Network Working Group
Internet Draft
Intended status: Standards Track
Expires: September 2013

S. Dhesikan
Cisco Systems
D. Druta, Ed.
AT&T
P. Jones
J. Polk
Cisco Systems

March 10, 2013

DSCP and other packet markings for RTCWeb QoS
draft-dhesikan-tsvwg-rtcweb-qos-01

Abstract

Many networks, such as service provider and private networks, can provide per packet treatments based on Differentiated Services Code Points (DSCP) on a per hop basis. This document provides the recommended DSCP values for browsers to use for various classes of traffic.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2013.

Table of Contents

1. Introduction

Differentiated Services Code Points (DSCP) [RFC2474] style packet marking can help provide QoS in some environments. There are many use cases where such marking does not help, but it seldom makes things worse if packets are marked appropriately. In other words, when attempting to avoid congestion by marking certain traffic flows, say all audio or all audio and video, marking too many audio and/or video flows for a given network's capacity can prevent desirable results. Either too much other traffic will be starved, or there is not enough capacity for the preferentially marked packets (i.e., audio and/or video).

This draft proposes how a browser and other VoIP applications can mark packets. This draft does not contradict or redefine any advice from previous IETF RFCs but simply provides a simple set of recommendations for implementers based on the previous RFCs.

There are some environments where priority markings frequently help. These include:

1. Private networks (Wide Area)

2. If the congested link is the broadband uplink in a Cable or DSL scenario, often residential routers/NAT support preferential treatment based on DSCP.
3. If the congested link is a local Wi-Fi network, marking may help.
4. In some cellular style deployments, markings may help in cases where the network does not remove them.

Traditionally, DSCP values have been thought of as being site specific, with each site selecting its own code points for each QoS level. However in the RTCWeb use cases, the browsers need to set them to something when there is no site specific information. This document describes a reasonable default set of DSCP code point values drawn from existing RFCs and common usage. These code points are solely defaults. Future drafts may define mechanisms for site specific mappings to override the values provided in this draft.

This draft defines some inputs that the browser can look at to determine how to set the various packet markings and defines a mapping from abstract QoS policies (media type, priority level) to those packet markings.

2. Relation to Other Standards

This specification does not change or override the advice in any other standards about setting packet markings. It simply provides a non-normative summary of them and provides the context of how they relate into the RTCWeb context. This document also specifies the requirements for the W3C WebRTC API to understand what it needs to control, and how the control splits between things the JavaScript application running in the browser can control and things the browser needs to control. In some cases, such as DSCP where the normative RFC leaves open multiple options to choose from, this clarifies which choice should be used in the RTCWeb context.

3. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

4. Inputs

The following are the inputs that the browser provides to the media engine:

- Type of flow: The browser provides this input as it knows if the flow is audio, video, or data. In this specification, both interactive and streaming media are included. They are treated

in different categories as their QoS requirements are slightly different.

If the type of flow is multiplexed content, then the input is a list of the type of flows that are multiplexed within the single stream.

- Session Context: This input provides the session context for the type of flow. For example, the type of flow may be audio. The flow may be part of a VoIP session or an audio/video session. Such session context information helps the media engine and the underlying network to make decisions on how to treat the audio flow which may differ based on the entire session to which the flow belongs. The browser should know this information.

- Relative priority: Another input is the relative treatment of the stream within that session. Many applications have multiple video flows and often some are more important than others. JavaScript applications can tell the browser whether a particular media flow is high, medium, or low importance to the application.

5. DSCP Mappings

Below is a table of DSCP markings for each media type in which RTCWeb is interested. These DSCPs for each media type listed are a reasonable default set of code point values taken from [RFC4594]. A web browser SHOULD use these values to mark the appropriate media packets. More information on EF can be found in [RFC3246]. More information on AF can be found in [RFC2597].

In a multiplexed session the flow may either be all mapped to a single or to multiple DSCPs one per flow. If some networks cannot handle multiplexed traffic, then [I-D.ietf-rtcweb-rtp-usage] provides the capability to disable multiplexing.

Media Type	Low	Medium	High
Audio	46 (EF)	46 (EF)	46 (EF)
Interactive Video	38 (AF43)	36 (AF42)	34 (AF41)
Non-Interactive Video	26 (AF33)	28 (AF32)	30 (AF31)
Data	8 (CS1)	0 (BE)	10 (AF11)

Table 1 - Media Type, Priority, and DSCP Mapping

6. QCI Mappings

Media Type	Low	Medium	High
Audio	1	1	1
Interactive Video	2	2	2
Non-Interactive Video	8	6	4
Data	9	9	3

Table 2 - Media Type, Priority, and QCI Values

This corresponds to the mapping provided in TODO REF which are:

Value			Use
1	GBR	2	Interactive Voice
2	GBR	4	Interactive Video
3	GBR	5	Non-Interactive Video
4	GBR	3	Real Time Gaming
5	Non-BG	R 1	IMS Signaling
6	Non-BG	R 7	interactive Voice, video, games
7-9	Non-BG	R 6	non interactive video / TCP web, email, / Platinum vs gold user

Table 3 - Media Type, Priority, and QCI Mapping (LTE)

7. Wi-Fi Mappings

Media Type	Low	Medium	High
Audio	6	6	6
Interactive Video	5	5	5
Non-Interactive Video	4	4	4
Data	1	0	3

Table 4 - Media Type, Priority, and 802.1d (UP) Values

This corresponds to the mappings from TODO REF of:

Value		Traffic Type	Access Category (AC)	Designation
1	BK	Background	AC_BK	Background
2	-	(spare)	AC_BK	Background
0	BE	Best Effort	AC_BE	Best Effort
3	EE	Excellent Effort	AC_BE	Best Effort
4	CL	Controlled Load	AC_VI	Video
5	VI	Video	AC_VI	Video
6	VO	Voice	AC_VO	Voice
7	NC	Network Control	AC_VO	Voice

Table 5 - Wi-Fi Priority (UP) / Access Category Mapping

8. W3C API Implications

To work with this proposal, the W3C specification SHOULD provide a way to specify the importance of media and data streams.

