**Developing a Software Defined CG-NAT**

Denis Plotnikov

<INTRO>

…

<WHAT IS NAT WHAT SHOULD IT DO (TYPES OF NAT and corresponding RFC overview)>

RFC

…

<WHAT IS CARRIER GRADE NAT, WHAT DOES IT DO, WHO CEARS>

RFC

…

<WHAT IS THE NOWADAYS APPROACH TO DO NAT>

Specific network devices

…

<WHY THIS APPROACH ISN’T GOOD: drawback of specialized NAT devices>

…

<WHAT ARE THEIR PRICES – the drawback #1>

My doc about prices with graph goes here

…

<HOW CAN WE DO BETTER>

Algorithmic(data structure) + technology(software/hardware architecture approaches (batching processing, NIC RSS queues, multithreading, locklessness, DDIO ) -> consequence DPDK already have it so let’s choose it)

…

<THE GOAL OF THIS WORK>

let’s try to make one

…

<WHAT OUR NAT SHOULD BE ABLE TO DO>

…

**Performance metrics of the NAT**

Our goal is to develop a working prototype of software defined carrier-grade network address translator (SD CG-NAT). To make sure that our SD CG-NAT is close to reality in terms of performance it is necessary to define the performance metrics and set their values. In order to get those metrics, a couple of sources are used. The first one is Rostelecom technical requirements to CG-NAT [ref\_TT\_ROS\_TEL]. The second one is the performance specification claimed by one of the on-market available NAT device producers[ref\_RDP.RU] which employ the same approach as this research does: **to use not task specific computer (a commodity server) to make a network specific solution using a mix of algorithmic and technological approaches. (our\_approach)**

<WHAT ARE THE NAT KEY METRICS>

The key characteristics are:

* ***Packets processing rate*** – (packets per second [PPS]) – the router’s maximum rate of packet processing. This is the main metric describing the packet processing abilities of a NAT device.
* ***Concurrent session support*** *–* (number) – the maximum number of sessions produced by served network. It describes the maximum network size which can be served by the NAT device. As described later in this document than bigger the network than harder to maintain translations to its nodes.
* ***Connections setups rate*** – (connection setups per second [csps]) the number of new NAT records to be created in a second. This metric shows the NAT ability to create new NAT records and could be a drawback of the NAT device in a certain modes of network work like when the networks nodes start creating of new connections actively, for example in the beginning working hours
* ***Throughput – (***bit per second[bps]***) –*** it isn’t very clear metric of the NAT device because it is mostly defined with NIC (network interface card) performance used by NAT device. If the NAT device won’t have enough of packet processing rate its throughput can’t achieve the maximum throughput provided with NIC and vice versa. The main sense of having it in the metric list is to make sure that NAT device is able to transfer needed amount of information.

<WHAT ARE THE METRICS VALUES>

**The target metrics values of the CG NAT**

Based on the sources of information the performance requirements of the NAT device are set following:

* Packet processing rate: 5.5 Mpps
* Concurrent session support: 65.5M (a B-class network with up to 1000 ports dedicated to each node)
* Connection setups rate: 3 Mcsps
* Throughput: 10 Gbps

In this document for evaluation of the performance another metrics are used: ***Cycles per packet*** – the amount of processors’ cycles spent on processing of one packet. This metric seems to be more descriptive than others while describing the NAT performance because there are a lot different processors. The processors differ to each other with CPU frequency and technologies used which makes it harder to estimate the performance of the NAT on different processors using the set of metrics described earlier in this chapter. Cycles per packet metric gives clearer estimation of the performance because at least it doesn’t depend on CPU frequency and may differ depending on model of processors. Thus the main performance metric used in this work is Cycles Per Packet while packet processing rate, concurrent session support, connections setups rate and throughput will play supplementary role and is mentioned where they role becomes important.

<HOW ARE WE GOING TO CHOSE THE DESIGN OF THE NAT>

…

<MY TEST NAT DESCRIPTION>

Functionality (CHECK SUM etc.)

==What I’ve included in my NAT and why==

For choosing the approach of building the NAT the testing application has been made. To simulate the NAT workflow several solutions have been implemented which use different data structures and software organization options. Conditionally the program can be split into 3 parts: measuring part, generation part and simulating part.

