Network Measurement over Cloud with perfSONAR

Li-Chi Ku[1] Hui-Lan Lee[2]
[1] [2]National Center for High-performance Computing
[1]lku@narlabs.org.tw [2]gracelee@narlabs.org.tw

Abstract

The quality of the cloud computing, especially those built upon inter-cloud and hybrid cloud mechanisms, relies heavily on the network quality. While network measurement tools can greatly ease the burden of maintaining network quality, the widely deployed NAT environment on most clouds hinders these tools from functioning properly. A double NAT technique has been developed and explained in this paper to make the NAT incompatible perfSONAR to work as fine as it is in on-premises networks.

Keywords: Double NAT, perfSONAR, Network measurement.

Introduction

The flourish of cloud computing has greatly reshaped the main usage of the Internet from its original peer to peer form into the nowaday client to datacenter form. The network quality between the user and the datacenters, in particular the network latency and packet loss rate, dictates the perceived quality of the cloud services. The development of inter-cloud and hybrid cloud services further elevates the importance of the network quality due to the involvement of more network segments behind the scene before the overall service can be delivered.

In current TCP/IP network architecture, the traffic is packed into packets before sending to the destination. Therefore a good network quality usually refers to the timely arrival of the packets at the destination, in the exact order and consistent interval as they are sent, without any packet loss. Depending on the type of the network application, the aforementioned characteristics of the network quality, namely network latency, packet order, jitter and packet loss rate may have different importance. But the latency and packet loss rate usually have the highest impact on the perceived cloud service quality.

However, the detection of network quality losses is not as straightforward as it might seem. The seven layers architecture[1] of TCP/IP network, designed from the beginning of the technology, was aimed at hiding the complexity of network transmission in modular layers thus the difficulty of designing network hardware and software became much easier. This design was so successful, leading to the huge growth of the Internet. However, it also hides the information regarding error under those layers, which makes network problems harder to detect.

Measurement tools are designed to obtain the network status to mitigate the aforementioned problem. Among different types of measurement, active network measurement[2] is the most common way to detect the end to end network quality. By actively sending packets to the destination, collecting the return packets and calculating the statistics from them, the characteristics of the network in between can be revealed without relying on the hidden information from the underlying network layers. This method requires the collaboration of both sides of the network, however.

Taking the most commonly used tools "ping" and "traceroute" as an example, they require the destination to send back "ICMP echo reply" or "ICMP port unreachable", respectively, in response to the sender side initiated measurement probes. These kinds of ICMP replies have been supported by all modern OSes and are taken care of by OS kernels. Thus extra software installation is not necessary in order to perform these kinds of measurements. However, if more complicated measurements, such as "one way ping" or bandwidth tests are demanded, since they are not supported by OSes, measurement software needs to be installed on both sides with matching settings configured.

Due to the aforementioned restriction, for measuring network quality to unspecific targets, our options are usually limited to basic measurements such as the ping and traceroute. In case more insight about the network is required, in order to have matching measurement software installed, both sides of the measurement often need to be in our control or under certain measurement collaboration.

The Challenge of The Cloud

Cloud computing imposes an additional challenge to network measurement because most cloud service providers adopt network address translation (NAT)[3] technology in their network environment, including Amazon AWS[4], Microsoft Azure[5] and Google Cloud[6]. With the NAT in place, the virtual machine (VM) gets a private IP address, such as 192.168.1.1, which is different from the out-facing public IP address when it talks with other machines on the Internet. Problem arises when this VM tells other measurement points its address and asks them to do measurement with it. Since the IP address that the VM knows is a private IP address, which is not an reachable IP address for other measurement points, the measurement is not going to be successfully made. Even if the out-facing public IP address is manually given to other measurement points instead, when they try to express their

intention of doing measurement with this VM, this VM only gets confused because the target they want to do measurement with (the out-facing public IP address) is obviously not the IP that it knows it posses. This problem becomes more apparent in peer to peer and scheduling based measurement solutions. While several network measurement tools are available in the market, many of them require public IP on both sides of measurement, or at least on the server side, if it's in client-server design. This requirement severely limits their usability in the cloud related network measurements.

perfSONAR

International research and education networks prefer open source network measurement tools for the following reasons:

- International research projects often involve multiple research institutions and universities across multiple countries and even continents. For the network measurement to be able to be made, every participating institution needs to have the same measurement tools. The openness and low-cost of open source tools make it easier to be widely adopted.
- To accommodate the quickly evolving new network technology and the different needs from the diversity of participating institutions, open source tools are much easier to modify accordingly and distribute the modified version to all other participants.

