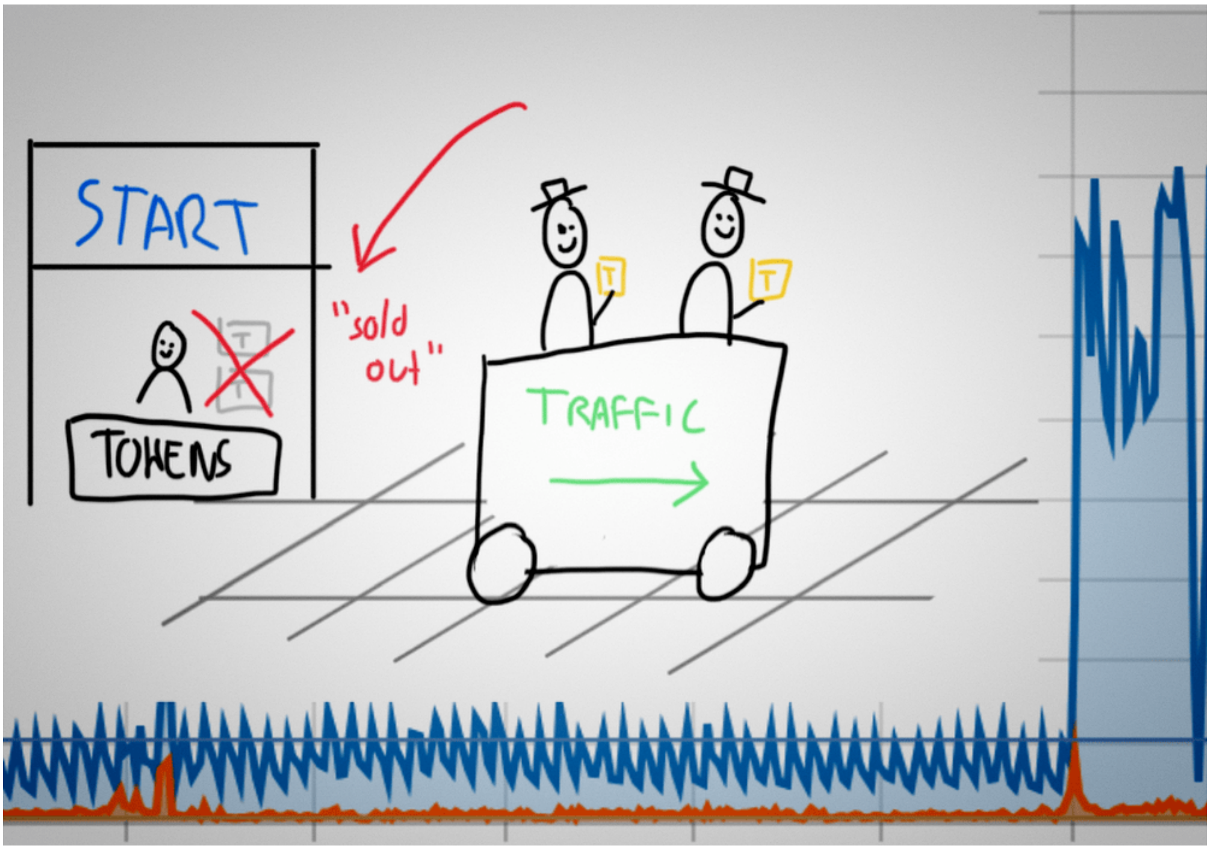
Traffic shaping using iptables and tc

Limiting outbound network bandwidth per client IP-address

Willem L. Middelkoop

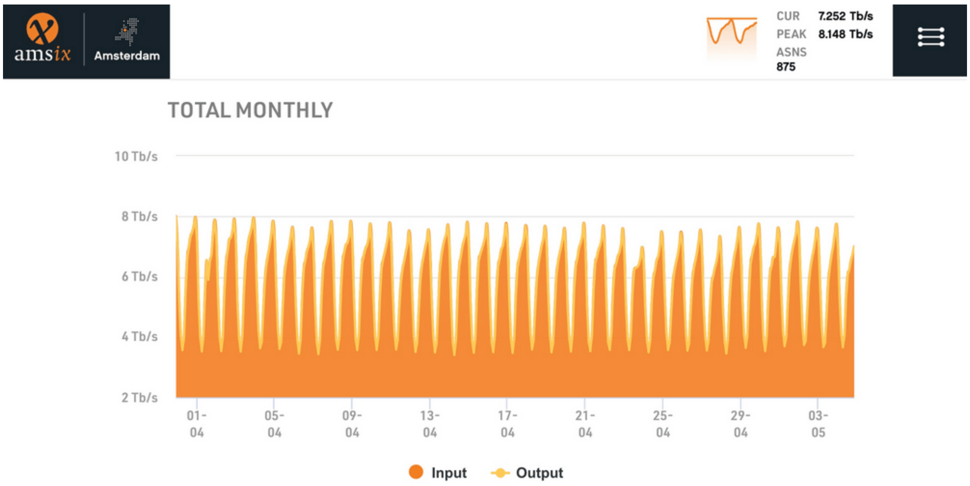
Apr. 1, 2020



Last month I received an automated alert indicating excessive bandwidth usage, usually a sign of trouble. When this happens, you should follow a standard incident procedure, trying to isolate the source of the traffic before shutting it down. The cause of this incident was not what I expected however... requiring a different kind of mitigation than a simple blockade.

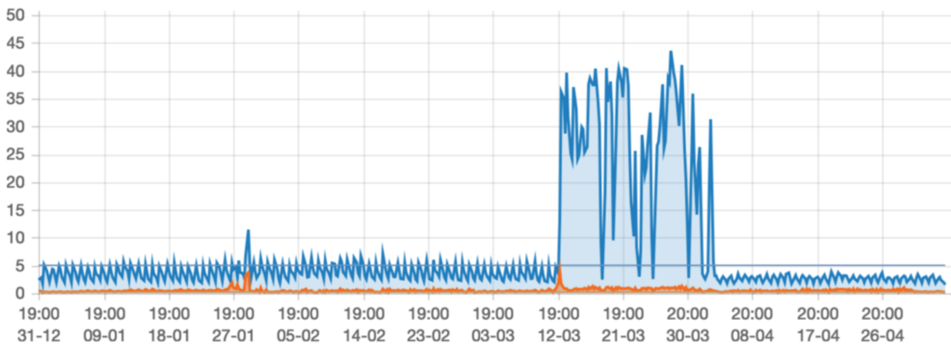
Excessive bandwidth alert

When you manage servers you'll notice that internet traffic usually occurs in predictable patterns. Just like real traffic on roads, there are regular times when it is busy and quite.



