

HSS

eNBs

Stream Framework on cloud
Why Realtime Stream Framework?

- Characteristics of future control messages
  - Tons of control messages from IoT devices
  - Generated by UE events (e.g., attach, mobility)
  - Unexpected loads
Why Realtime Stream Framework?

- Characteristics of future control messages
  - Tons of control messages from IoT devices
  - Generated by UE events (e.g., attach, mobility)
  - Unexpected loads

Future control messages == BIG DATA
Why Realtime Stream Framework?

- Realtime stream frameworks (RTSF)
  - Process unlimited data continuously
  - Event-based big data
  - Low latency and high throughput using computation parallelism
  - Fault-tolerance mechanism
  - Programming APIs and management tools
Why Realtime Stream Framework?

- Realtime stream frameworks (RTSF)
  - Process unlimited data continuously
  - Event-based big data
  - Low latency and high throughput using computation parallelism
  - Fault-tolerance mechanism
  - Programming APIs and management tools

Capabilities of RTSF meet control plane requirements (scalable, programmable, reliable) very well!
Design Challenges

- How can we design control plane application?
- What are the programming APIs?
- How can we manage user states?
Design Challenges

- How can we design control plane application?
- What are the programming APIs?
- How can we manage user states?
Stateless as much as Possible

- Scale out independently
- Easy scale-in/out
- Robust to failure
- Round-robin routing

MME supporting NBloT
Stateless as much as Possible

- Scale out independently
- Easy scale-in/out
- Robust to failure
- Round-robin routing

User context management

Remote Key-value storage

MME supporting NB-IoT
Stateless as much as Possible

- One synchronization point at entry point

User context management

MME supporting NBloT

Round-robin routing

Remote Key-value storage

SCTP

S1AP & UEContext Manager

NAS

SECURITY

HSS

AUC

NBloT Handler

MME supporting NBloT
Stateless as much as Possible

- One synchronization point at entry point

MME supporting NBloT

SCTP

S1AP & UEContext Manager

Remote Key-value storage

HSS

AUC

NBloT Handler

Data Format

<table>
<thead>
<tr>
<th>Destination ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source ID</td>
</tr>
<tr>
<td>Stream ID</td>
</tr>
<tr>
<td>Initial Source ID</td>
</tr>
</tbody>
</table>

UE Contexts
(S1 bearer info, nas security vector, etc)

Values (S1AP, NAS, user data)

Round-robin routing

Initial Source ID

Source ID

Stream ID

Destination ID

S1AP &
UEContext
Manager

SECURITY

UE Contexts
(S1 bearer info, nas security vector, etc)

Values (S1AP, NAS, user data)

Initial Source ID

Source ID

Stream ID

Destination ID
Stateless as much as Possible

<table>
<thead>
<tr>
<th>Data Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination ID</td>
</tr>
<tr>
<td>Source ID</td>
</tr>
<tr>
<td>Stream ID</td>
</tr>
<tr>
<td>Initial Source ID</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>S1AP</td>
</tr>
</tbody>
</table>

Round-robin routing

MME supporting NBloT

SCTP

S1AP & UEContext Manager

Remote Key-value storage

HSS

AUC

NBloT Handler

SECURITY

NAS

Initial Source ID

Source ID

Stream ID

Destination ID

Remote Key-value storage
### Stateless as much as Possible

**Data Format**
- Destination ID
- Source ID
- Stream ID
- Initial Source ID

**UE Contexts**
(S1 bearer info, nas security vector, etc)

**NAS**

**Remote Key-value storage**

- **SCTP**
- **S1AP & UEContext Manager**
- **SECURITY**
- **NAS**
- **HSS**
- **AUC**
- **NBioT Handler**

**Round-robin routing**

**MME supporting NBioT**

**Security**

**MME** supporting **NBioT**
Stateless as much as Possible

Data Format
- Destination ID
- Source ID
- Stream ID
- Initial Source ID

- UE Contexts
  - (S1 bearer info, nas security vector, etc)
- NAS

MME supporting NBloT

Round-robin routing

Remote Key-value storage

SCTP

S1AP & UEContext Manager

SECURITY

NAS

AUC

HSS

NBloT Handler
Intelligent Data Partitions

- Key-based routing
  - **UE-eNB-ID as a hash key** which is unique per an user
  - A local cache
  - Avoid complex context managements and frequent access to remote key value storages

**Key-based routing**

**Remote Key-value storage**

**SCTP**

**S1AP & UEContext Manager**

**NAS**

**SECURITY**

**HSS**

**AUC**

**NBloT Handler**

MME supporting NBloT

Round-robin routing
Intelligent Data Partitions

- Initial source aware routing
  - End-to-end transport connection with eNB

Initial Source ID

<table>
<thead>
<tr>
<th>Destination ID</th>
<th>Source ID</th>
<th>Stream ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Source ID</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

eNBs

Remote Key-value storage

HSS

AUC

NBloT Handler

S1AP & UEContext Manager

SECURITY

nas security vector, etc

values (S1AP, NAS, user data)

