Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Benchmarking Authoritative DNS Servers

G. Lencse[1]

[1]Department of Networked Systems and Services, Budapest University of Technology and Economics, Magyar tudósok körútja 2, H-1117 Budapest, Hungary

Corresponding author: G. Lencse (e-mail: lencse@hit.bme.hu).

ABSTRACT In this paper, we examine the performance of four authoritative DNS server implementations (BIND, NSD, knot DNS, and YADIFA). In our tests, we apply the measurement procedure defined in Section 9 of RFC 8219. Our aim is threefold: to provide DNS operators with ready to use measurement results to support their selection of the best fitting authoritative DNS server implementation for their needs, to assist researchers and DNS64 server developers in finding a suitable authoritative DNS server implementation for their DNS64 benchmarking measurements, and to advance the theory and practice of benchmarking DNS servers. We examine how the different conditions such as the number of active CPU cores, the size of the zone file, the applied timeout, and the type of the processor influence the performance of the tested authoritative DNS server implementations. The performance of all four tested DNS servers scales up more or less well with the number of CPU cores, except for YADIFA. The increase of the size of the zone file causes significant degradation only in the performance of BIND, which shows different anomalies described in the paper. The change of the timeout from 250ms (required by RFC 8219) to 100ms usually causes only a small performance degradation. We point out that NSD and Knot DNS can achieve an order of magnitude higher performance than BIND and YADIFA.

INDEX TERMS Benchmarking, DNS, DNS64, performance.

I. INTRODUCTION

DNS (Domain Name System) is an integral part of all commonly used Internet services, but it seems to be inconspicuous, when everything goes smooth. However, a failure or delay in DNS resolution results in poor QoE (Quality of Experience) for the users.

Although the performance of different authoritative DNS server implementations is an important issue, it still lacks of a standard benchmarking methodology. In this paper, we propose one. Whereas BIND is considered the *de facto industry standard DNS server*, and it was the most widely used one in 2004 [1], some other DNS implementations (e.g. NSD or Knot DNS) can provide multiple times higher authoritative DNS server performance than BIND. For a DNS server operator, higher performance results in less costs considering both CAPEX (Capital Expenditures, here: the price of the hardware) and OPEX (Operating Expenditure, here: the computing power requirement and thus, also the electricity bill). High performance can also be a kind of mitigation against DoS (Denial of Service) attacks [2].

As for a special usage of authoritative DNS servers, they are needed for benchmarking DNS64 [3] servers. (DNS64 is an important IPv6 transition technology [4], which is used together with stateful NAT64 [5] to enable IPv6-only clients to communicate with IPv4-only servers [6].) In section 9 of RFC 8219 [7], we defined a benchmarking methodology for DNS64 servers. This measurement procedure requires the use of an authoritative DNS server that can provide DNS resolution at 220% of the maximum testing rate of DNS64 servers. (Please refer to [8] for more details.) Thus, finding a sufficiently high performance authoritative DNS server is a prerequisite for performing DNS64 benchmarking tests. In 2017 we benchmarked three different DNS64 servers, and we had to choose an authoritative DNS server with high enough performance. Due to time constraint, then we have selected the first suitable one, which was YADIFA [9]. However, we considered the performance comparison of the different authoritative DNS servers an interesting research topic, especially, because we have found that there was a gap in research papers concerning both a standard methodology for benchmarking authoritative DNS servers and also measurement results. Although our original motivation was to support DNS64 benchmarking, we contend that the comparison of the performance of various authoritative DNS

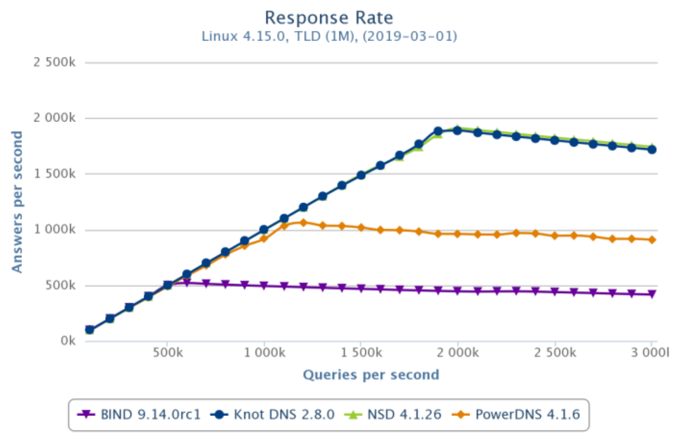

FIGURE 1. **Response rate of different authoritative DNS servers. [16]**

server implementations is even more important for DNS server operators due to the before mentioned cost and DoS mitigation issues. Therefore, we have set a threefold goal.

1. To provide DNS operators with ready to use measurement results to support their selection of the most proper authoritative DNS server implementation for their needs.
2. To assist researchers and DNS64 server developers in finding a suitable authoritative DNS server implementation for their DNS64 measurements depending on their required testing rate and available hardware resources.
3. To advance the theory and practice of benchmarking DNS servers.

In this paper, we examine the performance of four authoritative DNS server implementations (BIND, NSD, Knot DNS, and YADIFA) under different conditions including zone files of various sizes and different number of CPU cores.

The remainder of this paper is organized as follows. Section II surveys the available methods for benchmarking authoritative DNS servers and points out that the RFC 8219 compliant one suits better for the needs of DNS server operators than the other examined ones. Section III gives an introduction to the used measurement program, discloses our considerations behind the selection of the DNS server implementations to be tested, presents the test setups, and explains the types of measurements as well as their most important parameters and settings. Section IV discloses and evaluates our high number of results. Section V contains a brief discussion and our future plans. Section VI concludes our paper.

II. METHOD FOR BENCHMARKING AUTHORITATIVE DNS SERVERS

A. SURVEY OF RELATED WORKS

We were looking for both a standard method for benchmarking authoritative DNS servers and research papers with performance measurement results of the major authoritative DNS server implementations, but we have not found any of them. The majority of our hits were either about how to measure the performance of DNS servers operating at different parts of the world, like Sekiya et al. [10], or about the performance comparison of the operating (public) DNS servers, like Kesavan [11]. We have found only few related research papers. One of them gives a survey of the related papers, and it shows (in its Section 3.1) that they do not contain usable performance measurement results of different DNS server implementations [12]. It also summarizes the then (in 2014) available DNS performance measurement techniques in three groups: DNS performance testing software tools, traffic generator hardware appliances, and general purpose testing software. It mentions three DNS performance testing software tools: **`dnsperf`** and **`resperf`** of Nominum and **`queryperf`** of ISC and it also discloses their limitations as well as the limitations of the two other types of solutions.

Possible other methods include the use of **`tcpreplay`** for sending DNS queries on the basis of a **`pcap`** file and **`tcpdump`** for collecting the replies [13]. Although this method was actually used for benchmarking DNS resolvers, it can also be used for benchmarking different authoritative DNS server implementations.

Several software tools for DNS performance or security testing and also some benchmarking tests with actual results are listed at this web page [14].

We have found some technical publications (not peer reviewed research papers) describing practically usable methods for benchmarking authoritative DNS servers and also containing measurement results.

One of them is an NLnet Lab article from 2013 [15]. They used 5 computers to replay DNS queries at predefined rates and to collect the replies by **`tcpdump`**. They compared the performance of NSD to that of BIND, Knot DNS and YADIFA, and characterized their performance by the *proportion of the successfully answered queries* as a function of the query rate. As no timeout value was mentioned, we believe that they were concerned only if a query was answered or not. Unfortunately, their results are outdated today.

Another one is a still ongoing project and a still updated online article of the developers of Knot DNS aimed to compare the performance of several authoritative DNS servers [16]. They used the same measurement method as in [15] thus also lacking of timeout value. They characterized the performance of the authoritative DNS servers by their *response rate*. The results are graphs showing the number of received answers as a function of the number of sent queries, as illustrated in Fig. 1. (Alternatively, the results can also be viewed as response rate percentage, as in [15].) Whereas we admit that these graphs can be useful for DNS server developers, we contend that for a DNS operator, it is redundant, whether a given DNS server can answer 40% or 80% of the queries at a given rate: the DNS server is unusable in both cases. We consider that the lack of timeout value is another serious problem from the DNS operators' point of view: a response is completely useless for the users if it comes more than a second later than the query was sent: *the client software will timeout and resend the query*[1]. Although we contend that the results of our measurement method described in Section II.B are more suitable for DNS operators than that of the method used in [16], we have found their results valuable and we used them in Section III.B.

The developers of the YADIFA DNS server implementation have also published their performance comparison results of YADIFA 1.0.0, NSD 3.2.10, Knot DNS 1.0.5, and BIND 9.9.1. [17]. They also used **`tcpreplay`** and no timeout was mentioned. The date of their measurements has not been disclosed, but their results are now outdated due to the old software versions and obsolete hardware.

