Extending ZMap: Round-Trip Time Measurement via ICMP and TCP

Bernhard Vorhofer, Patrick Sattler*, Johannes Zirngibl*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: bernhard.vorhofer@tum.de, sattler@net.in.tum.de, zirngibl@net.in.tum.de

***Abstract*—ZMap is a high-performance network scanner for Internet-wide network surveys. Due to its modular architecture, it allows researchers to conduct a variety of different types of scans and can be extended with custom modules.**

The aim of this paper is to outline the process of how we implemented two modules for ZMap and the IPv6-enabled ZMapv6 for measuring round-trip time. The first sends ICMP echo request packets, the other is based on the existing SYN scan module and allows RTT measurement using TCP.

We give a brief overview of ZMap's implementation and discuss key architectural aspects that influenced our module design. Furthermore, we present the results of multiple tests we conducted using the new modules and compare them to an earlier study involving large scale round-trip time measurement. The experiments suggest that the implemented modules are working correctly. However, we found a discrepancy between ZMap's reported values and timestamps captured during the measurement using a network analyzer.

***Index Terms*—Network scanning, measurement, round-trip time**

1. Introduction

ZMap is a network scanner for Internet-scale network surveys. As Durumeric et al. have shown in [1], this tool is capable of scanning the entire IPv4 address space in under 45 minutes. ZMap was extended with IPv6 support along with three new modules by Gasser et al. [2], resulting in ZMapv6. The added IPv6-enabled modules allow scanning using ICMP echo request, TCP SYN or UDP messages.

In this paper, we present two new modules for ZMap and ZMapv6. One of them, the ICMP time module, has been ported from the existing IPv4 version. The other is based on the existing SYN scan module and allows RTT measurements via TCP without using the TCP Timestamps option. These modules, along with the existing ICMP time module, enable large-scale round-trip time measurements using ZMap. This can be useful in a variety of applications, for example for finding bottlenecks in large networks.

The ICMP time module provides the same functionality as the existing IPv4 version of this component, but does so using IPv6. Since the basic structure of the ICMPv6 header is identical to its version four counterpart, porting the ICMP time module to version six of the Internet Protocol was matter of refactoring the existing code. Therefore, the implementation details of this module are not described any further in this paper. However, we discuss the results of tests we performed using the ICMPv6 time module in Section 4.3.

The TCP time module accomplishes two things at once. First, it performs a standard SYN scan, determining whether a specific port is open for each target host. Second, the round-trip time is determined for each responding host by measuring the delay between the transmission of a SYN segment and the arrival of the corresponding SYN-ACK segment.

One noteworthy piece of related work is D. J. Bernstein's description of SYN cookies [3], a method of embedding information in the TCP sequence number field originally intended as a countermeasure against SYN flood attacks. ZMap uses a similar technique in its TCP time modules to include additional data in probe packets.

In the following sections, we first give a brief overview of ZMap's architecture before discussing the implementation details of the TCP time module in Section 3. Finally, we present the results of experiments we conducted in Section 4.

2. ZMap primer

This section gives a brief introduction to ZMap and its modular architecture, mostly summarized from the original paper by Durumeric et al. [1].

ZMap owes its exceptional speed at least in part to the stateless nature of its architecture – the tool was specifically designed with performance in mind. Target addresses are selected using a cyclic multiplicative group, which distributes them randomly across the IPv4 address space. This also eliminates the need for storing addresses already scanned because each address in the address space is reached exactly once.

To distinguish scan responses from background traffic, probe modules embed scan- and host-specific validation data in headers of probe packets (where possible). This allows responses to be validated on receipt without keeping additional per-connection state. ZMap generates a message authentication code (MAC) of the destination address for probe modules to use as validation data.

