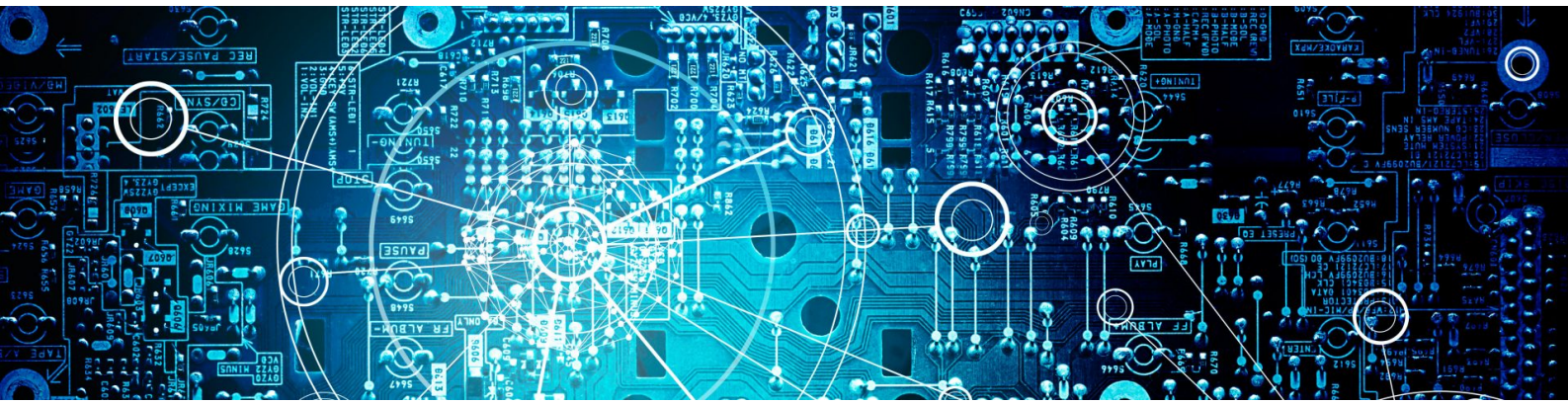

FIRST STEPS WITH ARISTA NETWORKS

January 20, 2014 Romain Aviolat Uncategorized 4 comments

Intro

We were looking for a fast layer 3 switch, I mean really fast. It should not only be able to switch frames at 10GbE line-rate but it should also be able to route packets (from one interface to another) at line-rate. With the smallest IP packet (84Bytes) it means that we can send it about 14'880'955 time per second to fill a 10GbE pipe.

And that's where it becomes a problem, most enterprise-class switches were able to route at line rate from a fixed source/destination (for example a "simple" file transfer) but where dying when we randomized the destination addresses and reduced the packets size to the minimum.

We heard about Arista few months ago, we knew that they were specialized into line-rate switching. We exchanged

few emails about our problems to find a switch that will fulfill our requirements, they kindly proposed to lend us a switch for one month. We were deeply impressed by the power of the switch we benchmarked, the features the provided, the openness of their products. I'll describe here the cool things I learned from them.

How I conducted the test

Basically one computer (10GbE card with one SFP+SR) into one VLAN behind the switch (10GbE port SFP+SR), generating packets with random destination addresses. Another computer behind the same switch (10GbE port SFP+SR) into another VLAN. The second computer (10GbE card with one SFP+SR) is the default gateway of the switch and the switch is the default gateway of the first computer. So when the first computer generates the packets it sends them to the switch that route them to the second computer, we have our setup.

[PC1 10GbE SFP+] <– om4 mm fiber –>
[VLAN1]SWITCH[VLAN2] <– om4 mm fiber –>
[PC2 10GbE SFP+]

Seems to be pretty simple on the paper but… Generating packets consume a lots of CPU if the packet needs to be constructed within the Linux networking stack, it will be hard to hit 1'000'000 pps with a decent Xeon processor. Same problem on the receiving machine, each time a packet is received it will generate an IRQ that will be processed by the kernel.

The solution is called pf_ring. It provides a way to talk directly to the networking card using a special driver. Of course standard userland applications wont work anymore. I used Masscan, it's an IP port scanner that can randomize destination IP/port using the pf_ring driver,

that's the one I used on the sending machine. On the receiving machine I used pf_ring built-in application called pf_count to count the number of received packets.