A group of experts from several research and education networks including Internet2[7], ESnet[8], GEANT[9], Indiana University[10], RNP[11] and University of Michigan[12] would like to address the common needs of cross domain network measurement and have eventually developed perfSONAR[13]. perfSONAR is a 100% open source github project under Apache License 2.0. It provides a nice web user interface, a scheduling mechanism to coordinate various measurement tests and an OpenSearch database to store the results. The tests it supports are quite abundant, as listed in table 1.

Table 1 perfSONAR supported test[14]

Test	Tool	Target
clock	psclock	Measure the clock difference between hosts
disk to disk	curl, globus	End to end, disk to disk data transfer rate
dns	dnspy	DNS transaction time
http	psurl	HTTP response time
latency	owping	One way network latency and packet loss rate
latencybg	powstream	Continuous one way latency and packet loss rate measurement
rtt	ping	Round-trip latency and packet loss rate
simplestream	simplestreamer	TCP connection test
throughput	iperf3, iperf2, nuttcp	Available bandwidth (network throughput)
s3throughput	s3-benchmark	S3 storage service transferring rate
trace	traceroute, tracepath, paris-traceroute	Network route tracing

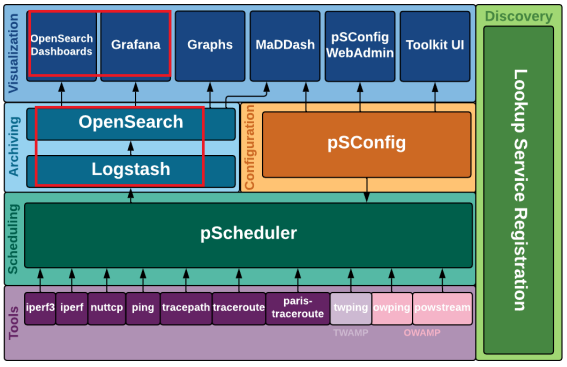

Figure 1 The perfSONAR component architecture

perfSONAR contains several roles[15], which are functionally independent, as follows. The architecture of the underlying components is shown in figure 1.

- Testpoint: a collection of test tools and a scheduling agent. The Testpoint can be installed independently and being coordinated by a central test plan publisher to do measurement with other Testpoints and then write the result remotely to an archive..
- Archive: an OpenSearch and Logstash backed database dedicated for storing measurement results.
- Toolkit: a combination of the aforementioned two, plus a web user interface. Thus it can do the measurement , store the result locally and provide a nice

visualization over the data, as shown in figure 2.

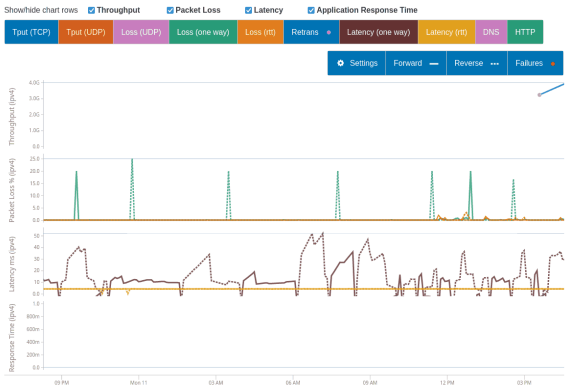

Figure 2 Packet loss rate and latency chart

The NAT Problem

Just like many other measurement tools, perfSONAR is a scheduling based system, which means there is only a scheduler running in the background and listening to port 443. Only when a test is requested from an internal schedule or from another testpoint through the https connection, the respective measurement tool is then launched.

