4G / 5G Open Source Cores

Current & Future States

June 27, 2019

OAI NA Workshop
Current Open Source Efforts

- EPC based
  - Makes sense given Release 15 just finished its 1.3 ASN.1 release
  - Scope of Release 15 is a start but not parity with 4G
  - Device & Production hardware availability

- Current efforts (major ones)
  - openair
  - magma
  - omec
Deployment Options

Intentional Squint Chart

Dashed Line – Control
Solid Line - User
Migration

Just picking on Option 3x ....

Option 1–Legacy Design Today

Option 3x–Non-standalone & LTE Assisted, EPC Connected

Option 3x / Option 2

Option 3x + Option 2

Option 2–Standalone NR & NGCN

LTE only devices
EN-DC devices

LTE only devices
EN-DC devices

Latest Trend – NGC NF + EPC NF
All previous devices + NR with LTE fallback

When?

Colors
Red – NR only
Black – LTE, LTE+NR

Line Type
Dashed – Control
Solid - User
The ‘+’ VoNR with VoLTE fallback

Only relevant components shown

Colors
- Red – NR only
- Black – LTE, LTE+NR

Line Type
- Dashed – Control
- Solid - User
So if you are developing a core....

- **Release 15+ EPC**
- **EPC to NGC Transition**
- **Standalone NGC**

**Need to support Tomorrow’s Devices Today**

**EPC + NGC**

**Need to support Today’s Devices**

(Volume of addressable market)
Constructing a Core for 4G + 5G

Deployment is being tackled by many, focus on functionality!

- Specification
- Test
- Develop
- Deploy
- Support

CI / CD

Continuous Specification Integration & Delivery

Cloud NFV SDN ZTM + ALTO

©2019 Sprint. This information is subject to Sprint policies regarding use and is the property of Sprint and/or its relevant affiliates and may contain restricted, confidential or privileged materials intended for the sole use of the intended recipient. Any review, use, distribution or disclosure is prohibited without authorization.
Continuous Specification Integration & Delivery (CSI/D)

Integrate as much as we can as early as possible (from specification stage)
➢ Not a new idea

Tenets:
➢ QA (code, performance, security, compliance) is required at every step!
➢ Look for operational (design) patterns
➢ Look for development patterns
➢ The framework will change over time (including languages and supporting technologies, e.g. VMs, cloud etc.)
➢ Make it efficient, reliable, scalable, available and repeatable!
➢ Automate where you can; develop if you must!

*The ONF Open Mobile Evolved Core (OMEC) is where we realize many of these ideas.*
OMEC’s Place in the ONF

- ONF (foundation)
- CORD (platform)
- COMAC (platform)
- OMEC (project)
  - NGIC-RTC (open source code)
  - C3PO (open source code)
OMEC – Present

Visible Projects

Single Frame (1 instance of each component)
40K Users
1K Control Plane TPS
42-80 CPU Cores

Note HSS supports IoT interfaces including S6t
## Features

<table>
<thead>
<tr>
<th>Now</th>
<th>Under Test (NGIC)</th>
<th>Next (NGIC)</th>
</tr>
</thead>
</table>
| • Default Bearers  
• Offline Billing secured by SGX  
• Child Protections (gating by domain or 5 tuple)  
• Basic MME support (initial attach, detach, etc.) | • GTP-C structure (auto generated)  
• PFCP structure (auto generated)  
• SGW, PGW, SAEGW roles  
• UPF lookup via DNS with Dedicated Core & Network Capability Support  
• CLI client to REST interface  
• REST interface for CLI and remote operation | • All TS 23.214-f40 Stage 2 Procedures that use S11 GTP with GTP based mobility  
• All SGW, PGW, SAEGW restoration procedures (TS 23.007)  
• Operational Measurements (via logs)  
• Alarms (via logs) |
Roles

MME

S11

SGW-C

S5/S8-C

PGW-C

eNB

S1-U

SGW-U

S5/S8-U

PGW-U

Inbound Roamer (acts as SGW)

SAEGW-C

S8-C

SAEGW-U

Outbound Roamer (acts as PGW)

S8-U

SAEGW-C

S8-U

SAEGW-U

PDN(s)

PDN(s)
Interesting Functions

DNS query tracking & refresh (API access under development)

- DNS client remembers the queries associated with answers
- Can automatically refresh based upon timers or commands
- Grow / contract following TS 29.303 (DDDS) and remain in sync with Orchestration decisions

Global Cluster

Local Cluster

Record Checkouts

Local Cluster Misses

2B+ Entries (design phase)

- Local Cluster (2B keys per space)
  - Holds individual records
  - Lists
- Global Cluster (2B key ranges per space)
  - checkout based
  - Each entry is a range of records

