Neutron DSCP
Policing Your Network

Nate Johnston - Principal Engineer - Comcast
David Shaughnessy - Network Software Engineer - Intel
Presentation Outline

- What is DSCP?
- DSCP Use Cases
- Implementing DSCP in Neutron QoS
- What’s Next?
- Conclusion
What is DSCP?
What is DSCP?

DSCP stands for “Differentiated Services Code Point”.

DSCP is a protocol for specifying and controlling network traffic by class so that certain types of traffic get precedence - for example, voice traffic, which requires a relatively uninterrupted flow of data, might get precedence over other kinds of traffic.

- DSCP is defined in RFC 2474 "Definition of the Differentiated Services Field (DS field) in the IPv4 and IPv6 Headers"

- DSCP for Tunnels is governed by RFC 2983 "Differentiated Services and Tunnels"
The DSCP Bits in the IP Header

DSCP is a six-bit field in the IP header – it comprises the high six bits of the eight bit DS (“DiffServ”) field in an IPv4 header. The DS field was formerly referred to as the ToS (“Type of Service”) field.

In IPv6, the DS field has been renamed the Traffic Class field. It functions identically to the DS field in IPv4.

In IPv4, DS is the third field in the IP header, and in IPv6 Traffic Class is the second, which indicates the importance of DSCP.
# DSCP in the IPv4 Packet

## IPv4 Header (RFC 791)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Version (4-bit)</td>
<td>Version of IP protocol</td>
</tr>
<tr>
<td>4</td>
<td>IP Header Length (4-bit)</td>
<td>Length of IP header in 32-bit units</td>
</tr>
<tr>
<td>8</td>
<td>DiffServ (8-bit)</td>
<td>DSCP for Quality of Service</td>
</tr>
<tr>
<td>16</td>
<td>Total Length (16-bit, in byteOffsets)</td>
<td>Total length of packet in bytes</td>
</tr>
<tr>
<td>24</td>
<td>IP Identification Number (16-bit)</td>
<td>Identification number for the packet</td>
</tr>
<tr>
<td>32</td>
<td>R</td>
<td>Don’t fragment (DF)</td>
</tr>
<tr>
<td>33</td>
<td>D</td>
<td>Don’t fragment (DF)</td>
</tr>
<tr>
<td>34</td>
<td>M</td>
<td>More Fragments (MF)</td>
</tr>
<tr>
<td>40</td>
<td>Fragmentation Offset (13-bit)</td>
<td>Offset of fragment within the larger packet</td>
</tr>
<tr>
<td>43</td>
<td>Time To Live (8-bit)</td>
<td>Time to live (TTL)</td>
</tr>
<tr>
<td>44</td>
<td>Protocol (8-bit)</td>
<td>Protocol number</td>
</tr>
<tr>
<td>48</td>
<td>Source IP Address (32-bit)</td>
<td>Source IP address</td>
</tr>
<tr>
<td>56</td>
<td>Destination IP address (32-bit)</td>
<td>Destination IP address</td>
</tr>
<tr>
<td>64</td>
<td>IP Options (variable… if any)</td>
<td>Options for IP packet</td>
</tr>
<tr>
<td>65</td>
<td>Data (variable)</td>
<td>Data of the packet</td>
</tr>
</tbody>
</table>
# DSCP in the IPv6 Packet

![IPv6 Header (RFC 2460)](image)

<table>
<thead>
<tr>
<th>Byte Offset 0</th>
<th>Byte Offset 1</th>
<th>Byte Offset 2</th>
<th>Byte Offset 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version (4-bit)</td>
<td>Traffic Class (8-bit)</td>
<td>Flow Label (20-bit, in byte Offsets)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte Offset 4</th>
<th>Byte Offset 5</th>
<th>Byte Offset 6</th>
<th>Byte Offset 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payload Length (16-bit)</td>
<td>Next Header (8-bit)</td>
<td>Hop Limit (8-bit)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte Offset 8</th>
<th>Byte Offset 9</th>
<th>Byte Offset 10</th>
<th>Byte Offset 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source IP Address (128-bit)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte Offset 12</th>
<th>Byte Offset 13</th>
<th>Byte Offset 14</th>
<th>Byte Offset 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte Offset 16</td>
<td>Byte Offset 17</td>
<td>Byte Offset 18</td>
<td>Byte Offset 19</td>
</tr>
<tr>
<td>Destination IP Address (128-bit)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte Offset 20</th>
<th>Byte Offset 21</th>
<th>Byte Offset 22</th>
<th>Byte Offset 23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data (variable)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The 8 bits of the DS byte are divided into 2 sections: DSCP has the high 6 bits, and ECN ("Explicit Congestion Notification") has the last 2 bits. For the purposes of this presentation, we are ignoring the ECN bits entirely.

