POC HTTP band limitation

A technical description

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Comments |
| March, 19th 2018 | 1.0 | Zardoni, Matteo[[1]](#footnote-1) | First draft |
| March, 20th 2018 | 1.1 | Zardoni, Matteo[[2]](#footnote-2)1 | Added subtitle |

The POC we have realized consists of a CentOS Virtual Machine having installed an instance of nginx server.

This instance of nginx server, version 1.12.2, has been built by using gcc 4.8.5 20150623 (Red Hat 4.8.5-16) (GCC), including the following modules:

1. lua-nginx-module-0.10.12rc2 and ngx\_devel\_kit-0.3.1rc1, which are applied in order to execute LUA source code within the nginx configuration file;
2. Nginx-limit-traffic-rate-module, which is applied in order to limit the rate of download requests. Since this module has been taken from a Git repository, the branch “master” has been included.

nginx version: nginx/1.12.2

built by gcc 4.8.5 20150623 (Red Hat 4.8.5-16) (GCC)

configure arguments: --add-module=../lua-nginx-module-0.10.12rc2 --add-module=../Nginx-limit-traffic-rate-module.master --add-module=../ngx\_devel\_kit-0.3.1rc1 --add-module=../echo-nginx-module --prefix=/opt/nginx --with-debug

The current implementation receives POST requests in order to set throttling rate values related to ip addresses. The command is the one which follows:

curl -d ipaddr=185.26.140.59 -d bps=100k http://35.205.84.208/limit/config

where the “bps” field specifies the throttling rate value that should be applied to the address specified in the “ipaddr” field. In addition, a web interface reachable at <http://35.205.84.208> has been developed, which permits to send the POST request without using the command above.

A portion of the nginx configuration file containing LUA source code.

location /limit/config {

if ($request\_method != POST) {

return 405;

}

content\_by\_lua\_block {

local string\_format = string.format

ngx.req.read\_body()

local args = ngx.req.get\_body\_data() or ""

local ip\_address = string.match(args, "%d+.%d+.%d+.%d+")

local bps\_speed = string.match(args, "%d+k")

ngx.say("Storing values to custom configuration file...")

             local file, err = io.open("/opt/nginx/../"..ip\_address..".dat", "w")

             if file == nil

then

ngx.say(err);

else

file:write(bps\_speed)

  file:close()

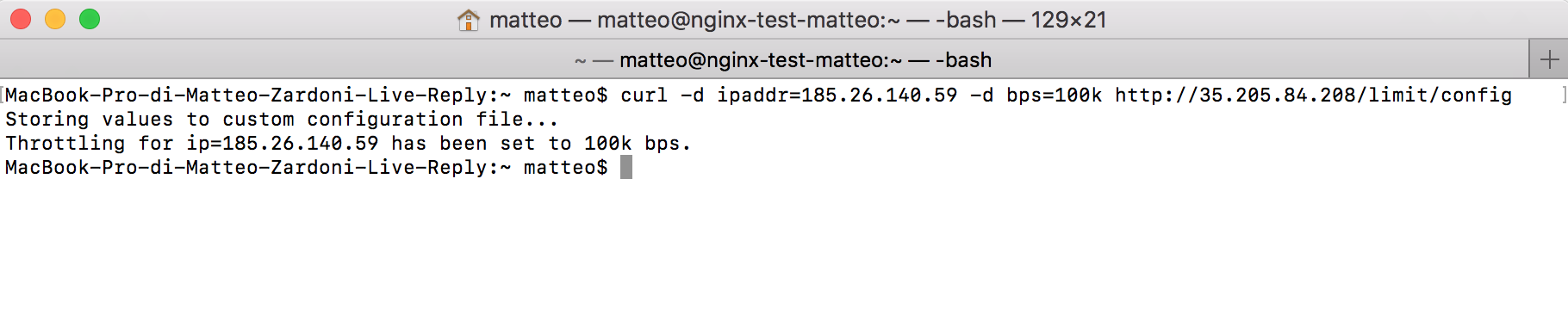
ngx.print("Throttling for ip=", ip\_address, " has been set to ", bps\_speed, " bps. \n")

                    end

        }

}

Once received this POST request, the nginx server instance defines a file having the extension “.dat”, which is named through the IP address. The “.dat” file contains the bps value that has been specified. For instance, considering the example command shown before, the nginx server creates a “185.26.140.59.dat” file, consisting of a unique row composed by “100k”. Everytime a new POST request arrives, a new file is created. In other words, if for a particular ip address the throttling rate has been previously set, this rate is overwritten by the new one just arrived. *This implementation has been realized by injecting LUA source code within the nginx configuration file, as shown in the previous text box*.