2.1. Architecture

ZMap's modular architecture can be split into three basic parts: the scanner core, output modules and probe modules. The scanner core handles packet transmission

doi: 10.2313/NET-2020-11-1_06

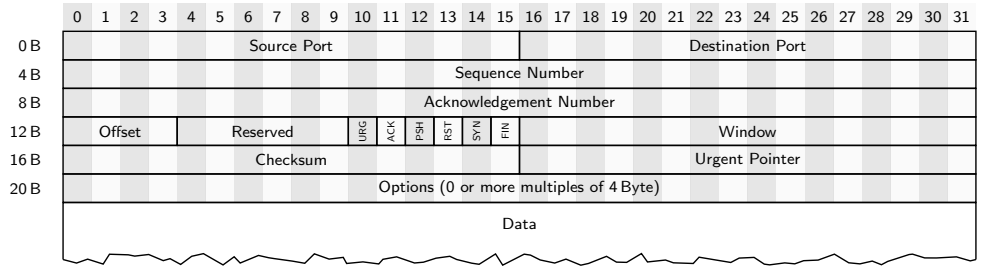


Figure 1: TCP Header

and receipt, address generation and other fundamental tasks. It also invokes probe- and output modules via callbacks, for example to generate packets before they are sent or to validate packets after they have been received. Output modules allow for multiple forms of scan output, e.g. CSV files, JSON files or writing entries to an in-memory Redis database. Since output modules are not immediately relevant for this work, we will not elaborate on the details of their implementation. Probe modules provide multiple different scan methods, for example TCP SYN scans or ICMP echo scans. Their implementation is further discussed in Section 2.3.

2.2. Transmission and reception

To avoid unnecessary kernel overhead, ZMap makes use of raw sockets for packet transmission. Consequently, probe packets are assembled on the data-link layer, which allows for the Ethernet header to be cached and reused because the source (local network interface) and destination (gateway) MAC addresses remain unchanged during a scan. Packet reception is achieved using libpcap, a C/C++ library for traffic capture. Furthermore, this library provides a first filter for incoming packets, so traffic clearly not related to the current scan is discarded before it reaches the active probe module.

2.3. Probe modules

Probe modules are responsible for constructing probe packets to be sent to and validating responses received from target hosts. The scanner core provides the active probe module with a single buffer which holds the data to be sent. Before a scan commences, the probe module populates the empty buffer with any necessary headers. Values set at this stage (e.g. source and destination MAC addresses in the Ethernet header) are not modified during the rest of the scan, as mentioned before.

For each scan target, the probe module in use fills all host-specific header fields in the buffer with data provided by the scanner core. This includes the network layer addresses as well as validation data. It is desirable to include as much of the validation string as possible in the probe packet while ensuring it is placed in header fields which allow recovery of the validation data from the probe response. The reason for this is that more validation data decreases the probability of a false positive, i.e., an incoming packet unrelated to ZMap being erroneously classified as a probe reply. After the probe module has populated the buffer, the scanner core handles transmission of the packet.

Whenever a packet is received, the scanner core calls two probe module callbacks in succession. First, the incoming message is validated, i.e., validation data is extracted from the header and compared to the expected validation string provided by the scanner core. Second, and only if validation succeeded, the received packet is processed. The primary purpose of this step is to pass values from the received packet to the active output module for formatting.

3. TCP time module

The TCP time module extends the existing TCP SYN scan module with the capability of measuring round-trip time during a partial three-way handshake. This is achieved in a manner similar to measuring RTT using ICMP: a timestamp is included in the probe packet in a way which allows it to be recovered from the response, thus allowing the time difference between transmission of the probe packet and reception of the response to be determined without keeping additional state.

In our research, we found two viable methods of measuring round-trip time using TCP. The first, which was employed for the TCP time module, utilizes the sequence number field (see Figure 1 for reference) to embed a timestamp in the TCP header. The second makes use of a TCP option intended for precisely this use case.

3.1. Embedding and recovering the timestamp

The problem we faced when first attempting to implement RTT measurement without using the TCP Timestamps option was that the sequence number field provides only four bytes of space for the timestamp, but the timestamp structure used in other modules is eight bytes in size. As a consequence of this, we decided to reduce the size of the timestamp by using a relative timestamp instead of an absolute one and decreasing its resolution from one microsecond to one millisecond.