Section 9 of RFC 8219 [7] defined a benchmarking methodology for DNS64 servers in 2017. Please refer to [8] for more details and considerations behind the method described in the RFC. They both state that during the self-test of the Tester, the *AuthDNS* (authoritative DNS server) *subsystem* of the Tester should be benchmarked with the same method as the DNS64 servers, but with a different timeout value. Thus it can be said that RFC 8219 has implicitly defined a benchmarking method for authoritative DNS servers (although it was intended for a special purpose only). As for tools, none of the before mentioned ones comply with the requirements of RFC 8219. As far as we know, the only RFC 8219 compliant DNS/DNS64 tester is Dániel Bakai's **`dns64perf++`** [18].

B. SELECTED BENCHMARKING METHOD

To support DNS64 benchmarking with our results, we had to perform RFC 8219 compliant benchmarking measurements. We shall examine, whether the conditions of RFC 8219 are appropriate, when the results are intended to support DNS operators in their authoritative DNS server selection.

The measurement procedure of the DNS64 benchmarking test defined in RFC 8219 follows RFC 2544 both in its wording and in its spirit:

- The Tester sends AAAA record requests at a constant rate for at least 60 seconds, it receives the replies, and checks if they are valid (they contain an AAAA record and arrived within timeout time).
  - If the number of valid replies is equal with the number of queries sent, then the query rate is increased and the test is rerun.
  - If the number of valid replies is less than the number of queries sent, then the query rate is decreased and the test is rerun.

In practice, a binary search is used to find the highest rate at, at which the number of valid replies is equal with the number of queries sent (as it is usually done by RFC 2544 compliant commercial testers, too).

For benchmarking DNS64 servers, the timeout time was chosen as 1 second, and 0.25s was specified for benchmarking the authoritative DNS server subsystem during the self-test of the Tester [8]. Let us consider, whether this 250ms timeout is a good one, when our results are intended to support DNS operators. Although, DNS is used by many types of applications, in a typical case, it is used for resolving domain names for web browsers. A typical URL looks like "http://acme.com", and the resolver (local caching only DNS server) has already cached the IP address of the DNS server responsible for the "com" domain, thus only one request and reply is needed. In this case, the maximum 250ms response time of the authoritative DNS server is acceptable. There may be cases, when multiple subdomains are used (like in the case of the URL: "http://www.hit.bme.hu") and thus more requests and replies are needed, thus a shorter timeout is required, therefore, we have checked our results using 100ms timeout, too.

The absolutely 0% loss required by RFC 8219 results is no problem, when the results are intended to support DNS operators, but this criterion might be too strict. In some cases, we also apply other criteria, which allow e.g. 0.01% or 0.1% packet loss.

As for measurement traffic, RFC 8219 requires all different queries to eliminate the effect of caching. We followed this approach, which often required the usage of huge zone files, see Section III.D.2 for details.

III. MEASUREMENTS

In this section, first, we give an introduction to **`dns64per++`**, the measurement program used for testing, second, we give our considerations, why the given four DNS implementations were selected, third, we describe our different measurement setups, finally, we disclose the details of our measurements.

A. INTRODUCTION TO *DNS64PERF++*

A detailed description of the original **`dns64perf++`** measurement program can be found in our open access paper [18], thus we mention only a few things, which are necessary to understand the rest of our current paper. However, we must give somewhat deeper description of its new properties, which were developed later and have not been published yet.

[1] The one second timeout is our experimental result, measured by using Firefox under Windows 7 [8] and now confirmed under Windows 10. Other software may behave somewhat differently, however, "human timeout" (that is, our patience) does not allow significantly higher timeout value than a few seconds for the majority of the client software.

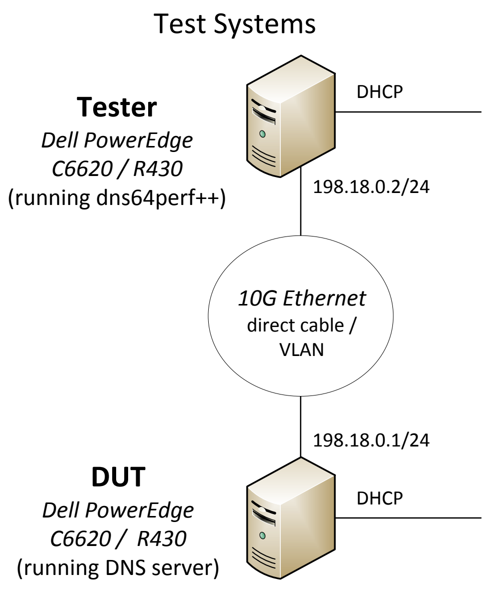

FIGURE 2. **Measurement setup for all types of measurements.**

The **dns64perf++** program has been written in C++14 and it is a command line tool running under Linux. It performs one test: it sends queries at the required rate, receives and validates the replies. The binary search is done by a bash shell script, which performs it at least 20 times as required by RFC 8219.

To be able to send all different queries during a given test, **dns64perf++** uses the following independent namespace: {000..255}-{000..255}-{000..255}-{000..255}.dns64perf.test.

When DNS64 testing is done, a subset of this namespace is used, and it has to be resolved to the corresponding IPv4 addresses by authoritative DNS server. The namespace to be used by the tester is described by the corresponding IPv4 network using CIDR (Classless Inter-Domain Routing) notation. For example, 10.0.0.0/8 identifies the following namespace:
10-{000..255}-{000..255}-{000..255}.dns64perf.test.

As **dns64perf++** sends queries for AAAA records, when an authoritative DNS server is benchmarked (called the "self-test of the tester" in RFC 8219 terminology), the authoritative DNS server has to be configured to provide AAAA records.

In this paper, we refer to the size of the zone file with the mask of the corresponding IPv4 network, e.g. "/8".

The original test program used a self-correcting timing algorithm, which caused significant inaccuracies at higher than 50,000qps (queries per second) rates. This problem was investigated and the timing algorithm was replaced by a simpler one [19], which is now included in its mainline version [20].

The original test program used only two threads, one for sending and one for receiving. Now, it can use *n* times two threads, providing much higher performance. It is also able to use a high number of different source ports for sending the queries, which proved to be a prerequisite of benchmarking at high query rates, see Section III.E.1. Originally, it used IPv6 and as transport protocol (for sending queries and receiving replies), which was completely adequate for benchmarking DNS64 servers. Now it can use also IPv4, which permits testing up to higher rates.

During our preliminary measurements, we have observed that its threads were moved among the CPU cores by the scheduler during the execution of the tests. Therefore, we have added a new feature that sets the affinity of the threads so that the sender threads and the receiver threads are pinned to cores 0 – (*n*-1) and to cores *n* – (2*n*-1), respectively. The modified files are available from [21].

We note that **dns64perf++** was designed to work in two phases. First, the requests are sent and the replies are received and the relevant information (e.g. sending and receiving timestamps) are stored in huge arrays. This is done by *n* times two threads. Then, the evaluation is done as a sequential process. At high rates (e.g. 1-2 million queries per second), the execution time is dominated by the second phase. Thus, the execution time highly depends on the actual rates. For example, when the performance of a DNS or DNS64 server is a few times 10,000qps, the initial range can be [0, 50,000]. The execution time of the required 20 repetitions of the binary search consisting of 16 steps each is typically 6-8 hours. On the other hand, when the performance of a DNS server is about 3,000,000qps, the initial range can be [0, 3,300,000]. The execution time of the required 20 repetition of the binary search consisting of 22 steps each is about a day. (Both, because the binary search requires more steps, and because the execution of a 60s long test lasts for several minutes due to the sequential evaluation process.) These numbers are important, when a high number of measurements are required e.g. due to testing with all possible parameter combinations.

B. SELECTED AUTHORITATIVE DNS SERVER IMPLEMENTATIONS

As for authoritative DNS server implementations to be tested, we have considered only free software [22] (also called open source [23]) for the same reasons given in [24].

ISC BIND [25] is the de facto industry standard DNS server, thus even if we knew that other implementations had higher performance, we considered important to include it.

NSD [27] was selected because of our good experience with it: its single core performance was found higher than the 16-core performance of BIND [9].

Knot DNS [26] was selected on the basis of the performance measurement results of its developers [16].

YADIFA [28] was included because RFC 8219 mentions it, and according to our previous experience it outperformed BIND [9].

As for further free software DNS implementations, we have also considered PowerDNS [29], but we did not select it. The results of the Knot DNS developers showed that PowerDNS could not comply with the 0% loss criterion of RFC 8219 even at 100,000qps rate (please see the original, interactive version of our Fig. 1: it could answer only 99,450 queries per second [16], and it produced 1.7% loss at 400,000qps rate, which we consider unacceptable for authoritative DNS server function). Unbound is also a well-known high performance recursive DNS server, but it does not have an authoritative DNS server function.

C. TEST SYSTEMS FOR BENCHMARKING

The benchmarking experiments were carried out in the NICT StarBED, Japan. We used two types of servers (*N nodes* and *P nodes*), because they both had their advantages over the other one. The N nodes were Dell PowerEdge C6620 servers with 16 physical cores, and their clock frequencies could be set to a fixed value, which was an important advantage [8]. The P nodes were Dell PowerEdge R430 servers with 32 physical cores, which was also important, when the performance was examined as a function of the number of CPU cores.

