Stream: Internet Engineering Task Force (IETF)
RFC: 9693
Category: Informational
Published: January 2025
ISSN: 2070-1721
Authors: G. Lencse, *Széchenyi István University*; K. Shima, *SoftBank Corp.*

RFC 9693
Benchmarking Methodology for Stateful NATxy Gateways

Abstract

RFC 2544 defines a benchmarking methodology for network interconnect devices. RFC 5180 addresses IPv6 specificities, and it also provides a technology update but excludes IPv6 transition technologies. RFC 8219 addresses IPv6 transition technologies, including stateful NAT64. However, none of them discuss how to apply pseudorandom port numbers from RFC 4814 to any stateful NATxy (such as NAT44, NAT64, and NAT66) technologies. This document discusses why using pseudorandom port numbers with stateful NATxy gateways is a difficult problem. It recommends a solution that limits the port number ranges and uses two test phases (phase 1 and phase 2). This document shows how the classic performance measurement procedures (e.g., throughput, frame loss rate, latency, etc.) can be carried out. New performance metrics and measurement procedures are also defined for measuring the maximum connection establishment rate, connection tear-down rate, and connection tracking table capacity.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[RFC2544] defines a comprehensive benchmarking methodology for network interconnect devices that is still in use. It is mainly independent of IP version, but it uses IPv4 in its examples. [RFC5180] addresses IPv6 specificities and also adds technology updates but declares IPv6 transition technologies are out of its scope. [RFC8219] addresses the IPv6 transition technologies, including stateful NAT64. It reuses several benchmarking procedures from [RFC2544] (e.g., throughput, frame loss rate), and it redefines the latency measurement and adds further ones (e.g., the Packet Delay Variation (PDV) measurement).

However, none of them discuss how to apply pseudorandom port numbers from [RFC4814] when benchmarking stateful NATxy gateways (such as NAT44 [RFC3022], NAT64 [RFC6146], and NAT66). (It should be noted that stateful NAT66 is not an IETF specification but refers to an IPv6 version of the stateful NAT44 specification.) The authors are not aware of any other RFCs that address this question.

First, this document discusses why using pseudorandom port numbers with stateful NATxy gateways is a difficult problem. Then, a solution is recommended.

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] defines 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 (such as NAT44, NAT64, and NAT66) solutions use the port numbers to identify connections. The usage of pseudorandom port numbers causes different problems depending on the direction:

- 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 [RFC4814]:
    - 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.

  It should be noted that the usage of different source and destination IP addresses further increases the number of connection tracking table entries.
- For the server-to-client direction, the stateful Device Under Test (DUT) would drop any packets that do not belong to an existing connection; therefore, the direct usage of pseudorandom port numbers from the ranges mentioned above is not feasible.

3. Test Setup and Terminology

Section 12 of [RFC2544] requires testing using a single protocol source and destination address pair first and then also using multiple protocol addresses. The same approach is followed: first, a single source and destination IP address pair is used, and then it is explained how to use multiple IP addresses.

3.1. When Testing with a Single IP Address Pair

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

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

Note that the private IP addresses from [RFC1918] are used to facilitate an easy understanding of the example, and the usage of the IP addresses reserved for benchmarking is 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 stateful NAT64 solution [RFC6146], which is also widely used, 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 the transport layer protocol, [RFC2544] recommended testing with UDP, and it was also kept in [RFC8219]. UDP is also kept for a general recommendation; thus, the port numbers in the following text are to be understood as UDP port numbers. The rationale and limitations of this approach are discussed in Section 8.

The most important elements of the proposed benchmarking system are defined as follows:

Connection: Although UDP itself is a connectionless protocol, stateful NATxy gateways keep track of their translation mappings in the form of a "connection" as well as in the case of UDP using the same kind of entries as in TCP.

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, and 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. A recommendation is given for the 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 only send 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 stores in its state table.

Test phase 1: The 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., a bidirectional test or a unidirectional test in any of the two directions).

