Arris Proposal: A QoS Gate Model for the OpenDaylight PCMM API

Erich Arnold (erich.arnold@arris.com)
Engineering Fellow
Network Solutions CTO Team

Why is Arris Proposing a QoS Gate Model for the ODL PCMM API?

- The problem is that the current ODL PCMM NB API is based on the OpenFlow flow programmer model
  - The ODL OF model abstracts network nodes and the ports on those nodes
  - OF apps, therefore, must be topology-aware to be capable of diverting specific packet flows through multiple network nodes
- However, the PCMM QoS Gate abstraction is different because it provides a DOCSIS QoS treatment on the RFI to a given subscriber's device connected between a CM and its hosting CCAP device rather than providing a flow forwarding function
  - PCMM apps are typically directly in contact with their client devices and thus are not topology-aware and, in particular, they do not know (or want to know) which CM or which CCAP is hosting these client devices
  - PCMM QoS Gates do not divert packet flows, but rather just provide a QoS bandwidth treatment for the selected packet streams transiting the CCAP via the cable RFI
  - Some PCMM apps may be able to construct complex DOCSIS bandwidth parameter specs on the fly, but most are likely to select from a set of pre-constructed, pre-provisioned, and pre-tested Service Class Names (SCNs) that are pre-installed on the target CMTSes
    - Therefore, even though PCMM supports other traffic profiles, the SCN profile is the easiest to use
  - Free benefit: when SCNs are used in PCMM Dynamic Service Flow activations under the control of PCMM Gates, the SCNs become an element in the IPDR record for those DSFs which allows billing systems to use the SCN to distinguish between statically provisioned HSD services and other PCMM dynamic services

Why is Arris Proposing a QoS Gate Model for the ODL PCMM API?

- Therefore, the current ODL PCMM NB API needs to be re-architected to provide a topology-agnostic QoS Gate abstraction rather than a topology-aware Packet Flow abstraction for the PCMM apps
- A proposed new model for the new ODL PCMM NB API follows and the ODL Helium SR2 packetcable plugin has been reimplemented to this model
  - The new ODL packetcable plugin is based on release/helium-sr2 and is basically a fork of the existing ODL packetcable project
  - The existing ODL packetcable code is malformed because it tries to conform to the OF API model (which is the wrong model) and was replaced
  - The COPS driver implementation (which mostly works) was saved with a few fixes for broken interfaces
  - The QoS Gate model simplifies the packetcable plugin code considerably
  - At the moment the packetcable-provider only uses the ODL config datastore for top-down Gate deployment and management
    - Future work to include the packetcable-consumer and the read-only operational datastore could be considered if it is deemed useful

Why is Arris Proposing a QoS Gate Model for the ODL PCMM API?

- The new PCMM QoS Gate API has 2 components: a CCAP config API for admins and a QoS Gate config API for apps
  - Network admins configure the topology information for all the CCAP devices including the COPS connection IP address:port, the COPS Application Manager Id, and the list of subscriber subnets and service class names served by each CCAP
    - The QoS Gate API uses this info to provide built-in topology mapping and SCN validation necessary to complete each PCMM Gate Set request to the correct target CCAP device
  - Topology-agnostic applications configure their gate requests which include the subscriberId that is used by the QoS Gate API to identify the hosting CCAP device
    - Creating a gate request results in a gate set operation to the CCAP while deleting a gate request results in a gate delete operation
  - Both APIs support only create and delete operations; update is not supported
- Using the PCMM CCAP Config API a network admin creates and deletes CCAP devices in the ODL config datastore via /restconf/config/packetcable:ccap URLs
  - Adding a CCAP to the ODL config datastore initiates a COPS session from the plugin to that CCAP
  - This is necessary because, unlike OF nodes, CCAPs do not report into their PCMM Application Managers (i.e. ODL)

Why is Arris Proposing a QoS Gate Model for the ODL PCMM API?