We used four test systems with the same logical construction as shown in Fig. 2. The components of the four test systems are detailed in Table 1.

As we have pointed out performance differences among our three test systems consisting N nodes during our earlier tests [9], we performed the same kinds of tests of each examined DNS implementation using the very same computers and used the different test systems for different kinds of tests.

As for the settings of the computers, hyper-threading was disabled in the BIOS of all computers to ensure stable results [8]. To make always the DUT (Device Under Test) the bottleneck, Turbo Mode was disabled in the DUTs and enabled in the Testers. The clock frequencies of the N series DUTs (n015 and n018) was set to fixed 2GHz using the same BIOS setting as in [9]. However, we could not do the same with the P series DUTs (p102 and p104), finding no such settings in their BIOS.

TS2 (Test System 2) and TS3 were our primary testbeds, which we used for the scale-up tests. TS1 and TS4 were their “back-ups”, they were added to offload some measurements (e.g. zone file size tests and also scale up tests of DNS servers with lower performance) from TS2 and TS3 and thus speed up experimentation.

We need to mention another difference between the two types of nodes: their cores are enumerated differently, which also influences their apparent NUMA (Non-Uniform Memory Access) architecture. In the N series nodes, CPU cores 0-7 (called cpu0 - cpu7 under Linux) belong to the first physical CPU and NUMA node 0, and cores 8-15 belong to the second physical CPU and NUMA node 1. In the P series nodes, the even number cores (cores 0, 2, 4, … 28, 30) belong to the first physical CPU and NUMA node 0, and the odd number cores (cores 1, 3, 5, … 29, 31) belong to the second physical CPU and NUMA node 1. Thus, in both cases, the first physical CPU is NUMA node 0 and the second physical CPU is NUMA node 1, the only difference is the order, in which the cores are enumerated by the Linux kernel. However, this order may make a difference, when the first *n* number of cores are enabled during the tests (see its effect for the scale up tests in Section IV).

TABLE I
THE BUILDING ELEMENTS OF THE TEST SYSTEMS

Test System	Tester node	Tester speed	DUT node	DUT speed	connection
1	n014	2-2.4GHz	n015	2GHz	direct cable
2	n017	2-2.4GHz	n018	2GHz	direct cable
3	p101	1.2-2.6GHz	p102	1.2-2.1GHz	VLAN 3251
4	p103	1.2-2.6GHz	p104	1.2-2.1GHz	VLAN 3253

D. TYPES OF TESTS AND THEIR MOST IMPORTANT PARAMETERS

The performance of the examined four authoritative DNS server implementations may depend on several factors, such as the number of active CPU cores of the DUT, the size of the zone file, the timeout value, and the hardware type of the DUT. The number of all combinations of their possible values were too high to use them all. Hence, we used those combinations, which we found meaningful during our preliminary tests.

1) SCALE UP TEST

The aim of these tests was to examine, how the performance of the four tested DNS server implementations scaled up as the number of CPU cores were increased. Following our earlier practice, we doubled the number of active CPU cores starting from one (instead of increasing them one by one) to reduce the necessary number of tests [9].

In order to support RFC 8219 compliant DNS64 benchmarking tests, we needed to ensure all different domain names for a 60s long test at all used query rates. It means that we had to choose large enough zone files for our tests. (Table 2 shows the number of entries and the maximum usable query rate of a 60s long test as a function of the zone file size.) To fulfill this requirement, we performed preliminary measurements to assess the performance of the DNS implementations using various number of CPU cores in order to be able to determine the necessary zone file size for the scale up tests. Both Knot DNS and NSD performed over 1.5Mqps and 2.5Mqps using TS2 and TS3, respectively. Therefore, we had to use /5 and /4 size zone files for their scale up tests on TS2 and TS3, respectively. Our preliminary tests showed that BIND and YADIFA could not achieve higher performance than

TABLE II
THE MAXIMUM QUERY RATE AS A FUNCTION OF THE ZONE FILE SIZE

Size	Number of entries	Maximum query rate
/11	2097152	34952
/10	4194304	69905
/9	8388608	139810
/8	**16777216**	**279620**
/7	33554432	559240
/6	67108864	1118481
/5	**134217728**	**2236962**
/4	**268435456**	**4473924**

260,000qps, thus a /8 size zone file was enough for them and their performance was measured using T1 and TS4.

Generally, the zero loss criterion of RFC 8219 was used, but in some cases we have experienced that the results of the 20 measurements were very much scattered, and when we have examined the measurement log files, we saw that the tests failed due to a low number of missing replies. In these cases we also used 99.99% valid answers as acceptance criterion.

2) ZONE FILE SIZE TEST

The aim of these tests was to examine, how the size of the used zone file influenced the performance of the four tested DNS server implementations. We performed these tests by doubling the size of the zone file (four times) from /8 to /4. In addition to that, the low performance of BIND allowed it to be tested with smaller zone files (from /11 to /9), too. For these tests, we used TS1 and TS4.

Our approach was to execute these tests using only a single CPU core, to reduce the performance of the tested DNS servers thus enabling the usage of smaller zone files. It was successful with both BIND and NSD. However, the single core results of Knot DNS and YADIFA were too much scattered, thus they were unsuitable for examining how the size of the used zone file influenced the performance of these DNS implementations.

For testing Knot DNS, we used 99.99% and 99.9% valid answers as acceptance criterion with TS1 and TS4, respectively.

For testing YADIFA, we used eight CPU cores, as its performance was still moderate enough allowing the usage of a “/8” size zone file.

3) TIME-OUT TESTS

The aim of these tests was to examine, how the specified timeout value influenced the performance of the four tested DNS server implementations. Our preliminary results showed that the applied 250ms and 100ms values resulted in very little differences (please see the actual values later), thus only a few tests were performed to check the validity of this observation in various critical working points. (All four tests systems were used.)

E. OTHER RELEVANT PARAMETERS

1) USAGE OF HIGH NUMBER OF DIFFERENT SOURCE PORT NUMBERS

When authoritative DNS servers are used in real life, the requests usually arrive from different source IP addresses and also from different source ports. (Even if the majority of the requests comes from a few recursive servers, the source port numbers used by a given recursive DNS server must be different to comply with the requirements of the RFC 5452 [30].)

When authoritative DNS servers are used to support DNS64 benchmarking, the requests are coming from the same IP address, but the source ports should be still different for the same reason as above. (In the case of the major DNS64 implementations, namely BIND, PowerDNS and Unbound [31], we have checked that they complied with this rule [32].)

Neither RFC 8219, nor our paper on the benchmarking methodology for DNS64 servers [8] mentions the usage of high number of different source port numbers as requirement for the Tester, but now we contend that they should have done so. We consider it important for both *theoretical* and *practical* point of view.

Under “theoretical” we mean that the usage of high number of different source ports is necessary for proper testing, because the tested DNS / DNS64 implementations may use the **`SO_REUSEPORT`** socket option for distributing the traffic among their multiple threads or processes [33]. (This socket option is supported since Linux kernel version 3.9.) A Tester without the capability of using a high number of source ports is simply not suitable for benchmarking of those DNS / DNS64 implementations that use the aforementioned socket option.

Under “practical” we mean that multi-core operating systems may use also source port numbers in the hash function to distribute the interrupts evenly among the CPU cores, which is a prerequisite for receiving several millions of packets per second [34]. We used the following command to distribute the interrupts evenly among the CPU cores:

```
ethtool -N interface rx-flow-hash udp4 sdfn
```

In this case, the source and destination IP addresses and port numbers are included in the hash function. As the other three numbers are constants during the tests, the interrupts cannot be distributed among the CPU cores without using a high number of different source port numbers and the capacity of the single core used by the interrupts becomes a bottleneck.

In all our measurements, we used the highest possible number of threads in **`dns64perf++`**, it means that 8 thread pairs on n014 and n017, and 16 thread pairs on p101 and p103. We used 4,000 different port numbers by each sending threads of **`dns64perf++`**, thus altogether 32,000 or 64,000 different source ports were used. (We note that the mainline version of **`dns64perf++`** starts the source port numbers from 10,000, thus we have changed it to 1024 in the source code.)

2) TO OPTIMIZE OR NOT TO OPTIMIZE?

On the one hand, one can argue that it is fair to consider the very best results of each tested DNS server implementations, suggesting that they should be optimized for benchmarking. This would include their recompilation and fine tuning.

However, on the other hand, this could be a never ending game: different tests would require different settings, and one could never be sure, if the best performance has already been found or not. Such optimization would also require a very deep knowledge of all tested implementations, which we do not have. Our most important argument against such tuning is that the results would be irrelevant for the majority of their users because the users usually use them as they are included in their favorite Linux distribution.

Therefore, we also used them as they were installed using Debian 9.6.

3) CONFIGURATION SETTINGS OF THE DNS SERVERS

In general, we have made only the absolutely necessary changes to the default configuration files of the DNS servers, which means the setting of the zone name and zone file. In the case of NSD, we had to make further changes, because otherwise it would have used only a single CPU core. The new settings were:

```
server:
  server-count: n # = no. of active cores
  reuseport: yes  # enable SO_REUSEPORT
```

The other three DNS implementations automatically used multiple threads.

