Dynamic On-Demand QoS
A Practical Approach to Intent-Based Networking
The Arqueopterix Project

- Objective: Improve user experience in interactive video applications
- Technologies
  - Dynamic On-Demand QoS
  - Q4S: Real time e2e Quality Monitoring
  - Logarithmical Hopping Encoder (Ultra Fast Video encoder/decoder)
Architecture

Northbound API

Network Controller

Southbound APIs

Actuator

Network QoS request

Q4S alert

App Quality Modification

Q4S Monitoring

Application

Virtual Networks

Physical Networks

OpenFlow

netconf

Virtual Networks

Physical Networks
Dynamic On-Demand QoS

• SDN: enabler for Dynamic QoS
• sub-second response to a QoS Requests
• Flow level QoS Requests
• Billing and Accounting Impact
Dynamic **On-Demand** QoS

- Northbound API to request QoS
- Request Originator: Client vs OTT Provider
- Billing and Accounting Impact
Use Cases & Scenarios

- Video Conference
Use Cases & Scenarios

• Video Streaming
Use Cases & Scenarios

- Cloud Gaming
QoS with OpenFlow 1.3

- Multiple tables and pipelining to support independent SDN applications
- Scalability through packet marking @edge and Traffic Class Queueing @core
- Traffic Shaping through OF Meters @edge
QoS with OpenFlow 1.3

**EDGE Switch Pipeline**

**Table 0 Dispatcher**
- IP: GOTO T10
- LLDP OUT: Controller
- ARP: Controller
- *: DROP

**Table 10 ACL**
- IP DST=X: METER, GOTO T20
- IP DST=Y: METER, GOTO T20
- IP_NET=Z: SET_DSCP, GOTO T20
- *: DROP

**Table 20 L2 Forward**
- MAC_DST=X: OUT=1
- MAC_DST=Y: OUT=2

**CORE Switch Pipeline**

**Table 0 Dispatcher**
- IP: GOTO T10
- LLDP OUT: Controller
- ARP: Controller
- *: GOTO T20

**Table 15 DiffServ**
- IP_DSCP=A: SET_QUEUE:1, GOTO T20
- IP_DSCP=B: SET_QUEUE:2, GOTO T20
- IP_DSCP=C: SET_QUEUE:3, GOTO T20
- *: SET_QUEUE:0, GOTO T20

**Table 20 L2 Forward**
- MAC_DST=X: OUT=1
- MAC_DST=Y: OUT=2
- *: DROP
Northbound QoS API

• Simple requests answering two questions:
  • What do we need?
    • Differentiated services
  • Which traffic should be treated differently?
    • Flow-application granularity (match by IP, transport protocol, port …)
Northbound QoS API

- QoS petitioners don’t know about network topology nor hardware capabilities.
- SDN controller maps QoS request into low-level QoS commands:
  - OpenFlow 1.3 rules
  - Netconf requests
Northbound QoS API

- HTTP REST API: [OpenAPI Definition](#)
- Intent-Based: High Abstraction Level
  - Flow-level resolution
  - Service level defined as a single integer

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Technology independency</td>
<td>Accounting and Billing for QoS</td>
</tr>
<tr>
<td>Simple QoS Requests: Give me more/less Quality</td>
<td>Cannot set detailed QoS Requirements in Request</td>
</tr>
</tbody>
</table>