Use the Zero-packet Internet Groper Network Utility

Zing This!

Zing Network Utility UserBook

William F. Gilreath

Zing This!

The Zero Packet Internet Groper

BY WILLIAM F. GILREATH

will@wfgilreath.xyz

ISBN: 9798224428151

Disclaimer

This e-book has been written for information purposes only. Every effort has been made to make this e-book as complete and accurate as possible. However, there may be mistakes in typography or content. Also, this e-book provides information only up to the publishing date. Therefore, this e-book should be a guide, not the ultimate source.

The purpose of this e-book is to educate. The author and the publisher do not warrant that the information contained in this e-book is fully complete and shall not be responsible for any errors or omissions. The author and publisher shall have neither liability nor responsibility to any person or entity concerning any loss or damage caused or alleged to be caused directly or indirectly by this e-book.

“Imagination is more important than knowledge.”

Albert Einstein (1879-1955)

Table of Contents

Dedication

This book and the text are dedicated to a very much adored cat, my special kitty:

Jaguaro or “Jaggie” who I lost by cruel caprice.

A very dear cat that was a friend, and companion—taken too soon, missed forever, and loved eternally.

Chapter 1. Rationale

This overview or introduction is the rationale, the reason I created, wrote, and am documenting the zing network utility.

1.1 Motivation

My original motivation to create zing, the zero-packet ping or zero-packet Internet groper was to create a tool or utility like ping, yet one that is “lighter” in terms of network packets.

Another motive was simply to try out the idea, to create and implement what I considered something that is in retrospect, so obvious, and deceptively simple that no one else had thought of it.

1.2 Goals

The overall goal of the zing network utility is to be not a replacement, but a complement in addition to ping in terms of network speed, performance, and presence of systems.

The function of the zing network utility is to provide information about the existence, presence, and time to reach a host on a network.

The three overall long-term goals of the zing network utility are:

- Ubiquity for the widespread use of the zing network utility on different computer systems.

- Knowledge for utilizing and using the zing network utility.

- Proliferation or porting the zing network utility to other computer systems.

This book is to spread knowledge so that others can add it as a tool to their toolbox, or another weapon in the arsenal.

A long-term goal of zing is to be ubiquitous, hence I have open-sourced the zing network utility, put the source code out on GitHub in a public repository, and encourage porting zing into other programming languages on other platforms.

The power is in the understanding, sharing, and proliferation of zing as a network utility. Excelsior!

Chapter 2. History and Origin

The zing network utility ultimately started (as some things do in high tech...) as a conversation between me and another colleague.

2.1 Origin of the Idea

For the backstory, at the time I was on a team where much of the software was network-focused, and one problem was that there was no easy way to detect or determine if a network cluster was present—active, available, running, and ready on a host computer. Also, the timing to send and receive any packets to a network cluster on a host system.

The network utility ping was an obvious, readily available tool as a solution, but was sending a lot of packets at a level of network communication. The idea was to have something lighter than an ICMP (Internet Control Message Packet) packet stream.

The answer for me was obvious, try to connect and disconnect with TCP/IP (Transmission Control Protocol/Internet Protocol) on a well-known port such as HTTP (HyperText Transport Protocol) port 80, or HTTPS (HyperText Transport Protocol Secure) port 443.

This colleague was smart, knew much about all the network hardware from vendors, and had worked with different network equipment, and devices.

Unfortunately, this colleague rejected the idea, and the rationale with "We don't want to pollute the runtime log of a system." Thus the core idea for the zing network utility was dismissed for "polluting" the runtime log for the host system on a network.

2.2 Implementing the Idea

Despite the rejection and dismissal by my obtuse, unimaginative, colleague, I decided to implement the idea for the zing network utility. The great thing about software, as a software development engineer, is that one can implement an idea readily.

For me it was easier and more fun to create the utility I had suggested, and viola! or shiver me timbers, there it is—the of zing is born as my brainchild.

Then it was a matter of design, and for familiarity, I decided to use a ping-style command-line interface and output. Once designed I implemented it in Java for simplicity (the network packages are easier to use) and to create a cross-platform network utility.

2.3 Article in Linux Magazine

From there, I reached out to several technical magazines, and *Linux Magazine* published the first article about Zing. The article was published in December of 2022 in *Linux Magazine*.

Linux Magazine December 2022 Cover

My original title 'Zing This!' was too colorful and cute, so then was "Zing Me!" as the article title.