4) SETTING THE NUMBER OF ACTIVE CPU CORES AT THE DUT

The number of active CPU cores at the DUT was set by using the **maxcpus=*n*** kernel parameter.

We note that first, we tried using the method for switching the CPU cores on and off on the fly described in [9], but then we have received scattered measurement results using the N nodes. Then we used the above mentioned method (with rebooting the operating system), but there were still problems with the scattered results. Then our colleague, Gábor Horváth, who teaches Computer Architecture at the Budapest University of Technology and Economics, advised us to completely power off the node (not only reboot it). It was done by using the “Hard Reset (Restart)” power control action of the “Dell Remote Manager Controller” of the given N node, which has solved the issue. We have used this power control action always, when the number of active CPU cores were changed. We have not tested whether it was necessary or not, rather we used its equivalent “Power Cycle System (cold boot)” with the P nodes. We plan to investigate this phenomenon later on.

5) HARDWARE PARAMETERS AND SOFTWARE VERSION NUMBERS

For the repeatability of our results, we give the most important hardware parameters and software version numbers.

The N nodes were Dell PowerEdge C6620 servers with two Intel Xeon E5-2650 2GHz CPUs, having 8 cores each, and 16x8GB 1333MHz DDR3 RAM. We used one of their Intel 10G 2P X520 (fiber) network adapters.

The P nodes were Dell PowerEdge R430 servers with two Intel Xeon E5-2683 v4 2.1GHz CPUs, having 16 cores each, and 12x32GB 2400MHz DDR4 RAM. We used one of their Intel 10G 2P X540 (copper) network adapters.

The version numbers of the tested DNS servers were the following:

- BIND 9.10.3-P4-Debian
- NSD 4.1.14
- Knot DNS 2.4.0
- YADIFA 2.2.3-6237

The earlier installed Debian Linux systems were upgraded to 9.6 on all nodes. As the update of Debian does not update the Linux kernel, the kernel release was 4.9.0-4-amd64 and 4.9.0-8-amd64 on the N nodes and on the P nodes, respectively.

As for **dns64perf++** [20], its multiport branch was used (commit d6fa119 on Oct 8 2018) with our aforementioned modifications (adding affinity, and starting the source ports from 1024). It was compiled by clang 3.8.1-24 enabling packet sending over IPv4 by using the “IPV4=1” make parameter.

IV. RESULTS AND EVALUATION

During the presentation and discussion of the results, first, we focus on the behavior each DNS server separately, and compare them in the end.

We usually begin the discussion of each DNS server with some general information about it. Next, the scale up test results are presented and discussed, and we deal with the zone file size test after them. The timeout test is not handled separately, it was rather integrated with the scale up and/or the zone file size tests.

As for summarizing function of the results of the 20 measurements, RFC 8219 requires to use median, and as for index of dispersion of the results, it requires the presentation of 1st percentile and 99th percentile, which are the minimum and maximum values, when we have less than 100 measurement results. When the given authoritative DNS implementation is intended to be used to support DNS64 benchmarking, then the 1st percentile should to be taken into consideration, so that the insufficient performance of the authoritative DNS server may not impact the DNS64 measurement results. When the results are intended to support DNS server operators, then we recommend to use the median.

In [9], we have introduced another measure as follows:

$$dispersion = \frac{99^{th}\ percentile - 1^{st}\ percentile}{median} \cdot 100\% \quad (1)$$

TABLE III
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF THE NUMBER OF CPU CORES, BIND, **"/8"**, **TS1**, 0.25S *(0.1S)*

Num. CPU cores	1	2	4	8	16	***1***	***16***
Median (qps)	19792	38928	37596	77244	140372	*17409*	*140720*
1st percentile (qps)	19477	37499	37565	76166	138621	*16383*	*137448*
99th percentile (qps)	19794	38963	37604	77539	141119	*17411*	*141358*
Dispersion (%)	1.60	3.76	0.10	1.78	1.78	*5.90*	*2.78*

TABLE IV
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF THE NUMBER OF CPU CORES, BIND, **"/8"**, **TS4**, 0.25S *(0.1S)*

Num. CPU cores	1	2	4	8	16	32	***1***	***32***
Median (qps)	28205	44082	47537	85816	153736	259935	*28203*	*259528*
1st percentile (qps)	28067	43895	43749	74168	152928	253127	*27977*	*249999*
99th percentile (qps)	29688	44269	48597	88090	156311	264362	*28640*	*265747*
Dispersion (%)	5.75	0.85	10.20	16.22	2.20	4.32	*2.35*	*6.07*

TABLE V
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF THE NUMBER OF CPU CORES, BIND, **"/5"**, **TS2**, 0.25S

Num. CPU cores	1	2	4	8	16
Median (qps)	8386	20798	18403	38286	68514
1st percentile (qps)	8383	18733	18067	37505	65624
99th percentile (qps)	8388	20827	18421	38910	68994
Dispersion (%)	0.06	10.07	1.92	3.67	4.92

TABLE VI
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF THE NUMBER OF CPU CORES, BIND, **"/4"**, **TS3**, 0.25S

Num. CPU cores	1	2	4	8	16	32
Median (qps)	8325	15424	9342	22455	48728	75654
1st percentile (qps)	8325	12492	9334	21751	47494	75324
99th percentile (qps)	8325	15426	9345	23444	49323	75780
Dispersion (%)	0.00	19.02	0.12	7.54	3.75	0.60

It can be used to judge the quality of the results. If it is low (e.g. below 5%) then the results are consistent. Its higher and higher values indicate more and more scattered results.

Unfortunately, scattered results may be either an inherent property of a given DNS server implementation, or they may come from somewhere else and thus indicate for example a hardware issue or even a bug or performance deficiency in the measurement software, **`dns64perf++`**. After the evaluation of the results, we show that the performance of **`dns64perf++`** is definitely enough up to 3.3 million queries per second rate, when it is executed by a P node with Turbo Mode enabled.

A. BIND

Before the presentation of the results, we need to touch an important feature of BIND. When BIND is started, it provides several pieces of useful information through syslog. Among others, it writes the following two lines:

```
found n CPUs, using n worker threads
using m UDP listeners per interface
```

The number of the worker threads equals the number of the active CPU cores, but it uses a special heuristic to set the number of the UDP listeners.

- When there are 1 or 2 active CPU cores, then the number of the UDP listeners equals the number of the CPU cores.
- When the number of the active CPU cores is 4 or higher, then the number of the UDP listeners equals the half of the number of the CPU cores.

This heuristic has significant consequences on the performance of BIND.

1) SCALE UP TEST

The authoritative DNS server performance results of BIND as a function of the number of active CPU cores using a "/8" size zone file measured by TS1 (Test System 1, the DUT is an N node) are presented in Table III. The performance of BIND visibly scales up very well from one to two cores, but there is conspicuous glitch at four cores. The bottleneck is deliberately the number of UDP listeners (two). At higher number of cores, the performance of BIND scales up well again. The five performance results measured using 250ms timeout are complemented with two results measured using 100ms timeout. Let us consider first the single CPU core result in the last but one column of the table. Due to the smaller timeout value, the median has decreased by 12% from 19,792qps to 17,409qps and the dispersion of the results increased from 1.6% to 5.9%. As for the result of the 16-core test with 100ms timeout, the value of the median did not change significantly (the slight increase must be a measurement error), only the dispersion increased from 1.78% to 2.78%. The second behavior can be explained by the fact that with four cores and above, the

TABLE VII
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF ZONE FILE SIZE, BIND, 1 CPU CORE, **TS1**, 0.25S

Num. CPU cores	/11	/10	/9	/8	/7	/6	/5	/4
Median (qps)	23520	24029	21437	19251	11729	14014	9374	6849
1st percentile (qps)	23484	23436	21406	18749	11729	13961	9370	6849
99th percentile (qps)	23560	24072	21485	19287	11731	14026	9374	6849
Dispersion (%)	0.32	2.65	0.37	2.79	0.02	0.46	0.04	0.00

TABLE VIII
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF ZONE FILE SIZE, BIND, 1CPU CORE, **TS4**, 0.25S

Num. CPU cores	/8	/7	/6	/5	/4	/3
Median (qps)	28645	22286	20450	13267	13809	4300
1st percentile (qps)	28124	22067	20435	13084	12352	4168
99th percentile (qps)	28839	23439	21100	13267	13811	4300
Dispersion (%)	2.50	6.16	3.25	1.38	10.57	3.07

bottleneck is the number of receivers, and thus if a request is successfully received, then it will be replied soon, thus the smaller timeout does not influence the achievable rate, which is also confirmed by our TS4 results below. (We mean it for the median. Of course, random events in the measurement system may influence the 1st percentile more, when the timeout is smaller).

The results measured by TS4 (the DUT is a P node) are presented in Table IV. Similar tendencies can be observed: BIND scales up well, and there is a glitch at 4 cores. However, there are significant differences, too. Especially at 4 and 8 cores, there is high (more than 10%) dispersion, which is also significant (more than 5%) at 1 core. The high dispersion could be attributed to the varying CPU clock frequency of the P nodes, but the dispersion is low (less than 1%) at two cores. The last two columns of the table show our results with 100ms timeout using 1 or 32 cores. The decrease of the medians is negligible in both cases (in our opinion it is very likely less than the error of the measurements).

