# VMware ESXi Networking Subsystem - HPSI Assignment #1

José Donato, donato@student.dei.uc.pt, 2016225043

***Abstract*—VMWare ESXi [1] is a bare metal hypervisor developed by VMWare to deploy, serve and manage virtual machines. Can be used to facilitate centralized management for data center applications. VMware ESXi can also provide different virtual networks for the virtual machines. This report is aimed at the networking subsystem of this hypervisor.**

## I. Introduction

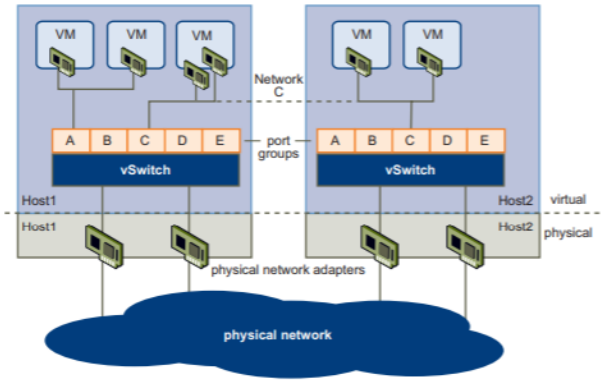
This report is divided into two sections. A first section where an overview of the VMware network subsystem is made including topics such as Networking Architecture, VLAN handling, Distributed vSwitches and Resource Control, more precisely, the Jumbo Frames. After the initial section, there is is a more practical section that tries to study the trustworthiness of the ESXi networking traffic shaping capabilities.

## II. Overview of the VMware Networking Subsystem

### A. Networking Architecture

VMware infrastructure provides a solution that makes networking the virtual machines as simple as in the physical environment [2] and enables new features like distributed vSwitches or Port Groups.

VMware makes virtual networking similar to physical networking, as we can see in the figure below:




On the ”virtual world”, we have virtual devices doing similar functions as the real ones. For example, we have virtual devices like virtual switches or virtual network interface controllers (vNIC). Each virtual machine has a vNIC as one physical computer has a physical NIC (also with a MAC address associated). Virtual Switches act a little bit different.

In the virtual environment, there is a new concept called Port Groups. It is a mechanism that allow us to set policies that govern the network. Each vSwitch can have multiple port groups and each port groups can have multiple vNICs associated. Instead of what happens in the physical environment (a network interface controller connects to a physical port of a switch), here the virtual NICs connect to one Port Group.

This has multiple advantages because in Port Groups we can configure multiple policies that provide more networking security, segmentation, performance, availability, etc. For example, we can set a policy to isolate compromised or malicious virtual machines preventing those from infecting other machines that are on the same network. Also, we can change traffic shaping policies to improve the management of the traffic in a network (used later in the practical section to calculate the trustworthiness of traffic shaping features).

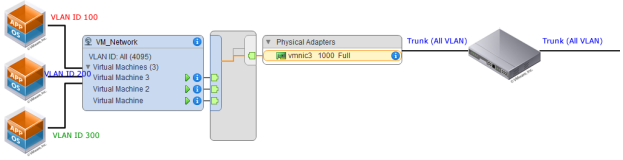
### B. VLAN handling

VLAN is a virtual local area network. It has the objective of handling a logical independent network. VMware ESXi recommends the implementation of VLAN because it guarantees security in network traffic, integrates the ESXi system in an existing network and reduces the network traffic congestion [3].