Furthermore, since the original TCP SYN scan module utilizes both the source port and the sequence number fields for validation data, the amount of validation data included in the probe packet needed to be reduced in order to accommodate the additional timestamp data. With the sequence number field occupied by the timestamp, this leaves only the source port field available for validation data. Because we chose to only allow ports from the standard Linux ephemeral port range (32 768 to 61 000), the effective validation string length (binary logarithm of the number of available ports) is approximately 14.8 bit, even though the source port field is two bytes in size.

doi: 10.2313/NET-2020-11-1_06

When the probe module is initialized before the scan starts, a timestamp is stored in a global variable which all of the relative timestamps included in probe packets use as a reference. For each probe packet, a 32 bit value representing the number of milliseconds passed since the scan started is used as the initial sequence number. In contrast to the `timeval` struct used in the ICMP time module, this timestamp provides only millisecond instead of microsecond resolution. With the information about libpcap's internal timestamp precision found in [4], we conclude that the prospect of achieving microsecond precision without employing specialized hardware is questionable at best. Coupled with the fact that most round-trip time measurement tools we have used (most notably *ping*) report results in millisecond precision, this supports our decision to reduce the timestamp resolution. When a SYN-ACK message is received, the timestamp can be recovered by subtracting 1 from the acknowledgement number field in its header.

3.2. Alternative method for RTT measurement

An alternative method of measuring round-trip time using TCP is by using the aforementioned TCP Timestamps option [5]. It was designed specifically for measuring round-trip time, among other uses. Although this would simplify the task at hand, the method we implemented in the TCP time module offers a significant advantage: it does not require any TCP extensions to be supported and enabled. Kühlewind et al. [6] tested a selection of popular web servers for support of certain TCP options and extensions. The results of their tests from 2012 show that in total, roughly 80 % of responding hosts supported the timestamp option.

Additionally, some administrators might decide to disable this option for security reasons. The reason for this is that, as Beverly et al. [7] describe, TCP timestamps can be used to estimate the uptime of a host, potentially allowing attackers to discern whether a security patch is installed or not. With the technique employed in the TCP time module, measurements can be made regardless of TCP timestamp support on the target host.

3.3. Implications of using less validation data

When reducing the amount of validation data included in the probe packet TCP header, the question of whether the remaining data is sufficient for discriminating between scan responses and background traffic arises. For this application, there is only one specific scenario we need to be concerned about: an unsolicited incoming TCP segment leading to a false positive and, consequently, an incorrect round-trip time value. More specifically, only an incoming SYN-ACK or RST message can lead to such an error, because any other type of message will be discarded by libpcap before it reaches the validation function due to the packet filter set up for the module.

Since a SYN-ACK is received only during an outgoing connection attempt in normal operation, it is safe to assume that these messages would only be received due to another program in the process of establishing a TCP connection while a scan is being performed. Outgoing TCP connections are usually assigned an ephemeral port from the same range ZMap uses for probe packets. Even though it is possible that a program is assigned the exact port that would lead to a false positive in terms of response validation, the probability of this happening should be rather low considering the large number of ports available. To further reduce the odds of such an error occurring, network activity unrelated to ZMap should be reduced on the scanning machine where possible.

Another case that should be considered is the effect a scan has on existing TCP connections. If a connection to a target host exists on the same port ZMap uses for its probe of that specific host – which is unlikely, as mentioned above – can this lead to an RST message? According to the TCP protocol specification [8], a SYN received in one of the synchronized states does not lead to a reset. Rather, an ACK message containing the next expected sequence number would be sent. Since such a packet would be discarded by the libpcap filter, existing connections do not interfere with the network scanner.