1. > Q: foo.com
2. < A: s1.foo.com
3. > Update remove s1 & add s2
4. > Refresh
5. > Q: foo.com
6. < A: s2.foo.com
OMEC CI / CD

OMEC CI/CD is on github (https://github.com/omec-project)

We want OMEC to be as much of a worry free codebase as soon as possible

Submissions will pass basic checks on github then
- Code is brought over to ONF lab, built and deployed
- Tested against commercial, 3rd party tools for functional tests
- Tested against commercial, 3rd party and open source tools for load tests
- Periodic scans by commercial tool for
  - License compliance
  - Code match to other projects (for various violations)
  - Static security scanning

3rd Parties have also performed AI based testing and other methods
Architecture Pattern - Control Plane Functions

Message Library

- Protocol Interface
  - Packet
  - Protocol Decode
- Decoded Information
  - Event ID
  - Common (Cross Protocol) State Machine
- OM Publication
  - Read
  - Update (Function, Proposed State)
- Event Dispatcher
  - Contexts include Metrics
  - System Context is catch all for date that does not fit in other contexts
  - UE Context lookup (as required)
  - Event + State maps to function
  - Check Preconditions (Guard for transition)
- Event Handler
  - Decoded Information
  - Next Action
  - Create or Update
- Data To Encode
- Timer Queue
- Timer Manager

NOTES
1 – Decode does NOT occur for FreeDiameter or fired Timers
2 - May not always be present, e.g. retransmission

LEGEND
- Activity
- Information
- Text in BOLD – Code Stubs that must be customized

Command
Config
CLI

S11, S5/S8, Sx,
FreeDiameter,
Timers (fired), CLI,
SBA (future)

Protocol
Interface

Packet

Protocol Decode

Decoded Information

Event ID

Common (Cross Protocol)
State Machine

OM
Publication

Event Dispatcher

Decoded Information

Next Action

Event Handler

Data To
Encode

Timer Queue

Timer Manager

Protocol
Encoder

Message Library

Packet

Protocol Interface

Server Context

Interface Context

Peer Context

Session Context

System Context

UE, Peer or Interface Procedures
Can follow Stage 2 procedure and/or protocol procedure

Procedures

Common (Cross Protocol)

Packet

Message Library

OM Publication

Check Preconditions

Guard for transition

Decoded Information

Next Action

Create or Update

Data To Encode

Timer Queue

Timer Manager

Protocol Encoder

Packet

 Protocol Interface

Server Context

Interface Context

Peer Context

Session Context

System Context

UE, Peer or Interface Procedures
Can follow Stage 2 procedure and/or protocol procedure

Procedures

Common (Cross Protocol)

Packet

Message Library

OM Publication

Check Preconditions

Guard for transition

Decoded Information

Next Action

Create or Update

Data To Encode

Timer Queue

Timer Manager

Protocol Encoder

Packet

Protocol Interface

Server Context

Interface Context

Peer Context

Session Context

System Context

UE, Peer or Interface Procedures
Can follow Stage 2 procedure and/or protocol procedure

Procedures

Common (Cross Protocol)

Packet

Message Library

OM Publication

Check Preconditions

Guard for transition

Decoded Information

Next Action

Create or Update

Data To Encode

Timer Queue

Timer Manager

Protocol Encoder

Packet

Protocol Interface

Server Context

Interface Context

Peer Context

Session Context

System Context

UE, Peer or Interface Procedures
Can follow Stage 2 procedure and/or protocol procedure

Procedures

Common (Cross Protocol)

Packet

Message Library

OM Publication

Check Preconditions

Guard for transition

Decoded Information

Next Action

Create or Update

Data To Encode

Timer Queue

Timer Manager

Protocol Encoder

Packet

Protocol Interface

Server Context

Interface Context

Peer Context

Session Context

System Context

UE, Peer or Interface Procedures
Can follow Stage 2 procedure and/or protocol procedure

Procedures

Common (Cross Protocol)

Packet

Message Library

OM Publication

Check Preconditions

Guard for transition

Decoded Information

Next Action

Create or Update

Data To Encode

Timer Queue

Timer Manager

Protocol Encoder

Packet

Protocol Interface

Server Context

Interface Context

Peer Context

Session Context

System Context

UE, Peer or Interface Procedures
Can follow Stage 2 procedure and/or protocol procedure

Procedures

Common (Cross Protocol)

Packet

Message Library

OM Publication

Check Preconditions

Guard for transition

Decoded Information

Next Action

Create or Update

Data To Encode

Timer Queue

Timer Manager

Protocol Encoder

Packet

Protocol Interface

Server Context

Interface Context

Peer Context

Session Context

System Context

UE, Peer or Interface Procedures
Can follow Stage 2 procedure and/or protocol procedure

Procedures

Common (Cross Protocol)

