Workgroup:
Benchmarking Methodology Working Group
Internet-Draft:
draft-ietf-bmwg-benchmarking-stateful-05
Published: 23 January 2024
Intended Status: Informational
Expires: 26 July 2024
Authors: G. Lencse                    K. Shima
         Széchenyi István University   SoftBank Corp.

# Benchmarking Methodology for Stateful NATxy Gateways using RFC 4814 Pseudorandom Port Numbers

## Abstract

RFC 2544 has defined a benchmarking methodology for network interconnect devices. RFC 5180 addressed IPv6 specificities and it also provided a technology update but excluded IPv6 transition technologies. RFC 8219 addressed IPv6 transition technologies, including stateful NAT64. However, none of them discussed how to apply RFC 4814 pseudorandom port numbers to any stateful NATxy (NAT44, NAT64, NAT66) technologies. We discuss why using pseudorandom port numbers with stateful NATxy gateways is a difficult problem. We recommend a solution limiting the port number ranges and using two test phases (phase 1 and phase 2). We show how the classic performance measurement procedures (e.g. throughput, frame loss rate, latency, etc.) can be carried out. We also define new performance metrics and measurement procedures for maximum connection establishment rate, connection tear-down rate, and connection tracking table capacity measurements.

## Status of This Memo

## Copyright Notice

## Table of Contents

## 1. Introduction

[RFC2544] has defined a comprehensive benchmarking methodology for network interconnect devices, which is still in use. It was mainly IP version independent, but it used IPv4 in its examples. [RFC5180] addressed IPv6 specificities and also added technology updates, but declared IPv6 transition technologies out of its scope. [RFC8219] addressed the IPv6 transition technologies, including stateful NAT64. It has reused several benchmarking procedures from [RFC2544] (e.g. throughput, frame loss rate), it has redefined the latency measurement and added further ones, e.g. the PDV (packet delay variation) measurement.

However, none of them discussed, how to apply [RFC4814] pseudorandom port numbers, when benchmarking stateful NATxy (NAT44, NAT64, NAT66) gateways. We are not aware of any other RFCs that address this question.

First, we discuss why using pseudorandom port numbers with stateful NATxy gateways is a hard problem.

Then we recommend a solution.

### 1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 2. Pseudorandom Port Numbers and Stateful Translation

In its appendix, [RFC2544] has defined a frame format for test frames including specific source and destination port numbers. [RFC4814] recommends using pseudorandom and uniformly distributed values for both source and destination port numbers. However, stateful NATxy (NAT44, NAT64, NAT66) solutions use the port numbers to identify connections. The usage of pseudorandom port numbers causes different problems depending on the direction.

- As for the client-to-server direction, pseudorandom source and destination port numbers could be used, however, this approach would be a denial of service attack against the stateful NATxy gateway, because it would exhaust its connection tracking table capacity. To that end, let us see some calculations using the recommendations of RFC 4814:

  - The recommended source port range is: 1024-65535, thus its size is: 64512.

  - The recommended destination port range is: 1-49151, thus its size is: 49151.

  - The number of source and destination port number combinations is: 3,170,829,312.

  We note that the usage of different source and destination IP addresses further increases the number of connection tracking table entries.

- As for the server-to-client direction, the stateful DUT (Device Under Test) would drop any packets that do not belong to an existing connection, therefore, the direct usage of pseudorandom port numbers from the above-mentioned ranges is not feasible.

## 3. Test Setup and Terminology

Section 12 of [RFC2544] requires testing first using a single protocol source and destination address pair an then also using multiple protocol addresses. We follow the same approach: first, we use a single source and destination IP address pair, and then we explain how to use multiple IP addresses.

### 3.1. When Testing with a Single IP Address Pair

Our methodology works with any IP versions to benchmark stateful NATxy gateways, where x and y are in {4, 6}. To facilitate an easy understanding, we use two typical examples: stateful NAT44 and stateful NAT64.

The Test Setup for the well-known stateful NAT44 (also called NAPT: Network Address and Port Translation) solution is shown in Figure 1.

Note: We are fully aware of [RFC6890] special purpose IP address ranges. The [RFC1918] private IP addresses are used to facilitate an easy understanding of the example. And we consider the usage of the IP addresses reserved for benchmarking absolutely legitimate.

```
              +--------------------------------------+
     10.0.0.2 |Initiator                    Responder| 198.19.0.2
+-------------|                Tester                |<------------+
| private IPv4|                         [state table]| public IPv4 |
|             +--------------------------------------+             |
|                                                                  |
|             +--------------------------------------+             |
|    10.0.0.1 |                 DUT:                 | 198.19.0.1  |
+------------>|        Stateful NAT44 gateway        |-------------+
  private IPv4|     [connection tracking table]      | public IPv4
              +--------------------------------------+
```

Figure 1: Test setup for benchmarking stateful NAT44 gateways

The Test Setup for the also widely used stateful NAT64 [RFC6146] solution is shown in Figure 2.

```
              +--------------------------------------+
    2001:2::2 |Initiator                    Responder| 198.19.0.2
+-------------|                Tester                |<------------+
| IPv6 address|                         [state table]| IPv4 address|
|             +--------------------------------------+             |
|                                                                  |
|             +--------------------------------------+             |
|   2001:2::1 |                 DUT:                 | 198.19.0.1  |
+------------>|        Stateful NAT64 gateway        |-------------+
  IPv6 address|     [connection tracking table]      | IPv4 address
              +--------------------------------------+
```

Figure 2: Test setup for benchmarking stateful NAT64 gateways

As for transport layer protocol, [RFC2544] recommended testing with UDP, and it was kept also in [RFC8219]. For the general recommendation, we also keep UDP, thus the port numbers in the following text are to be understood as UDP port numbers. We discuss the limitations of this approach in Section 8.