Initial source aware routing

MME supporting NBloT

Round-robin routing
Intelligent Data Partitions

- Initial source aware routing
  - End-to-end transport connection with eNB
Hints for Degree of Parallelisms

LTE/EPC Initial attachment procedure
Hints for Degree of Parallelisms

LTE/EPC Initial attachment procedure

- Attach request
- Authentication request
- Authentication response
- Security mode command
- Security mode complete
- Attach accept
- Initial context setup
- Attach complete

eNB

MME

```
Samples: 40 of event 'cycles:ppp', Event count (approx.): 14069063

<table>
<thead>
<tr>
<th>Children</th>
<th>Self</th>
<th>Command</th>
<th>Shared Object</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>87.44%</td>
<td>0.00%</td>
<td>mmmme::menas::handle_nas_msg</td>
<td>libmmme_mme.so</td>
<td>mmmme::menas::attach_sm</td>
</tr>
<tr>
<td>87.47%</td>
<td>0.00%</td>
<td>mmmme::menas::handle_nas_msg</td>
<td>libmmme_mme.so</td>
<td>mmmme::menas::attach_sm</td>
</tr>
<tr>
<td>87.47%</td>
<td>0.00%</td>
<td>mmmme::menas::handle_nas_msg</td>
<td>libthread-2.23.so</td>
<td>mmmme::menas::attach_sm</td>
</tr>
<tr>
<td>87.11%</td>
<td>0.00%</td>
<td>mmmme::menas::handle_nas_msg</td>
<td>libmmme_mme.so</td>
<td>mmmme::menas::attach_sm</td>
</tr>
</tbody>
</table>

- 37.42% mmmme::menas::send_security_mode_command
  - libmmme_mme.so
  - mmmme::menas::security_mode_command
    - mmmme::menas::security_mode_command
      - mmmme::menas::security_mode_command

- 19.80% mmmme::menas::send_attach_accept
  - libmmme_mme.so
  - mmmme::menas::attach_accept
    - mmmme::menas::attach_accept
      - mmmme::menas::attach_accept

- 6.58% mmmme::menas::send_authentication_request
  - libmmme_mme.so
  - mmmme::menas::authentication_request
    - mmmme::menas::authentication_request
      - mmmme::menas::authentication_request

- 24.11% mmmme::menas::parse_security_mode_complete
  - libmmme_mme.so
  - mmmme::menas::security_mode_complete
    - mmmme::menas::security_mode_complete
```
Intelligent Data Partitions

- Assign "SECURITY" blocks more than other blocks
Design Challenges

- How can we design control plane application?
- **What are the programming APIs?**
- How can we manage user states?
Design Programming APIs

- What are basic functions for control planes?
  - For request: Decode incoming packets & update UE contexts
  - For response: Encode outgoing packets with UE contexts
Design Programming APIs

- What are basic functions for control planes?
  - For request: Decode incoming packets & update UE contexts
  - For response: Encode outgoing packets with UE contexts

- Design an interface for control planes?

```java
public interface ControlMessageHandler {
    public StreamValues ProcessRequestMessage (byte[] PDU, byte[] ueContexts)
    public StreamValues ProcessResponseMessage (byte[] PDU, byte[] ueContexts)
}

class StreamValues{
    int result;
    byte[] PDU;
    byte[] ueContext;
}
```
Programmability: Basic Building Blocks

- Implement **ControlMessageHandler** Interface

```java
/* Creating instances */
MMESCTP sctp = new MMESCTP();
MMES1AP s1ap = new MMES1AP();
MMENAS nas = new MMENAS();
MMENAS Security = new MMESecurity();
MMEAUC auc = new MMEAUC();
MMEUEManager ueManager = new MMEUEManager();
MMES1APDecorder s1apDecorder = new MMES1APDecorder();
MMENASDecorder nasDecorder = new MMENASDecorder();
MMENBIoT nbIoT = new MMENBIoT();

/* Process PDU and update UE contexts */
StreamValue streamValue = s1ap.ProcessRequestMessage(pdu, ueContexts);
StreamValue streamValue = nas.ProcessRequestMessage(pdu, ueContexts);
```
Programmability: Assemble Them