Packet

Message Library

OM Publication

Check Preconditions

Guard for transition

Decoded Information

Next Action

Create or Update

Data To Encode

Timer Queue

Timer Manager

Protocol Encoder

Packet

Protocol Interface

Server Context

Interface Context

Peer Context

Session Context

System Context

UE, Peer or Interface Procedures
Can follow Stage 2 procedure and/or protocol procedure

Procedures

Common (Cross Protocol)

Packet

Message Library

OM Publication

Check Preconditions

Guard for transition

Decoded Information

Next Action

Create or Update

Data To Encode

Timer Queue

Timer Manager

Protocol Encoder

Packet

Protocol Interface

Server Context

Interface Context

Peer Context

Session Context

System Context

UE, Peer or Interface Procedures
Can follow Stage 2 procedure and/or protocol procedure

Procedures

Common (Cross Protocol)

Packet

Message Library

OM Publication

Check Preconditions

Guard for transition

Decoded Information

Next Action

Create or Update

Data To Encode

Timer Queue

Timer Manager

Protocol Encoder

Packet

Protocol Interface

Server Context

Interface Context

Peer Context

Session Context

System Context

UE, Peer or Interface Procedures
Can follow Stage 2 procedure and/or protocol procedure

Procedures

Common (Cross Protocol)

Packet

Message Library

OM Publication

Check Preconditions

Guard for transition

Decoded Information

Next Action

Create or Update

Data To Encode

Timer Queue

Timer Manager

Protocol Encoder

Packet

Protocol Interface

Server Context

Interface Context

Peer Context

Session Context

System Context

UE, Peer or Interface Procedures
Can follow Stage 2 procedure and/or protocol procedure

Procedures

Common (Cross Protocol)
Note: Decode is skipped in CP due to the fact that FreeDiameter has already decoded

**Considerations:** Would Multiple Diameter Applications ever result in multiple UNIX_FD socket in each direction?

1 – Message is not decoded but the Event is assigned

Cross Message (Request / Answer) checking can take place
## Stacks Pattern

### Services

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability of Message Delivery</td>
<td></td>
</tr>
<tr>
<td>Endpoint Path Failure Detection</td>
<td></td>
</tr>
<tr>
<td>Piggyback messages</td>
<td></td>
</tr>
<tr>
<td>Protocol Errors</td>
<td></td>
</tr>
<tr>
<td>Unsupported Versions</td>
<td></td>
</tr>
<tr>
<td>Message Invalid Length</td>
<td></td>
</tr>
<tr>
<td>Unknown Message</td>
<td></td>
</tr>
<tr>
<td><strong>Unexpected Message</strong></td>
<td><em>Invalid Length Information Element</em> can be executed in the Communications Layer but will not be able to pinpoint the offending IE. <em>Common Structure Error Handling</em> is protocol specific.</td>
</tr>
<tr>
<td>Missing Information Element</td>
<td></td>
</tr>
<tr>
<td>Semantically Incorrect Information Element</td>
<td></td>
</tr>
<tr>
<td>Unknown or Unexpected Information Element</td>
<td></td>
</tr>
<tr>
<td>Repeated Information Elements</td>
<td></td>
</tr>
<tr>
<td><strong>Common Structure Error Handling</strong></td>
<td></td>
</tr>
<tr>
<td>Detection of Peer Reset</td>
<td></td>
</tr>
</tbody>
</table>

Items in **bold** can only partially be verified at the Communications Layer.
Automation of Components

Number of customization locations ~ 5
- State Machine
- Control Plane Handler
- Contexts
- Message Handlers (within stacks)
  - Received Messages
  - Sent Messages

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Code Gen</th>
<th>Test Gen</th>
<th>Spec QA</th>
</tr>
</thead>
<tbody>
<tr>
<td>eGTP</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>Diameter</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>S1AP</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>PFCP</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>OpenAPI</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

Specifications (IETF, 3GPP) → Model Abstraction → Code → Test Plans → Reports
The Mute-Notification AVP (AVP code 2809) is of type Enumerated.

The following values are defined:
- MUTE_REQUIRED(0)
- ...

21 ways this is expressed in specifications

6 ways this is expressed in specifications

 dia file entry
Mute-Notification 2089 VM ...
@enum Mute-Notification MUTE_REQUIRED 0

AVP Flags
Example – Message to Code (1/2)

From S6a (TS 29.272)

```
< Cancel-Location-Request > ::= < Diameter Header: 317, REQ, PXY, 16777251 >
  < Session-Id >
  [ DRMP ]
  [ Vendor-Specific-Application-Id ]
  { Auth-Session-State }
  { Origin-Host }
  { Origin-Realm }
  { Destination-Host }
  { Destination-Realm }
  { User-Name }
  *[Supported-Features ]
  { Cancellation-Type }
  [ CLR-Flags ]
  *[ AVP ]
  *[ Proxy-Info ]
  *[ Route-Record ]
```