The W3C API SHOULD also provide a way for the application to find out the source and destination IP and ports of any flow as well as the DSCP value or other markings in use for that flow. The JavaScript application can then communicate this to a web service that may install a particular policy for that flow.

The W3C API SHOULD NOT provide a way for the JavaScript to arbitrarily set the marking to any value of the JavaScript choosing as this reduces the security provided by the browser knowing the media type.

9. Security Considerations

TODO - discuss implications of what browser can set and what JavaScript can set

10. IANA Considerations

This specification does not require any actions from IANA.

11. Downward References

This specification contains a downwards reference to [RFC4594]. However, the parts of that RFC used by this specification are sufficiently stable for this downward reference.

12. Document History

Note to RFC Editor: Please remove this section.

This document was originally an individual submission in RTCWeb WG. The RTCWeb working group selected it to be become a WG document. Later the transport ADs requested that this be moved to the TSVWG WG as that seemed to be a better match. This document is now being submitted as individual submission to the TSVWG with the hope that WG will select it as a WG draft and move it forward to an RFC.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, August 2006.

13.2. Informative References

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, June 1999.

[RFC3246] Davie, B., Charny, A., Bennet, J., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, March 2002.

[I-D.ietf-rtcweb-rtp-usage] Perkins, C., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", draft-ietf-rtcweb-rtp-usage-05 (work in progress), October 2012.

14. Acknowledgements

Cullen Jennings was one of the authors of this text in the original individual submission but was unceremoniously kicked off by the chairs when it became a WG version. Thanks for hints on code to do

this from Paolo Severini, Jim Hasselbrook, Joe Marcus, and Erik Nordmark.

This document was prepared using 2-Word-v2.0.template.dot.

Appendix A. Code Hints

On windows setting the source interface works but BSD, OSX, Linux use weak end-system model and will route out different interface if that looks like a better route. (TODO - Can someone verify this with specific versions?)

In windows you might be able to tell something about priority of an interface for ICE purposes with WlanQueryInterface or GetIfTable.

The specific mechanisms required to set DSCP code points depend on the application platform.

In windows, setting the DSCP is not easy. See Knowledge Base Article KB248611. TODO - add more information about what can be done for windows.

For most Unix variants, the following program can set DSCP.

TODO - make this work in V6. For v6 have a look at IPv6_TCLASS or better the tclass part of sin6_flowid for IPv6

TODO - Can someone test and report back results of program in iOS, Android, Linux, OSX, BSD.

Example test program:

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>

#define MSG "Hello, World!"

int main(void) {
    int sock = -1;
    struct sockaddr *local_addr = NULL;
    struct sockaddr_in sockin, host;
    int tos = 0x60; /* CS3 */
    socklen_t socksiz = 0;
    char *buffer = NULL;
    sock = socket(AF_INET, SOCK_DGRAM, 0);
    if (sock < 0) {
```

```
            fprintf(stderr,"Error: %s\n", strerror(errno));
            exit(-1);
        }
        memset(&sockin, 0, sizeof(sockin));
        sockin.sin_family = PF_INET;
        sockin.sin_addr.s_addr = inet_addr("11.1.1.1");
        socksiz = sizeof(sockin);

        local_addr = (struct sockaddr *) &sockin;

        /* Set ToS/DSCP */
        if (setsockopt(sock, IPPROTO_IP, IP_TOS,  &tos,
                       sizeof(tos)) < 0) {
           fprintf(stderr,"Error setting TOS: %s\n", strerror(errno));
        }

        /* Bind to a specific local address */
        if (bind(sock, local_addr, socksiz) < 0) {
           fprintf(stderr,
                   "Error binding to socket: %s\n", strerror(errno));
           close(sock); sock=-1;
           exit(-1);
        }

        buffer = (char *) malloc(strlen(MSG) + 1);
        if ( buffer == NULL ) {
            fprintf(stderr,
                    "Error allocating memory: %s\n", strerror(errno));
            close( sock ); sock=-1;
            exit(-1);
        }
        strlcpy(buffer, MSG, strlen(MSG) + 1);
        memset(&host, 0, sizeof(host));
        host.sin_family = PF_INET;
        host.sin_addr.s_addr = inet_addr("10.1.1.1");
        host.sin_port = htons(12345);

        if (sendto(sock, buffer, strlen(buffer), 0,
                   (struct sockaddr *) &host, sizeof(host)) < 0) {
            fprintf(stderr,
                    "Error sending message: %s\n", strerror(errno));
            close(sock); sock=-1;
            free(buffer); buffer=NULL;
            exit(-1);
        }

        free(buffer);
        buffer=NULL;
```

```
       close(sock);
```

```
        sock=-1;

        return 0;
    }
```

Authors' Addresses

Subha Dhesikan
Cisco Systems, Inc.
170 W Tasman St
San Jose, CA 95134
USA

Email: sdhesika@cisco.com

Dan Druta
AT&T

Email: dd5826@att.com

Paul E. Jones
Cisco Systems, Inc.
7025 Kit Creek Rd
Research Triangle Park, NC 27709
USA

Email: paulej@packetizer.com

James Polk
Cisco Systems, Inc.
3913 Treemont Circle
Colleyville, Texas 75082
USA

Email: jmpolk@cisco.com