- Use stream framework APIs

```java
/* Building Topology */
TopologyBuilder builder = new TopologyBuilder();

/* API to add components into topology*/
builder.setBolt("component name", "component class", "parallelism")
  .routingType("destination component name", "stream type")

/* Example SCTP -> S1AP -> NAS */
builder.setBolt("SCTP", new SCTPServerBolt(host, port), 1).shuffleGrouping("S1AP","S1AP STREAM");
builder.setBolt("S1AP", new S1APBolt(), 1).fieldsGrouping("NAS","NAS STREAM", new Fields(eNBID))
builder.setBolt(NAS, new NASBolt(), 1).shuffleGrouping("S1AP", "NAS STREAM");
```
Programmability: Assemble Them

- Use stream framework APIs

/* Building Topology */

MME supporting NBloT

Initial source aware routing

Key-based routing

Round-robin routing

builder.setBolt("SCTP", new SCTPServerBolt(host, port), 1).shuffleGrouping("S1AP","S1AP STREAM");
builder.setBolt("S1AP", new S1APBolt(), 1).fieldsGrouping("NAS", "NAS STREAM", new Fields(eNBID))
builder.setBolt(NAS, new NASBolt(), 1).shuffleGrouping("S1AP", "NAS STREAM");
Programmability : Assemble Them

- Use stream framework APIs

/* Building Topology */
TopologyBuilder builder = new TopologyBuilder();

/* API to add components into topology*/

Give flexibilities for your pipeline design
and degree of parallelisms

// Example SCTP -> S1AP -> NAS /
builder.setBolt("SCTP", new SCTPServerBolt(host, port), 1).shuffleGrouping("S1AP","S1AP STREAM");
builder.setBolt("S1AP", new S1APBolt(), 1).fieldsGrouping("NAS", “NAS STREAM”, new Fields(eNBID))
builder.setBolt(NAS, new NASBolt(), 1).shuffleGrouping("S1AP”, “NAS STREAM”);
Programmability: Building Your Own Blocks

- Add your own blocks for new services by implementing `ControlMessageHandler` interfaces
- Still reuse other components

```java
Public class NBloTNASHandler extends ControlMessageHandler {
    Public StreamValues ProcessRequestMessage (byte[] PDU, byte[] ueContexts){
        /* your logic to handle NBloT */
    }
    Public StreamValues ProcessResponseMessage (byte[] PDU, byte[] ueContexts){
        /* your logic to handle NBloT */
    }
}
```
Programmability: Building Your Own Blocks

- Add your own blocks for new services by implementing `ControlMessageHandler` interfaces
- Still reuse other components

```java
Public class NBloTNASHandler extends ControlMessageHandler {
    Public StreamValues ProcessRequestMessage (byte[] PDU, byte[] ueContexts) {
        /* your logic to handle NBloT */
    }

    Public StreamValues ProcessResponseMessage (byte[] PDU, byte[] ueContexts) {
        /* your logic to handle NBloT */
    }
}
```

Give flexibilities to add your own functions for new services
Design Challenges

- How can we design control plane application?
- What are the programming APIs?
- How can we manage user states?
Managing UE Contexts

- Remote key-value storages
- A transaction
  - One events (e.g., attachment procedures)
  - Updates remote storages after one transaction
- Two key-value storages
  - Active user
  - Idle user
MobileStream Prototype

- **Control plane library**
  - Written by C++
  - Support most of events (e.g., attach, detach, Service req, S1 release, TAU, dedicated bearers)
  - Minimal S/PGW control plane functions (e.g., Tunnel assignment, PCO informations)

- **Storm extension:**
  - Extended SCTP transport library
  - Support Initial source aware routing

- **Storm-based control plane**
  - Used JNI to bridge C++ and Java layer
  - Redis as remote key-value storages

- **UE contexts**
  - Using google protocol buffers
Evaluations

- Validate control plane standards compliance
- Evaluate CIoT vs MEC
Evaluations

- Validate control plane standards compliance
- Evaluate CIoT vs MEC
Validating MobileStream Standard Compliance

- Storm-based control planes in PhantomNet testbed

<table>
<thead>
<tr>
<th>UE</th>
<th>eNB</th>
</tr>
</thead>
<tbody>
<tr>
<td>srsUE &amp; Nexus 5</td>
<td>srs eNB, OAI eNB, Commercial ip.access small cell</td>
</tr>
<tr>
<td>OAI &amp; OpenEPC emulator</td>
<td>OAI &amp; OpenEPC eNB Emulator</td>
</tr>
</tbody>
</table>

http://phantomnet.org/
Future work

- Large scale evaluation
- Evaluate robust functions
- Apply policies or priorities based on service or requests
- Explore possibility to use this design for data planes (i.e., S/PGWs)
- Dynamic reconfigurations
MobileStream: A new scalable, programmable, evolvable and reliable mobile control plane platform

Key-value storages

Get/set UE contexts

Mobile Control Planes

S/P-GWs

HSS

eNBs

Stream Framework on cloud