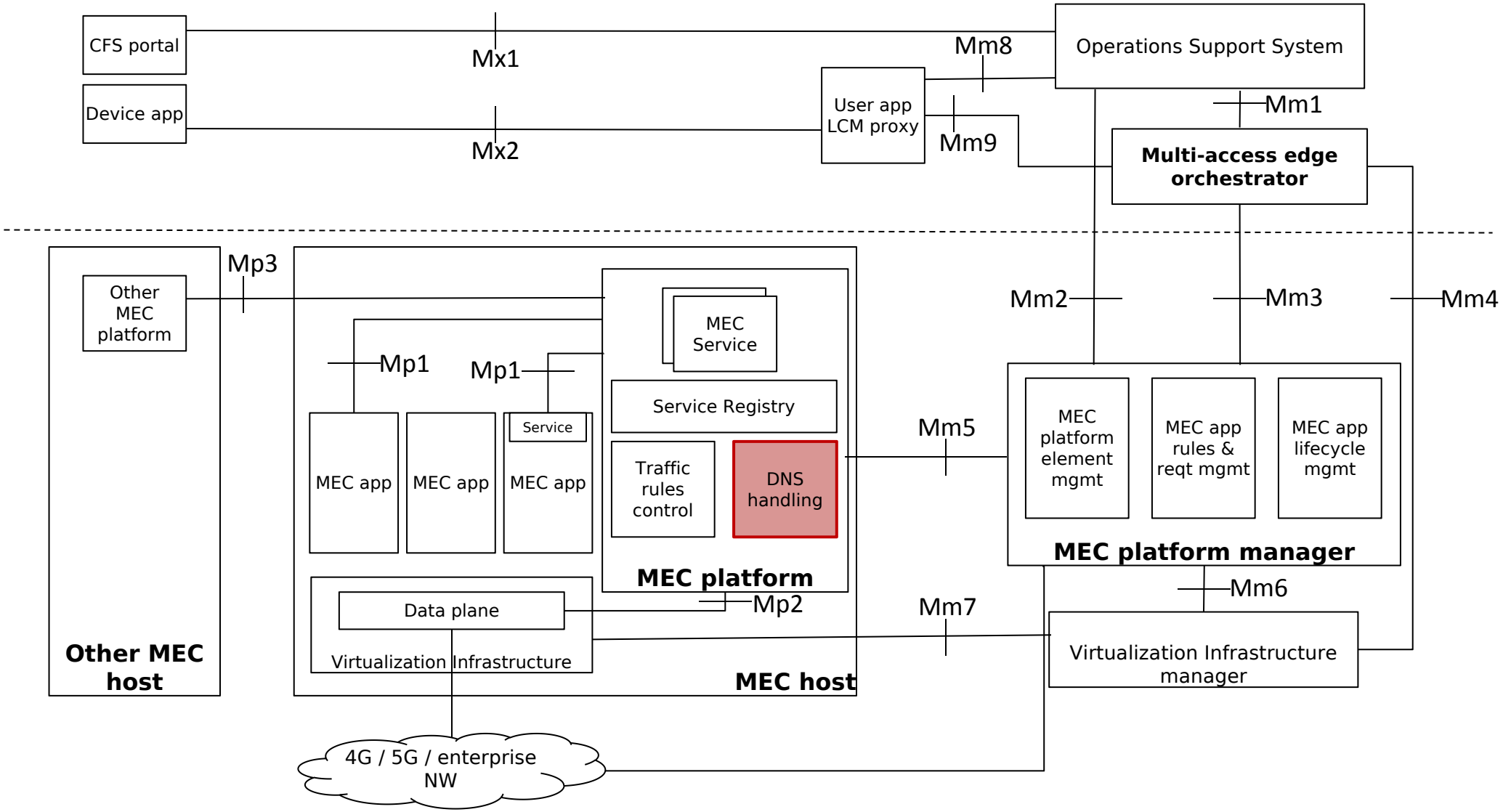
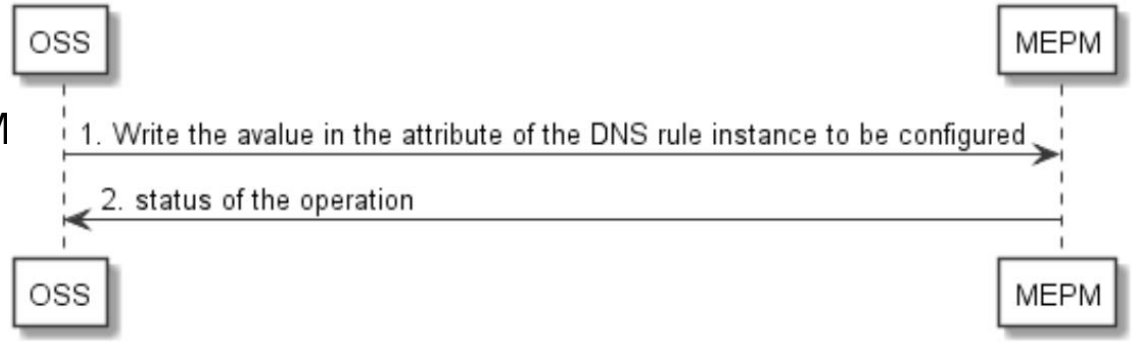
MEC DNS solution