If a test request is received but it is not targeted at the IP which the scheduler thinks itself to be, the request will simply be ignored. This is exactly what happens when perfSONAR is installed in a NAT environment, because the out-facing IP everybody else is seeing, is different from the IP which perfSONAR sees itself to be, as shown in the figure 3.

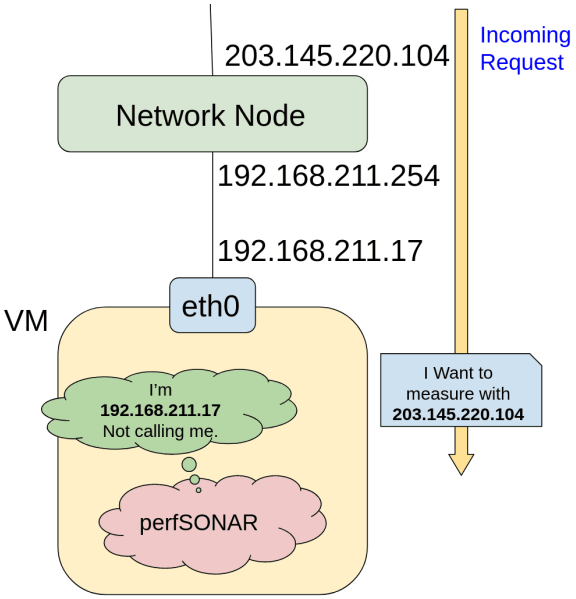

Figure 3 IP mismatch in the NAT environment

In such an environment, the VM gets assigned a private IP, such as the 192.168.211.17 in this example. When traffic goes to or comes from the Internet, it passes a device which translates the private IP into a public one or vice versa. This kind of device is called network nodes in OpenStack, or edge in VMware NSX-T. In this way, however, the IP seen from the inside and from the outside must be different. Because of the aforementioned scheduling mechanism, perfSONAR requires that the IP seen from each testpoints, including the testpoint itself, must be the same. Therefore the perfSONAR installed in most cloud environments does not work at all.

The Double NAT Solution

Since the NAT provided by the cloud service provider is inevitable and the measurement software requires the inner and outer IP to be the same, the workaround is to deploy a second NAT to translate the inner private IP back to the outer public IP. Thus the inner perfSONAR possesses the same IP as what other testpoints may see.

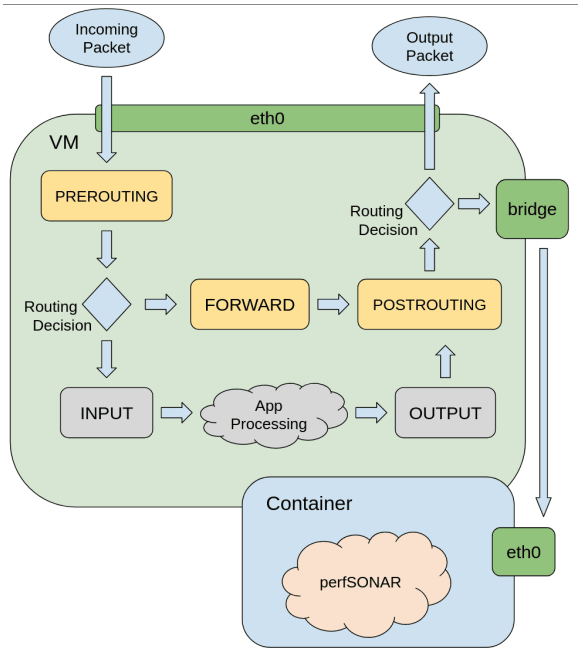

Figure 4 Iptables flow chart

This is possible because perfSONAR officially releases a docker container version of its testpoint. When docker containers are created in "bridge" network mode, they naturally come with a layer of NAT between the container and the host. We can then take advantage of this mechanism and create our perfSONAR container inside of the VM we obtain from the cloud service provider. The NAT introduced by docker between the VM and the perfSONAR container is then served as the second NAT we need to convert the private IP back to the outside public IP. In order to do that, we need to add rules to the following three iptables chains: PREROUTING, FORWARD and POSTROUTING, as shown in the yellow boxes in the figure 4.