- The config for each CCAP device also contains a list of subscriber subnets (IPv4 and IPv6) that are attached to it and a list of SCNs that are provisioned there
  - The internal subscriber subnet LPM table and the SCN table are used to validate PCMM API QoS Gate Requests and to map them to the correct CCAP device where an open COPS session is available
  - The subscriber subnets and the SCNs that are available to the PCMM apps are managed by the network admins; in particular these may be limited to a subset of all the subnets and SCNs actually configured on the CCAP target

- Using the PCMM QoS Gate Config API an application developer creates and deletes QoS Gates in the ODL config datastore via /restconf/config/packetcable:qos URLs
  - This provides a straight-forward QoS Gate service abstraction that maps directly onto the PCMM COPS gate model
  - The QoS Gate API is topology-agnostic which makes it easy for an app to provide services to subscribers anywhere in the network
  - There are no collisions among apps because each app has its own namespace for its own gates and these gates are organized by SubscriberId to make it easier to manage bandwidth services on a per subscriber basis

ODL PCMM Restconf API Operation

- Note well that the ODL /restconf/config API is asynchronous
  - An HTTP PUT request to the /resconf/config URL will immediately respond with HTTP 200 OK as long as the json object passes the basic yang model syntax; otherwise it will be rejected with a syntax error
    - In particular, the yang model does not validate any values in the config object
    - Therefore a successful PUT config request must be followed by a GET config request to verify that the PUT operation actually succeeded
  - The PCMM API will process the PUT config object and will then update the config object with an HTTP “response” message object that is only visible in a subsequent GET config request
    - The API will validate the PUT request config object for correct parameter values
    - If validation fails the “response” message object will indicate the failure and the app must delete the defunct config object and fix & retry the PUT config operation
    - The validated PUT request config object will then be used to either open a connection to the CCAP device or to send a gate set request
    - The result of this operation will be recorded in the “response” message object

Arris Proposal:
ODL PCMM CCAP Config API for Admins

- CCAP ID: a string identifier for the CCAP device
- Application Manager ID Container
  - AM tag: a locally unique identifier for the ODL PCMM plugin
  - AM type: allows multiple ODL controllers to communicate with a CCAP device
- Connection Container
  - Management IP Address of the CCAP device (IPv4 only for now)
  - The port number for the COPS connection (default 3918)
- Subscriber Subnets List
  - List of attached subscriber subnet prefix/len values (IPv4 and IPv6)
  - Each subscriberId for a gate request must be member of a known subnet
- Upstream and Downstream Service Class Name Lists
  - List of SCNs available at the CCAP device
  - SCNs have built-in direction attributes which must match the gate requests

Arris Proposal:
ODL PCMM CCAP Config API for Admins

- HTTP Response Message
  - The HTTP response from the API for the PUT operation which is only visible in a subsequent GET response json object
  - If 200 OK, then the request is active until the config object is deleted
    - The COPS connection will remain open with Keep-Alive messages until the CCAP object is deleted
  - If 400 Bad Request, then the message will identify the parameters that are malformed
  - If 404 Not Found, then the COPS connection could not be opened to the given IP address:port
  - If 4xx failure message, the config object is defunct and must be deleted
- Note well that deleting a CCAP object only closes the COPS connection and does not remove any QoS Gate Requests that are active on the target CCAP
  - The app may recreate the CCAP object thereby opening a new COPS connection and then it may operate on existing or new QoS Gate Requests on that CCAP

Arris Proposal : ODL PCMM CCAP Config API Example: PUT Request

PUT http://localhost:8181/restconf/config/packetcable:ccap/ccaps/E6000-CTO

```
{"ccaps": [{
        "ccapId": "E6000-CTO",
        "connection": {
            "ipAddress": "192.168.1.2",
            "port": 3918
        },
        "amId": {
            "am-tag": "0xcada",
            "am-type": "1"
        },
        "subscriber-subnets": [
            "44.137.0.0/16",
            "2001:4978:030d:1000:0:0:0:0/52"
        ],
        "downstream-scns": [
            "ipvideo_dn",
            "extrm_dn",
            "SCNC"
        ],
        "upstream-scns": [
            "SCNA",
            "extrm_up"
        ]
    }]
}
```