1. Architecture

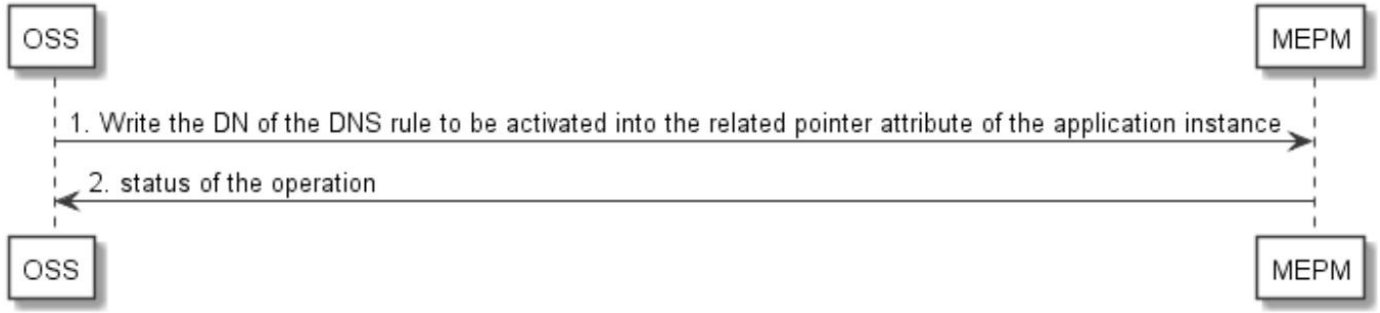


2.1 DNS Requirements from MEC 010-1 v1.1.1

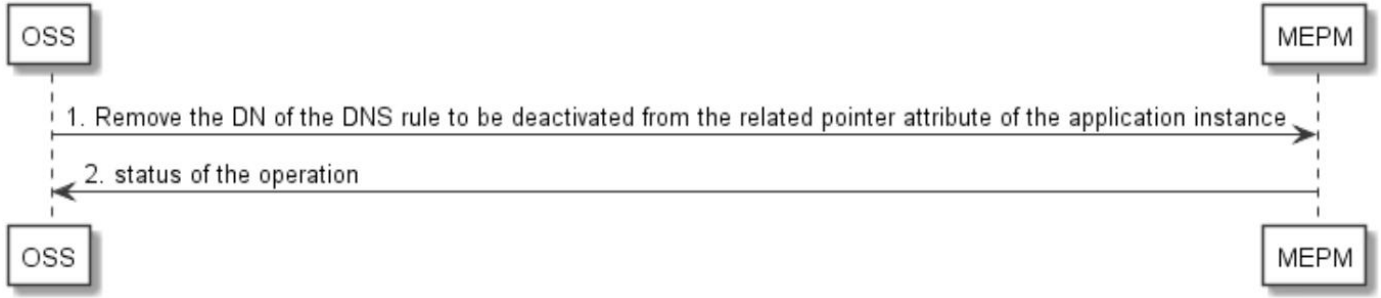
- 6.1.2 Configuration of DNS rules
  - The DNS rules can be configured by the NM



- 6.1.3 Activation of DNS rules
  - The DNS rules serving a particular application instance can be activated by the NM.



- 6.1.4 Deactivation of DNS rules
  - The DNS rules serving a particular application instance can be deactivated by the NM.