In this example, the out-facing IP of the cloud VM is 203.145.220.104. The NAT mechanism in the network nodes modifies the destination IP of the inbound packets to 192.168.211.17, which is the private IP of our cloud VM, as shown in figure 5.

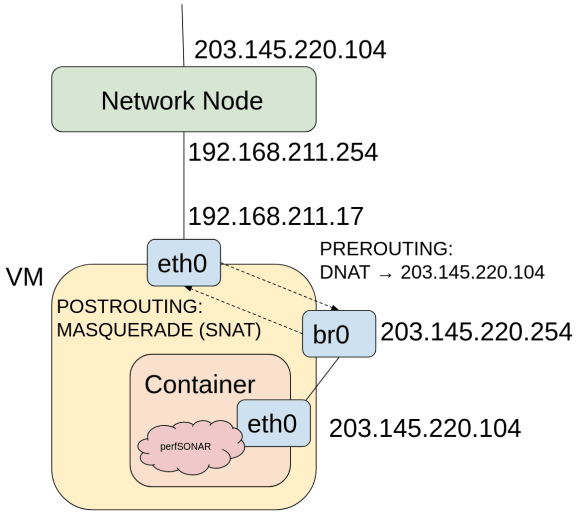

Figure 5 The second NAT in the VM

The DNAT rules we created in the PREROUTING chain translate the destination IP of the incoming packets to 203.145.220.104, which is exactly the out-facing public IP that others see. This works because a bridge br0 is created with 203.145.220.254, the gateway IP of this public IP subnet, which turns the 203.145.220.0/24 as a local attached subnet. Since it's a local subnet, the incoming packets are forwarded to br0, and eventually reach the inner perfSONAR container, which is also on br0 and has exactly the destination IP the packets are destined to go to.

When the perfSONAR container sends packets outward, those packets go to the br0 first because br0 has the gateway IP. On their way being forwarded, the masquerade SNAT rules automatically coming with the docker translate the source IP to 192.168.211.17. Thus when those packets leave the VM, they already look like regular packets that the VM generates. When they reach the network node, the NAT of the cloud service translates their source IP to 203.145.220.104, the out-facing IP, which is also the IP our perfSONAR thinks it has. With this double NAT setup, the IP gets translated back to what it is supposed to be, and the perfSONAR behaves exactly like it has that public IP in the first place.

The double NAT design won't cause IP duplication problems because the "fake public IP" given to the inner perfSONAR container is completely shielded inside the VM. Any traffic between the VM and the outside world behaves exactly like they are from the private IP of the VM. The only problem of the double NAT, if at all, is that since 203.145.220.0/24 becomes a local attached subnet in the perspective of the VM, any inner traffic targeting the real 203.145.220.0/24 of the outside world will only be forwarded to br0, staying in the VM "inner world" and never reach the real world targets. The real world 203.145.220.0/24 subnet is possessed by that same cloud provider and very likely assigned to other VMs of this cloud. This is unlikely to be of any problem to measurement use because other measurement testpoints, usually widely spread on other networks, are seldom on the same subnet.

The docker command to create the container inside the VM is as follows.

- docker network create -d bridge --subnet=203.145.220.0/24 --gateway=203.145.220.254 br0
- docker run -d --net=br0 --ip 203.145.220.104 -h hostname --name hostname perfsonar/testpoint

When the container is created, docker automatically sets up masquerade rules in iptables to allow the container outbound traffic and related return traffic to pass. Therefore only the inbound rules need to be added to allow the measurement container to receive connections from other testpoints. The ports used in perfSONAR supported tools are shown in table 2.