Bandwidth graph of AMS-IX shows a predictable pattern - notice the wave-like pattern

The predictable pattern makes it possible to detect anomalies automatically. Just like a traffic jam on a highway at an unusual time can be the result of an accident, unexpected swings in internet traffic can be an indicator for an incident in cyberspace.



Bandwidth graph with unusual spike indicating that something is wrong - you don't need to be Sherlock Holmes to find it

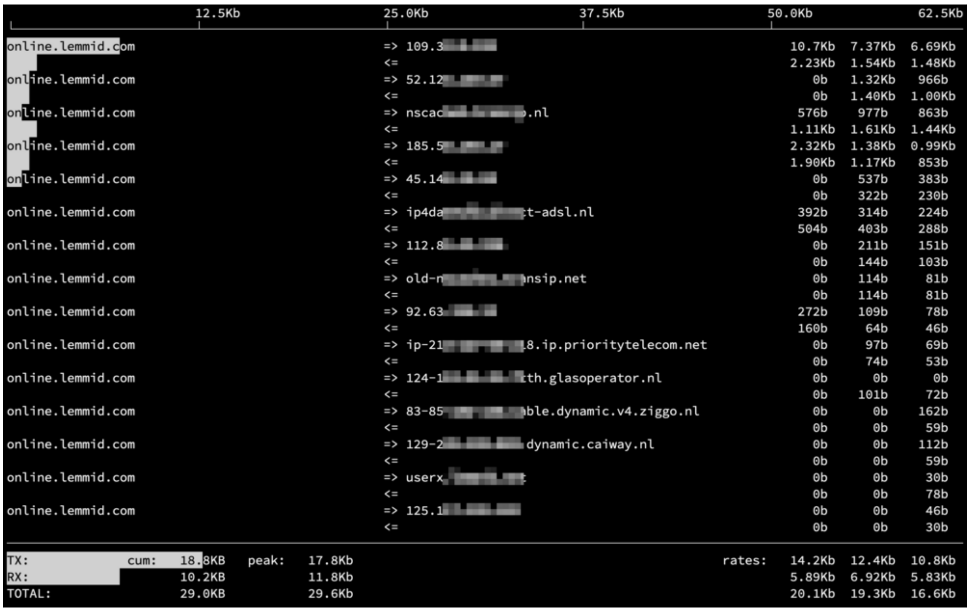
Isolating the source

The first thing you should do is to find the source of the problem. Do this by analysing the anomaly by:

- Determine if the extra traffic is inbound or outbound (up or download?)
- Determine the associated source and target IP addresses (what server is affected? Where is the traffic going to or coming from?)
- Determine the kind of traffic (email, web or something else?)