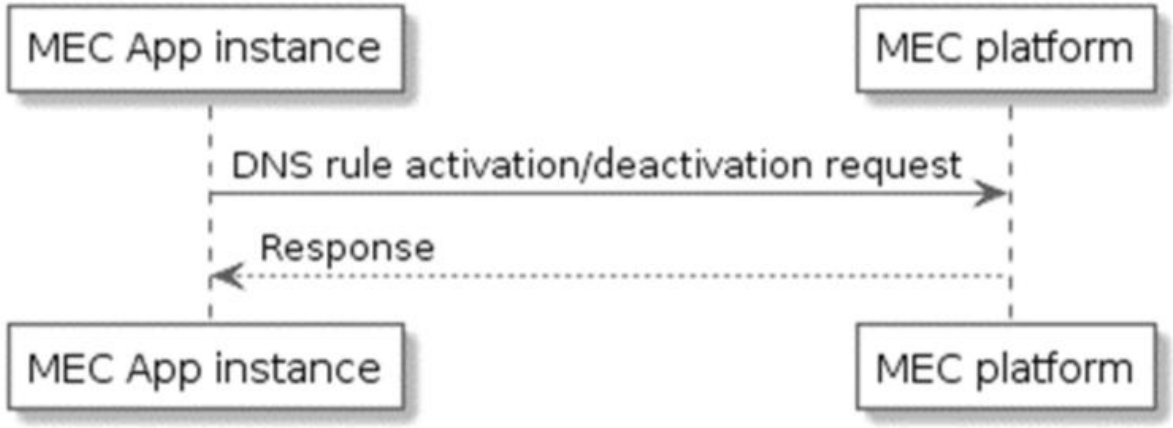


2.2 DNS Requirements from MEC 010-2 v2.1.1

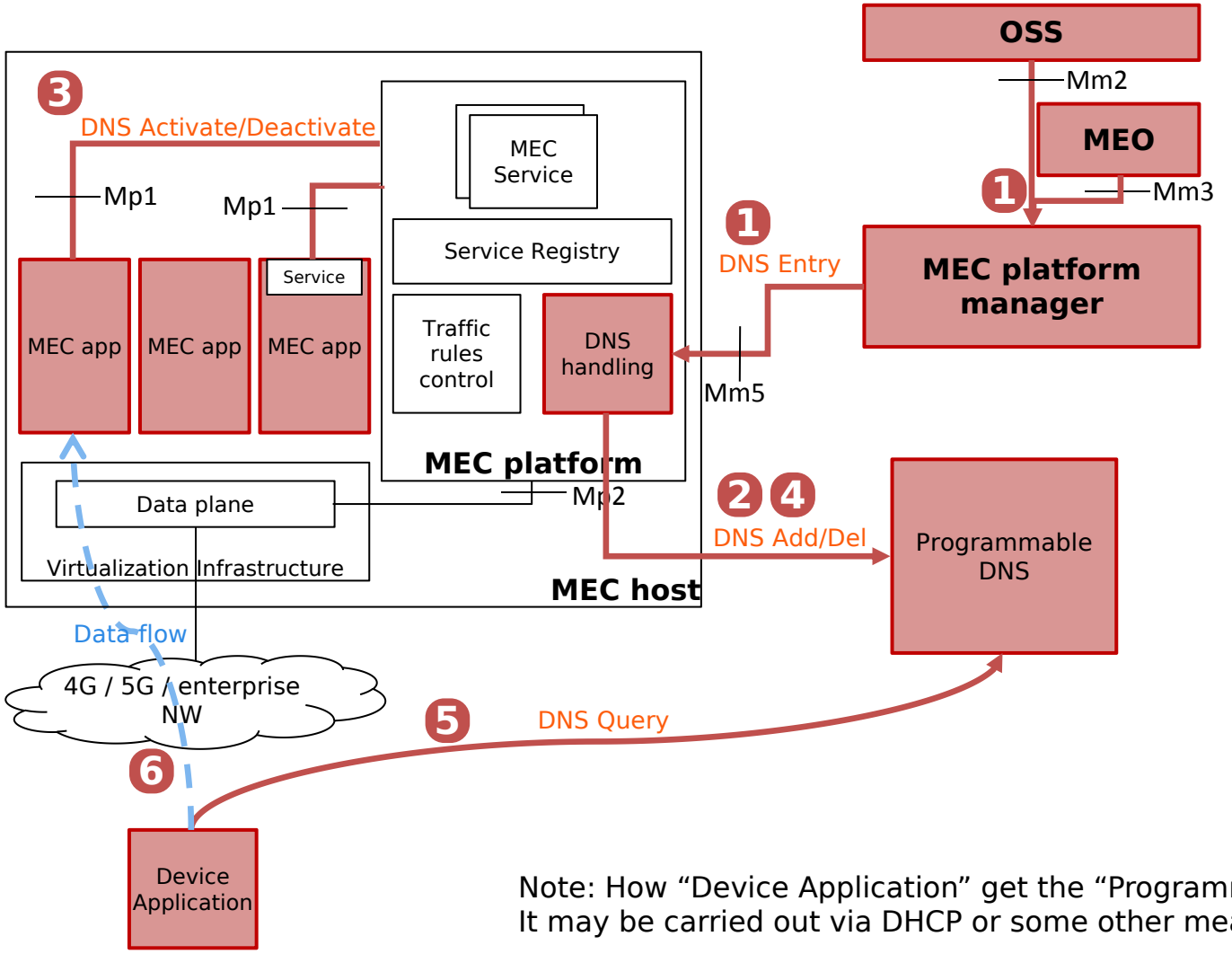
- 4.3.2 Application descriptor requirements
  - AppDesc.009: The application descriptor shall support a description of DNS Rules which provide specific FQDNs to be registered into the MEC system (e.g. for redirection of traffic to local host).

