         Benchmarking Methodology for Stateful NATxy Gateways using RFC 4814
                         Pseudorandom Port Numbers
                draft-lencse-bmwg-benchmarking-stateful-03

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 NAT (NAT44,
   NAT64, NAT66) technologies. We discuss why using pseudorandom port
   numbers with stateful NAT gateways is a hard problem and recommend a
   solution.

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
BCP14 [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 to use 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 private to public 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.
   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 section 12 of [RFC2544] also requires testing with
      256 destination networks, which further increases the number of
      connection tracking table entries.

   * As for the public to private 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

   Our methodology works with any IP version. We use IPv4 in the Test
   Setup shown in Figure 1 to facilitate its easy understanding based on
   the well-known stateful NAT44 (also called NAPT: Network Address and
   Port Translation) solution.

                   +---------------------------------------+
        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 |
      +------------>| Sateful NATxy gateway |------------+
        private IPv4| [connection tracking table] | public IPv4
                   +---------------------------------------+
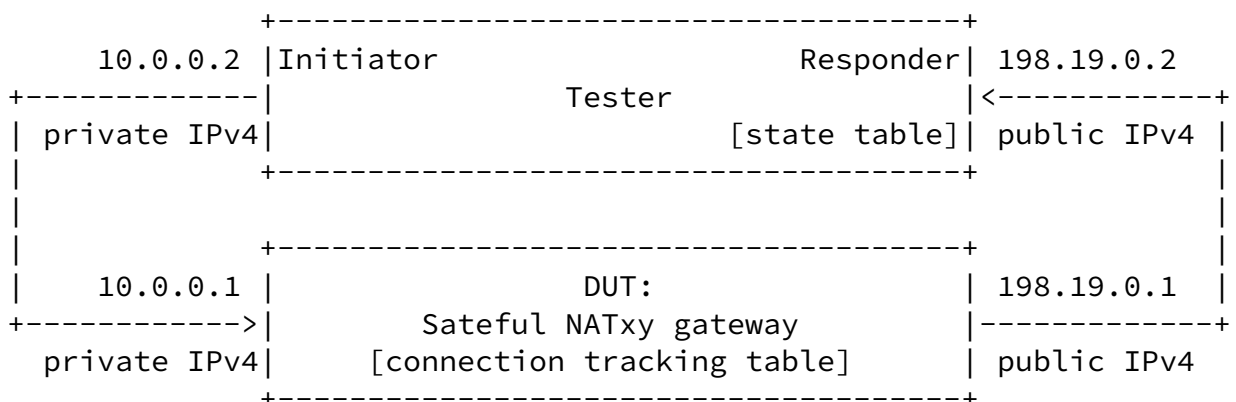
Figure 1: Test Setup for benchmarking stateful NATxy gateways

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 limitation of this approach in Section 6.

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 public to private direction. Its size, policy and content are unknown for 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 4.8.

*   Initiator: The port of the Tester that may initiate a connection through the stateful DUT in the private to public 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 public to private 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.

* Preliminary test phase: 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 phase is explained in
      [Section 4.2](#).

   * Real test phase: The actual test (e.g. throughput, latency, etc.)
      is performed in this phase after the completion of the preliminary
      test phase. Test frames are sent as required (e.g. bidirectional
      test or unidirectional test in any of the two directions).

4. Recommended Benchmarking Method

4.1. Restricted Port Number Ranges

   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](#). 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].

   The product of the sizes of the two ranges can be used as a
   parameter. The performance of the stateful NATxy gateway MAY be
   examined as a function of this parameter.

4.2. Preliminary Test Phase

   The preliminary phase 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 exist in the connection tracking table of the DUT.

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

A preliminary test phase MUST be performed before all tests performed in the real test phase. In this phase, the following things happen:

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 combinations 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 the preliminary phase.

When the preliminary test phase is performed in preparation to the real test phase, the applied frame rate and the duration of the preliminary phase SHOULD be carefully selected so that:

* The applied frame rate be safely lower than the maximum connection establishment rate.

* Enough four tuples be stored in the state table of the Responder so that it can generate frames with the proper distribution of the four tuples.

Please refer to [Section 4.4](#) for further conditions regarding timeout and port number combinations.

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 private to public 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 public to private direction. ACTION: The packet is discarded.

3. EVENT: A packet belonging to an existing connection arrives (in any dicection). 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 entires 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.

4.4. Control of the Connection Tracking Table Entries

It is necessary to control the connection tracking table entries of the DUT in order 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 real test phase measurements, like throughput, latency, etc.