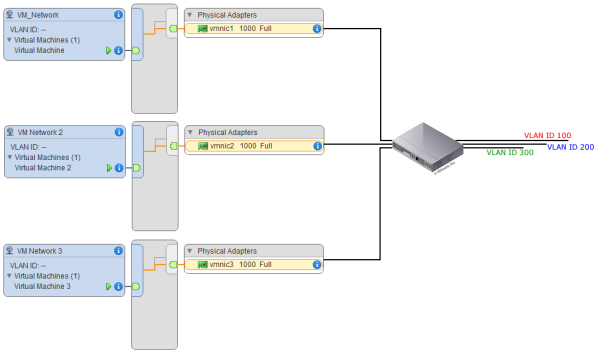
To use VLANs we need VLAN tagging. This is the process that helps identifying packets (also called Ethernet frames) travelling through links, i.e., to tell which packet belongs to which VLAN. To be more specific, VLAN Tagging puts an ID called VLAN ID into the header of the packets telling which network it belongs to. This way, it’s easier to tell where each packet should be sent to [4].

There are three methods of implementing VLAN Tagging on ESXI: Virtual Guest Tagging, External Switch Tagging and Virtual Switch Tagging [5]).

1) **Virtual Guest Tagging (VGT)**: The guest operating system (the OS running on the guest VMware) decides the association between the VLAN tags and its outbound traffic, i.e., the tagging is done at guest OS level. Guest OS inserts the VLAN ID into the frames before they leave the virtual machine virtual network interface controller. Since the PortGroups on the virtual switches will be carrying frames from any VLAN, they need to be configured with VLAN ID 4095.

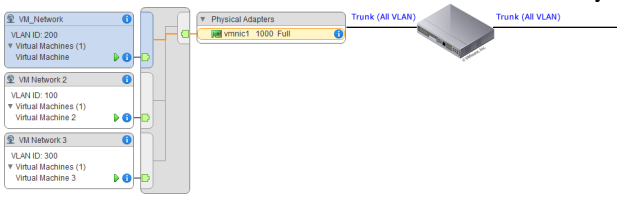



2) **External Switch Tagging (EST)**: Physical switch handles the VLAN tagging of the packets. Virtual switches are unaware of VLANs. Each physical network interface controller can only carry one VLAN each time. That is the reason why this way of tagging is only viable for small networks with low number of VLANs. VLAN is configured only at physical switch level.




3) **Virtual Switch Tagging (VST)**: It is the most common of tagging. Virtual switch performs the VLAN tagging before leaving the ESXi host. Although physical switch needs to be configured to carry more than one VLAN simultaneously, since PortGroups can tag frames on a specific

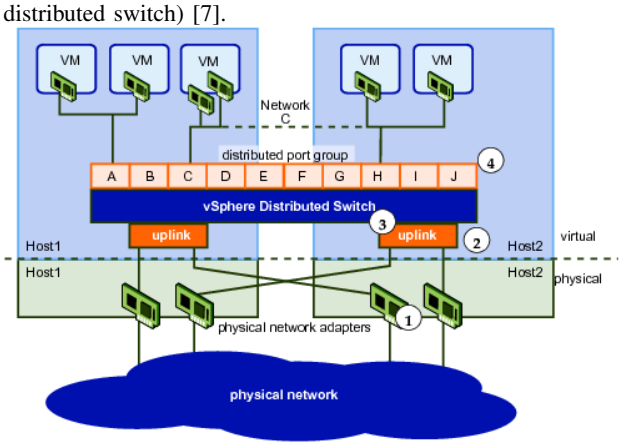
VLAN from 1 to 4094, the 1:1 problem (no more than one VLAN simultaneously) we have seen at EST vanishes and we can have more than one VLAN carried simultaneosly.




### C. Distributed vSwitches

Standard Switches are similar to a physical Ethernet switch in the way that the switches detect the relations between the multiple virtual machines and use this information to forward the traffic in the network in a single ESXi host.

On the other hand, a distributed switch provides this same functionality accross multiple ESXi hosts. It is a single virtual switch that allows the different ESXi hosts to use it as they exist in the same host [6]. This way, it is provided a centralized management and monitoring of the network configuration in a data center running multiple ESXi hosts (as long as this hosts are connected to the same distributed switch) [7].