We define the most important elements of our proposed benchmarking system as follows.

*Connection tracking table: The stateful NATxy gateway uses a connection tracking table to be able to perform the stateful translation in the server to client direction. Its size, policy, and content are unknown to the Tester.

- *Four tuple: The four numbers that identify a connection are source IP address, source port number, destination IP address, destination port number.

- *State table: The Responder of the Tester extracts the four tuple from each received test frame and stores it in its state table. Recommendation is given for writing and reading order of the state table in Section 4.10.

- *Initiator: The port of the Tester that may initiate a connection through the stateful DUT in the client-to-server direction. Theoretically, it can use any source and destination port numbers from the ranges recommended by [RFC4814]: if the used four tuple does not belong to an existing connection, the DUT will register a new connection into its connection tracking table.

- *Responder: The port of the Tester that may not initiate a connection through the stateful DUT in the server-to-client direction. It may send only frames that belong to an existing connection. To that end, it uses four tuples that have been previously extracted from the received test frames and stored in its state table.

- *Test phase 1: Test frames are sent only by the Initiator to the Responder through the DUT to fill both the connection tracking table of the DUT and the state table of the Responder. This is a newly introduced operation phase for stateful NATxy benchmarking. The necessity of this test phase is explained in Section 4.2.

- *Test phase 2: The measurement procedures defined by [RFC8219] (e.g. throughput, latency, etc.) are performed in this test phase after the completion of test phase 1. Test frames are sent as required (e.g. bidirectional test or unidirectional test in any of the two directions).

One further definition is used in the text of this document:

- *Black box testing: It is a testing approach when the Tester is not aware of the details of the internal structure and operation of the DUT. It can send input to the DUT and observe the output of the DUT.

### 3.2. When Testing with Multiple IP Addresses

We make considerations regarding the necessary and available IP addresses.

In Figure 1, we used the single 198.19.0.1 IPv4 address on the WAN side port of the stateful NAT44 gateway. However, in practice, not a single IP address, but an IP address range is assigned to the WAN side port of the stateful NAT44 gateways. Its required size depends on the number of client nodes and on the type of the stateful NAT44 algorithm. (The traditional algorithm always replaces the source port number, when a new connection is established. Thus it requires a larger range than the extended algorithm, which replaces the source port number only when it is necessary. Please refer to Table 1 and Table 2 of [LEN2015].)

When router testing is done, section 12 of [RFC2544] requires testing first using a single source and destination IP address pair, and then using destination IP addresses from 256 different networks. The 16-23 bits of the 198.18.0.0/24 and 198.19.0.0/24 addresses can be used to express the 256 networks. As we do not do router testing, we do not need to use multiple destination networks, therefore, these bits are available for expressing multiple IP addresses that belong to the same "/16" network. Moreover, both the 198.18.0.0/16 and the 198.19.0.0/16 networks can be used on the right side of the test setup as private IP addresses from the 10.0.0.0/16 network are used on its left side.

```
10.0.0.2/16 – 10.0.255.254/16      198.19.0.0/15 - 198.19.255.254/15
           \  +--------------------------------------+  /
            \ |Initiator                    Responder| /
+-------------|                Tester                |<------------+
| private IPv4|                         [state table]| public IPv4 |
|             +--------------------------------------+             |
|                                                                  |
|             +--------------------------------------+             |
| 10.0.0.1/16 |                 DUT:                 | public IPv4 |
+------------>|        Stateful NAT44 gateway        |-------------+
  private IPv4|     [connection tracking table]      | \
              +--------------------------------------+  \
                                   198.18.0.1/15 - 198.18.255.255/15
```

Figure 3: Test setup for benchmarking stateful NAT44 gateways using multiple IPv4 addresses

A possible solution for assigning multiple IPv4 addresses is shown in Figure 3. On the left side, the private IP address range is abundantly large. (We used the 16-31 bits to generate nearly 64k potential different source addresses, but the 8-15 bits are also available if needed.) On the right side, the 198.18.0.0./15 network is used, and it was cut into two equal parts. (Asymmetric division is also possible, if needed.)

We note that these are the potential address ranges. We discuss the actual address ranges to be used in Section 4.1.