The result obtained by sending the POST request

In order to read the “.dat” created, the module b) has been modified by defining a new function which sets the throttling rate value within the module. In this manner it is possible to set the value *both* for new incoming requests and requests already opened. The command useful to download a resource is the following:

curl -X GET http://35.205.84.208/filter/RM/live/ABR/bt/sample\_24.mp4   
-O

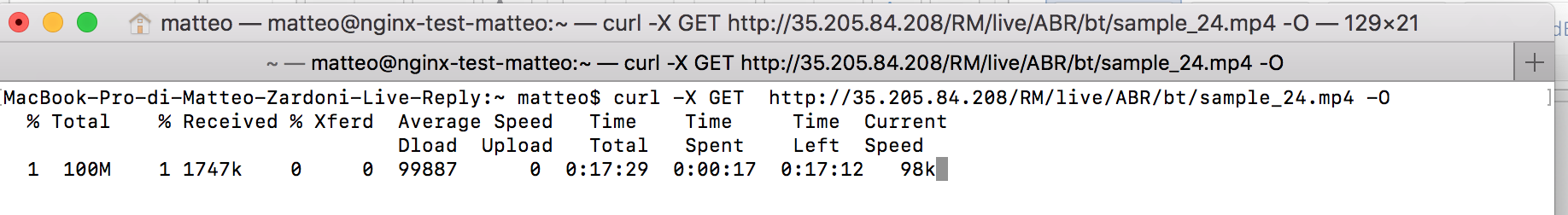
In the command above we set a URL referring to the following one

[*http://vod-soip.pcdn.skycdn.it/RM/live/ABR/bt/sample\_24.mp4*](http://vod-soip.pcdn.skycdn.it/RM/live/ABR/bt/sample_24.mp4).   
Hence this structure is followed:

[http://35.205.84.208/filter**/x/y/z**](http://35.205.84.208/filter/x/y/z) is mapped to [http://mydomain.com**/x/y/z**](http://mydomain.com/x/y/z) .

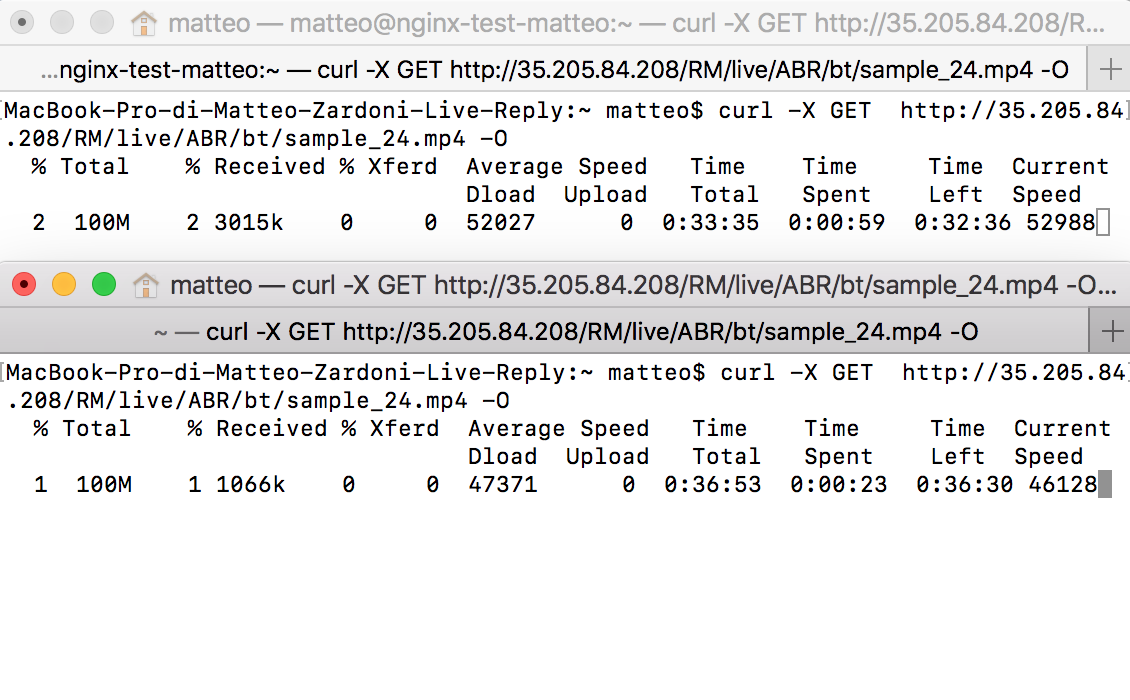
Our server receives the GET request, it sends it to the original URL and receives back the response. In this manner, it is possible to download the “sample\_24.mp4” resource by considering the throttling rate value specified before.

The result of the application of the command is shown in the picture below.



The result obtained by sending the GET request

If more than one connection is open, the speed is divided by the number of connections opened.



Since 2 connections are open, the 100k rate value is divided properly.

If no throttling rate value is set, the throttling rate value considered by the module b) is 50000k.

1. [ma.zardoni@reply.it](mailto:ma.zardoni@reply.it) [↑](#footnote-ref-1)
2. [↑](#footnote-ref-2)