Comparing the results of TS1 and TS4, we can see that the performance gain of the newer system highly depend on the number of CPU cores used. With a single core, the median performance grows by 42.5% from 19,792qps to 28,205qps, whereas the increase from 140,372qps to 153,736qps is only 9.5% with 16 cores, which we consider inconsistent.

We have executed the scale up test measurements also with a “/5” size zone file using TS2. The results are shown in Table V. The tendencies are very similar to that of the results produced by TS1, since both DUTs were N nodes.

The results measured by TS3 using a “/4” size zone file are presented in Table VI. Here, the situation is even worse than in the case of TS4 in two aspects.

1. There is a high dispersion at 2 cores, which is caused by a single outlier. We have performed this test 4 times, and always there was a single outlier, which fell in the 12,300qps – 15,500qps range.
2. The performance sharply falls back at 4 cores.

We attribute this phenomenon to design problems of BIND and did not invest any more effort into its investigation for the following reasons:

1. Similar problem is identified in the next subsection.
2. As we have pointed it out in [9], BIND had also a serious performance problem, when it was used as a DNS64 server. (Its performance did not scale up over 4 cores at all. We have reported it to the developers as [ISC-Bugs #46924] in 2017, but we have not received any reply so far.)
3. As it is shown later in this paper, BIND was significantly outperformed by other DNS implementations.

Thus, we believe that it is not worth the effort to do a deeper analysis of the anomalies of BIND.

2) ZONE FILE SIZE TEST

The authoritative DNS server performance results of BIND as a function of the size of the zone file measured by TS1 using a single CPU core are presented in Table VII. The overall tendency is exactly, what we expected on the basis of the previous results: the performance decreases as the size of the zone file increases. It can be easily explained by computer architectural causes. Considering, that BIND uses somewhat more than 4GB memory, when it loads a /8 zone file and the CPU has only 20MB cache, the explanation has nothing to do with caching but the reason can be the decreasing TLB (Translation Lookaside Buffer) coverage.

However, similarly to the scale up test, we can also observe a glitch: when the size of the zone file is doubled from “/7” to “/6” the performance shows a significant increase. We surmise that there can be some kind of technology change behind, which is somewhat ill positioned by an inappropriate heuristic. (Such as changing form a linked list representation to a B-tree representation at too high number of elements in date storage and retrieval. But that is intended to be a simile only, nothing more.)

The results measured by TS4 are presented in Table VIII. Similar tendencies can be observed: the performance globally decreases as the zone file size increases, but there is a glitch at the “/4” size zone file. We have also performed

TABLE IX
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF THE NUMBER OF CPU CORES, NSD, **"/5"**, **TS2**, 0.25S *(0.1S)*

Num. CPU cores	1	2	4	8	16	***16***
Median (qps)	177432	327260	615192	1062615	1454661	*1453490*
1st percentile (qps)	176512	324999	599950	999999	1399999	*1399974*
99th percentile (qps)	178126	328828	619800	1160155	1500001	*1500001*
Dispersion (%)	0.91	1.17	3.23	15.07	6.87	*6.88*

TABLE X
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF THE NUMBER OF CPU CORES, NSD, **"/5"**, **TS2**, 0.25S *(0.1S)*
ACCEPTANCE CRITERION: 99.99%, NON-RFC 8219 COMPLIANT!

Num. CPU cores	1	2	4	8	16	***16***
Median (qps)	177735	327515	617180	1089275	1537105	*1538946*
1st percentile (qps)	177342	324999	612108	1065220	1523387	*1524971*
99th percentile (qps)	178130	328321	619674	1168751	1550318	*1553129*
Dispersion (%)	0.44	1.01	1.23	9.50	1.75	*1.83*

TABLE XI
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF THE NUMBER OF CPU CORES, NSD, **"/4"**, **TS3**, 0.25S

Num. CPU cores	1	2	4	8	16	32
Median (qps)	166715	268721	405410	802359	1552845	2442195
1st percentile (qps)	165226	190624	387499	734374	1413446	2085936
99th percentile (qps)	168909	275079	425001	817398	1665530	2812523
Dispersion (%)	2.21	31.43	9.25	10.35	16.23	29.75

TABLE XII
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF THE NUMBER OF CPU CORES, NSD, **"/4"**, **TS3**, 0.25S *(0.1S)*
ACCEPTANCE CRITERION: 99.99%, NON-RFC 8219 COMPLIANT!

Num. CPU cores	1	2	4	8	16	32	***32***
Median (qps)	178255	277996	448608	811774	1740020	3099333	*2954180*
1st percentile (qps)	174999	198338	437499	799950	1656004	3074752	*2936185*
99th percentile (qps)	181446	282940	457032	818754	1752930	3125013	*2966732*
Dispersion (%)	3.62	30.43	4.35	2.32	5.57	1.62	*1.03*

a measurement with a "/3" size zone file and its performance result was about one third of the performance measured with a "/4" size zone file.

B. NSD

Unlike the other three authoritative DNS server implementations, NSD does not use multiple threads, it rather uses multiple processes. We set the **`server-count`** value always to the number of the active CPU cores. When NSD was started using *n* number of active CPU cores, NSD always started *n*+2 processes listening on port 53, however, only *n* of them were taking part in the service of the DNS queries.

1) SCALE UP TEST

The authoritative DNS server performance results of NSD as a function of the number of active CPU cores using a "/5" size zone file measured by TS2 are presented in Table IX. NSD scales up well up to four CPU cores. However, significant problems can be observed at 8 cores, were the dispersion of the results is 15.07%. We have investigated its cause and found that some of the tests failed due to very small differences between the number of the sent requests and the number of the valid answers (less than 0.01%). Therefore, we have repeated our tests with the 99.99% acceptance criterion. The results, which are shown in Table X, confirmed our hypothesis: the dispersion has decreased at any number of cores, although in a different measure. For the results of 1-4 cores, the performance increase over the results in Table IX is very small (below 3% concerning any of the values). Although the increase of the 1st percentile is significant at 16 cores, it does not really matter, because DNS64 benchmarking, for which the 1st percentile is used, does not tolerate packet loss. The increase of the median, which we consider important for DNS operators is only 5.67% (from 1,454,661qps to 1,537,105qps).

The authoritative DNS server performance results of NSD as a function of the number of active CPU cores using a "/4" size zone file measured by TS3 are presented in Table XI. Unfortunately, the results of the newer and higher performance DUT are lower than that of the older one from 1 to 8 cores, but the situation changes at 16 cores. For an easier comparison of the performance of the two systems we have performed our measurements with the 99.99% acceptance criterion (to reduce the dispersion of the results). The results are shown in Table XII. Unfortunately the dispersion remained very high at 2 cores (30.43%), which was caused by the low 1st percentile value

TABLE XIII
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF ZONE FILE SIZE, NSD, 1 CPU CORE, **TS1**, 0.25S *(0.1S)*

Num. CPU cores	/8	/7	/6	/5	/4	***/8***
Median (qps)	184468	178115	178905	184019	181240	*184498*
1st percentile (qps)	184031	176951	177733	182420	179686	*184288*
99th percentile (qps)	184772	178517	179701	184571	181604	*184741*
Dispersion (%)	0.40	0.88	1.10	1.17	1.06	*0.25*

TABLE XIV
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF ZONE FILE SIZE, NSD, 1CPU CORE, **TS4**, 0.25S *(0.1S)*

Num. CPU cores	/8	/7	/6	/5	/4	***/8***
Median (qps)	165594	166067	167712	162409	163195	*167873*
1st percentile (qps)	163929	162495	165526	160440	161692	*149999*
99th percentile (qps)	169006	171924	169647	164941	165747	*170568*
Dispersion (%)	3.07	5.68	2.46	2.77	2.48	*12.25*

(198,338qps). The dispersion was low at any other number of cores, thus we could check the effect of the timeout change. When the timeout value was decreased from 250ms to 100ms, the median changed from 3,099,333qps to 2,954,180qps, which is only a 4.7% decline. Now, let us return to the performance comparison of the two types of nodes. We consider the median values of the results of the 99.99% acceptance criterion tests of TS2 and TS3, shown in Table X and Table XII, respectively. The medians are approximately the same at a single core (177,735qps and 178,255qps). At two cores, the result of TS2 is 327,515qps, which is a good scale up (84% growth), whereas the result of TS3 is 277,996qps, which is a significantly lower scale up (only 56% growth). We attribute this difference to the fact that the cores of the two types of CPUs are enumerated in a different order, as we detailed it at the end of Section III.C. It means that core 0 and core 1 belong to the same physical CPU (and NUMA node) in the DUT of TS2, whereas they belong to two different physical CPUs (and NUMA nodes) in TS3. Let us check our hypothesis: what happens, when the CPU (and NUMA) situation changes in TS2 from homogeneous to heterogeneous and it does not change in TS3 (as it is already heterogeneous in both cases). The median grows only by 41% from 8 cores (1,089,275qps) to 16 cores (1,537,105qps) in TS2. The increase of the median from 16 cores (1,740,020qps) to 32 cores (3,099,333qps) is still 78% in TS3, which is nearly the double of the before mentioned 41%. Thus, we consider our hypothesis as confirmed.