In the case of stateful NAT64, a single "/64" IPv6 prefix contains a high number of bits to express different IPv6 addresses. Figure 4 shows an example, where we used bits 96-111 for that purpose.

```
2001:2::[0000-ffff]:0002/64       198.19.0.0/15 - 198.19.255.254/15
           \  +--------------------------------------+  /
  IPv6      \ |Initiator                    Responder| /
+-------------|                Tester                |<------------+
| addresses   |                         [state table]| public IPv4 |
|             +--------------------------------------+             |
|                                                                  |
|             +--------------------------------------+             |
| 2001:2::1/64|                 DUT:                 | public IPv4 |
+------------>|        Stateful NAT64 gateway        |-------------+
 IPv6 address |     [connection tracking table]      | \
              +--------------------------------------+  \
                                   198.18.0.1/15 - 198.18.255.255/15
```

Figure 4: Test Setup for benchmarking stateful NAT64 gateways using multiple IPv6 and IPv4 addresses

## 4. Recommended Benchmarking Method

### 4.1. Restricted Number of Network Flows

When a single IP address pair is used for testing then the number of network flows is determined by the number of source port number destination port number combinations.

The Initiator SHOULD use restricted ranges for source and destination port numbers to avoid the denial of service attack like event against the connection tracking table of the DUT described in Section 2. If it is possible, the size of the source port number range SHOULD be larger (e.g. in the order of a few times ten thousand), whereas the size of the destination port number range SHOULD be smaller (may vary from a few to several hundreds or thousands as needed). The rationale is that source and destination port numbers that can be observed in the Internet traffic are not symmetrical. Whereas source port numbers may be random, there are a few very popular destination port numbers (e.g. 443, 80, etc., see [IIR2020]), and others hardly occur. And we have found that their role is also asymmetric in the Linux kernel routing hash function [LEN2020].

However, in some special cases, the size of the source port range is limited. E.g. when benchmarking the CE and BR of a MAP-T [RFC7599] system together (as a compound system performing stateful NAT44), then the source port range is limited to the number of source port numbers assigned to each subscriber. (It could be as low as 2048 ports.)

When multiple IP addresses are used, then the port number ranges should be even more restricted, as the number of potential network flows is the product of the size of the source IP address range, the size of the source port number range, the size of the destination IP address range, and the size of the destination port number range. And our method requires the enumeration of all their possible combinations in test phase 1 as described in Section 4.4.

The number of network flows can be used as a parameter. The performance of the stateful NATxy gateway MAY be examined as a function of this parameter as described in Section 5.1.

### 4.2. Test Phase 1

Test phase 1 serves two purposes:

1. The connection tracking table of the DUT is filled. It is important, because its maximum connection establishment rate may be lower than its maximum frame forwarding rate (that is throughput).

2. The state table of the Responder is filled with valid four tuples. It is a precondition for the Responder to be able to transmit frames that belong to connections that exist in the connection tracking table of the DUT.

Whereas the above two things are always necessary before test phase 2, test phase 1 can be used without test phase 2. It is done so when the maximum connection establishment rate is measured (as described in Section 4.5).

Test phase 1 MUST be performed before all tests performed in test phase 2. The following things happen in test phase 1:

1. The Initiator sends test frames to the Responder through the DUT at a specific frame rate.

2. The DUT performs the stateful translation of the test frames and it also stores the new connections in its connection tracking table.

3. The Responder receives the translated test frames and updates its state table with the received four tuples. The responder transmits no test frames during test phase 1.

When test phase 1 is performed in preparation for test phase 2, the applied frame rate SHOULD be safely lower than the maximum connection establishment rate. (It implies that maximum connection establishment rate measurement MUST be performed first.) Please refer to Section 4.4 for further conditions regarding timeout and the enumeration of all possible four tuples.

### 4.3. Consideration of the Cases of Stateful Operation

We consider the most important events that may happen during the operation of a stateful NATxy gateway, and the Actions of the gateway as follows.

1. EVENT: A packet not belonging to an existing connection arrives in the client-to-server direction. ACTION: A new connection is registered into the connection tracking table and the packet is translated and forwarded.

2. EVENT: A packet not belonging to an existing connection arrives in the server-to-client direction. ACTION: The packet is discarded.

3. EVENT: A packet belonging to an existing connection arrives (in any direction). ACTION: The packet is translated and forwarded and the timeout counter of the corresponding connection tracking table entry is reset.

4. EVENT: A connection tracking table entry times out. ACTION: The entry is deleted from the connection tracking table.

Due to "black box" testing, the Tester is not able to directly examine (or delete) the entries of the connection tracking table. But the entries can be and MUST be controlled by setting an appropriate timeout value and carefully selecting the port numbers of the packets (as described in Section 4.4) to be able to produce meaningful and repeatable measurement results.

We aim to support the measurement of the following performance characteristics of a stateful NATxy gateway:

1. maximum connection establishment rate

2. all "classic" performance metrics like throughput, frame loss rate, latency, etc.

3. connection tear-down rate

4. connection tracking table capacity

### 4.4. Control of the Connection Tracking Table Entries

It is necessary to control the connection tracking table entries of the DUT to achieve clear conditions for the measurements. We can simply achieve the following two extreme situations:

1. All frames create a new entry in the connection tracking table of the DUT and no old entries are deleted during the test. This is required for measuring the maximum connection establishment rate.

2. No new entries are created in the connection tracking table of the DUT and no old ones are deleted during the test. This is ideal for the measurements to be executed in phase 2, like throughput, latency, etc.

From this point, we use the following two assumptions:

1. The connection tracking table of the stateful NATxy is large enough to store all connections defined by the different four tuples.

2. Each experiment is started with an empty connection tracking table. (It can be ensured by deleting its content before the experiment.)