The 6 DSCP bits are arranged into 4 sections:

- The highest 3 bits are used as the “Precedence” setting, which defines the Class Selector.
- The next two bits designate “Delay” and “Throughput”, and collectively define the “Assured Forwarding” (AF) setting.
- The lowest bit designates “Reliability” and is unused.

The contents of the DSCP bits are collectively referred to as a “mark” or “codepoint”.

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>DSCP</td>
<td>ECN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Precedence</td>
<td>Delay</td>
<td>Throughput</td>
<td>Reliability</td>
<td>ECT</td>
<td>CE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Per-Hop Behavior

- Per-Hop Behavior (PHB) describes how traffic is handled at each hop based on the DSCP value that is set.

- In cases where packets will be dropped because of congestion, traffic with a lower DSCP mark, or none at all, will be dropped before traffic with a higher DSCP mark.

- There are 4 kinds of PHB setting: none (the default), Class Selector 1-7, three Assured Forwarding subclasses for Class Selectors 1-4, and Expedited Forwarding.

- Before you implement DSCP, you need to really know how these things work or you may cause unintended effects. Contact a network engineer to verify your DSCP implementation details.
# All DSCP Marks

## DSCP Bit Chart

<table>
<thead>
<tr>
<th>Class</th>
<th>Precedence</th>
<th>D</th>
<th>T</th>
<th>R</th>
<th>DSCP decimal</th>
<th>Entire DS Byte (ToS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>CS0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CS1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AF11</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>AF12</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AF13</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CS2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AF21</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>AF22</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AF23</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CS3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AF31</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>AF32</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AF33</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CS4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AF41</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>AF42</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AF43</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CS5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EF</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CS6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CS7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
DSCP Use Cases
Typically, networks operate on a best-effort delivery basis, which means that all traffic has equal priority and an equal chance of being delivered in a timely manner. When congestion occurs, all traffic has an equal chance of being dropped.

DSCP allows you to select specific network traffic for prioritization according to its relative importance and use congestion-management and congestion-avoidance techniques to provide preferential treatment.

This is what DSCP was designed for.
Use Case 2: DSCP Marks as Security Policy

- DSCP marks, as part of the TCP header, can be used as criteria in firewall rules and network device ACLs. Here is a Cisco example:

```plaintext
access-list 101 permit ip any any dscp cs1
```

- Therefore you could come up with a convention, for example:
  - CS4 is production guests
  - CS3 is QA guests
  - And so forth...

- Then ensure that the ACLs permit traffic with the given marks to the networks corresponding to their function.
Implementing DSCP in Neutron QoS
The Neutron DSCP code did not make it into the Mitaka release of Neutron because of last minute issues.

Those issues have been resolved, and the DSCP changes have been merged into Neutron master.

DSCP functionality will be available in the Newton release.
QoS Object Relationships

Port

QosPolicy

QosRule

QosDscpMarkingRule

QosBandwidthLimitRule

extends

extends
Attaching a QoS Policy to a Port

- Create a QoS policy

```
# openstack network qos-policy-create 'urgent' \
    --description 'Deliver now'
```

- Create a DSCP rule for a QoS policy

```
# openstack network qos-dscp-marking-rule-create urgent \
    --dscp-mark 26
```

- Assign a QoS policy to a port

```
# openstack network port-update \
    48c6256f-9123-4e39-a321-108782807cf0 --qos-policy urgent
```
QoS Policy Adds and Updates

Controller

"DSCP mark 26, please"

"Update a port, please."

"What are the port details?"

"Here, including QoS policy <uuid>.