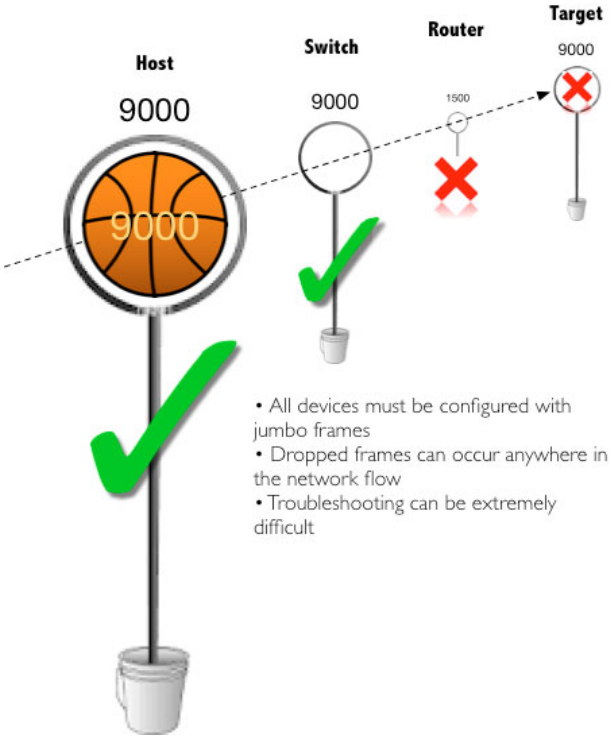



Because of distributed vSwitches, virtual machines network configurations remain consistent as they migrate across multiple hosts. This introduce another term called Networking vMotion that can only be associated to distributed switches. This term boils down to tracking a virtual machine networking state (as a machine moves across different hosts as told before).

### D. Resource Control

**Jumbo Frames**: are a specific type of Ethernet frames with a payload of 9000 bytes. They reduce the CPU load caused by transferring data and enhance network throughput (throughput is the measurement taken by unit of time between two devices: Kbps, Mbps, Gbps, etc).

In the VMware world, Jumbo Frames allow ESXi hosts to send larger packets to the physical network. Although every device in the route physical or virtual must have been configured to support packets with MTU (maximum transmission unit) of 9000 bytes otherwise the Jumbo Frames will be dropped or lost. It doesn't matter if the first and the last points of the route (my host and the other host vmkernel) support Jumbo Frames if the rest of the network has the default MTU of 1500 bytes. When the jumbo frame reaches that "unsupported" device (device that is not configured with MTU of 9000 bytes), the packet/frame will be dropped. In the figure below there is a good analogy [8] using basketballs and hoops. Even if the first and last hops are bigger than the ball (Host and Target), if one in the middle is smaller than the ball (Router), the ball will not pass and, consequently, will not reach the target.

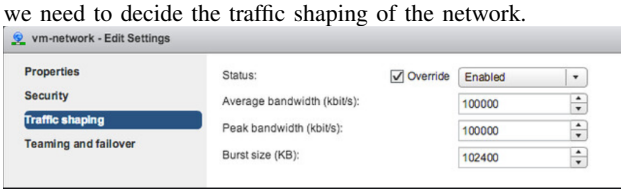



To conclude this topic, we can see that the implementation of Jumbo Frames in a VMware Networking Subsystem can bring great advantages such as enhancing the networking throughput and reduce the CPU load but its implementation must be planned and done carefully (every physical network adapter, switches, storage devices and virtual switches must be configured and support jumbo frames).

## III. Trustworthiness of ESXi Traffic Shaping Capabilities

### A. VLAN configuration on ESXi

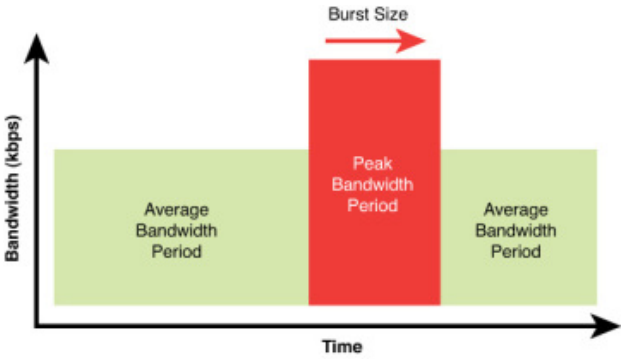
When we create a Port Group on ESXi, in addition to other settings, we need to decide the traffic shaping of the network.