The first extreme situation can be achieved by

*using different four tuples for every single test frame in test phase 1 and

*setting the UDP timeout of the NATxy gateway to a value higher than the length of test phase 1.

The second extreme situation can be achieved by

*enumerating all possible four tuples in test phase 1 and

*setting the UDP timeout of the NATxy gateway to a value higher than the length of test phase 1 plus the gap between the two phases plus the length of test phase 2.

[RFC4814] REQUIRES pseudorandom port numbers, which we believe is a good approximation of the distribution of the source port numbers a NATxy gateway on the Internet may face with.

We note that although the enumeration of all possible four tuples is not a requirement for the first extreme situation and the usage of different four tuples in test phase 1 is not a requirement for the second extreme situation, pseudorandom enumeration of all possible four tuples in test phase 1 is a good solution in both cases. It may be computing efficiently generated by preparing a random permutation of the previously enumerated all possible four tuples using Dustenfeld's random shuffle algorithm [DUST1964].

The enumeration of the four tuples in increasing or decreasing order (or in any other specific order) MAY be used as an additional measurement.

### 4.5. Measurement of the Maximum Connection Establishment Rate

The maximum connection establishment rate is an important characteristic of the stateful NATxy gateway and its determination is necessary for the safe execution of test phase 1 (without frame loss) before test phase 2.

The measurement procedure of the maximum connection establishment rate is very similar to the throughput measurement procedure defined in [RFC2544].

Procedure: The Initiator sends a specific number of test frames using all different four tuples at a specific rate through the DUT. The Responder counts the frames that are successfully translated by the DUT. If the count of offered frames is equal to the count of received frames, the rate of the offered stream is raised and the test is rerun. If fewer frames are received than were transmitted, the rate of the offered stream is reduced and the test is rerun.

The maximum connection establishment rate is the fastest rate at which the count of test frames successfully translated by the DUT is equal to the number of test frames sent to it by the Initiator.

Note: In practice, we RECOMMEND the usage of binary search.

### 4.6. Validation of Connection Establishment

Due to "black box" testing, the entries of the connection tracking table of the DUT may not be directly examined, but the presence of the connections can be checked easily by sending frames from the Responder to the Initiator in test phase 2 using all four tuples stored in the state table of the Tester (at a low enough frame rate). The arrival of all test frames indicates that the connections are indeed present.

Procedure: When all the desired N number of test frames were sent by the Initiator to the Receiver at frame rate R in test phase 1 for the maximum connection establishment rate measurement, and the Receiver has successfully received all the N frames, the establishment of the connections is checked in test phase 2 as follows:

- The Responder sends test frames to the Initiator at frame rate r=R*alpha, for the duration of N/r using a different four tuple from its state table for each test frame.

- The Initiator counts the received frames, and if all N frames are arrived then the R frame rate of the maximum connection establishment rate measurement (performed in test phase 1) is raised for the next iteration, otherwise lowered (as well as in the case if test frames were missing in the preliminary test phase).

Notes:

- The alpha is a kind of "safety factor", it aims to make sure that the frame rate used for the validation is not too high, and test may fail only in the case if at least one connection is not present in the connection tracking table of the DUT. (So alpha should be typically less than 1, e.g. 0.8 or 0.5.)

- The duration of N/r and the frame rate of r means that N frames are sent for validation.

- The order of four tuple selection is arbitrary provided that all four tuples MUST be used.

- Please refer to Section 4.9 for a short analysis of the operation of the measurement and what problems may occur.

### 4.7.  Test Phase 2

As for the traffic direction, there are three possible cases during test phase 2:

- bidirectional traffic: The Initiator sends test frames to the Responder and the Responder sends test frames to the Initiator.

- unidirectional traffic from the Initiator to the Responder: The Initiator sends test frames to the Responder but the Responder does not send test frames to the Initiator.

- unidirectional traffic from the Responder to the Initiator: The Responder sends test frames to the Initiator but the Initiator does not send test frames to the Responder.

If the Initiator sends test frames, then it uses pseudorandom source port numbers and destination port numbers from the restricted port number ranges. (If it uses multiple source and/or destination IP addresses, then their ranges are also limited.) The responder receives the test frames, updates its state table, and processes the test frames as required by the given measurement procedure (e.g. only counts them for the throughput test, handles timestamps for latency or PDV tests, etc.).

If the Responder sends test frames, then it uses the four tuples from its state table. The reading order of the state table may follow different policies (discussed in Section 4.10). The Initiator receives the test frames and processes them as required by the given measurement procedure.

As for the actual measurement procedures, we RECOMMEND to use the updated ones from Section 7 of [RFC8219].

## 4.8. Measurement of the Connection Tear-down Rate

Connection tear-down can cause significant load for the NATxy gateway. The connection tear-down performance can be measured as follows:

1. Load a certain number of connections (N) into the connection tracking table of the DUT (in the same way as done to measure the maximum connection establishment rate).

2. Record TimestampA.

3. Delete the content of the connection tracking table of the DUT.

4. Record TimestampB.

The connection tear-down rate can be computed as:

connection tear-down rate = N / ( TimestampB - TimestampA)

The connection tear-down rate SHOULD be measured for various values of N.

