Accelerating downloads with Trickle

TRAFFIC CONTROL

Is your Internet connection groaning under the load of too many simultaneous downloads? If so, try Trickle, a simple application that gives you more granular control over network traffic.

BY OLIVER FROMMEL

High-volume downloads can make simultaneous interactive connections practically unusable. Just like in the bad old days of modems, users press keys and then wait for seconds for the server to finally echo the character on screen.

TCP/IP needs enough free capacity in both directions. No matter whether you are uploading or downloading, packets still travel in both directions. The device at the receiving end of the download is expected to acknowledge the receipt of every packet it receives, and obviously the receipt travels in the upload direction towards the source.

Traffic Shaping made easy

To use the bandwidth of a broadband connection more efficiently, it is important to have more granular control over the packets entering or leaving your machine. Linux has queuing management facilities to handle this, and they provide a wide variety of traffic shaping approaches based on various algorithms. Traffic shaping is a technique for controlling incoming and outgoing network traffic to make optimum use of line capacity. Needless to say, the system is as complicated as it sounds (see [1].) The Trickle tool [2] solves this problem by giving users a simple command with just a few options. Trickle is normally quite easy to build; it requires only Libevent [3] and a few other standard libraries.

Non-Privileged

Trickle does not even need administrative privileges to control traffic. Normally, Glibc provides applications with a number of network functions that allow them to establish Internet connections, transfer files, and so on. When Trickle launches, it uses the *LD_PRELOAD* environment variable to load its own dynamic library, which includes network functions. They work just like the originals, but additionally record traffic flows. Trickle works with most programs, but there are a few exceptions. If you have difficulty getting Trickle to work with your favorite application, you might like to check the notes in the "Restrictions" box.

Some applications do not use the mechanism by default. To subject a program to bandwidth control, you need to launch the program with the *trickle* command. The *-d* switch lets you specify a limit for the download rate:

```
trickle -d 50 ↵
ftp ftp.redhat.com
```

This tells Trickle to ensure that the download stream *FTP* connection will not consume more than 50 Kbps of bandwidth on average. The *-u* option does the same for the upload stream. Other FTP programs running at the same

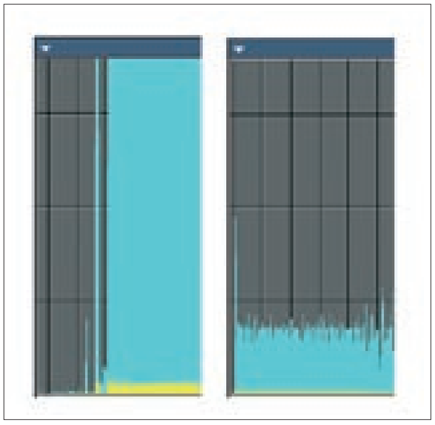
Figure 1: Without Trickle control, the FTP download consumes the total available bandwidth. Setting a limit keeps enough capacity for other network applications.

time are not affected by Trickle and could continue to hog the wire. To provide more sensible controls, you need to allow trickle to control any programs that send or receive Internet data. The Trickle daemon, *trickled*, takes care of this by logging all Trickle connections and controlling the total bandwidth consumption.

Global Controls

Again, the daemon supports the two simple parameters that we just looked at, *-d* and *-u*, but in this case they apply to the total bandwidth:

```
trickled -d 50 -u 5 -f -N 5
```

This command sets the download rate to 50 Kbps and the upload rate to 5 Kbps. The *-f* parameter tells the daemon to run in the foreground. *-N 5* tells the daemon to output statistics every 5 seconds. If you omit the *-d* parameter, Trickle will set the upload and download rates to 10 Kbps. You can store permanent settings in the */etc/trickled.conf* configuration file, or set the *-c* flag and specify a filename.

Trickle programs talk to the daemon when launched (normally via a **Unix socket** called */tmp/.trickled*.sock), and control the client programs based on your rules. If the server connection is interrupted, Trickle keeps on running, but it will not honor the set limits. If the server is running with a download limit of 10 Kbps, entering *trickle wget http://www.w3.org* will keep to the limit. But if you quit the server while the download is running, the download rate will increase.

Smoothing

You can use the */etc/trickled.conf* configuration file to define priorities for individual services. Lower values represent higher priorities. Internally, Trickle manipulates the packet order in a queue. This approach means that you can achieve fairly good download speeds and still use an interactive *ssh* session. A simple example of a configuration file follows:

```
[ssh]
Priority = 1
[www]
Priority = 8
```

The documentation also recommends the *Time-Smoothing* and *Length-Smoothing* parameters to avoid transfer rate fluctuation. In contrast to the command line options, *-d* and *-l*, you can set the parameters in the configuration file individually for each service (SSH, FTP, WWW, ...):

```
[ssh]
Priority = 1
Time-Smoothing = 0.1
Length-Smoothing = 2
[www]
Priority = 8
Time-Smoothing = 5
Length-Smoothing = 20
```

These values define the time and length normalization that Trickle applies to a program that it is controlling. Larger values are recommended for high-volume transfers, whereas interactive applications should use smaller values. Neither of these values had much effect in our tests, but you might like to do some experimenting of your own.

Limits

There are limits to what a system of this kind can do. No matter whether it runs in kernel or user space, buffers – which are bound to hold a few packets – and the dynamic nature of today's networks set natural boundaries for traffic shaping. Trickle calculates the rates over a specific period of time. As the rate starts with a low value and increases over time, the tool might overshoot the mark and climb way above the value you are aiming for.

-w specifies the number of bytes in which Trickle should attempt to avoid burstiness, although it must be said that this setting had very little effect in our testing. The difference to the default setting of 512 bytes was hardly noticeable. This said, Trickle still does a good job without fine tuning.

Good for Home Use

Trickle is mainly designed to give individual users more control over the traffic flows generated or received by individual applications. Multi-user use is conceivable, but it would involve writing wrapper scripts as a catch-all for the applications involved, so a QDisc-based solution might be preferable (see [1]). Trickle also relies on user cooperation: each program has to be launched via the *trickle* binary.

Trickle can only handle TCP stream connections, which are typical of most high-volume services. Most people will be able to live with the fact that Trickle can't handle DNS traffic, for example. The Trickle traffic shaping tool is definitely a good solution for readers who need a spontaneous and simple approach for maintaining an interactive session while transferring high volumes of data. ■

Restrictions

Because the *LD_PRELOAD* mechanism uses dynamic libraries, Trickle will not work with so-called static binaries that do not use Glibc. The *ldd* command will tell you what kind of program you are asking Trickle to control; at the same time, *ldd* gives you a list of dynamic libraries:

```
ldd /usr/bin/wget | grep libc.so
libc.so.6 => ➘
/lib/libc.so.6 (0x00add000)
```

If *ldd* does not give you any output, you can assume that the test candidate is a static program. You can verify this by giving the *file /usr/bin/wget* command.

Additionally, Trickle can't handle SUID programs, which disable the *LD_PRELOAD* mechanism for security reasons.

THE AUTHOR

For several years Oliver was a sysop and programmer at Ars Electronica Center in Linz/Austria.

After finishing his studies in Philosophy, Linguistics and Computer Science he became an editor for the Bavarian Broadcasting Corporation. Today he is head of the Editorial Competence Center for Software and Programming at Linux New Media AG.

INFO

[1] Advanced Routing and Traffic Control Howto: *http://lartc.org*

[2] Trickle: *http://monkey.org/~marius/pages/?page=trickle*

[3] Libevent: *http://www.monkey.org/~provos/libevent*