Arris Proposal: ODL PCMM CCAP Config API Example: GET Request

GET http://localhost:8181/restconf/config/packetcable:ccap/ccaps/E6000-CTO


```
{"ccaps":[
         {"ccapId":"E6000-CTO",
          "response": "200 OK – CCAP connected: E6-CTO @ 192.168.1.2:3918",
          "upstream-scns":["SCNA","extrm_up"],
          "subscriber-subnets":[
                 "2001:4978:030d:1000:0:0:0:0/52",
                 "44.137.0.0/16"],
          "amId":{"am-tag":51930,"am-type":1},
          "connection":{"ipAddress":"192.168.1.2","port":3918},
          "downstream-scns":["extrm_dn","ipvideo_dn","SCNC"]
          }
    ]
}
```

Arris Proposal:
ODL PCMM QoS Gate Config API for Apps

- List of Application Ids
  - The appID is the key to all PCMM QoS Gates owned by a given application
  - This permits multiple apps to operate independently within their own subId/gateId namespaces
- List of Subscriber Ids
  - The subId is the network identity of the subscriber's device that will receive the bandwidth allocation via PCMM Qos Gates
  - Typically the subID is the CPE IP address of the device
    - May be either a public IPv4 address or routeable-scope IPv6 address
    - If IPv4, the subId may be a shared NAT public IP address, not a private LAN address
    - May also be a CM IP address if known
  - The PCMM QoS Gate API locates the subscriber's IP address in the network
    - The PCMM app does not need to know the network topology
    - The subscriber's IP address subnet prefix will be verified to belong to a known CCAP subscriber subnet
- List of Gate Ids
  - A gateID is a unique name for each QoS Gate relative to <appId>/<subId>
  - QoS Gates are unidirectional: thus a bidirectional service requires 2 PCMM QoS Gates while a downstream-only video service requires only 1 PCMM QoS Gate

Arris Proposal:
ODL PCMM QoS Gate Config API for Apps

- QoS Gate Request Parameters
  - Provided in a json object that is carried as the data of the HTTP PUT request
  - Associates the <appId>/<subId>/<gateId> with its PCMM QoS Gate traffic profile and the packet classifier that identifies which packets will be carried by the dynamic service flow in the CCAP
  - The traffic profile provides a reference to a DOCSIS Service Class Name (SCN) that is pre-provisioned in the target CCAP
  - The QoS Gate packet classifier is either a (legacy) classifier, an extended classifier or an IPv6 classifier as defined by the PCMM spec
- HTTP Response Message
  - The HTTP response message from the API for the PUT operation; this message is only visible in a subsequent GET response json object
  - If 200 OK, then the request is active until the config object is deleted
  - If 400 Bad Request, then the message will identify the parameters that are malformed
  - If 404 Not Found, then either the CCAP cannot be matched to the subscriberId or the SCN is not provisioned on the CCAP
  - If 4xx failure message: the config object is defunct and must be deleted

Arris Proposal : ODL PCMM QoS Gate Config API - Legacy Classifier API Example

PUT http://localhost:8181/restconf/config/packetcable:qos/apps/testApp/subs/44.137.1.20/gates/ipvideo_dn-1

```
{ "gates": [{
        "gateId": "ipvideo_dn-1",
        "gate-spec": {"dscp-tos-overwrite": "0xa0",
                      "dscp-tos-mask": "0xff"},
        "traffic-profile": {"service-class-name": "ipvideo_dn"},
        "classifier": {
            "srcIp": "10.10.10.1",
            "dstIp": "44.137.1.20",
            "protocol": "0",
            "srcPort": "1234",
            "dstPort": "4321",
            "tos-byte": "0xa0",
            "tos-mask": "0xe0"
        }
    }]
}
```

Arris Proposal : ODL PCMM QoS Gate Config API – Legacy Classifier API Example

GET http://localhost:8181/restconf/config/packetcable:qos/apps/testApp/subs/44.137.1.20/gates/ipvideo_dn-1


```
{"gates":[
         {"gateId":"ipvideo_dn-1",
          "response": "200 OK – created QoS gate for E6000-CTO/testApp/
44.137.1.12/ipvideo_dn-1 with COPS GateId e2090029",
          "traffic-profile":{"service-class-name":"ipvideo_dn"},
          "classifier":{
                  "dstPort":4321,
                  "tos-mask":224,
                  "srcIp":"10.10.10.1",
                  "srcPort":1234,
                  "protocol":0,
                  "dstIp":"44.137.1.12",
                  "tos-byte":160},
         "gate-spec":{"dscp-tos-overwrite":160,"dscp-tos-mask":255}
         }]
}
```