"What are policy <uuid>’s rules?"

"Here they are."

"I’m subscribing to policy <uuid>."

"Hey, policy <uuid> changed!"

Compute

ML2 Plugin and OVS Mechanism Driver

L2 Agent and QoS Agent Extension
QoS Extension Architecture

User assigns QoS policy containing a DSCP mark rule, to port

OVS driver sends RPC message

QoS agent extension receives RPC message

QoS agent extension notifies OVS agent

OVS agent sets DSCP mark on port

"DSCP mark 26, please"
Provider Network with OVS

**Legend**
- **Provider network**
- **Generic network (vlans)**
- **Management network**
- **External network**

(Parenthetical numbers indicate OpenFlow ofport.)
Introduction to OpenFlow

1. Find highest-priority matching flow entry
2. Execute instructions:
   - apply action **list**
   - update (clear or write) action **set**
3. Apply action **set**

**Match Fields**
- ingress port
- packet headers
- metadata

**Actions**
- modify packet headers
- update metadata
- send packet to another table or out of the pipeline
DSCP in OVS

OVS Flow Table Before DSCP Marking Added

cookie=1234, table=0, priority=10, arp, in_port=6 actions=resubmit(,24)
cookie=1234, table=0, priority=0 actions=NORMAL

cookie=1234, table=24, priority=2, arp, in_port=6, arp_spa=10.251.2.136 actions=NORMAL

OVS Flow Table After DSCP Marking Added

cookie=1234, table=0, priority=10, arp, in_port=6 actions=resubmit(,24)
cookie=1234, table=0, priority=1, in_port=6 actions=mod_nw_tos:104,NORMAL
cookie=1234, table=0, priority=0 actions=NORMAL

cookie=1234, table=24, priority=2, arp, in_port=6, arp_spa=10.251.2.136 actions=NORMAL
DSCP Mark Witnessed

OVS Flow Table After DSCP Marking Added

- cookie=1234, table=0, priority=10, arp, in_port=6 actions=resubmit(,24)
- cookie=1234, table=0, priority=1, in_port=6 actions=mod_nw_tos:104,NORMAL
- cookie=1234, table=0, priority=0 actions=NORMAL
- cookie=1234, table=24, priority=2, arp, in_port=6, arp_spa=10.251.2.136 actions=NORMAL

tcpdump Output Reflecting DSCP Mark

03:36:15.516084 fa:16:3e:41:90:36 > fa:16:3e:41:90:37, ethertype IPv4 (0x0800), length 73: (tos 0x68, ttl 10.251.2.136.58321 > 10.251.2.132.8999: Flags [P.], cksum 0x7a0e (correct), seq 78:85, ack 1, win 229,
03:36:15.516156 fa:16:3e:41:90:37 > fa:16:3e:82:32:e0, ethertype IPv4 (0x0800), length 66: (tos 0x0, ttl 10.251.2.132.8999 > 10.251.2.136.58321: Flags [], cksum 0x1b28 (incorrect -> 0x6528), seq 1, ack 85,
DSCP Mark Witnessed: Wireshark

Frame 1: 72 bytes on wire (576 bits), 72 bytes captured (576 bits)
Ethernet II, Src: fa:16:3e:50:00:02 (fa:16:3e:50:00:02), Dst: fa:16:3e:72:9a:aa (fa:16:3e:72:9a:aa)
Internet Protocol Version 4, Src: 10.251.2.158, Dst: 10.251.2.154

0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes

Differentiated Services Field: 0x68 (DSCP: AF31, ECT: Not-ECT)

0110 10.. = Differentiated Services Codepoint: Assured Forwarding 31 (26)
.... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
Total Length: 58
Identification: 0x64ec (25836)
Flags: 0x02 (Don't Fragment)
Fragment offset: 0
Time to live: 64
Protocol: TCP (6)

Header checksum: 0xba3c [validation disabled]
Source: 10.251.2.158
Destination: 10.251.2.154
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]

Transmission Control Protocol, Src Port: 55421 (55421), Dst Port: 8999 (8999), Seq: 1, Ack: 1, Len: 6

Data (6 bytes)

0000 fa 16 3e 72 9a aa fa 16 3e 50 c0 02 08 00 45 88
0010 00 3a 64 ec 40 00 40 00 6d ba 3c 0a fb 02 9e 0a fa
0020 02 9a d8 7d 23 27 8f af 0f 6a aa b0 cf 70 80 18
0030 00 e5 8b 30 00 00 01 01 08 0a 00 00 48 56 03 cf
0040 9b 83 61 73 64 66 0d 0a
The session ID of the L2 agent is used as the cookie value in OVS flow entries.