As we can see on the figure above, we have four settings to fill. First, we need to decide if the network inherit the parent configurations. If not, we need to assign three values: Average bandwith, peak bandwith and burst size [9].

1) **Average bandwidth**: Measured in kbit/s, establishes the number of kilobits per second allowed for the switch to send. There are moments that this value can be exceeded.
2) **Peak bandwidth**: Measured also in kbit/s, establishes the maximum number of kilobits per second that a switch is allowed to let through. When the traffic volume is lower than the average limit, we gain a "burst bonus". This burst bonus can go up until it reaches the burst size limit (covered below).
3) **Burst size**: The maximum number of kilobytes to allow in a burst (measured in KB). Whenever a port needs more

bandwidth than the value specified by average bandwidth, it may be allowed to temporarily transmit data at a higher speed if a burst bonus is available. The burst bonus keep summing up until the burst size is reached.




To explain this better I will use a math example with real numbers [9] where the number of seconds that the traffic is peaking is calculated in a "best case" scenario. In this specific case:

- Average bandwidth is 1000 Kbps
- Peak bandwidth is two times this value, 2000 Kbps
- Burst size is 1000 KB, i.e., 8000 Kb

In the "best case" scenario, the bust bonus is fulfilled (8000Kb). This way, the traffic can peak, i.e., send traffic at a value specified in the Peak Bandwidth for 4 seconds because 8000Kb/2000Kbps = 4 seconds.

To perform the tests I will make different combinations of this three values (Average Bandwidth, Peak Bandwidth and Burst Size).

### B. Machines To Perform the tests

We have two different machines on the same network on the same ESXi host:

1) Machine A running Kali Linux operating system with IP 10.254.0.166

```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.254.0.166  netmask 255.255.255.0  broadcast 10.254.0.255
        inet6 fe80::20c:29ff:fe4a:1de5  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:4a:1d:e5  txqueuelen 1000  (Ethernet)
        RX packets 17371  bytes 1115172075 (1.0 GiB)
        RX errors 0  dropped 3  overruns 0  frame 0
        TX packets 14142  bytes 934304 (912.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2) Machine B running CentOS operating system with the IP 10.254.0.168

```
[josedonato@localhost ~]$ ifconfig
ens192: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.254.0.168  netmask 255.255.255.0  broadcast 10.254.0.255
        inet6 fe80::9402:7e5e:290d:1087  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:51:e5:df  txqueuelen 1000  (Ethernet)
        RX packets 8602  bytes 575261 (561.7 KiB)
        RX errors 0  dropped 74  overruns 0  frame 0
        TX packets 11586  bytes 749369097 (714.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

To show that both machines are on the same network, I used a command line tool named a nmap. The goal of this tool is to discover hosts and services running on a specific network by sending packets and analyzing the responses. The command I ran is presented below:

```
$ nmap -sn 10.254.0.0./24 > nmap_vmnetwork.txt
```

And then, another command to check if both IPs were in the output file.

```
$ cat nmap_network.txt
\\| grep -a1 -E '10.254.0.168|10.254.0.166'
```

It is presented below a figure with the output of the commands:

```
root@kali:~# nmap -sn 10.254.0.0/24 > nmap_vmnetwork.txt
root@kali:~# cat nmap_vmnetwork.txt | grep -a1 -E '10.254.0.168|10.254.0.166'
MAC Address: 00:0C:29:48:FF:7B (VMware)
Nmap scan report for 10.254.0.168
Host is up (0.00012s latency).
--
MAC Address: 80:EE:73:DB:3B:E6 (Shuttle)
Nmap scan report for 10.254.0.166
Host is up.
```

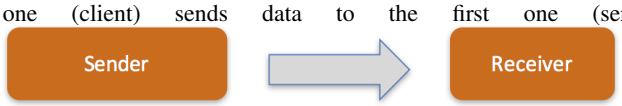
As we can see, since both machines are presented in the output of the nmap, is confirmed that they are in the same network. Since they are in the same network and they can communicate with each other (ping from one machine to another and vice-versa was done, everything is ready to start sending packets from one machine to other.