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

Black box testing: 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

This section considers the number of the necessary and available IP addresses.

In Figure 1, the single 198.19.0.1 IPv4 address is used on the WAN side port of the stateful NAT44 gateway. However, in practice, it is not a single IP address, but rather an IP address range that 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 Tables 1 and 2 of [LEN2015].)

When router testing is done, Section 12 of [RFC2544] requires testing using a single source and destination IP address pair first 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 this document does not deal with router testing, no multiple destination networks are needed; 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. (The 16-31 bits were used for generating 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.)

It should be noted that these are the potential address ranges. The actual address ranges to be used are discussed 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 bits 96-111 are used 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 and destination port number combinations.

The Initiator **SHOULD** use restricted ranges for source and destination port numbers to avoid the exhaustion of the connection tracking table capacity of the DUT as 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 tens of thousands), whereas the size of the destination port number range **SHOULD** be smaller (e.g., it 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 Internet traffic are not symmetrical. Whereas source port numbers may be random, there are a few very popular destination port numbers (e.g., 443 or 80; see [IIR2020]), and others hardly occur. Additionally, it was 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. For example, when benchmarking the Customer Edge (CE) and Border Relay (BR) of a Mapping of Address and Port using Translation (MAP-T) system [RFC7599] together (as a compound system performing stateful NAT44), 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 source port number range,

- the destination IP address range, and
- the destination port number range.

In addition, the recommended 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. This is important because its maximum connection establishment rate may be lower than its maximum frame forwarding rate (that is, its 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. This is done when the maximum connection establishment rate is measured (as described in Section 4.5).

Test phase 1 **MUST** be performed before all tests are 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

The authors 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. However, the entries can 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.

This document aims to support the measurement of the following performance characteristics of a stateful NATxy gateway:

- maximum connection establishment rate
- all "classic" performance metrics like throughput, frame loss rate, latency, etc.
- connection tear-down rate
- 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. One 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, the following two assumptions are used:

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. (This 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.

As described in [RFC4814], pseudorandom port numbers are **REQUIRED**, which the authors believe is a good approximation of the distribution of the source port numbers a NATxy gateway on the Internet may be faced with.

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. Pseudorandom enumeration of all possible four tuples may be generated in a computationally efficient way by using Durstenfeld's random shuffle algorithm [DUST1964] to prepare a random permutation of the previously enumerated all possible four tuples.

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].

The procedure is as follows:

- 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, the usage of binary search is **RECOMMENDED**.

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. However, 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.

The procedure is as follows:

When all the desired N number of test frames are 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 have 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, it is lowered (as well as in the case that test frames were missing in the preliminary test phase, as well).

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 the test may fail only in the case of if at least one connection is not present in the connection tracking table of the DUT. (Therefore, 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:

1. Bidirectional traffic: The Initiator sends test frames to the Responder, and the Responder sends test frames to the Initiator.
2. 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.
3. 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, the usage of the updated ones from Section 7 of [RFC8219] is **RECOMMENDED**.

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.

It is assumed 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).

It is noted 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:

- 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:
  - 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 one cannot measure the maximum connection establishment without 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.)
  - 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).
- 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 2. 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:
     - 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 if the new connections are discarded. Some connections are lost anyway, and it makes validation fail.)
     - The R frame sending rate used by the Initiator was too high in test phase 1; 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.)
     - The r frame sending rate used by the Responder was too high in test phase 2; 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.)

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

Experience shows that the DUT may collapse if its memory is exhausted. Such a situation may make the connection tracking table capacity measurements rather inconvenient. This possibility is included in the recommended measurement procedure, but the detection and elimination of such a situation is not addressed (e.g., how the algorithm can reset the DUT).

For the connection tracking table size measurement, first, one needs 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, an exponential search is performed 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, three possible replacement policies for the NATxy gateway are considered:

1. The gateway does not delete any live connections until their timeout expires.
2. The gateway replaces the live connections according to the Least Recently Used (LRU) 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 LRU connections, where K is greater than 1.