We assume that the content of the connection tracking table may be deleted by an out-of-band control mechanism specific to the given NATxy gateway implementation. (E.g. by removing the appropriate kernel module under Linux.)

We are aware that the performance of removing the entire content of the connection tracking table at one time may be different from removing all the entries one by one.

### 4.9. Measurement of the Connection Tracking Table Capacity

The connection tracking table capacity is an important metric of stateful NATxy gateways. Its measurement is not easy, because an elementary step of a validated maximum connection establishment rate measurement (defined in Section 4.6) may have only a few distinct observable outcomes, but some of them may have different root causes:

1. During test phase 1, the number of test frames received by the Responder is less than the number of test frames sent by the Initiator. It may have different root causes, including:

    1. The R frame sending rate was higher than the maximum connection establishment rate. (Note that now the maximum connection establishment rate is considered unknown because we can not measure the maximum connection establishment without our assumption 1 in Section 4.4!) This root cause may be eliminated by lowering the R rate and re-executing the test. (This step may be performed multiple times, while R>0.)

    2. The capacity of the connection tracking table of the DUT has been exhausted. (And either the DUT does not want to delete connections or the deletion of the connections makes it slower. This case is not investigated further in test phase 1.)

2. During test phase 1, the number of test frames received by the Responder equals the number of test frames sent by the Initiator. In this case, the connections are validated in test phase 1. The validation may have two kinds of observable results:

    1. The number of validation frames received by the Initiator equals the number of validation frames sent by the Responder. (It proves that the capacity of the connection tracking table of the DUT is enough and both R and r were chosen properly.)

    2. The number of validation frames received by the Initiator is less than the number of validation frames sent by the Responder. This phenomenon may have various root causes:

        1. The capacity of the connection tracking table of the DUT has been exhausted. (It does not matter, whether some existing connections are discarded and new ones are stored, or the new connections are discarded. Some connections are lost anyway, and it makes validation fail.)

2. The R frame sending rate used by the Initiator was too high in test phase 1 and thus some connections were not established, even though all test frames arrived at the Responder. This root cause may be eliminated by lowering the R rate and re-executing the test. (This step may be performed multiple times, while R>0.)

3. The r frame sending rate used by the Responder was too high in test phase 2 and thus some test frames did not arrive at the Initiator, even though all connections were present in the connection tracking table of the DUT. This root cause may be eliminated by lowering the r rate and re-executing the test. (This step may be performed multiple times, while r>0.)

And here is the problem: as the above three root causes are indistinguishable, it is not easy to decide, whether R or r should be decreased.

We have some experience with benchmarking stateful NATxy gateways. When we tested iptables with a very high number of connections, the 256GB RAM of the DUT was exhausted and it stopped responding. Such a situation may make the connection tracking table capacity measurements rather inconvenient. We include this possibility in our recommended measurement procedure, but we do not address the detection and elimination of such a situation. (E.g. how the algorithm can reset the DUT.)

For the connection tracking table size measurement, first we need a safe number: C0. It is a precondition, that C0 number of connections can surely be stored in the connection tracking table of the DUT. Using C0, one can determine the maximum connection establishment rate using C0 number of connections. It is done with a binary search using validation. The result is R0. The values C0 and R0 will serve as "safe" starting values for the following two searches.

First, we perform an exponential search to find the order of magnitude of the connection tracking table capacity. The search stops if the DUT collapses OR the maximum connection establishment rate severely drops (e.g. to its one tenth) due to doubling the number of connections.

Then, the result of the exponential search gives the order of magnitude of the size of the connection tracking table. Before disclosing the possible algorithms to determine the exact size of the connection tracking table, we consider three possible replacement policies for the NATxy gateway:

1. The gateway does not delete any live connections until their timeout expires.

2. The gateway replaces the live connections according to LRU (least recently used) policy.

3. The gateway does a garbage collection when its connection tracking table is full and a frame with a new four tuple arrives. During the garbage collection, it deletes the K least recently used connections, where K is greater than 1.

Now, we examine, what happens and how many validation frames arrive in the there cases. Let the size of the connection tracking table be S, and the number of preliminary frames be N, where S is less than N.

1. The connections defined by the first S test frames are registered into the connection tracking table of the DUT, and the last N-S connections are lost. (It is another question if the last N-S test frames are translated and forwarded in test phase 1 or simply dropped.) During validation, the validation frames with four tuples corresponding to the first S test frames will arrive at the Initiator and the other N-S validation frames will be lost.

2. All connections are registered into the connection tracking table of the DUT, but the first N-S connections are replaced (and thus lost). During validation, the validation frames with four tuples corresponding to the last S test frames will arrive to the Initiator, and the other N-S validation frames will be lost.

3. Depending on the values of K, S, and N, maybe less than S connections will survive. In the worst case, only S-K+1 validation frames arrive, even though, the size of the connection tracking table is S.

If we know that the stateful NATxy gateway uses the first or second replacement policy and we also know that both R and r rates are low enough, then the final step of determining the size of the connection tracking table is simple. If the Responder sent N validation frames and the Initiator received N' of them, then the size of the connection tracking table is N'.