When an L2 agent reboots, modifies a port or updates a firewall it removes any flows which flow_cookie_value != my_session_ID.

Because L2 agent extensions, such as the QoS extension, manage their own OVS flow entries, the challenge is to ensure that an agent does not remove extensions’ flow entries.
L2 Agent Updates: Solution

- Agent assigns each extension its own cookie value

- Upon agent reboot or port update extension-owned flow entries are preserved
QoS Agent Extension

In_port=6, actions=mod_nw_tos:104, Normal

OVSCookieBridge

Cookie = 0x1234

OVSAgentBridge

Cookie=0x1234, In_port=6, actions=mod_nw_tos:104, Normal

In_port=6, actions=mod_nw_tos:104, Normal
Feature Isolation: Challenges

- We need to allow Neutron to use OVS to apply multiple features (e.g., DSCP marking and VLAN tagging) to a single packet.

- We need to ensure that a feature doesn’t hijack other OVS processing by removing a packet from the pipeline.

- We need to ensure that feature flows don’t affect, and aren’t affected by, any other pipeline processing, including any existing or future features.
● Use metadata to corral packets for feature application

● A packet entering the pipeline has all metadata fields set to 0

● A feature’s table0 flow entry will match (in part) on a particular metadata field and resubmit matching packets to a “feature table”

● The feature-table flow logic will apply the feature to the packet, set a non-0 value in the packet’s feature-specific metadata field, and resubmit the packet back to table0 for further processing
OpenStack’s DSCP in OVS

**OVS Flow Table Before DSCP Marking Added**

- cookie=1234, table=0, priority=10, arp, in_port=6, actions=resubmit(,24)
- cookie=1234, table=0, priority=0, actions=NORMAL
- cookie=1234, table=24, priority=2, arp, in_port=6, arp_spa=10.251.2.136, actions=NORMAL

**OVS Flow Table After Feature’s DSCP Marking Added**

- cookie=abcd, table=0, priority=65535, reg2=0x0, in_port=6, actions=resubmit(,10)
- cookie=1234, table=0, priority=10, arp, in_port=6, actions=resubmit(,24)
- cookie=1234, table=0, priority=0, actions=NORMAL
- cookie=abcd, table=10, priority=0, actions=load:0x37->NXM_NX_REG2[0..5], mod_nw_tos:104, resubmit(,0)
- cookie=1234, table=24, priority=2, arp, in_port=6, arp_spa=10.251.2.136, actions=NORMAL
Server-Agent Communications: Challenges

- Additional rule types may come available during a server upgrade
- Different agents may use different API objects (e.g., one agent may not know about DSCP rules while another does)
- Different agents may use differently versioned objects (e.g., one agent knows about QoS policy version 1.0 while another agent knows about QoS policy 1.1)
- An agent needs to know about a given policy *instance* id
Server-Agent Communications: Solution

- Neutron OVS agent
  - QosRuleType 1.0
  - QosRuleType 1.1

- RabbitMQ
  - QosRuleType-1.0 fanout
  - QosRuleType-1.1 fanout

- Neutron Server

Create Versioned Fanout Queue
Report Versioned Objects
What’s Next?
Ingress DSCP filtering: We discussed a change to Security Groups that would allow ingress DSCP filtering, but Security Group changes are fraught with peril. This could become a roadmap item for FWaaS at some point however.