### C. iPerf3

iPerf3 is an open source network benchmarking tool with the goal of measuring the bandwidth and the quality of a network.

To use this tool we need to have one server and one client. We assume machine A is the server and the second machine B is the client.

Next, TCP and UDP connections can be made between this server and client where the last one (client) sends data to the first one (server).




To achieve the scenario presented in the figure above [10] we need to execute two commands.

1) We execute this command on the machine A to set it as a server/receiver listening to connections on port 5021 (-1 to serve one client and close):

```
$ iperf3 -s -p -1 5021 > output_result.txt
```

2) And in the machine B we execute the command to set it as client/sender and send data to the ip and port of machine A (in some cases parameter -t 50 was added to send traffic for 50 seconds instead of the default of 10 seconds):

```
$ iperf3 -c <ip_machineA> -p 5021
\\ > output_result.txt
```

### D. Cases (different traffic shaping configurations)

I simulated eight cases with different traffic shaping configurations of the network.

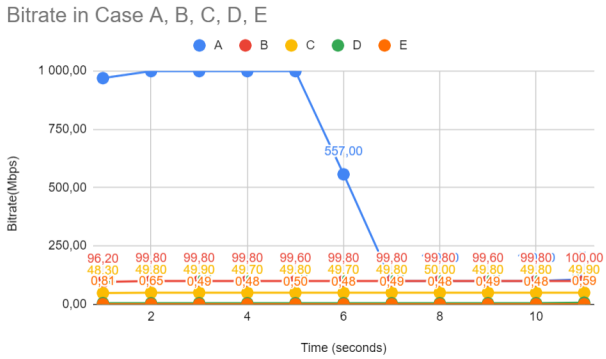
| Traffic Shaping | | | | |
|---|---|---|---|---|
| Case | Average Bandwidth (Kbps) | Peaking Bandwidth (Kbps) | Burst Size (KB) | Time (seconds) |
| A | 100000 | 1000000 | 1024000 | 10 |
| B | 100000 | 100000 | 110000 | 10 |
| C | 50000 | 50000 | 60000 | 10 |
| D | 5000 | 5000 | 6000 | 10 |
| E | 500 | 500 | 600 | 10 |
| A2 | 100000 | 1000000 | 1024000 | 50 |
| F | 1000000 | 1000000 | 1024000 | 50 |
| G | 1000000 | 2000000 | 1000000 | 50 |

With the first 5 cases (A, B, C, D and E) the goal was to see the bitrate of the transfer reduce when the average bandwidth keeps reducing. With the last two (F and G), the goal was to analyse the Peaking Bandwidth and Burst Size influence on the traffic throughput. A2 is the same as A but with a duration of 50 seconds.

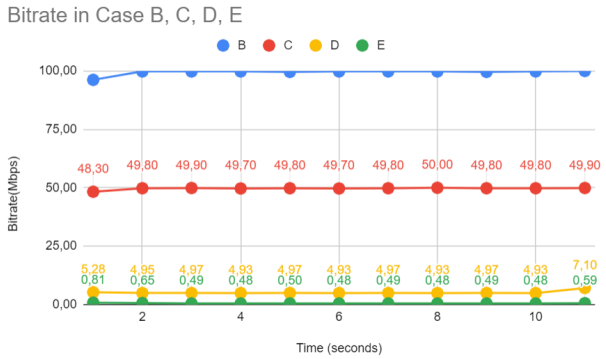
### E. Results

All the results can be seen in this link (https://docs.google.com/spreadsheets/d/1q65y6UC74LimvKgHqKunEGwwDtq-PP0rdf09fqWUNcI/edit?usp=sharing).