2) ZONE FILE SIZE TEST

The authoritative DNS server performance results of NSD as a function of the size of the zone file using a single CPU core measured by TS1 and TS4 are presented in Table XIII and Table XIV, respectively. They are in a complete agreement that the performance of NSD shows no significant decrease as the size of the zone file increases. They both confirm that the 100ms timeout value caused no change in the measured performance comparing to measurements with 250ms timeout value.

C. KNOT DNS

According to the Knot DNS server documentation, the `udp-workers` directive, which should be placed into the `server` section of the configuration file, can be used to set the number of UDP workers (threads). In accordance with our approach disclosed in Section III.E.2, we did not set it, thus its default value was used, which is an "auto-estimated optimal value based on the number of online CPUs" [26].

1) SCALE UP TEST

The authoritative DNS server performance results of Knot DNS as a function of the number of active CPU cores using a "/5" size zone file measured by TS2 are presented in Table XV. Unfortunately, the performance of the Tester was unsatisfactory for the tests with 16 cores (high number of received packets were reported to be lost by the Ethernet interface). For this reason, the values in this column of the table do not reflect the true performance of Knot DNS. We exclude them from the detailed analysis, but we still present them to show that they are higher than the results of NSD. The performance of Knot DNS scales up well up to 8 cores (and very likely up to 16 cores, too) considering both the median and the 1st percentile, but the results are very scattered from 1 to 4 CPU cores, which was caused by a small number of lost replies, as confirmed by our measurements using 99.99% acceptance criterion, shown in Table XVI. In the last column of this table, we included the 100ms timeout values measured with 8 cores (as the results with 16 cores are limited by the performance of the Tester). The lower timeout value does not have a significant influence on the performance of Knot DNS (the very small increase from 1,167,716qps to 1,168,724qps is deliberately a measurement error).

The authoritative DNS server performance results of Knot DNS as a function of the number of active CPU cores using a "/4" size zone file measured by TS3 are presented in Table XVII. The most salient problem is the 99,999qps 1st percentile value at 2 cores. This test was executed three times and this value occurred each time, thus it is not a once happened random event, but an inherent property of Knot DNS, which we must count on if Knot DNS is used for DNS64 benchmarking. However, it is caused by a few missing answers, and thus it is absent from Table XVIII, which shows the result with 99.99% acceptance criterion measurements. Otherwise Knot DNS scaled up well. We would like to point out that although Knot DNS produced highly scattered results with TS2 from 1 to 4 cores, and its results with TS3 are extremely scattered at 2 cores, they are much better with higher numbers of cores. Considering TS3, the dispersion is under 10% from 4 to 32 cores and it is quite low at 4 and 32 cores. We note that its excellent

TABLE XV
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF THE NUMBER OF CPU CORES, KNOT DNS, **"/5"**, **TS2**, 0.25S
(THE RESULTS WITH 16 CPU CORES WERE LIMITED BY THE PERFORMANCE OF THE TESTER!)

Num. CPU cores	1	2	4	8	(16)
Median (qps)	163170	300454	594585	1164253	(1678872)
1st percentile (qps)	137495	224999	449999	1099884	(1562491)
99th percentile (qps)	163901	300977	596876	1166016	(1750001)
Dispersion (%)	16.18	25.29	24.70	5.68	(11.17)

TABLE XVI
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF THE NUMBER OF CPU CORES, KNOT DNS, **"/5"**, **TS2**, 0.25S *(0.1S)*
ACCEPTANCE CRITERION: 99.99%, NON-RFC 8219 COMPLIANT!
(THE RESULTS WITH 16 CPU CORES WERE LIMITED BY THE PERFORMANCE OF THE TESTER!)

Num. CPU cores	1	2	4	8	(16)	***8***
Median (qps)	166514	301323	596910	1167716	(1630054)	*1168724*
1st percentile (qps)	162492	299999	549999	1162495	(1550780)	*1162499*
99th percentile (qps)	167242	301758	598047	1169532	(1750977)	*1171045*
Dispersion (%)	2.85	0.58	8.05	0.60	(12.28)	*0.73*

TABLE XVII
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF THE NUMBER OF CPU CORES, KNOT DNS, **"/4"**, **TS3**, 0.25S

Num. CPU cores	1	2	4	8	16	32
Median (qps)	119039	202235	355285	761424	1500439	2923327
1st percentile (qps)	112474	99999	348411	699999	1484312	2894053
99th percentile (qps)	125049	204369	358319	766632	1633057	2983339
Dispersion (%)	10.56	51.61	2.79	8.75	9.91	3.05

TABLE XVIII
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF THE NUMBER OF CPU CORES, KNOT DNS, **"/4"**, **TS3**, 0.25S *(0.1S)*
ACCEPTANCE CRITERION: 99.99%, NON-RFC 8219 COMPLIANT!

Num. CPU cores	1	2	4	8	16	32	***32***
Median (qps)	142197	235694	444991	839760	1773973	3233456	*3230968*
1st percentile (qps)	124999	228124	437499	835541	1749999	3196773	*3196773*
99th percentile (qps)	144262	238874	453209	845800	1812501	3238372	*3238800*
Dispersion (%)	13.55	4.56	3.53	1.22	3.52	1.29	*1.30*

TABLE XIX
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF ZONE FILE SIZE, KNOT DNS, 1 CPU CORE, **TS1**, 0.25S *(0.1S)*
ACCEPTANCE CRITERION: 99.99%, NON-RFC 8219 COMPLIANT!

Num. CPU cores	/8	/7	/6	/5	/4	***/8***
Median (qps)	162577	164552	156941	164549	161519	*161967*
1st percentile (qps)	157811	163267	149997	149999	159374	*149999*
99th percentile (qps)	162939	164868	157422	164848	161914	*162262*
Dispersion (%)	3.15	0.97	4.73	9.02	1.57	*7.57*

TABLE XX
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF ZONE FILE SIZE, KNOT DNS, 1CPU CORE, **TS4**, 0.25S *(0.1S)*
ACCEPTANCE CRITERION: 99.9%, NON-RFC 8219 COMPLIANT!

Num. CPU cores	/8	/7	/6	/5	/4	***/8***
Median (qps)	191877	186085	191420	190843	184676	*191754*
1st percentile (qps)	191014	185532	190624	187499	184227	*190624*
99th percentile (qps)	192578	186816	191845	191407	185022	*192969*
Dispersion (%)	0.82	0.69	0.64	2.05	0.43	*1.22*

TABLE XXI
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF THE NUMBER OF CPU CORES, YADIFA, **"/8"**, **TS1**, 0.25S

Num. CPU cores	1	2	4	8	16
Median (qps)	133492	176214	195494	209600	147353
1st percentile (qps)	96874	149901	187499	209251	146874
99 percentile (qps)	133930	190872	196094	210181	147852
Dispersion (%)	27.76	23.25	4.40	0.44	0.66

TABLE XXII
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF THE NUMBER OF CPU CORES, YADIFA, **"/8"**, **TS4**, 0.25S

Num. CPU cores	1	2	4	8	16	32
Median (qps)	118366	129681	150536	166416	186528	168315
1st percentile (qps)	95304	85529	85874	149217	149217	149999
99 percentile (qps)	131251	131695	156251	175001	197729	171948
Dispersion (%)	30.37	35.60	46.75	15.49	26.01	13.04

TABLE XXIII
AUTHORITATIVE DNS SERVER PERFORMANCE AS A FUNCTION OF ZONE FILE SIZE, YADIFA, **8** CPU CORES, **TS1**, 0.25S *(0.1S)*

Num. CPU cores	/8	/7	/6	/5	/4	/3	***/8***
Median (qps)	209596	209383	197134	194700	196673	192876	*208538*
1st percentile (qps)	209325	208585	196776	193700	196287	192176	*207615*
99 percentile (qps)	210059	209766	197754	195340	196924	193214	*209376*
Dispersion (%)	0.35	0.56	0.50	0.84	0.32	0.54	*0.84*

results at 32 cores made it possible for us to check the performance of **dns64perf++**, please refer to Section IV.F for more details.

2) ZONE FILE SIZE TEST

Due to the high dispersion of the results of Knot DNS at any number of CPU cores, the zone file size test was executed with the 99.99% acceptance criterion using Test System 1. The bar was lowered to 99.9% with Test System 4 to produce non-scattered results. By doing so we do not state that the 99.9% reply rate would be acceptable for anyone, we used this value to be able to produce non-scattered results for the comparison.

The authoritative DNS server performance results of Knot DNS as a function of the size of the zone file using a single CPU core measured by TS1 and TS4 are presented in Table XIX and Table XX, respectively. Although there are some fluctuations, both tables show that neither the size of the zone file nor the timeout value have significant effect on the performance of Knot DNS.