The result was impressive, the switch could handle the load without any problems, and as the routing / switching is made in hardware (using ASICs) the CPU was idling all the time.

Linux in the core

The core of Arista switches runs Linux, I mean it's not a stripped version with an interface on top of it, like most other embeded devices, but a full x86 operating system. Redhat is the distribution they choosed. By default when you first connect to the switch you are dropped to the CLI (command line interface), you can issue a simple "bash" command and you're dropped to a shell, then a simple "sudo" and you're root ! Isn't it cool ? As far as I know there is no limitation, you can tweak / install what you want on top of it.

EOS (Extensible Operating System)

Arista provides a CISCO like CLI, when I say "CISCO like" I mean it's 95% the same commands + a lots of improvements. EOS is on top of the Linux distribution and the link between the operating system and the CLI is very strong, you can easilly switch or combine commands from one to another. For example you can "grep" results from the CLI (**#show running-config | grep "your keyword"**) or redirect output to the OS (**#show interface 43 > /mount/flash/interface_43.out**).

EOS is "extensible", means that you can add new features to your devices. As it's a Redhat based OS the package management is made with RPMs. Arista provides a place

where developers can add their extensions called EOS central.

The CLI is written in python, if you want to add new features to the CLI then... go for it ! You can hack into the sources directly. It could be interesting for some usage to add a specific feature for the technician that access the switch if you don't want them to access it beeing root.

Last but not least the OS is the same for every switches. Download one time, apply many.

Designed for performances ?

Lots of compute work is made in hardware using ASICs to offload CPU usage and to improve latency. According to Arista and the book "Arista Warrior" from Gary A. Donahue (that I recommend, it's written by an idenpendant network engineer) switch backplanes (aka the "switch fabric") are made in a way where ports can't be oversubscribed. The term "non-blocking switch" is often used to describe those switches, it means that each port is able to send and receive traffic at line-rate to and from another interface. So If we have a 24 10GbE switch (The 7150S-24 for example) the fabric througput is about 240 Gbps (full duplex). On the previous paragraph I wrote that we could send about 15Mpps per second on a 10GbE pipe, so on a 24 ports switch it should be 15Mpps * 24 = 360Mpps, and guess what... That the number they give on the paper ! Believe me or not it was hard enought to hit the 15Mpps in a lab with "few" hardware so I'll believe them for now...

Arista doesn't develop their own ASICs like CISCO, they use for example the Broadcom Trident+ into some of their switches. Some of them also include an FPGA on dedicated ports, they are called "application switches" the end user can programm it to suit it's applications needs.

Another of their war horse is latency, they became famous in financial environments where a second could means millions. If I refer to the 7150S-24 documentation they give an impressive value of 350ns switching latency (from ingress to egress). Under the microsecond, wow.

cost per port

The cost per port seems to be cheaper than other competitors in the same product range, here's some personal assumption on how they achieve this:

– They don't put end-user features in their switches, like PoE or VoIP for example. It's designed for a data-center usage.
– They have a small product range
– Only one OS to maintain
– Merchant silicon, they don't develop their ASICs
– They don't add proprietary features into their switches

interesting features

When I started to read documentation about Arista I discovered some interesting features worth mentioning:

VARP (Virtual ARP): very simple gateway load-balancing mechanism
MLAG (MultiChassis Link Aggregation): link a port-channel to multiple switches instead of one. Two or more switches participates to the same mlag group and fools the switch at the other end of the port-channel who thinks that there is only one switch at the other end. Providing a simple high availability mechanism.
VM-Tracer (SDN feature): A hypervisor can dynamically allocate or remove VLANs on the switch.
LANZ (Latency Analyzer): A way to debug switch buffers at microsecond level

SysDB: logs and states / counters are stored into a in-memory database. If a process dies it can learn its last values from there instead of starting collecting them from scratch again.
Event handler: you configure triggers on your switch based on certain events, for example: “send me an email if interface 03 goes down”.

Conclusions

After playing for one month with one of their product I can say that I’m pleased to have discovered them. Arista seems to be an innovative company that not only tries to follow the leaders but add strong values to their products, that as far as I know nobody else offer.

references

Arista website: http://www.aristanetworks.com
Arista Warrior: from Gary A. Donahue ISBN: 978-1-449-31453-8