2.3 DNS Requirements from MEC 011 v2.1.1

- DNS rule activation/deactivation
  - MEC application instance sends DNS activation/deactivation request to the MEC platform. Platform install or remove the DNS rule(s) from the DNS server/proxy.
  - MEC platform sends response to the MEC application instance, contains the result.



3. How DNS fit in the MEC Architecture?



1. MEPM get the DNS mapping from OSS/MEO and add an entry in the MEP server.
2. If entry status is active, send a request to the “Programmable DNS”. If inactive, then keep the request in db and do not send the request.
3. MEC app can send activate or deactivate request to MEP server.
4. As per the request from MEC app, create or delete the entry from “Programmable DNS”.
5. Device application performs DNS Query to get the MEC app IP.
6. Once domain name resolved, device app directly communicate with MEC app.

Note: How “Device Application” get the “Programmable DNS Server” IP is out out scope of this document. It may be carried out via DHCP or some other means in the dataplane.

4. Problems to discuss

- Realization of the programmable DNS server
- Configuration flow and Requirements

5 Programmable DNS

Requirements

- Req 1: DNS server IP should be reachable from device applications.
- Req 2: Must have a programmable interface(REST/gRPC etc) to add or remove entries. MEP-server use this programmable interface to configure the DNS.
- Req 3: DNS server should use port number 53 always. This is required because of two reasons,
  - Client OS may not have capabilities to select the port number of DNS server.
  - DHCP protocol doesn't have option to carry DNS port number along with IP.

5.1 Can we use CoreDNS in K8S?

- CoreDNS deployed in kubernetes gets an internal IP to k8s which will not be accessable in outside.
- The other option is to expose the coredns service using NodePort. But this has multiple problems.
  - Exposing the k8s dns server will cause multiple security problems and will lead to service un-availability in the cluster.
  - Always need to access the DNS server using the IP + exposed node-port number, but protocols like dhcp doesn't support carrying port-number besides the dns ip.
- Tight dependency with k8s

- **Conclusion**: It is not feasible to use same CoreDNS with k8s and external.