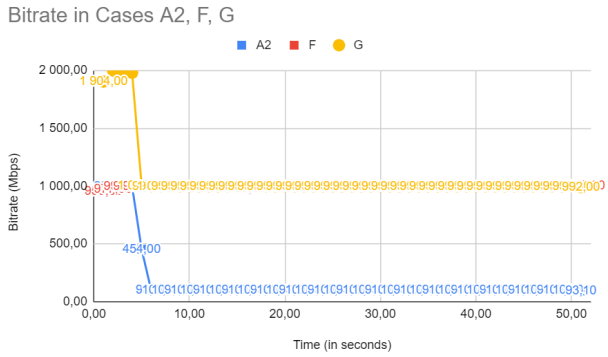
First I compared the first 5 cases. The result is presented below:



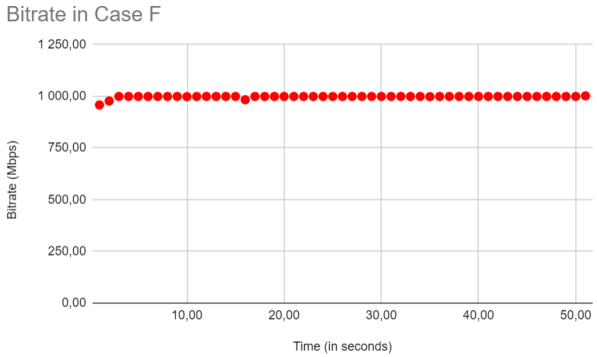

Because the case A has much higher values (making harder to analyse the results), I decided to make another graphic without case A, i.e., with B, C, D, E cases. The result is presented below:




Since A time was only 10 seconds, A2 was added to the cases with same traffic shaping as A but during 50 seconds to match cases F and G:




Because case F results got overlapped by A2 and G (after G uses all the peak bandwidth), I isolated the case F to have a better chance to analyse results:

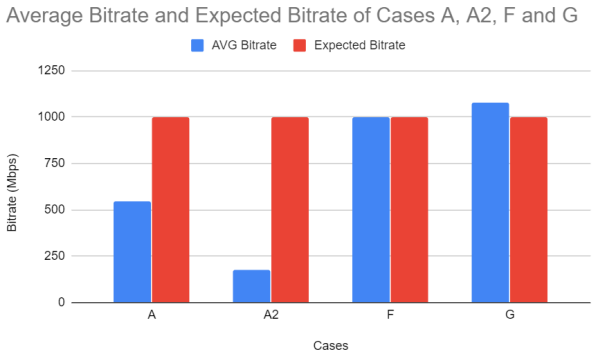


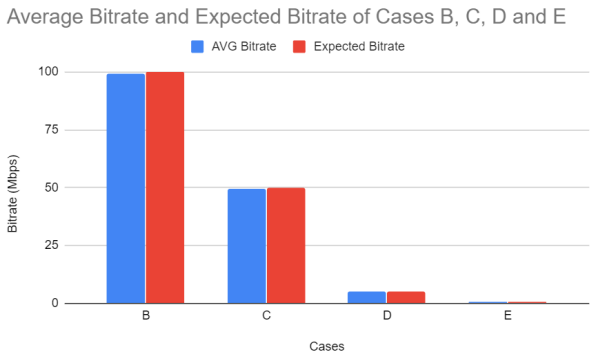

### F. Results Analysis

The bitrate should more or less match the average bandwidth in all cases except the ones that peaking bandwidth and burst size are calculated carefully to favour the traffic (in case G) or in the cases that peaking bandwidth and burst size are wrongly calculated (in case A and A2).

Only case G has traffic shaping properties that should reflect on having a better performance than the value specified in the average bandwidth and this is reflected in the results below.