Arris Proposal : ODL PCMM QoS Gate Config API – Extended Classifier Example

PUT http://localhost:8181/restconf/config/packetcable:qos/apps/testApp/subs/44.137.1.20/gates/ipvideo_dn-1

```
{ "gates": [{
        "gateId": "ipvideo_dn-1",
        "gate-spec": {"dscp-tos-overwrite": "0xa0",
                      "dscp-tos-mask": "0xff"},
        "traffic-profile": {"service-class-name": "ipvideo_dn"},
        "ext-classifier": {
             "srcIp": "10.10.10.0",
             "srcIpMask":"255.255.255.0",
             "dstIp":"44.137.1.12",
             "dstIpMask":"255.255.255.255",
             "tos-byte": "0xa0",
             "tos-mask": "0xe0",
             "protocol": "0",
             "srcPort-begin": "1234",
             "srcPort-end": "1235",
             "dstPort-begin": "4321",
             "dstPort-end": "4322"
         }
    }]
}
```

Arris Proposal : ODL PCMM QoS Gate Config API – Extended Classifier Example

GET http://localhost:8181/restconf/config/packetcable:qos/apps/testApp/subs/44.137.1.20/gates/ipvideo_dn-1


```
{"gates":[
        {"gateId":"ipvideo_dn-1",
         "response": "200 OK – created QoS gate for E6000-CTO/testApp/44.137.1.12/ipvideo_dn-1 with COPS GateId e2090028",
         "traffic-profile":{"service-class-name":"ipvideo_dn"},
         "ext-classifier":{
                "dstPort-end":4322,
                "dstPort-start":4321,
                "tos-mask":224,
                "srcIp":"10.10.10.0",
                "srcPort-start":1234,
                "protocol":0,
                "dstIpMask":"255.255.255.255",
                "dstIp":"44.137.1.12",
                "srcPort-end":1235,
                "srcIpMask":"255.255.255.0",
                "tos-byte":160},
        "gate-spec":{"dscp-tos-overwrite":160,"dscp-tos-mask":255}
        }]
}
```

Arris Proposal : ODL PCMM QoS Gate Config API – IPv6 Classifier Example

PUT http://localhost:8181/restconf/config/packetcable:qos/apps/testApp/subs/44.137.1.20/gates/ipvideo_dn-1

```
{ "gates": [{
        "gateId": "ipvideo_dn-1",
        "gate-spec": {"dscp-tos-overwrite": "0xa0",
                      "dscp-tos-mask": "0xff"},
        "traffic-profile": {"service-class-name": "ipvideo_dn"},
        "ipv6-classifier": {
            "srcIp6": "2001:4978:030d:1000:0:0:0:0/64",
            "dstIp6": "2001:4978:030d:1101:0:0:0:0/64",
            "flow-label": "0",
            "tc-low": "0xa0",
            "tc-high": "0xa0",
            "tc-mask": "0xe0",
            "next-hdr": "256",
            "srcPort-begin": "1234",
            "srcPort-end": "1235",
            "dstPort-begin": "4321",
            "dstPort-end": "4322"

        }
    }]
}
```

Arris Proposal : ODL PCMM QoS Gate Config API – IPv6 Classifier Example

GET http://localhost:8181/restconf/config/packetcable:qos/apps/testApp/subs/44.137.1.20/gates/ipvideo_dn-1


```
{"gates":[
        {"gateId":"ipvideo_dn-1",
         "response": "200 OK – created QoS gate for E6000-CTO/testApp/
44.137.1.12/ipvideo_dn-1 with COPS GateId e2090027",
         "traffic-profile":{"service-class-name":"ipvideo_dn"},
         "ipv6-classifier":{
                "dstPort-end":4322,
                "dstPort-start":4321,
                "flow-label":0,
                "tc-mask":224,
                "srcIp6": "2001:4978:030d:1000:0:0:0:0/64",
                "srcPort-start":1234,
                "next-hdr":256,
                "dstIp6": "2001:4978:030d:1101:0:0:0:0/64",
                "srcPort-end":1235,
                "tc-low":160,
                "tc-high":160},
        "gate-spec":{"dscp-tos-overwrite":160,"dscp-tos-mask":255}
        }]
}
```