IN-DEPTH
Zing

Introducing the Zing zero-packet network utility

Zing Me

Zing is a lightweight, zero-packet network utility similar to ping that provides ping functionality without the payload. *By William F. Gilreath*

The ping networking utility (backronym Packet InterNet Groper) [1, 2, 3] is a common tool used to determine network delay and whether a host is active at an IP address [4]. Ping sends an Internet Control Message Packet (ICMP) [5] to a host as an echo request, which then receives a response from the remote host. Ping sends data about the time and the host sends the data back – to put it at its simplest, plainest explanation.

Ping is popular with admins around the world, however, the ping utility is not without problems. For instance, ping can be used to send a packet storm to overload and crash a host system – a denial-of-service (DOS) attack called a *ping flood*. Some sites simply do not respond to ping requests, and some firewalls block the ICMP packets sent by the ping utility.

A problem I faced was the need for ping functionality, but without sending packets of data through the network. One of the best things about being a software development engineer is that, for interesting problems, it only takes a little creativity to find a solution. Driven by the need, I created my own network utility.

My requirements for this utility were:

- No data payload is sent or received, to avoid network congestion.
- Ability to check that both a host and the ports are alive and available
- Able to report the time to reach a host connect and disconnect
- Available to run on multiple platforms, in this case, Windows, macOS, and Linux
- Support for both Internet Protocol version 4 and version 6 addresses
- Parameters and reporting similar to ping – a familiar look and feel

With these objectives in mind, I created Zing. The name *Zing* is an acronym for Zero packet pING. I wanted the utility to have a name that sounds like ping, but I also liked the reference to *zinging* a system, much as an improv comic will *zing* a person in humor.

Application

Zing is implemented as both a command-line interface utility and a library function in Java. Using Java achieves cross-platform availability. Then, for each platform, I created a launch app (or *kicker app*) to run the Zing network utility.

Zing works by a simple approach: Measure the amount of time to connect and disconnect from specified ports for some limit to the number of operations. The idea behind Zing is not uniquely mine. I have read source code in network applications and libraries that takes a similar approach. Unlike the other tools I've seen, though, Zing is designed to look and operate more like ping.

This idea of connecting, binding, and disconnecting avoids sending any ICMP packets or other data. Zing simply requests to connect, connects, then disconnects. More measurements of each op give greater accuracy in the network timing. During this process, a host and port are determined to be reachable, available, and responding. Also, in some cases, a connection request can pass through a firewall to a remote host and port.