D. YADIFA

1) SCALE-UP TESTS

The authoritative DNS server performance results of YADIFA as a function of the number of active CPU cores using a "/8" size zone file measured by TS2 are presented in Table XXI. At 1 and 2 cores, the results are very much scattered (dispersion is more than 20%). They improve at 4 cores (dispersion is 4.4%), and the dispersion is only 0.44 at 8 cores, where YADIFA reaches its highest performance. Its performance not only scales up poorly, but it also significantly degrades at 16 cores, which we consider a fundamental problem.

Table XXII shows the results of YADIFA produced by TS4. They are even worse in the sense that they are always very scattered. We have included them only to show their quality and the performance degradation of YADIFA at 32 cores.

We have also executed the benchmarking tests with TS2 and TS3, using "/5" and "/4" size zone files, respectively, but we do not include their results because they are very similar to that of TS1 and TS4 and thus they would not lead to any further conclusion.

Because of the poor scale up of YADIFA, we did not see any point in producing more non-RFC 8219 compliant results, thus we did not test it with non-zero frame loss criterion.

2) ZONE FILE SIZE TEST

The authoritative DNS server performance results of YADIFA as a function of the size of the zone file using 8 CPU cores measured by TS1 are presented in Table XXIII. They show that neither the increase of the size of the zone file, nor the decrease of the timeout value from 250ms to 100ms causes a significant change in the performance a YADIFA.

E. COMPARISON

As for their performance, the examined four authoritative DNS server implementations evidently fall into two categories. BIND and YADIFA have shown moderate performance (less than 300,000qps), whereas NSD and Knot DNS gave an excellent performance, reaching 2-3 million qps depending on the given conditions. Thus we concentrate on the latter two.

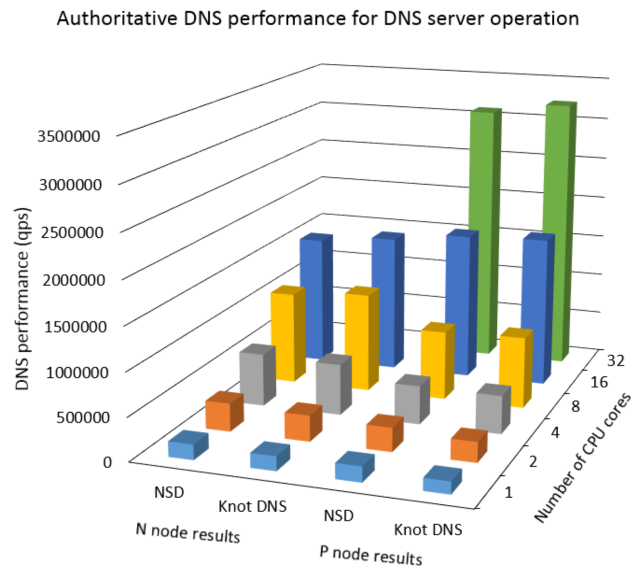

FIGURE 3. **Comparison of NSD and Knot DNS for DNS server operation. (The N node result of Knot DNS at 16 cores are limited by Tester performance.)**

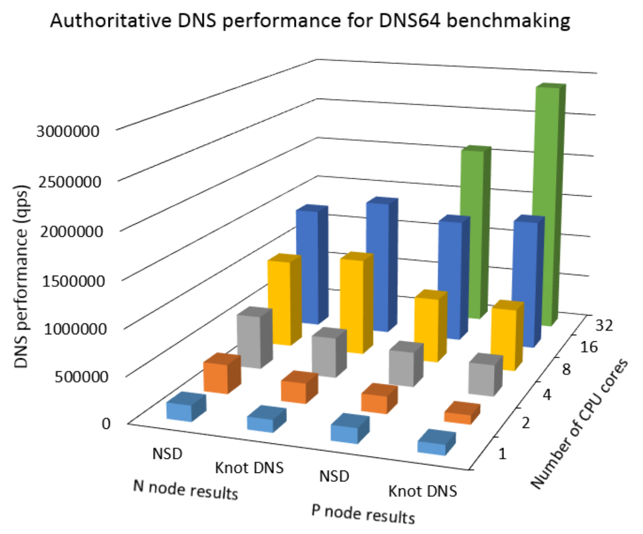

FIGURE 4. **Comparison of NSD and Knot DNS for DNS64 benchmarking. (The N node result of Knot DNS at 16 cores are limited by Tester performance.)**

1) OUR RECOMMENDATION FOR AUTHORITATIVE DNS SERVER OPERATION

We contend that a DNS service may be acceptable for many ISPs and their users if a single query is lost from 10,000 queries, therefore, we used the median values from Table X (NSD, TS2), Table XVI (Knot DNS, TS2), Table XII (NSD, TS3), and Table XVIII (Knot DNS, TS3) for comparison. (For those DNS operators, who prefer higher standards, we recommend the usage of our comparison in the next subsection.) Our final results are shown in Fig. 3. Both implementations performed excellently, whereas NSD was somewhat better at low number of cores (1-4), Knot DNS was somewhat better at high number of cores (8-32). As for their performance, we recommend the usage of both servers.

When selecting a DNS server implementation, operators need to consider several factors, including the following ones:

- Functionality (e.g. authoritative, recursive, DNSSEC, DNS64)
- Performance
- Security, reliability, maturity of the code
- Documentation and support
- Experience with the software

As both NSD and Knot DNS are used with some root DNS servers, we believe that they are both suitable for DNS server operators, too. We hope that our results will encourage DNS server operators to upgrade from BIND and thus achieve higher performance and/or save costs.

2) OUR RECOMMENDATION FOR DNS64 BENCHMARKING

To support DNS64 benchmarking, only the results of RFC 8219 compliant measurements can be used and the 1st percentiles should be taken into consideration, therefore, we used the 1st percentile values from Table IX (NSD, TS2), Table XV (Knot DNS, TS2), Table XI (NSD, TS3), and Table XVII (Knot DNS, TS3) for comparison. Our final results are shown in Fig. 4. Both implementations performed excellently. As for TS2 (DUT: Dell PowerEdge C6620), NSD performed significantly better with 1-4 cores, and Knot DNS produced higher results with 8-16 cores. On TS3 (DUT: Dell PowerEdge R430) NSD performed significantly better on 1-2 number of cores, their performance was similar on 4-16 number of cores, and Knot DNS performed significantly better on 32 cores.

Considering the actual performance of existing DNS64 servers [9], any of them and even BIND or YADIFA would do, but when high performance is needed (e.g. when testing a new, high performance DNS64 implementation), then it is worth selecting either NSD or Knot DNS, depending on the actual hardware environment. We also note that NSD requires a significant amount of time for starting as it builds its own database, for which it needs large amount of disk space, e.g. nearly 200GB for a "/4" size zone file.

F. CHECKING THE PERFORMANCE OF `DNS64PERF++`

The excellent performance of Knot DNS (using all 32 cores of a P node) made it possible for us to check the performance of **`dns64perf++`**. TS4 was used, however, Turbo Mode was enabled in the DUT, too. We found that **`dns64perf++`** could send and receive packets reliably at 3.3 million qps rate.

Without this test we could not be sure that the results in the last column of Table XVII and in the last two columns of Table XVIII reflect the performance of Knot DNS or that of our tester program **`dns64perf++`**.

V. DISCUSSION AND FUTURE WORK

On the one hand, the number of processes of NSD were set exactly to the number of active CPU cores. Whereas it worked well with low number of active CPU cores, the situation was different with higher number of cores. On the other hand, Knot DNS used an auto-estimated optimal number of threads. Although it did not seem to work well at a low number of active CPU cores, it was excellent at 32 cores. As the number of CPU cores is continuously growing, it seems that the developers of Knot DNS follow a good approach.

We note that the selected RFC 8219 compliant benchmarking method using the latest version of **`dns64perf++`** is not only the most suitable one for benchmarking DNS servers for DNS operators, but it is also the most economic one. Whereas the other solutions used five additional computers for testing a single server (please refer to [15] and [16]), we needed only a single computer as Tester, though we admit that Turbo Mode was enabled on the Tester and it was disabled on the DUT to make it the bottleneck. Disabling Turbo Mode was important also from *analytical point of view*. We mean it as follows. When Turbo Mode is enabled, a few number of cores may operate at the maximum turbo frequency, however, when all cores are enabled and have high load, their clock frequency is limited by the power budget determined by TDP (Thermal Design Power). Thus, when Turbo Mode is enabled, the doubling of the number of online CPUs does not always double the available computing power.

We contend that the usage of high number of different source ports is a very important condition for a proper testing of DNS or DNS64 servers, thus we are considering to initiate an update to RFC 8219. We are also examining the possibility of writing and Internet Draft on benchmarking methodology for DNS servers (possibly including both authoritative and recursive ones).

We are also considering to examine the computing power relative performance of the best performing DNS servers according to the methodology defined in [9] to assist energy efficiency aware DNS server administrators with another important factor for their DNS implementation selection.