Marking encapsulating packets with the DSCP mark of the encapsulated traffic: We believe it very unlikely that DSCP-based filtering will occur between parts of a region, because so much of that will be tunneled; the assumption is that any DSCP-related behavior will become relevant only after the traffic exits the spines.
Future Roadmap

- Neutron support for Explicit Congestion Notification (ECN)
- Neutron traffic classification
- Min bandwidth guarantees
- Ingress bandwidth limiting
Conclusion
 Configuring QoS

- **Neutron server** *(neutron.conf)*
  
  ```
  service_plugins = neutron.services.qos.qos_plugin.QoSPlugin
  notification_drivers = message_queue
  ```

- **ML2 plugin and L2 agent** *(ml2_conf.ini)*
  
  ```
  [ml2]
  extension_drivers = qos
  [agent]
  extensions = qos
  ```
● **DevStack** *(local.conf)*

```bash
enable_plugin neutron
git://git.openstack.org/openstack.neutron
enable_service q-qos

[[post-config|$NEUTRON_CONF]]
[DEFAULT]
 service_plugins=neutron.services.qos.qos_plugin.QoSPlugin

[[post-config|/$Q_PLUGIN_CONF_FILE]]
[ml2]
 extension_drivers=qos
[agent]
 extensions=qos
```

Configuring QoS, cont.
Other OpenStack Resources

General Documentation

- Networking Guide: Using OpenStack Networking with QoS
- Tokyo presentation “QoS - A Neutron n00bie”

Prerequisites for the DSCP Change

- L2 agent extensions implementation: agent API
- RPC callbacks rolling upgrades implementation
- RPC callbacks rolling upgrades implementation: reporting and integration

Changes Associated with the DSCP Change

- Original QoS API extension specification
- QoS API extension with DSCP specification
- Server and agent DSCP QoS rule implementation
- DSCP implementation in neutronclient
- DSCP in Heat: specification
- DSCP in Heat: implementation of the QosDscpMarkingRule resource
If You Want to Read More about DSCP...

**DiffServ RFCs**
- [RFC 2474](https://tools.ietf.org/html/rfc2474) — *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*
- [RFC 2475](https://tools.ietf.org/html/rfc2475) — *An Architecture for Differentiated Services*
- [RFC 2597](https://tools.ietf.org/html/rfc2597) — *Assured Forwarding PHB Group*
- [RFC 2983](https://tools.ietf.org/html/rfc2983) — *Differentiated Services and Tunnels*
- [RFC 3086](https://tools.ietf.org/html/rfc3086) — *Definition of Differentiated Services per Domain Behaviors and Rules for their Specification*
- [RFC 3140](https://tools.ietf.org/html/rfc3140) — *Per Hop Behavior Identification Codes* (replaces RFC 2836)
- [RFC 3247](https://tools.ietf.org/html/rfc3247) — *Supplemental Information for the New Definition of the EF PHB (Expedited Forwarding Per-Hop Behavior)*
- [RFC 4594](https://tools.ietf.org/html/rfc4594) — *Configuration Guidelines for DiffServ Service Classes*
- [RFC 5865](https://tools.ietf.org/html/rfc5865) — *A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic* (updates RFCs 4542 and 4594)

**DiffServ Management RFCs**
- [RFC 3290](https://tools.ietf.org/html/rfc3290) — *An Informal Management Model for DiffServ Routers*
- [RFC 3317](https://tools.ietf.org/html/rfc3317) — *Differentiated Services Quality of Service Policy Information Base*
DSCP Use Cases in Patents

All patents listed are U.S. patents.

- [US20070199064](https://example.com) — Method and system for quality of service based web filtering
- [US20080089324](https://example.com) — Indicating or remarking of a dscp for rtp of a flow (call) to and from a server
- [US20080144502](https://example.com) — In-band quality-of-service signaling to endpoints that enforce traffic policies at traffic sources using policy messages piggybacked onto DiffServ bits
- [US8767569 B2](https://example.com) — Dynamic DSCP availability request method
- [US20130283379](https://example.com) — System, method and apparatus that employ virtual private networks to resist ip qos denial of service attacks
It was a collective effort ... so our thanks goes to:

Margaret Frances
Miguel Angel Ajo
Ihar Hrachyshka
Victor Howard
James Reeves
Gary Kotton
John Schwarz
● Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

● No computer system can be absolutely secure.

● Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

● Intel, the Intel logo and others are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.

● © 2016 Intel Corporation.

● © 2016 Comcast Corporation.
Thank You
Want these slides?