

RFC 10055 : Configuring UDP Sockets for Explicit Congestion Notification (ECN) for Common Platforms

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 16 octobre 2026

Date de publication du RFC : Octobre 2026

<https://www.bortzmeyer.org/10055.html>

Un RFC un peu inhabituel que voici. Il ne spécifie pas un protocole ou un format, il explique simplement comment activer la notification de congestion (ECN) sur les systèmes d'exploitation courants.

Cette notification (ECN = "*Explicit Congestion Notification*") est normalisée dans le RFC 3168¹ et permet de signaler en aval de la machine qui met l'ECN qu'elle voit de la congestion. Elle est surtout utilisée avec TCP (où, de toute façon, c'est le noyau qui s'en occupe) mais la normalisation de QUIC (RFC 9000) a suscité davantage d'intérêt pour son utilisation avec UDP, où l'application doit gérer cela elle-même. Apparemment, un certain nombre de programmeuses ne savent pas bien comment mettre de l'ECN sur UDP donc l'auteur de ce RFC a testé pour vous, dans le cadre du développement de Chromium, sur Linux, FreeBSD, MacOS et Windows (par la suite, je vais surtout parler de Linux car c'est ce que j'ai testé; voyez le RFC pour les autres systèmes). Le code de Chromium est en ligne <https://source.chromium.org/chromium/chromium/src/+/main:net/socket/udp_socket_posix.cc>.

Si vous avez la flemme de relire le RFC 3168, il définit deux bits pour l'ECN, qui se trouvent après le champ DSCP ("*Differentiated Services Code Point*", RFC 2474, l'octet qui les contient était appelé ToS - "*Type of Service*" - au millénaire précédent) de l'en-tête IP. Si ces deux bits sont à 1, c'est que de la congestion a été vue. (Les options UDP du RFC 9868 sont autre chose, et ne sont pas dans la même couche.)

Outre l'UDP utilisé seul, ce RFC est une bonne lecture si vous mettez en œuvre QUIC (RFC 9000), ou SCTP sur UDP (RFC 9260). Pour ce dernier cas, il semble que ça n'ait pas encore été utilisé. Il y a aussi RTP sur UDP (RFC 6679 et RFC 8888).

Ce RFC ne développe pas de nouvelles API, il explique juste ce qu'on peut faire avec les API actuelles, en C et C++.

Donc, d'abord, si vous voulez **recevoir** les notifications (section 3 du RFC), vous devez utiliser `setsockopt` avec les options `IP_RECVTOS` (en IPv4) ou `IP_RECVTCLASS` (en IPv6) :

1. Pour voir le RFC de numéro NNN, <https://www.ietf.org/rfc/rfcNNN.txt>, par exemple <https://www.ietf.org/rfc/rfc3168.txt>

```
result =  
    setsockopt(socketfd, IPPROTO_IPV6, IPV6_RECVTCLASS, &enable, sizeof(enable));
```

Il faudra ensuite lire le paquet avec `recvmsg` et utiliser les messages auxiliaires avec `cmsg` (type de message `IPV6_TCLASS`):

```
numbytes = recvmsg(socketfd, &mh, 0);  
for (struct cmsghdr * cmsg = CMSG_FIRSTHDR(&mh);  
     cmsg != NULL; cmsg = CMSG_NXTHDR(&mh, cmsg)) {  
    if (cmsg->cmsg_level != IPPROTO_IPV6 || cmsg->cmsg_type != IPV6_TCLASS) {  
        continue;  
    }  
    tclass = (int *) CMSG_DATA(cmsg);
```

(Dans le monde Apple, il faudra plutôt lire leur documentation <<https://developer.apple.com/documentation/network/nwprotocolip/metadata>>.)

Et pour **envoyer** de l'ECN? La section 4 du RFC nous dit qu'il faut utiliser `setsockopt` pour indiquer la valeur (apparemment, il faut écrire l'octet complet, DSCP et ECN):

```
result = setsockopt  
    (sd, IPPROTO_IPV6, IPV6_TCLASS, &dscp_value, sizeof(dscp_value))
```

Si vous voulez le code complet, je l'ai mis dans les programmes `et` (uniquement IPv6). Ils ont évidemment été écrits en lisant le livre de Stevens <<https://www.bortzmeyer.org/unix-network-programming.html>> et le RFC 3542. Compilons-les et testons-les :

```
% make udp-ecn-receiver  
cc      udp-ecn-receiver.c  -o udp-ecn-receiver  
% ./udp-ecn-receiver
```

et dans une autre fenêtre :

```
% make udp-ecn-sender  
cc      udp-ecn-sender.c  -o udp-ecn-sender  
% ./udp-ecn-sender --verbose ::1  
Connecting to ::1 port 4242...  
Sending 11 bytes with DSCP 3...
```

et on voit dans la première fenêtre :

```
Message received, 11 bytes.  
Received DSCP 3
```

Analysé avec tshark, cette communication montre :

```
Frame 1: 73 bytes on wire (584 bits), 73 bytes captured (584 bits)
...
Internet Protocol Version 6, Src: ::1, Dst: ::1
...
    .... 0000 0011 .... = Traffic Class: 0x03 (DSCP: CS0, ECN: CE)
        .... 0000 00.. .... = Differentiated Services Codepoint: Default (0)
            .... ..11 .... = Explicit Congestion Notification: Congestion Experienced (3)
```

On voit bien que l'expéditeur a dit qu'il voyait de la congestion. Vous pouvez jouer avec l'option `--ecn` de l'expéditeur pour essayer d'autres valeurs que 3 (les deux bits ECN à 1).

Ces programmes devraient (?) fonctionner sur FreeBSD (mais je n'ai pas testé). Si vous les portez sur Rust ou Zig, le résultat m'intéresse.