To do this you should inspect traffic charts, they usually indicate distinct input/output. Then you should try to find the affected IP addresses. This might involve looking at more (individual server) charts and statistics inside switches, routers and servers. Then look at CPU usage and log files to determine what application is affected, like mail or web. The larger the anomaly is, the easier it is to find.

You may be tempted to kill the anomaly once you have found it, immediately stopping the associated traffic. But you should really keep it alive (at least a little longer) to learn as much as you can from it.



Using the iftop tool to see bandwidth usage per connected IP-address

Use a tool like iftop to see a realtime overview of bandwidth usage per connected IP-address. It is very useful to get a grasp of what's going on, especially in combination with log files (linking IP-address to individual accounts).

Unusual cause

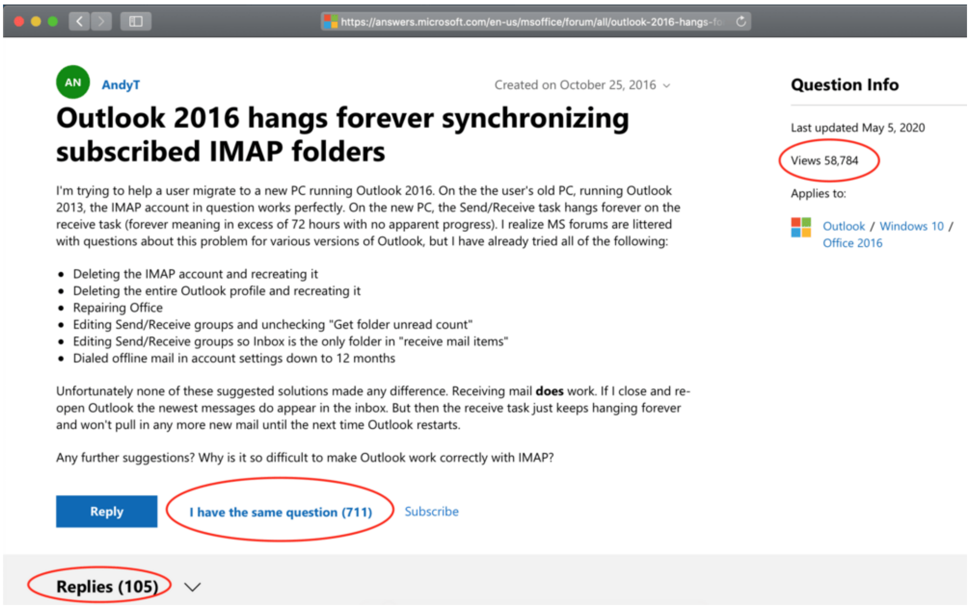
The server that produced the excessive bandwidth usage was a mail server. Often they are targeted by hackers to turn them into a spam relay server. This can happen in various ways, usually by the spammers capturing a valid login. This results in lots of outbound traffic as the spammers will push out many emails. If a spam run happens on your sever, you probably see this in the logs as spam messages frequently bounce, causing the mail queues to fill up quickly. It gets messy quickly, but on this server there was no such mess - just a lot of unusual bandwidth usage.

Mail (and spam) is sent using the SMTP protocol, but traffic on this server's SMTP port was very normal. This was very unusual as it meant that the excessive traffic came from something else. But what? After analysing various logfiles on the mail server I determined that the offending protocol was IMAP. Somehow a client connected to the mail server using IMAP was causing excessive amounts of network traffic. IMAP is a protocol that is used to *read* email, not to send it, making IMAP related anomalies very rare.

By analysing the mail server's log, I found the specific user that caused the traffic. I contacted the customer to ask if he noticed anything weird on his end. Not surprisingly, he noticed that his computer felt a little sluggish lately.

Microsoft Outlook synchronisation loop

After some trial and error by phone, we determined that something caused Microsoft Outlook to keep synchronising IMAP folders in a loop. This caused his computer to download the entire contents of his mailbox over and over again! As his mailbox was over 40 gigabyte in size, this caused the substantial traffic. Apparently this is caused by a bug in Microsoft Outlook, unfortunately there is no easy fix for it.