4. Test results

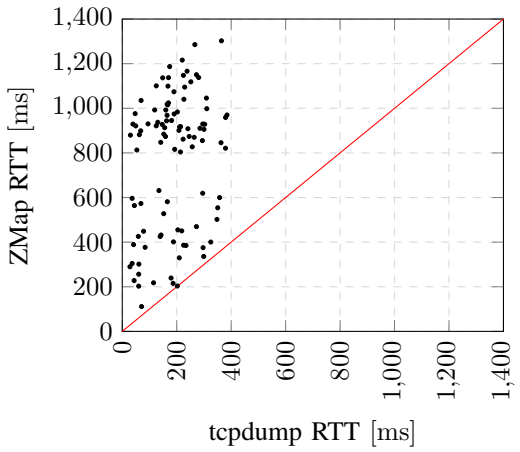


Figure 2: tcpdump vs. ZMap ICMPv4 RTT (red line represents both measurements reporting the same value)

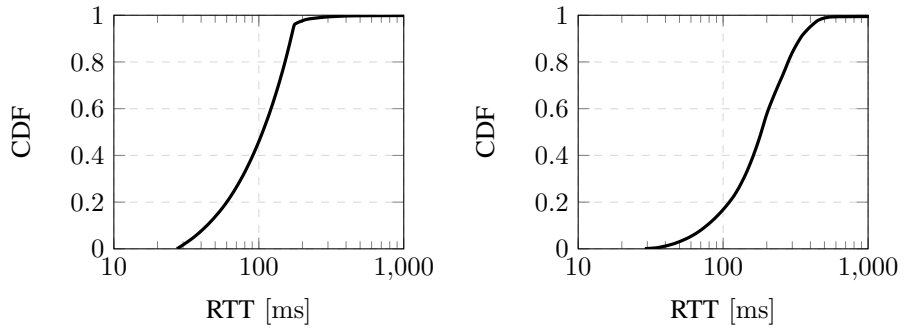
In order to confirm the functionality of the newly implemented modules, we conducted several tests and recorded more than 600 000 responses. We compare our results to those presented by Padmanabhan et al. [9] by plotting the cumulative distribution function (CDF) of the RTT measurements.

4.1. Testing methodology

Two series of ZMap measurements were conducted to verify the functionality of the implemented modules. For the TCP module test, probes were sent to randomly selected hosts and nearly 400 000 replies have been recorded. For ICMPv6, hosts from the Alexa[1] Top 1 Million web server list (which is not offered anymore at the time of writing) were probed and approximately 250 000 replies have been recorded. These two tests have been carried out from inside the university network.

1. https://www.alexa.com/topsites

doi: 10.2313/NET-2020-11-1_06



(a) ICMPv6 (b) TCP (IPv4)

Figure 3: Cumulative distributions of ZMap RTT measurements

During our tests, network traffic was captured using tcpdump, an open-source network protocol analyzer. This allowed us to cross-reference the timestamps of outgoing and incoming packets as recorded by tcpdump with ZMap's round-trip time measurements. Naturally, the expected result would be the tools reporting similar values, but we found that a significant number of Zmap's measurements diverge substantially from the RTT values indicated by the captured timestamps (see Section 4.2).

4.2. Validation of RTT measurements

To assess the overall quality of ZMap's round-trip time measurements, we conducted an additional test using the existing ICMP time module. ZMap was configured to probe randomly selected hosts until 100 echo replies were collected. As mentioned before, tcpdump was capturing all network traffic throughout the test so we could compare the RTT calculated using the timestamps in the capture file with ZMap's reported values. Even though libpcap timestamps may not be the most precise [4] (tcpdump captures packets using libpcap), they suffice for a rough estimate of the actual RTT.

Figure 2 shows the results of this test. It is clear that ZMap tends to report higher RTT values than those measured by tcpdump. Furthermore, there is a considerable inconsistency in the ZMap measurements – numerous values deviate significantly from those obtained from the capture file. Closer examination of the timestamps revealed that the receive timestamps are the cause of this inaccuracy, which leads us to believe that somewhere in the receiving process, a variable amount of delay causes unreliable receive timestamps. Further investigation to find the source of this delay is out of scope for this paper.