To make **`dns64perf++`** even better, we plan the following improvements:

- Enable it for using different local IP addresses for each thread pairs, thus provide each thread pair with 64,000 source ports (potentially).
- Test different placements using CPU affinity. (E.g. to place the sender and corresponding receiver on neighboring cores.)
- Parallelize the processing of the information in the second phase, which may significantly decrease execution time at high rates (e.g. over 1 million qps).

We also plan to test and document the new features of **`dns64perf++`** in a research paper.

VI. CONCLUSION

We have surveyed the available methods for benchmarking authoritative DNS servers, and found that the one we defined in RFC 8219 for a special purpose (to support DNS64 benchmarking) is the most appropriate one also for examining the performance of the authoritative DNS servers for real authoritative DNS server usage (with some additions or modifications, such testing also with 100ms timeout and allowing a small non-zero loss rate, like 0.01%).

We have carefully examined how the performance of BIND, NSD, Knot DNS, and YADIFA depends on different factors, such as the number of active CPU cores, the size of the zone file, the CPU architecture, and the timeout value.

We have provided ready to use measurement results for selecting the most suitable DNS implementation both for authoritative DNS server usage and for DNS64 benchmarking.

ACKNOWLEDGMENT

The experiments were carried out by remotely using the resources of NICT StarBED, 2-12 Asahidai, Nomi-City, Ishikawa 923-1211, Japan.

The author would like to thank Shuuhei Takimoto for the possibility to use StarBED, as well as to Masatoshi Enomoto, and Satoru Gonno for their help and advice in StarBED usage related issues.

The author thanks Gábor Horváth, Budapest University of Technology and Economics, Budapest, Hungary for advising to completely power off the N nodes (not only reboot them) in order to bring them into a “clear” state.

The author thanks Dániel Bakai, Budapest University of Technology and Economics, Budapest, Hungary for the additional developments of **`dns64perf++`** in his free time.

The author thanks Attila Pivoda, Széchenyi István University, Győr, Hungary for recommending the usage of **`ethtool`** to distribute the interrupts evenly among the CPU cores.

REFERENCES

[1] D. Moore “DNS server survey”, May 23, 2004. [Online]. Available: http://mydns.bboy.net/survey/
[2] G. Lencse and Y. Kadobayashi, “Methodology for the identification of potential security issues of different IPv6 transition technologies: Threat analysis of DNS64 and stateful NAT64”, Computers & Security, vol. 77, no. 1, pp. 397-411, Aug. 2018, DOI: 10.1016/j.cose.2018.04.012
[3] M. Bagnulo, A Sullivan, P. Matthews and I. Beijnum, “DNS64: DNS extensions for network address translation from IPv6 clients to IPv4 servers”, IETF RFC 6147, Apr. 2011. DOI: 10.17487/RFC6147

[4] G. Lencse and Y. Kadobayashi, "Comprehensive survey of IPv6 transition technologies: A subjective classification for security analysis", IEICE Transactions on Communications, vol. E102-B, no. 10, pp. 2021–2035. Oct. 2019, DOI: 10.1587/transcom.2018EBR0002

[5] M. Bagnulo, P. Matthews and I. Beijnum, "Stateful NAT64: Network address and protocol translation from IPv6 clients to IPv4 servers", IETF RFC 6146, Apr. 2011. DOI: 10.17487/RFC6146

[6] M. Bagnulo, A. Garcia-Martinez and I. Van Beijnum, "The NAT64/DNS64 tool suite for IPv6 transition", IEEE Commun. Magazine, vol. 50, no. 7, pp. 177–183, Jul. 2012. DOI: 10.1109/MCOM.2012.6231295

[7] M. Georgescu, L. Pislaru and G. Lencse, "Benchmarking methodology for IPv6 transition technologies", IETF RFC 8219, Aug. 2017. DOI: 10.17487/RFC8219

[8] G. Lencse, M. Georgescu, and Y. Kadobayashi, "Benchmarking methodology for DNS64 servers", Computer Communications, vol. 109, no. 1, pp. 162–175, Sep. 2017, DOI: 10.1016/j.comcom.2017.06.004

[9] G. Lencse and Y. Kadobayashi, "Benchmarking DNS64 implementations: Theory and practice", Computer Communications, vol. 127, no. 1, pp. 61–74, Sep. 2018, DOI: 10.1016/j.comcom.2018.05.005

[10] Y. Sekiya, K. Cho, A. Kato, and J. Murai, "Research of method for DNS performance measurement and evaluation based on benchmark DNS servers", *Electronics and Communications in Japan, Part I: Communications*, *vol. 89, no.* 10, pp. 66–75. Oct. 2006, DOI: 10.1002/ecja.20211

[11] A. Kesavan, "Comparing the performance of popular public DNS providers", Network World, May, 2017. [Online]. Available: https://www.networkworld.com/article/3194890/comparing-the-performance-of-popular-public-dns-providers.html

[12] E. Ahmad, K. Sarwar, "A comparative analysis on existing DNS performance measurement mechanisms", Int. J. of Computer Networks and Communications Security, vol. 2, no. 5, pp. 158–167. May 2014.

[13] H. Boulakhrif, "Analysis of DNS resolver performance measurements", MSc thesis, University of Amsterdam, Jul. 2015. [Online]. Available: https://www.nlnetlabs.nl/downloads/publications/os3-2015-rp2-hamza-boulakhrif.pdf

[14] B. R. Greene, "DNS latency and performance test tools", [Online]. Available: http://www.senki.org/network-operations-scaling/dns-latency-and-performance-test-tools/

[15] NLnet Labs, "NSD4 performance measurements", Jul. 2013, [Online]. Available: https://medium.com/nlnetlabs/nsd4-performance-measurements-9e224bc4fa0f

[16] Knot DNS, "Benchmarking", [Online]. Available: https://www.knot-dns.cz/benchmark/

[17] EURid, "Benchmark", [Online]. Available: https://www.yadifa.eu/benchmark/

[18] G. Lencse, D. Bakai, "Design and implementation of a test program for benchmarking DNS64 servers", IEICE Transactions on Communications, vol. E100-B, no. 6. pp. 948–954, Jun. 2017. DOI: 10.1587/transcom.2016EBN0007

[19] G. Lencse and A. Pivoda, "Checking and increasing the accuracy of the dns64perf++ measurement tool for benchmarking DNS64 servers", Int. J. of Advances in Telecommunications, Electrotechnics, Signals and Systems, vol. 7. no. 1. pp. 10–16. (2018.) DOI: 10.11601/ijates.v7i1.255

[20] D. Bakai, "A C++14 DNS64 performance tester", source code, [Online]. Available: https://github.com/bakaid/dns64perfpp

[21] G. Lencse, Modified source files of dns64perf++, [Online]. Available: http://www.hit.bme.hu/~lencse/dns64perfpp/

[22] Free Software Foundation, "The free software definition", [Online]. Available: http://www.gnu.org/philosophy/free-sw.en.html

[23] Open Source Initiative, "The open source definition", [Online]. Available: http://opensource.org/docs/osd

[24] G. Lencse and S. Répás, "Performance analysis and comparison of four DNS64 implementations under different free operating systems", Telecommunication Systems, vol. 63, no. 4, pp. 557–577, Nov. 2016, DOI: 10.1007/s11235-016-0142-x

[25] Internet Systems Consortium, "BIND: Versatile, classic, complete name server software", [Online]. Available: https://www.isc.org/downloads/bind

[26] Cz Nic, "Knot DNS: High-performance authoritative-only DNS server", [Online]. Available: https://www.knot-dns.cz/

[27] NLnet Labs, "NSD: Name Server Daemon", [Online]. Available: https://www.nlnetlabs.nl/projects/nsd/

[28] EURid, "YADIFA", [Online]. Available: https://www.yadifa.eu

[29] Powerdns.com BV, "PowerDNS", [Online]. Available: http://www.powerdns.com

[30] A. Hubert, R. van Mook, Measures for making DNS more resilient against forged answers, IETF RFC 5452 (2009). doi:10.17487/RFC5452

[31] NLnet Labs, "Unbound", [Online]. Available: http://unbound.net

[32] G. Lencse and Y. Kadobayashi, "Methodology for DNS cache poisoning vulnerability analysis of DNS64 implementations", Infocommunications Journal, vol. 10, no. 2, pp. 13–25. (2018.)

[33] A. Kleen, M, Wilcox, "SOCKET(7)" in Linux Programmer's Manual, [Online]. Available: http://man7.org/linux/man-pages/man7/socket.7.html

[34] M. Majkowski, "How to receive a million packets per second", Cloudflare Blog, Jun. 2015. [Online]. Available: https://blog.cloudflare.com/how-to-receive-a-million-packets/

Gábor Lencse received his MSc and PhD in computer science from the Budapest University of Technology and Economics, Budapest, Hungary in 1994 and 2001, respectively.

He has been working full time for the Department of Telecommunications, Széchenyi István University, Győr, Hungary since 1997. Now, he is a Professor. He has been working part time for the Department of Networked Systems and Services, Budapest University of Technology and Economics as a Senior Research Fellow since 2005. His research interests include the performance and security analysis of IPv6 transition technologies. He is a co-author of RFC 8219.