Bugs in Microsoft Outlook cause it to keep synchronising IMAP folders, a problem experienced by many people (see the number of views!)

Mitigating by traffic shaping

I faced the difficult decision to either block the (normal, legitimate, paying) customer or to allow the excessive traffic to continue (incurring serious costs to the network provider). Blocking legitimate users that operate their business using email is a very bad idea, but allowing excessive traffic to continue is bad too. Luckily I found an alternative way to reduce the excessive traffic while still allowing the customer access to his mail (using his trusted, but flawed Outlook app).

Traffic Shaping

As a bandwidth management technique you can use traffic shaping to delay some (or all) data packets to bring them into compliance with a desired traffic profile. If applied, you are - quite literally - shaping the traffic graphs, hence the name.

This technique is not without controversy as it is quite the opposite of the often hailed ”net neutrality” principle. With traffic shaping you can intentionally block or slow down (or charge extra money) for specific types of traffic. By principle I am against anything that hurts net neutrality, but for this particular situation I had no feasible alternative. I needed to seriously slow down email traffic for this particular customer, reducing the amount of bandwidth while continuing to provide access to his account.

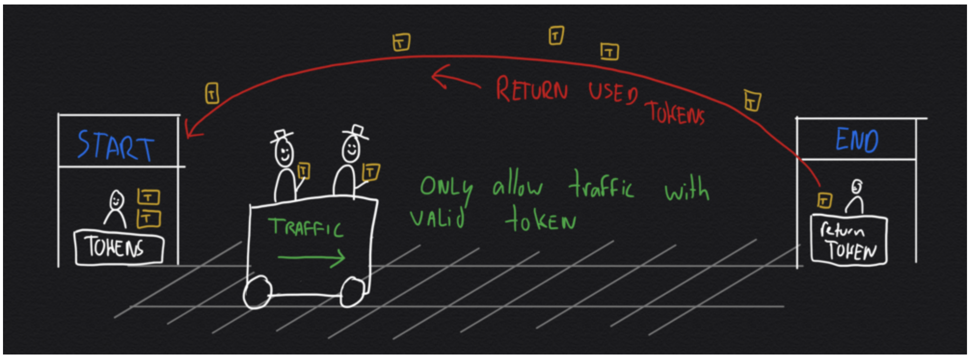
Implementing traffic shaping

To shape traffic you need to do two things:

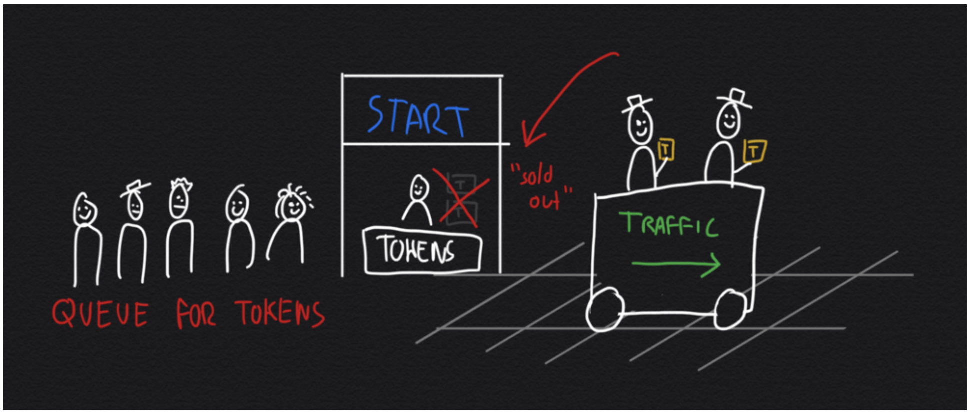
- **mark traffic**: You only want to affect particular network traffic, in this case IMAP access for a given client. Other traffic should be unaffected. This is done by marking packets that match particular characteristics, such as client IP or port numbers
- **enforce shape policy**: Using traffic control tools like 'tc' you then enforce the shaping policy on the marked traffic. There are different ways to do this, but a common one is to work with so-called "Hierarchy Token Buckets" or HTB's.

Hierarchy Token Buckets (HTB's)

In principle a token bucket is similar to the principle of limiting the amount of passengers in a train ride by only distributing a fixed number of available tokens. When passengers (or data packets) enter the train (or network) they take a token. When they disembark (or reach their destination) the token is returned. Using the token bucket principle you can control the amount of concurrent traffic in a system.