5.2 Seperate instance of DNS

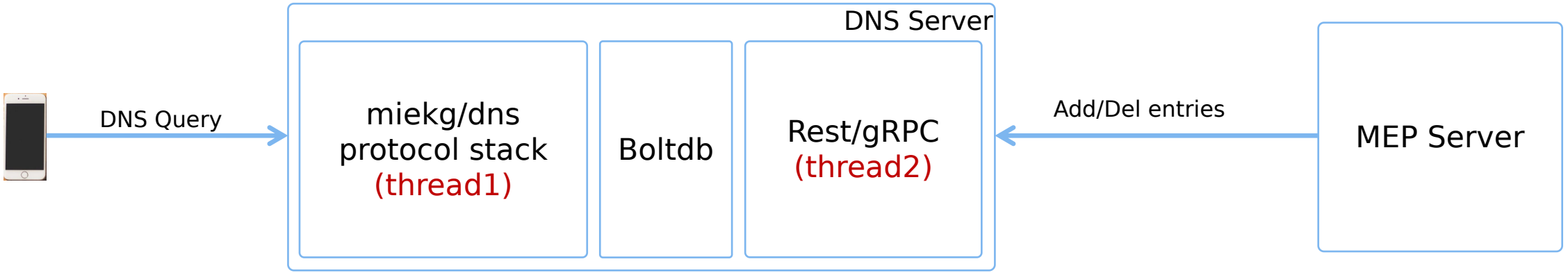
- Req 1: Reachability
  - Deployment on k8s cluster:
    - Better scalability and reliability.
    - Need a LB(internal) + FW(external) to expose the DNS service.
  - Deployed outside(external) k8s cluster:
    - Can be deployed on an external node or k8s node itself.
    - No scalability as deployed outside k8s cluster.
    - No need for a loadbalancer, but a FW is required.

- Req 2: Programmability
  - Deailed analysis on next page

- Req 3: Listening on port 53
  - K8s cluster deployement can leverage the loadbalancer for port mapping.
  - External deployement can use the port directly.

5.2.1 Solution: New DNS server

- CoreDNS is built on top of “miekg/dns” package.
- “miekg/dns” package has the complete implementation of DNS protocols. CoreDNS just build a plugin framework on top of this package.
- We can use this “miekg/dns” package and build a DNS solution which is programmable.



- Bolt is a stable, embedded key-value store maintained by etcd, used in many production environments. We can use boltdb as the backend of DNS to store the entries.
- Bolt uses files to store data for persistence, we can provide persistence by mounting local directory or pv in k8s.
- miekg/dns handles dns-query and check entries in boltdb.
- rest/grpc module handles programmable interfaces from mep server and add entries into the boltdb.

Note:- This solution is already implemented as part of plug-test and available to use.

5.2.2 Solution Comparison

- Pros
  - Trimmed down version reduces the overhead like plugin frameworks in CoreDNS.
  - In-memory embedded database provides faster query results.
  - Can avoid external etcd/db cluster for backend.
- Cons
  - Additional synchronizing mechanism required for achieving scalability.

5.2.3 miekg/dns Resource Consumption

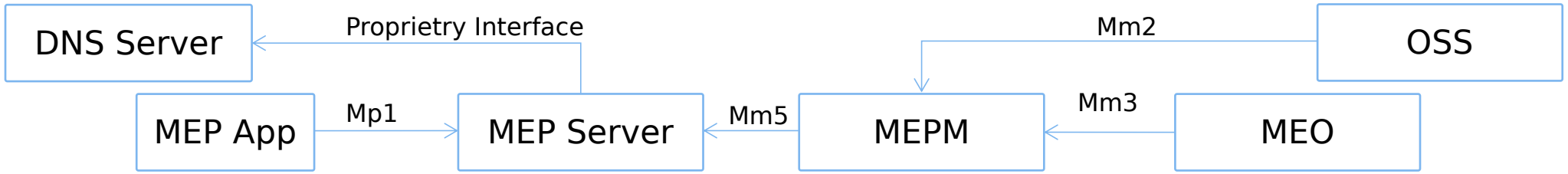
- TestCase: Idle scenario
  - Queries : 0
  - CPU: ~1m(0 cpu)
  - Memory: ~8MB

- TestCase: 100000 queries using single client
  - Queries: 8740 requests/second
  - CPU: ~1000m(1cpu)
  - Memory: ~30MB

5.3 Deployment Scenarios

- Deployment on k8s cluster:
  - Pros:
    - Better scalability and reliability, k8s orchestrates the deployments.
    - Better security as management interfaces are unexposed by k8s.
  - Cons:
    - Need an additional loadbalancer to expose the DNS service.
- Deployed outside(external) k8s cluster:
  - Pros:
    - No need for additional loadbalancer.
  - Cons:
    - No auto scalability.
    - Security is a real concern here.
      - Need FW to limit connecting to other ports.
      - Mep-server to DNS Server management interface authentication and encryption.
      - Configuring the DNS server certificates and keys.