Case G has similar traffic shaping as the example provided above when the differences between average bandwidth, peaking bandwidth and burst size were being explained (Peak bandwidth two times the value of average bandwidth and burst size eight times the value of average bandwidth).




The trust was calculated by dividing the average bitrate by the expected bitrate and multiplying the result by 100.

| Trustworthiness of the traffic on the VMs | |
|---|---|
| Case | Trust (in %) |
| A | 54.80 |
| B | 99.45 |
| C | 99.36 |
| D | 103.51 |
| E | 107.36 |
| A2 | 17.66 |
| F | 99.75 |
| G | 107.52 |

This means that, for example, case A perform way below as expected, almost half of the bitrate expected. On the other hand, case G performed around 0.07 times better the expected. Other cases like case B and F have almost the same performance as expected (almost same average and expected bitrate).

With the exception of the A cases (A and A2), all other cases guarantee bitrates with at most errors of less than 1%.

Once again, as expected, the only case that calculated the peak bandwidth and the burst size carefully achieved the best results (case G with 107.52). This was expected because it uses the peak bandwidth for around 5 seconds to send packets at 2000 Mbps of bitrate and only then starts sending with the average bandwidth (can be seen in the graphic that compares Bitrate in cases A2, F and G in the previous section).

With the normal cases B, C, D, E, F where I grabbed the default values given by VMware ESXi (of average bandwidth, peak bandwidth and burst size) and decreased them gradually (indicar nos casos), the network is always reliable. With, in the worst case, a loss of performance of 0.64% in Case C and with, in the best case, a gain of performance of 7.36% in case E.

We can see that both A and A2 cases has the worst values. In this cases, the traffic shaping was wrongly configured. It has a peak bandwidth 10 times the value of average bandwidth and a burst size 80 times. As we can see in the results, in case A2 when we should be getting an average of 1000 Mbps, we only get around 100 Mbps after 5 seconds and until the end of the 50 seconds (the expected 1000 Mbps was only achieved in the first 5 seconds).

This shows that VMWare ESXi network subsystem is reliable and has a high level of trustworthiness in case it is not wrongly configured.

## IV. Conclusion

In this report, I managed, in a first part, to research and describe some of the most important concepts on the VMware ESXi networking subsystem. Then, in a second part, to evaluate on a practical way how reliable the networking subsystem (more precisely the traffic shaping features) of VMware ESXi is. I have concluded that it has a high level of reliability if the networking subsystem is well configured and implemented.

## References


[1] VMware. VMware ESXi. https://www.vmware.com/products/esxi-and-esx.html.

[2] VMware. VMware Infrastructure Architecture Overview. https://www.vmware.com/pdf/vi_architecture_wp.pdf.

[3] VMware. VLAN configuration on virtual switches, physical switches, and virtual machines. https://kb.vmware.com/s/article/1003806.

[4] techopedia. VLAN tagging. https://www.techopedia.com/definition/32105/vlan-tagging.

[5] hostilecoding. VMware: VLAN Tagging - EST, VST VGT. https://hostilecoding.blogspot.com/2014/02/vmware-vlan-tagging-est-vst-vgt.html.

[6] VMware. Networking Concepts Overview. https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.vsphere.networking.doc/GUID-2B11DBB8-CB3C-4AFF-8885-EFEA0FC562F4.html.

[7] Saranya Venkatraman. vSphere client 6.5: Standard switch vs Distributed switch. https://www.vembu.com/blog/vsphere-client-6-5-standard-switch-vs-distributed-switch-2/.

[8] Jason Massae. What's the Big Deal with Jumbo Frames? https://blogs.vmware.com/virtualblocks/2019/01/30/whats-the-big-deal-with-jumbo-frames/.

[9] Christopher Wahl and Steven Pantol. Networking for VMware Administrators: The vSphere Standard Switch. http://www.pearsonitcertification.com/articles/article.aspx?p=2190191&seqNum=7.

[10] Zaib Kaleem. iPerf vs iPerf3. https://www.accessagility.com/blog/iperf-vs-iperf3.