From this point we use the following three assumptions:

1. A single source address destination address pair is used for all tests. We make this assumption for simplicity. Of course, we are aware that [RFC2544] requires testing also with 256 different destination networks.

2. The connection tracking table of the stateful NATxy is large enough to store all connections defined by the different source port number destination port number combinations.

3. 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 all different source port number destination port number combinations in the preliminary phase and

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

The second extreme situation can be achieved by

*   using all different source port number destination port number combinations in the preliminary phase and

*   enumerating all the possible source port number destination port number combinations in the preliminary phase and

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

[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 pseudorandom all different source port number destination port number combinations may be computing efficiently generated by preparing a random permutation of the previously enumerated all possible source port number destination port number combinations using Dustenfeld's random shuffle algorithm [DUST1964]. This method also satisfies the criterion for the second case that all possible source port number destination port number combinations must be enumerated during the preliminary phase.

Important warning: in normal (non-NAT) router testing, the port number selection algorithm, whether it is pseudo-random or enumerated in increasing (or decreasing) order does not affect final results. However, our experience with iptables shows that if the connection tracking table is filled using port number enumeration in increasing order, then the maximum connection establishment rate of iptables degrades significantly compared to its performance using pseudorandom port numbers [LEN2021].

The enumeration of the source port number destination port number combinations 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 the preliminary test phase (without frame loss) before the real test phase.

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 source port number destination port number combinations
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.

Notes:

1. In practice, we RECOMMEND the usage of binary search.

2. As for the successful translation, the Responder MAY (or SHOULD?)
    check that the source IP address is different than the original
    source IP address set by the Initiator.

4.6. Real Test Phase

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

* 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. 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 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 4.8). 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].

4.7. 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
   (important) 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.

4.8. Writing and Reading Order of the State Table

As for 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. Implementation and Experience

The "stateful" branch of siitperf [SIITPERF] is an implementation of
this concept. It is documented in a paper currently under second
review [LEN2022].

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

6. 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 usig TCP (like HTTP and HTTPS) can be performed as
described in [I-D.ietf-bmwg-ngfw-performance]. This approach also
solves the potential problem of protocol helpers may be present in
the stateful DUT.

7. Acknowledgements

The authors would like to thank Al Morton, Sarah Banks, Edwin
Cordeiro, Lukasz Bromirski and Sandor Repas for their comments.

8. IANA Considerations

This document does not make any request to IANA.

9. 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.

10. References

10.1. Normative References

[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>.

   [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>.

10.2. 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.ietf-bmwg-ngfw-performance]
              Balarajah, B., Rossenhoevel, C., and B. Monkman,
              "Benchmarking Methodology for Network Security Device
              Performance", Work in Progress, Internet-Draft, draft-
              ietf-bmwg-ngfw-performance-13, 12 January 2022,
              <https://www.ietf.org/archive/id/draft-ietf-bmwg-ngfw-

performance-13.txt>.

[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-01, 21 February 2022,
          <https://www.ietf.org/archive/id/draft-lencse-v6ops-
          transition-scalability-01.txt>.

[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>.

[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>.

[LEN2021]   Lencse, G., "Design and Implementation of a Software
          Tester for Benchmarking Stateful NAT64 Gateways: Theory
          and Practice of Extending Siitperf for Stateful
          Tests", it was under review in Computer
          Communications, then it was significantly rewritten,
          2021, <http://www.hit.bme.hu/~lencse/publications/SFNAT64-
          tester-for-review.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", revised version, under second review in Computer

               Communications, may be revised or removed without notice,
               2022, <http://www.hit.bme.hu/~lencse/publications/SFNATxy-
               tester-revised.pdf>.

   [SIITPERF] Lencse, G., "Siitperf: An RFC 8219 compliant SIIT
               (stateless NAT64) tester written in C++ using
               DPDK", source code, available from GitHub, 2019-2022,
               <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 obsolated by the
   previous re-writing of Section 4.4.

   Added Section 4.7 about the method for measuring connection tear down
   rate.

   Updates for Section 5 about the implementation and experience.

Authors' Addresses

Gabor Lencse
Szechenyi Istvan University
Gyor
Egyetem ter 1.
H-9026
Hungary
Email: lencse@sze.hu

Keiichi Shima
IIJ Innovation Institute
Iidabashi Grand Bloom, 2-10-2 Fujimi, Tokyo
102-0071
Japan
Email: keiichi@iijlab.net