6. Requirements



- Requirements:
  - MECM
    1. Implement Mm2, and Mm3 interfaces for DNS support.
    2. Use Mm5 interface to configure the DNS rules in MEP.
    3. MECM portal should have the option to configure the exposed application IP(Ingress/LB IP).
    4. Configure the DNS via MEP server, after every application deployment and termination over Mm5 interface.
  - MEP
    1. Add Mp1 interface support for MECAPP to activate, deactivate and query the DNS rules.
    2. Add Mm5 interface support to create, modify and delete the DNS rules.
    3. Implement a programmable DNS server and deploy it in the MEC host. Add support in MEP to configure the DNS server.
  - MEP Agent
    1. New interface to call the DNS activation/deactivation

Note: Except Mm5 all other interfaces are already defined by ESTI, we can follow same.

MEP Server to DNS server interface is proprietry.

7.1 Interface Design- Mp1

GET /applications/{appInstanceId}/dns_rules

```
[
  {
    "dnsRuleId": "dnsRule1",
    "domainName": "www.example.com",
    "ipAddressType": "IP_V6",
    "ipAddress": "192.0.2.0",
    "ttl": "?",
    "state": "ACTIVE"
  }
]
```

GET /applications/{appInstanceId}/dns_rules/{dnsRuleId}

```
{
  "dnsRuleId": "dnsRule1",
  "domainName": "www.example.com",
  "ipAddressType": "IP_V6",
  "ipAddress": "192.0.2.0",
  "ttl": "?",
  "state": "ACTIVE"
}
```

PUT /applications/{appInstanceId}/dns_rules/{dnsRuleId}

```
{
  "dnsRuleId": "dnsRule1",
  "domainName": "www.example.com",
  "ipAddressType": "IP_V6",
  "ipAddress": "192.0.2.0",
  "ttl": "?",
  "state": "ACTIVE"
}
```

Attribute name	Data type	Cardinality	Description
dnsRuleId	String	1	Identifies the DNS Rule
domainName	String	1	FQDN resolved by the DNS rule
ipAddressType	Enum (inlined)	1	Specify the IP address type, value: IP_V6, IP_V4
ipAddress	String	1	IP address associated with the FQDN resolved by the DNS rule
ttl	Int	0..1	Time to live value, in seconds
state	Enum (inlined)	1	Contains the DNS rule state: ACTIVE, INACTIVE. This attribute may be updated using HTTP PUT method
NOTE: If no ttl value is provided, the DnsRule shall not expire.			

7.2 Interface Design- Mm5

Root Path: /mepcfg/mec_app_config/v1

GET /rules/{appInstanceId}/dns_rules

Response: [{
"dnsRuleId": "ac4cd7be-debc-4244-a2a5-1598225927b9",
"domainName": "www.example.com",
"ipAddressType": "IP_V4",
"ipAddress": "172.16.189.14",
"ttl": 30,
"state": "ACTIVE"
}]

GET /rules/{appId}/dns_rules/{dns_rule}

Response: {
"dnsRuleId": "ac4cd7be-debc-4244-a2a5-1598225927b9",
"domainName": "www.example.com",
"ipAddressType": "IP_V4",
"ipAddress": "172.16.189.14",
"ttl": 30,
"state": "INACTIVE"
}

POST /rules/{appInstanceId}/dns_rules

Request: {
"dnsRuleId": "ac4cd7be-debc-4244-a2a5-1598225927b9",
"domainName": "www.example.com",
"ipAddressType": "IP_V4",
"ipAddress": "172.16.189.14",
"ttl": 30,
"state": "INACTIVE"
}

PUT /rules/{appId}/dns_rules/{dns_rule}

Request: {
"state": "INACTIVE"
}

DELETE /rules/{appId}/dns_rules/{dns_rule}

7.3 Interface Design- MEP to DNS

PUT /mep/dns_server_mgmt/v1/rrecord

Request: [

```
{
 "zone": ".",
 "rr": [
  {
   "name": "www.example.com.",
   "type": "A",
   "class": "IN",
   "ttl": 30,
   "rData": [ "172.16.104.22"]
  },
  {
   "name": "www.example1.com.",
   "type": "A",
   "class": "IN",
   "ttl": 30,
   "rData": ["172.16.104.40", "172.16.104.38", "172.16.104.27"]
  }
 ]
}
]
```

DELETE /mep/dns_server_mgmt/v1/rrecord/{fqdn}/{rrType}

8. Open Points

- Need to adopt the authentication and encrption between MEP and DNS server same as MEP and MEP-Agent.
- Consider network isolation for DNS in Mm5 and other interfaces.
- Add deployment and installation requirements.

Discussions