In the general case, we perform a binary search to find the exact value of the connection tracking table capacity within E error. The search chooses the lower half of the interval if the DUT collapses

OR the maximum connection establishment rate severely drops (e.g. to its half) otherwise it chooses the higher half. The search stops if the size of the interval is less than the E error.

The algorithms for the general case are defined using C like pseudocode in Figure 5. In practice, this algorithm may be made more efficient in a way that the binary search for the maximum connection establishment rate stops, if an elementary test fails at a rate under RS*beta or RS*gamma during the external search or during the final binary search for the capacity of the connection tracking table, respectively. (This saves a high amount of execution time by eliminating the long-lasting tests at low rates.)

```
// The binary_search_for_maximum_connection_establishment_rate(c,r)
// function performs a binary search for the maximum connection
// establishment rate in the [0, r] interval using c number of
// connections.

// This is an exponential search for finding the order of magnitude
// of the connection tracking table capacity
// Variables:
//   C0 and R0 are beginning safe values for the connection tracking
//     table size and connection establishment rate, respectively
//   CS and RS are their currently used safe values
//   CT and RT are their values for the current examination
//   beta is a factor expressing an unacceptable drop in R (e.g.
//     beta=0.1)
R0=binary_search_for_maximum_connection_establishment_rate(C0,maxrate);
for ( CS=C0, RS=R0;  1; CS=CT, RS=RT )
{
  CT=2*CS;
  RT=binary_search_for_maximum_connection_establishment_rate(CT,RS);
  if ( DUT_collapsed || RT < RS*beta )
    break;
}
// Here the size of the connection tracking table is between CS and CT

// This is the final binary search for finding the connection tracking
// table capacity within E error
// Variables:
//   CS and RS are the safe values for connection tracking table size
//     and connection establishment rate, respectively
//   C and R are the values for the current examination
//   gamma is a factor expressing an unacceptable drop in R
//     (e.g. gamma=0.5)
for ( D=CT-CS;  D>E; D=CT-CS )
{
  C=(CS+CT)/2;
  R=binary_search_for_maximum_connection_establishment_rate(C,RS);
  if ( DUT_collapsed || R < RS*gamma)
    CT=C; // take the lower half of the interval
  else
    CS=C,RS=R; // take the upper half of the interval
}
// Here the size of the connection tracking table is CS within E error
```

Figure 5: Measurement of the Connection Tracking Table Capacity

### 4.10.  Writing and Reading Order of the State Table

As for the writing policy of the state table of the Responder, we RECOMMEND round robin, because it ensures that its entries are automatically kept fresh and consistent with that of the connection tracking table of the DUT.

The Responder can read its state table in various orders, for example:

- pseudorandom

- round-robin

We RECOMMEND pseudorandom to follow the spirit of [RFC4814]. Round-robin may be used as a computationally cheaper alternative.

## 5. Scalability Measurements

As for scalability measurements, we do not define any new type of performance metrics, but we recommend to perform measurement series through which the value of one or more parameter(s) is/are changed to discover how the various values of the given parameter(s) influence the performance of the DUT.

### 5.1. Scalability Against the Number of Network Flows

The scalability measurements aim to quantify how the performance of the stateful NATxy gateways degrades with the increase of the number of network flows.

As for the actual values for the number of network flows to be used during the measurement series, we RECOMMEND to use some representative values from the range of the potential number of network flows the DUT may be faced with during its intended usage.

It is important, how the given number of network flows are generated. The sizes of the ranges of the source and destination IP addresses and port numbers are essential parameters to be reported together with the results. Please see also Section 6 about the reporting format.

If a single IP address pair is used, then we RECOMMEND to use

- a fixed, larger source port number range (e.g., a few times 10,000)

- a variable size destination port number range (e.g. 10; 100; 1,000; etc.), where its expedient granularity depends on the purpose.

### 5.2. Scalability Against the Number of CPU Cores

Stateful NATxy gateways are often implemented in software that are not bound to a specific hardware but can be executed by commodity servers. To facilitate the comparison of their performance, it can be useful to determine

*the performance of the various implementations using a single core of a well-known CPU

*the scale-up of the performance of the various implementations with the number of CPU cores.

If the number of the available CPU cores is a power of two, then we recommend to perform the tests with 1, 2, 4, 8, 16, etc. number of active CPU cores of the DUT.

## 6. Reporting Format

Measurements MUST be executed multiple times to achieve statistically reliable results. The report of the results MUST contain the number of repetitions of the measurements. We RECOMMEND median as the summarizing function of the results complemented with the first percentile and the 99th percentile as indices of the dispersion of the results. Average and standard deviation MAY also be reported.