4.3. ICMPv6 time module

For the ICMPv6 test, around 200 000 probe replies were collected. The cumulative distribution function in Figure 3a was plotted using a subsample of roughly 1000 data points. Systematic sampling was used to select the samples, i.e., the replies were ordered by round-trip time before values were selected at regular intervals to yield the desired sample size.

The function has similar characteristics to that in [9, Figure 7]. It seems that our dataset contains a larger portion of comparatively low values, though it should be noted that our sample size is several orders of magnitude smaller. Nevertheless, we believe this shows that the ICMPv6 time module is working as intended.

4.4. TCP time module

Around 400 000 replies were collected for the TCP over IPv4 test, roughly 1000 of which were selected (using systematic sampling) to plot the CDF in Figure 3b. This graph shows even more similarity to that in [9, Figure 7], with significantly more values above 200 ms. Keeping in mind the restricted sample size and the aforementioned variability of receive timestamps, we believe this shows that the values reported by the TCP time module are plausible.

5. Conclusion and future work

With the new modules we implemented for ZMap, two additional scan types now have the capability to report round-trip time measurements – ICMPv6 and TCP. This allows researchers to collect one more metric from large-scale network surveys, determining not only reachability but also response time.

During our tests, we discovered an unexpected disparity between RTT values reported by ZMap and measurements we recorded simultaneously using tcpdump. This was observed with all of the modules, including the existing ICMP time module. Before the source of this measurement error is found and eliminated, we are unable to reliably determine the precision of the implemented modules' measurements. The reason for this is that ZMap's measurements can not be directly compared to values calculated using the captured timestamps. However, we compared the distributions of round-trip time measurements to [9, Figure 7] and found that they exhibit very similar characteristics, which suggests that the modules themselves do indeed work as intended.

Other than finding and resolving the cause of the observed measurement error, possible future work includes implementing a ZMap module for RTT measurement using the TCP Timestamps option. Although we decided against employing this protocol extension for our TCP time module, it might be useful in some situations to have both methods at one's disposal.

doi: 10.2313/NET-2020-11-1_06

References

[1] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast Internet-wide Scanning and Its Security Applications," in *22nd USENIX Security Symposium (USENIX Security 13)*. Washington, D.C.: USENIX Association, Aug. 2013, pp. 605–620.

[2] O. Gasser, Q. Scheitle, S. Gebhard, and G. Carle, "Scanning the IPv6 Internet: Towards a Comprehensive Hitlist," *CoRR*, vol. abs/1607.05179, 2016.

[3] D. J. Bernstein, "SYN cookies." [Online]. Available: http://cr.yp.to/syncookies.html

[4] "Manpage of PCAP-TSTAMP." [Online]. Available: https://www.tcpdump.org/manpages/pcap-tstamp.7.html

[5] D. Borman, B. Braden, V. Jacobson, and R. Scheffenegger, "TCP Extensions for High Performance," Internet Requests for Comments, RFC Editor, RFC 7323, September 2014. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7323.txt

[6] M. Kühlewind, S. Neuner, and B. Trammell, "On the State of ECN and TCP Options on the Internet," in *Passive and Active Measurement*, M. Roughan and R. Chang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 135–144.

[7] R. Beverly, M. Luckie, L. Mosley, and K. Claffy, "Measuring and Characterizing IPv6 Router Availability," in *Passive and Active Measurement*, J. Mirkovic and Y. Liu, Eds. Cham: Springer International Publishing, 2015, pp. 123–135.

[8] J. Postel, "Transmission Control Protocol," Internet Requests for Comments, RFC Editor, RFC 793, September 1981. [Online]. Available: http://www.rfc-editor.org/rfc/rfc793.txt

[9] R. Padmanabhan, P. Owen, A. Schulman, and N. Spring, "Timeouts: Beware Surprisingly High Delay," in *Proceedings of the 2015 Internet Measurement Conference*. Association for Computing Machinery, 2015, pp. 303–316.