Now, it is examined what happens and how many validation frames arrive in the three 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 one knows that the stateful NATxy gateway uses the first or second replacement policy and one also knows 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, a binary search is performed 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 the 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 binarySearchForMaximumConnectionCstablishmentRate(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)
//   maxrate is the maximum frame rate for the media
R0=binarySearchForMaximumConnectionCstablishmentRate(C0,maxrate);
for ( CS=C0, RS=R0; 1; CS=CT, RS=RT )
{
  CT=2*CS;
  RT=binarySearchForMaximumConnectionCstablishmentRate(CT,RS);
  if ( DUT_collapsed || RT < RS*beta )
    break;
}
// At this point, 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=binarySearchForMaximumConnectionCstablishmentRate(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
}
// At this point, 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, round robin is **RECOMMENDED**, 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

Pseudorandom is **RECOMMENDED** to follow the approach of [RFC4814]. Round robin may be used as a computationally cheaper alternative.

5. Scalability Measurements

As for scalability measurements, no new types of performance metrics are defined, but it is **RECOMMENDED** to perform measurement series through which the value of one or more parameter(s) 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, it is **RECOMMENDED** 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 also see Section 6 about the reporting format.

If a single IP address pair is used, then it is **RECOMMENDED** to use:

- a fixed, larger source port number range (e.g., a few times 10,000) and
- 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 is 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 and
- 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 it is **RECOMMENDED** 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. The necessary number of repetitions to achieve statistically reliable results may depend on the consistent or scattered nature of the results. The report of the results **MUST** contain the number of repetitions of the measurements. The median is **RECOMMENDED** as the summarizing function of the results complemented with the first percentile and the 99th percentile as indices of the dispersion of the results. The 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.)				

Table 1: Example Table of the Maximum Connection Establishment Rate of Iptables Against the Number of Sessions

Table 1 shows an example of table headings for reporting the measurement results regarding the scalability of the iptables stateful NAT44 implementation against the number of sessions. The table indicates the required fields (req.) and the implementation-specific ones (i.s.). A computed value was also added in row 6; it is 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, the table contains 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 the "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 the following:

- computer type
- CPU type
- number of active CPU cores
- memory type, size, and speed
- network interface card type (also reflecting the speed)
- the fact that direct cable connections were use or the type of switch used for interconnecting the Tester and the DUT

The operating system type and version, kernel version, and version of the NATxy gateway implementation (including the 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. Its first version that only supports multiple port numbers is documented in this (open access) paper: [LEN2022]. Its extended version that also supports multiple IP addresses is documented in this (open access) paper: [LEN2024b].

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

Further experience with this methodology of using siitperf for measuring the scalability of the iptables stateful NAT44 and Jool stateful NAT64 implementations are described in [SCALABILITY].

This methodology was successfully applied for the benchmarking of various IPv4-as-a-Service (IPv4aas) technologies without the usage of technology-specific Testers by reducing the aggregate of their Customer Edge (CE) and Provider Edge (PE) devices to a stateful NAT44 gateway documented in this (open access) paper: [LEN2024a].

8. Limitations of Using UDP as a Transport Layer Protocol

The test frame format defined in [RFC2544] exclusively uses UDP (and not TCP) as a transport layer protocol. Testing with UDP was kept in both [RFC5180] and [RFC8219] regarding the standard benchmarking procedures (throughput, latency, frame loss rate, etc.). The benchmarking methodology proposed in this document follows this long-established benchmarking tradition using UDP as a transport layer protocol, too. The rationale for this is that the standard benchmarking procedures require sending frames at arbitrary constant frame rates, which would violate the flow control and congestion control algorithms of the TCP protocol. TCP connection setup (using the three-way handshake) would further complicate testing.