All parameters and settings that may influence the performance of the DUT MUST be reported. Some of them may be specific to the given NATxy gateway implementation, like the "hashsize" (hash table size) and "nf_conntrack_max" (number of connection tracking table entries) values for iptables or the limit of the number of states for OpenBSD PF (set by the "set limit states number" command in the pf.conf file).

| | | | | |
|---|---:|---:|---:|---:|
| number of sessions (req.) | 0.4M | 4M | 40M | 400M |
| source port numbers (req.) | 40,000 | 40,000 | 40,000 | 40,000 |
| destination port numbers (req.) | 10 | 100 | 1,000 | 10,000 |
| "hashsize" (i.s.) | 2^17 | 2^20 | 2^23 | 2^27 |
| "nf_conntrack_max" (i.s.) | 2^20 | 2^23 | 2^26 | 2^30 |
| num. sessions / "hashsize" (i.s.) | 3.05 | 3.81 | 4.77 | 2.98 |
| number of experiments (req.) | 10 | 10 | 10 | 10 |
| error of binary search (req.) | 1,000 | 1,000 | 1,000 | 1,000 |
| connections/s median (req.) | | | | |
| connections/s 1st perc. (req.) | | | | |
| connections/s 99th perc. (req.) | | | | |

Figure 6: Example table: Maximum connection establishment rate of iptables against the number of sessions

Figure 6 shows an example of table headings for reporting the measurement results for the scalability of the iptables stateful NAT44 implementation against the number of sessions. We have indicated the always required fields (req.) and the implementation-specific ones (i.s.). In row 6, we also added a computed value, the number of sessions per hashsize ratio, which helps the reader to interpret the achieved maximum connection establishment rate. (A lower value results in shorter linked lists hanging on the entries of the hash table thus facilitating higher performance. The ratio is varying, because the number of sessions is always a power of 10, whereas the hash table size is a power of 2.) To reflect the accuracy of the results, we have also added the value of the "error" of the binary search, which expresses the stopping criterion for the binary search. The binary search stops, when the difference between the "higher limit" and "lower limit" of the binary search is less than or equal to "error".

The table MUST be complemented with reporting the relevant parameters of the DUT. If the DUT is a general-purpose computer and some software NATxy gateway implementation is tested, then the hardware description SHOULD include: computer type, CPU type, and number of active CPU cores, memory type, size and speed, network interface card type (reflecting also the speed), the fact that direct cable connections were used or the type of the switch used for interconnecting the Tester and the DUT. Operating system type and version, kernel version, and the version of the NATxy gateway implementation (including last commit date and number if applicable) SHOULD also be given.

## 7. Implementation and Experience

The stateful extension of siitperf [SIITPERF] is an implementation of this concept. It is documented in this (open access) paper [LEN2022].

The proposed benchmarking methodology has been validated by performing benchmarking measurements with three radically different stateful NAT64 implementations (Jool, tayga+iptables, OpenBSD PF) in (open access) paper [LEN2023].

Our further experience with this methodology using siitperf for measuring the scalability of the iptables stateful NAT44 and Jool stateful NAT64 implementations are described in [I-D.lencse-v6ops-transition-scalability].

## 8. Limitations of using UDP as Transport Layer Protocol

Stateful NATxy solutions handle TCP and UDP differently, e.g. iptables uses 30s timeout for UDP and 60s timeout for TCP. Thus benchmarking results produced using UDP do not necessarily characterize the performance of a NATxy gateway well enough when they are used for forwarding Internet traffic. As for the given example, timeout values of the DUT may be adjusted, but it requires extra consideration.

Other differences in handling UDP or TCP are also possible. Thus we recommend that further investigations are to be performed in this field.

As a mitigation of this problem, we recommend that testing with protocols using TCP (like HTTP and HTTPS) can be performed as described in [RFC9411]. This approach also solves the potential problem of protocol helpers may be present in the stateful DUT.

## 9. Acknowledgements

The authors would like to thank Al Morton, Sarah Banks, Edwin Cordeiro, Lukasz Bromirski, Sándor Répás, Tamás Hetényi, Timothy Winters, Eduard Vasilenko, Minh Ngoc Tran, Paolo Volpato, Zeqi Lai, and Bertalan Kovács for their comments.

This work was supported by the Japan Trust International Research Cooperation Program of the National Institute of Information and Communications Technology (NICT), Japan.

## 10. IANA Considerations

This document does not make any request to IANA.

## 11. Security Considerations

We have no further security considerations beyond that of [RFC8219]. Perhaps they should be cited here so that they be applied not only for the benchmarking of IPv6 transition technologies but also for the benchmarking of stateful NATxy gateways.

## 12. References

### 12.1. Normative References

**[RFC1918]** Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <https://www.rfc-editor.org/info/rfc1918>.

**[RFC2119]** Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

**[RFC2544]** Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <https://www.rfc-editor.org/info/rfc2544>.

**[RFC4814]** Newman, D. and T. Player, "Hash and Stuffing: Overlooked Factors in Network Device Benchmarking", RFC 4814, DOI 10.17487/RFC4814, March 2007, <https://www.rfc-editor.org/info/rfc4814>.

**[RFC5180]** Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, DOI 10.17487/RFC5180, May 2008, <https://www.rfc-editor.org/info/rfc5180>.

**[RFC6146]** Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <https://www.rfc-editor.org/info/rfc6146>.

**[RFC6890]** Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <https://www.rfc-editor.org/info/rfc6890>.

**[RFC7599]** Li, X., Bao, C., Dec, W., Ed., Troan, O., Matsushima, S., and T. Murakami, "Mapping of Address and Port using Translation (MAP-T)", RFC 7599, DOI 10.17487/RFC7599, July 2015, <https://www.rfc-editor.org/info/rfc7599>.