Measuring part consists of the environment that performs testing routine and calculates the performance results. The metric produced by this part is cycles per packet. This metric is acquired by using **rdtsc** function which reads the internal processor tick counters. The measuring part performs the number of tests set by the user and as an output calculates the average value of cycles per packet achieved including standard deviation.

Generation part generates a packet set to be processed by the simulation part. It imitates uniformly distributed network node activity and stores generated packets in a one-dimensional array of structures which is the input to the simulation part. Time of packet set generation isn’t taken into account when calculating the performance of the algorithm.

Simulation part is a core of the testing application and consists of NAT necessary actions. In turn the simulation part consists of the actions that must be performed by NAT such as calculation of the check sum, setting time stamp and saving/acquiring translation information in a NAT lookup data structure. The last point is the main exploration area of this document.

There are couple of necessary action to be performed by the NAT in order to perform address translation properly besides changing of packet’s IP address and number of TCP/UDP port in the corresponding headers. They are: calculation of the checksum for IP and TCP/UDP headers and storing the timestamp of the particular translation.

The checksum calculation is related to the packet processing. This action should be performed each time when the packet translation occurs and a packet IP and port number changed in order to be consistent with the requirements of the IP[1.4 of ref\_rfc791] and TCP/UDP protocols [1.5 of ref\_rfc793 and rfc\_768].

The storing of the timestamp translation in the NAT translation data structure is necessary and cannot be eliminated because of the Mapping Refresh requirement in [ref\_rfc4787].

For the testing purposes in the NAT testing program the following function implementations are used. For checksum calculation <function\_name> from Linux kernel is used. For getting the timestamp the gettimeofday() Linux system call is used. These functions might be potential targets of performance optimization but are out of the scope of this document.

The most interesting part in the NAT system is the data structure for storing the address translation information. In fact, two of the data structures are needed because of the necessity to store two pieces of data for a single address translation. The first one is the data about translation from LAN to WAN and second one is the data about translation from WAN to LAN. From the first glance it isn’t clear how to organize them well.

==Investigation of the base line performance of my NAT==

Before starting the exploration of the NAT translation data structures it is essential to estimate the performance of the system which uses the ideal NAT translation data structure. By word “ideal” the zero-time lookup data structure is implied. To get this estimation the bogus data structure was used which returns deterministic result and requires computation time tends to zero. Another words it is a function which cyclically returns the same sequence of results.

On the figure 1 there are some results explaining the result cost of one packet processing having the ideal lookup data structure.

Описать из чего это все состоит

==How much the NAT spends on what==

Add graph comparision of w/o check\_sum, w/o timestamps

==What is the NAT main problem -> look up table==

==Why is it so -> prove it by adding linear search to NAT==

Add chart of NAT performance with linear search for num\_of\_nodes [100, 250, 500, 1000, 1500, 2000, 2500, 5000 ]

<DATA STRUCTURES OVERVIEW>

1. tree
2. tree-array
3. rb-tree(balanced) array
4. hash
5. parallel

<PERFORMANCE OF EXPLORED APPROACHES>

(packets per second, memory usage, limitations)

1. Base line
2. Linear
3. Tree
   * 1. Tree-tree
     2. Tree - plain array
     3. RB-tree - plain array
4. Hash+array
5. Parallel hash+array
   * 1. Size
     2. Cores

Summary

**REFERENCES**

[ref\_TT\_ROS\_TEL] file:TT CGNAT 2014\_26\_06v1.doc

[ref\_RDP.RU] <http://rdp.ru/>

[ref\_rfc4787] Network Address Translation (NAT) Behavioral Requirements for Unicast UDP <https://tools.ietf.org/html/rfc4787#page-5>

[ref\_rfc791] Internet protocol <https://www.ietf.org/rfc/rfc791.txt>

[ref\_rfc793] Transmission Control Protocol <https://www.ietf.org/rfc/rfc793.txt>

[ref\_rfc768] User Datagram Protocol https://www.ietf.org/rfc/rfc768.txt