Table 2 perfSONAR tests and their ports[16]

Tool	Port
owamp (control)	861
owamp (test)	8760-9960/UDP
twamp (control)	862
twamp (test)	18760-19960/UDP
pscheduler	443
traceroute	33434-33634/UDP
simplestream	5890-5900
nuttcp	5000, 5101
iperf3	5201
iperf2	5001
ntp	123/UDP
ping	ICMP

In our current setup, the tools in use are:

- owamp (twamp follows automatically)
- iperf3
- ping
- traceroute

Thus the ports that need to be forwarded to the container are 443,861-862, 5201, 8760-9960/UDP and 18760-19960/UDP. The reasons that the port requirements for ping and traceroute are not included are that the traceroute 33434-33634/UDP belongs to the ephemeral port range, which is likely to be used by the applications in the VM. The VM also needs ICMP to perform various functions. Forwarding all of them to the container causes unexpected problems in the VM. Besides, the OS kernel of the VM can already take care of them, forwarding them further to the container is not necessary. Thus the ports being forwarded are simplified as:

- TCP: 443,861-862, 5201
- UDP: 8760-19960

and the final commands to add those rules are:

(sudo)

iptables -t nat -A PREROUTING -i ens3 -p tcp -m multiport --dports 443,861,862,5201 -j DNAT --to-destination 203.145.220.104

iptables -t nat -A PREROUTING -i ens3 -p udp -m multiport --dports 8760:19960 -j DNAT --to-destination 203.145.220.104

iptables -A DOCKER -j ACCEPT

Cautions should be taken that any traffic forwarded to the container will simply bypass the VM. Thus the service in the VM and those in the container can not overlap in their port ranges, otherwise only the service in the container will get the connection. Port 443, in particular, needs to be taken care of because the scheduler of perfSONAR needs it to work. Any application in the VM which uses port 443 will not get any inbound traffic after the DNAT rules are in effect.

Since cloud services usually come with their own firewall, the aforementioned ports and the requirements for ping and traceroute need to also be allowed in the cloud firewall service.

Conclusion

By using the double NAT technique to bring those “public IP only” measurement tools to the cloud, the network quality between the user and the cloud, and even between clouds and on premises facilities can be constantly monitored and any degradation of the overall quality can be immediately notified and rectified. By extending the already powerful perfSONAR from backbone network to the cloud, an overall picture of the network can be much easier to understand and maintain.

Reference

[1] Wikipedia, “OSI model,” https://en.wikipedia.org/wiki/OSI_model , Last modified: Apr 10 2024.

[2] Venkat Mohan. , Y. R. Janardhan Reddy, K. Kalpana, "Active and Passive Network Measurements: A Survey," International Journal of Computer Science and Information Technologies, Vol. 2 (4) , 1372-1385, 2011.

[3] Wikipedia, “Network address translation,” https://en.wikipedia.org/wiki/Network_address_translation , Last modified: Apr 8 2024.

[4] Amazon, Amazon Web Services, https://aws.amazon.com/ , Last viewed: Apr 12 2004.

[5] Microsoft, Azure Cloud Service, https://azure.microsoft.com/ , Last viewed: Apr 12 2004.

[6] Google, Google Cloud Platform, https://cloud.google.com/ , Last viewed: Apr 12 2004.

[7] Internet2, https://internet2.edu/ , Last viewed: Apr 12 2004.

[8] Department of Energy, Energy Sciences Network, https://www.es.net/ , Last viewed: Apr 12 2004.

[9] GEANT, https://geant.org/ , Last viewed: Apr 12 2004.

[10] Indiana University, https://www.iu.edu/index.html , Last viewed: Apr 12 2004.

[11] RNP, https://www.rnp.br/en , Last viewed: Apr 12 2004.

[12] University of Michigan, https://umich.edu/ , Last viewed: Apr 12 2004.

[13] perfSONAR, https://www.perfsonar.net/ , Last viewed: Apr 12 2004

[14] perfSONAR, “Test and Tool Reference,” perfSONAR 5.0.8 Documentation, 2024.

[15] perfSONAR, “perfSONAR Installation Options,” perfSONAR 5.0.8 Documentation, 2024.

[16] perfSONAR, “Firewalls and Security Software,” perfSONAR 5.0.8 Documentation, 2024.