Further potential transport layer protocols, e.g., the Datagram Congestion Control Protocol (DCCP) [RFC4340] and the Stream Control Transmission Protocol (SCTP) [RFC9260], are outside of the scope of this document, as the widely used stateful NAT44 and stateful NAT64 implementations do not support them. Although QUIC [RFC9000] is also considered a transport layer protocol, QUIC packets are carried in UDP datagrams; thus, QUIC does not need a special handling.

Some stateful NATxy solutions handle TCP and UDP differently, e.g., iptables use a 30s timeout for UDP and a 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, the authors recommend that further investigations should be performed in this field.

As a mitigation of this problem, this document recommends that testing with protocols using TCP (like HTTP and HTTPS up to version 2) can be performed as described in [RFC9411]. This approach also solves the potential problem of protocol helpers that may be present in the stateful DUT.

As for HTTP/3, it uses QUIC, which uses UDP as stated above. It should be noted that QUIC is treated as any other UDP payload. The proposed measurement method does not aim to measure the performance of QUIC, rather, it aims to measure the performance of the stateful NATxy gateway.

9. IANA Considerations

This document has no IANA actions.

10. Security Considerations

This document has no further security considerations beyond that of [RFC8219]. They should be cited here so that they can be applied not only for the benchmarking of IPv6 transition technologies but also for the benchmarking of any stateful NATxy gateways (allowing for x=y, too).

11. References

11.1. 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>.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <https://www.rfc-editor.org/info/rfc3022>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <https://www.rfc-editor.org/info/rfc4340>.

[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>.

[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>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9260] Stewart, R., Tüxen, M., and K. Nielsen, "Stream Control Transmission Protocol", RFC 9260, DOI 10.17487/RFC9260, June 2022, <https://www.rfc-editor.org/info/rfc9260>.

[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>.

11.2. 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/pdf/10.1145/364520.364540>.

[IIR2020] Kurahashi, T., Matsuzaki, Y., Sasaki, T., Saito, T., and F. Tsutsuji, "Periodic Observation Report: Internet Trends as Seen from IIJ Infrastructure - 2020", Internet Initiative Japan Inc., 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 10.1587/transcom.E98.B.1580, August 2015, <https://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, November 2020, <http://ijates.org/index.php/ijates/article/view/291>.

[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. 192, pp. 75-88, DOI 10.1016/j.comcom.2022.05.028, August 2022, <https://www.sciencedirect.com/science/article/pii/S0140366422001803>.

[LEN2023] Lencse, G., Shima, K., and K. Cho, "Benchmarking methodology for stateful NAT64 gateways", Computer Communications, vol. 210, pp. 256-272, DOI 10.1016/j.comcom.2023.08.009, October 2023, <https://www.sciencedirect.com/science/article/pii/S0140366423002931>.

[LEN2024a] Lencse, G. and Á. Bazsó, "Benchmarking methodology for IPv4aaS technologies: Comparison of the scalability of the Jool implementation of 464XLAT and MAP-T", Computer Communications, vol. 219, pp. 243-258, DOI 10.1016/j.comcom.2024.03.007, April 2024, <https://www.sciencedirect.com/science/article/pii/S0140366424000999>.

[LEN2024b] Lencse, G., "Making stateless and stateful network performance measurements unbiased", Computer Communications, vol. 225, pp. 141-155, DOI 10.1016/j.comcom.2024.05.018, September 2024, <https://www.sciencedirect.com/science/article/abs/pii/S0140366424001993>.

[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>.

[SIITPERF] "Siitperf: An RFC 8219 compliant SIIT and stateful NAT64/NAT44 tester", commit 165cb7f, September 2023, <https://github.com/lencsegabor/siitperf>.

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.

The authors thank Warren Kumari, Michael Scharf, Alexey Melnikov, Robert Sparks, David Dong, Roman Danyliw, Erik Kline, Murray Kucherawy, Zaheduzzaman Sarker, and Éric Vyncke for their reviews and comments.

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

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, Minato-ku, Tokyo
105-7529
Japan
Email: shima@wide.ad.jp
URI: https://softbank.co.jp/