Cancel-Location-Request is mapped to CALR
HSS does not receive CALR (it sends it)

CODE - https://raw.githubusercontent.com/omec-project/c3po/master/hss/src/s6as6d_impl.cpp

```
// Member functions that customize the individual application
Application::Application( DataAccess &dbobj )
  : ApplicationBase()
  , m_cmd_uplr( *this )
  //, m_cmd_calr( *this )
  ...
  {
    registerHandlers();
  }
  ...
void Application::registerHandlers()
  {
    // Remove the comments for any Command this application should
    // listen for (receive).
    ...
    //registerHandler( m_cmd_calr );
    ...
  }
```

©2019 Sprint. This information is subject to Sprint policies regarding use and is the property of Sprint and/or its relevant affiliates and may contain restricted, confidential or privileged materials intended for the sole use of the intended recipient. Any review, use, distribution or disclosure is prohibited without authorization.
Example – Message to Code (2/2)

// CALR Request (req) Command member functions
// Sends a CALR Request to the corresponding Peer
bool Application::sendCALRreq(FDPeer &peer)
{
    // TODO - This code may be modified based on specific
    // processing needs to send the CALR Command
    CALRreq *s = createCALRreq(peer);

    try
    {
        if ( s )
        {
            s->send();
        }
    }
    catch (FDException &ex)
    {
        Logger::s6as6d().error("EXCEPTION - %s", ex.what());
        delete s;
        s = NULL;
    }

    // A factory for CALR requests
    CALRreq *Application::createCALRreq(FDPeer &peer)
    {
        // creates the CALRreq object
        CALRreq *s = new CALRreq(*this);

        // TODO - Code must be added to correctly
        // populate the CALR request object
        return s;
    }

    // A handler for Answers corresponding to this specific
    Request
    void CALRreq::processAnswer(FDMessageAnswer &ans)
    {
        // TODO - This code must be implemented if the application
        // receives Answers for this command, i.e. it sends the
        // CALR Command
        // CALR Command (cmd) member function
        // Function invoked when a CALR Command is received
        int CALRcmd::process(FDMessageRequest *req)
        {
            return -1;
        }
    }

    // TODO - This code must be implemented if the application
    // receives the CALR command.
    return -1;
}

// DO NOT free the newly created CALRreq object!!
// It will be deleted by the framework after the
// answer is received and processed.
return s != NULL;

// TODO - This code must be
// receives Answers for this
// CALR Command

// A handler for Answers corresponding to this specific
// Request
void CALRreq::processAnswer(FDMessageAnswer &ans)
{
    // CODE -
    // https://raw.githubusercontent.com/omec-project/c3po/master/hss/src/s6as6d_impl.cpp
Progression of Automation

Support
- Diameter
- GTP-C/PFCP
- 4G/5G Languages
  - NAS
  - OpenAPI

Ingest Speed
- 90 minutes
- 30 minutes
- 6 seconds!

Capabilities
- Diameter
  - Code Gen
  - AI Test Gen
- GTP-C/PFCP
  - Error Logs
  - Generation (Code)
  - Limited Test Gen
  - Reports
- Common Support
  - Ingest
  - Generation (YANG, Code, Test)
  - Reports (Error, Analysis)

Gen 1 - 2017
Gen 2 – 2018/19
Gen 3 – 2019/20
Automation and Quality

Generation 1
• Gen 1 requires **absolute** perfection
• 3700 Errors
• ~900 categories of errors
Generation 2 (2 specs – PFCP and eGTP)
• ~1 dozen CRs & Counting
• 3 workarounds that defy computer science
• Many minor editorial corrections
• System can tolerate some errors
Generation 3
• Combines language support
• Tolerates errors (as much as possible)
• Reports errors
• Centralized Service & Plug-ins
Research

We have begun using techniques such as term substitution, AI trained natural language tools and other techniques with interesting results.

1) Inconsistencies discovered by AI. Three CRs have been created (and accepted) for Stage 2 (sequence diagrams). Including sequence diagrams that reference sequence diagrams.
   - Two were issues present for 10+ years in base LTE spec
   - One was in 23.214 based on referencing.

2) Automated Test Generation with the goal for complete coverage

3) OSS / BSS support
What’s Next?

Construction
• TODOs to backlog – answer the question ‘where can I help?’
• Gen 3
• Capturing all known references in generated stubs
  • You know *every* location a protocol message is referenced in the comments
• Differential Comment Generation
  • Ability to add to already modified files
• OMEC Design Patterns for
  • User Planes
  • Billing

Field Test Ready Functionality

- Interop (EPC ↔ NGC)
- HSS, PCRF, IoT?, NGC components
- ePDG
- SAEGW, PGW, SGW
- NGC NF + EPC NF