**[RFC8174]** Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

**[RFC8219]** Georgescu, M., Pislaru, L., and G. Lencse, "Benchmarking Methodology for IPv6 Transition Technologies", RFC 8219, DOI 10.17487/RFC8219, August 2017, <https://www.rfc-editor.org/info/rfc8219>.

**[RFC9411]** Balarajah, B., Rossenhoevel, C., and B. Monkman, "Benchmarking Methodology for Network Security Device Performance", RFC 9411, DOI 10.17487/RFC9411, March 2023, <https://www.rfc-editor.org/info/rfc9411>.

### 12.2. Informative References

**[DUST1964]** Durstenfeld, R., "Algorithm 235: Random permutation", Communications of the ACM, vol. 7, no. 7, p.420., DOI

10.1145/364520.364540, July 1964, <https://dl.acm.org/doi/10.1145/364520.364540>.

**[I-D.lencse-v6ops-transition-scalability]**
Lencse, G., "Scalability of IPv6 Transition Technologies for IPv4aaS", Work in Progress, Internet-Draft, draft-lencse-v6ops-transition-scalability-05, 14 October 2023, <https://datatracker.ietf.org/doc/html/draft-lencse-v6ops-transition-scalability-05>.

**[IIR2020]** Kurahashi, T., Matsuzaki, Y., Sasaki, T., Saito, T., and F. Tsutsuji, "Periodic observation report: Internet trends as seen from IIJ infrastructure - 2020", Internet Infrastructure Review, vol. 49, December 2020, <https://www.iij.ad.jp/en/dev/iir/pdf/iir_vol49_report_EN.pdf>.

**[LEN2015]** Lencse, G., "Estimation of the Port Number Consumption of Web Browsing", IEICE Transactions on Communications, vol. E98-B, no. 8. pp. 1580-1588, DOI DOI: 10.1587/transcom.E98.B.1580, 1 August 2015, <http://www.hit.bme.hu/~lencse/publications/e98-b_8_1580.pdf>.

**[LEN2020]** Lencse, G., "Adding RFC 4814 Random Port Feature to Siitperf: Design, Implementation and Performance Estimation", International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems, vol 9, no 3, pp. 18-26., DOI 10.11601/ijates.v9i3.291, 2020, <http://www.hit.bme.hu/~lencse/publications/291-1113-1-PB.pdf>.

**[LEN2022]** Lencse, G., "Design and Implementation of a Software Tester for Benchmarking Stateful NAT64xy Gateways: Theory and Practice of Extending Siitperf for Stateful Tests", Computer Communications, vol. 172, no. 1, pp. 75-88, August 1, 2022, DOI 10.1016/j.comcom.2022.05.028, 2022, <http://www.hit.bme.hu/~lencse/publications/ECC-2022-SFNAT64xy-Tester-published.pdf>.

**[LEN2023]** Lencse, G., Shima, K., and K. Cho, "Benchmarking methodology for stateful NAT64 gateways", Computer Communications, vol. 210, no. 1, pp. 256-272, October 1, 2023, DOI 10.1016/j.comcom.2023.08.009, 2023, <http://www.hit.bme.hu/~lencse/publications/ECC-2023-BM-SFNAT64-published.pdf>.

**[SIITPERF]** Lencse, G., "Siitperf: An RFC 8219 compliant SIIT and stateful NAT64/NAT44 tester written in C++ using DPDK", source code, available from GitHub, 2019-2023, <https://github.com/lencsegabor/siitperf>.

## Appendix A. Change Log

### A.1. 00

Initial version.

### A.2. 01

Updates based on the comments received on the BMWG mailing list and minor corrections.

### A.3. 02

Section 4.4 was completely re-written. As a consequence, the occurrences of the now undefined "mostly different" source port number destination port number combinations were deleted from Section 4.5, too.

### A.4. 03

Added Section 4.3 about the consideration of the cases of stateful operation.

Consistency checking. Removal of some parts obsoleted by the previous re-writing of Section 4.4.

Added Section 4.8 about the method for measuring connection tear-down rate.

Updates for Section 7 about the implementation and experience.

### A.5. 04

Update of the abstract.

Added Section 4.6 about validation of connection establishment.

Added Section 4.9 about the method for measuring connection tracking table capacity.

Consistency checking and corrections.

### A.6. 00 - WG item

Added measurement setup for Stateful NAT64 gateways.

Consistency checking and corrections.

### A.7. 01

Added Section 4.5.1 about typical types of measurement series and reporting format.

### A.8. 02

Added the usage of multiple IP addresses.

Section 4.5.1 was removed and split into two Sections: Section 5 about scalability measurements and Section 6 about reporting format.

### A.9. 03

Updated the usage of multiple IP addresses.

Test phases were renamed as follows:

*preliminary test phase --> test phase 1

*real test phase --> test phase 2.

### A.10. 04

Minor updates to Section 3.2 and Section 7.

### A.11. 05

Minor updates addressing WGLC nits (adding the definition of "black box", and performing a high amount of grammatical corrections).

## Authors' Addresses

Gábor Lencse
Széchenyi István University
Győr
Egyetem tér 1.
H-9026
Hungary

Email: lencse@sze.hu

Keiichi Shima
SoftBank Corp.
1-7-1 Kaigan, Tokyo
105-7529
Japan

Email: shima@wide.ad.jp
URI: https://softbank.co.jp/