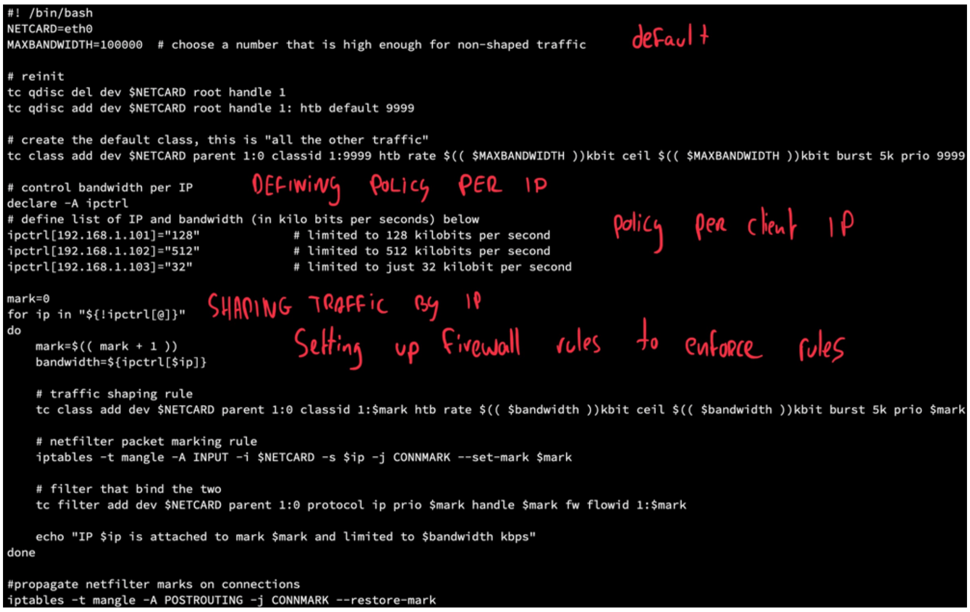


Using tokens to control traffic - only passengers (or data packets) with a valid token are allowed. Tokens are returned as traffic reaches its destination.



Traffic must wait for tokens to become available when the maximum number of tokens is given away, enforcing the maximum concurrent traffic

You can implement this principle using the network tools "tc" (for traffic control) in combination with "iptables". You can use a script to set the rules for marking and enforcing. You can find various samples online, I used this one by Julien Vehent.



```
#! /bin/bash
NETCARD=eth0
MAXBANDWIDTH=100000  # choose a number that is high enough for non-shaped traffic

# reinit
tc qdisc del dev $NETCARD root handle 1
tc qdisc add dev $NETCARD root handle 1: htb default 9999

# create the default class, this is "all the other traffic"
tc class add dev $NETCARD parent 1:0 classid 1:9999 htb rate $(( $MAXBANDWIDTH ))kbit ceil $(( $MAXBANDWIDTH ))kbit burst 5k prio 9999

# control bandwidth per IP
declare -A ipctrl
# define list of IP and bandwidth (in kilo bits per seconds) below
ipctrl[192.168.1.101]="128"            # limited to 128 kilobits per second
ipctrl[192.168.1.102]="512"            # limited to 512 kilobits per second
ipctrl[192.168.1.103]="32"             # limited to just 32 kilobit per second

mark=0
for ip in "${!ipctrl[@]}"
do
    mark=$(( mark + 1 ))
    bandwidth=${ipctrl[$ip]}

    # traffic shaping rule
    tc class add dev $NETCARD parent 1:0 classid 1:$mark htb rate $(( $bandwidth ))kbit ceil $(( $bandwidth ))kbit burst 5k prio $mark

    # netfilter packet marking rule
    iptables -t mangle -A INPUT -i $NETCARD -s $ip -j CONNMARK --set-mark $mark

    # filter that bind the two
    tc filter add dev $NETCARD parent 1:0 protocol ip prio $mark handle $mark fw flowid 1:$mark

    echo "IP $ip is attached to mark $mark and limited to $bandwidth kbps"
done

#propagate netfilter marks on connections
iptables -t mangle -A POSTROUTING -j CONNMARK --restore-mark
```


Sample traffic shaping script

Conclusion

Sometimes network anomalies are not what you expect them to be, therefore you should always take the time to investigate them! Blindly blocking traffic is blunt, sometimes you need to refine your methods to mitigate problems.

Applying unorthodox filtering techniques is not something that I like, but sometimes they are the only means to an end. Know what you're doing and why you're doing it is very important!