Photo by Johannes Plenio on Unsplash

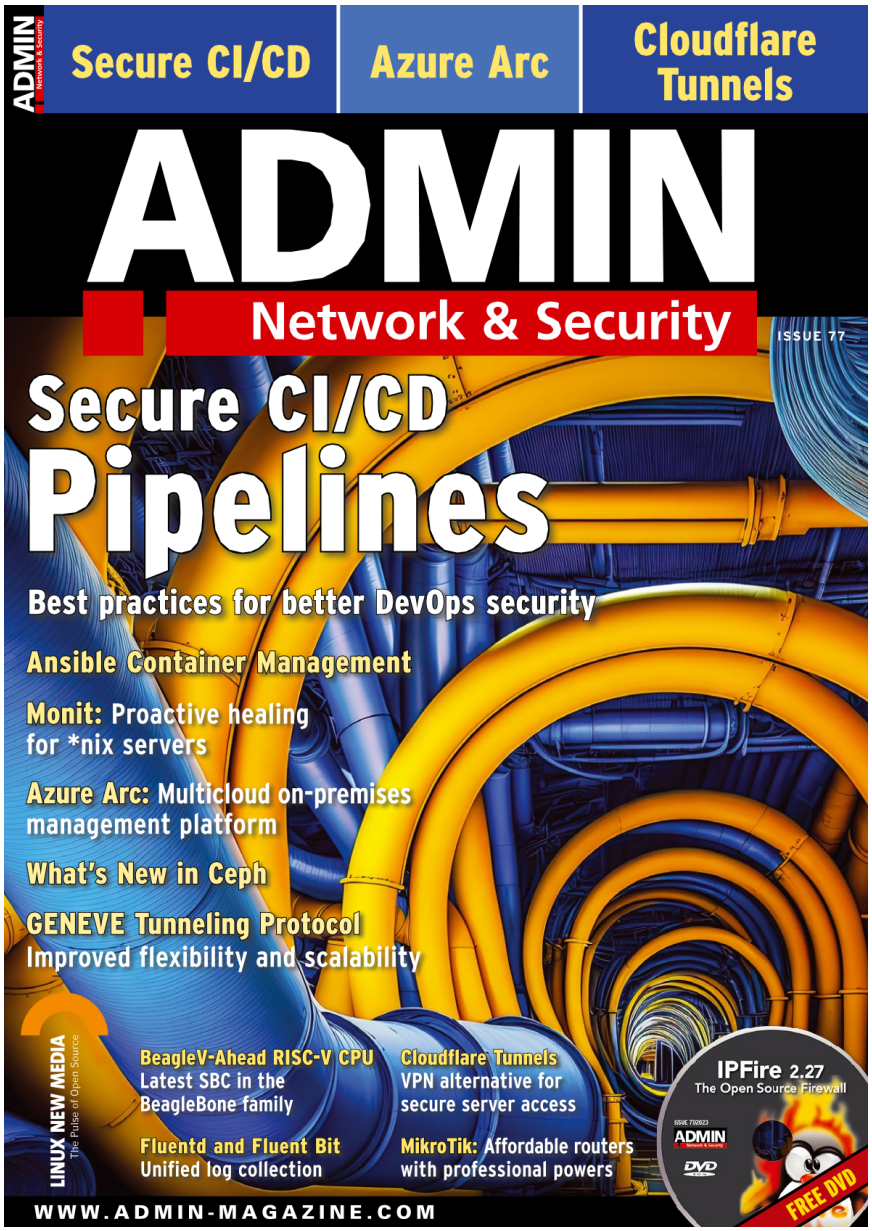
Linux Magazine Zing Article First Page

2.4 Porting to Other Systems

Once I had a working version both designed and implemented in Java, I then ported the zing network utility to other languages and platforms. At the time of this writing, the zing network utility has been ported to nine programming languages across the platforms of Linux, MacOS, and Windows.

2.5 Article in Admin Magazine

The C implementation of Zing is for Linux or MacOS and is more for scripting with a shell script. The bash implementation is an implementation using the netcat network utility and runs on Linux, MacOS, and Windows (sub-system for Linux).



Admin Magzine October 2023

The bash implementation uses netcat, and an article describing the bash version of the zing utility was published in October 2023 in *Admin Magazine*.

TOOLS | Zing Bash Script

Have a Bash with the Zing network utility

Zinger

We show you how to implement the zero-packet Ping utility, Zing, as a Bash shell script. By William F. Gilreath

Zing is a zero-packet network utility similar to Ping that I first introduced in December 2022 [1]. Zing has the distinct advantage that hosts without Ping capability can be zinged without sending network packets. Thus, zinging a remote host with multiple `zing` requests is not a "Ping flood" [2] attack, making Zing a network-safe utility that does not add network traffic. Zing is implemented in a variety of programming languages, is open source, and is available for download [3]. Yet the idea occurred to me to implement `zing` as a Bash shell script, which has the advantage of avoiding recompilation and of customization or tailoring for specific idiosyncrasies of a system platform or application.

Netcat

The netcat utility (`nc`) is a key command-line application and "Swiss army knife" of network utilities [4]. Netcat is described as a simple Unix utility that reads and writes data across network connections by the TCP or UDP protocol [5]. Originally developed for Unix, netcat has since been ported to many other computer platforms. Thus, netcat has the advantage of being somewhat ubiquitous across different operating systems.

The decision to implement `zing` with netcat in a shell script left one major design consideration decision. Put simply: Which shell to use? I develop on macOS, so Zsh seemed a likely obvious choice. However, I also work with Linux, so other shells were possible.

Ultimately I opted to go with Bash because it is ubiquitous, and I have done much more scripting for work and tinkering with the Bash shell. Moreover, Bash scripting is powerful, and with many users and a large pool of questions and answers on many tech boards, I was likely to find the solutions to unexpected problems, errors, and anomalies quickly. Thus, a Zing Bash script was the final choice.

Bash Shell Script for Zing

Having settled on netcat as the core of a Bash shell script to implement `zing`, two design principles were determined:

- Explicit specified parameters to eliminate ambiguity; an erroneous parameter then terminates the script.
- Early Failure; when a required explicit parameter is missing, the Bash script terminates and exits.

The Bash implementation with netcat has four essential sections: (1) initialize, check, and process command-line interface (CLI) arguments; (2) check for the host and get the hostname and host address; (3) echo and perform the `zing` operation and accumulate time; and (4) calculate simple statistics and report the results. Each section can then be broken down into specific operations and functions. To keep the code and design simple, all of these operations are centered on netcat as the only network tool.

Initializing sets the `zing` variable defaults for the count, operations limit, network timeout, port list, and remote host. After initialization, the CLI arguments then re-initialize specific `zing` parameters. This re-initialization requires some checks on the CLI arguments and number of arguments. The initialization section also checks for at least one parameter and whether the argument is asking for help. If the CLI argument is not `-h` or `--help`, then the arguments submitted are processed. The processing is a loop to identify the argument and its

Photo by Micah Tindell on Unsplash

Admin Magazine Zing Article First Page

Chapter 3. Idea and Concept

The idea and concept of the zing network utility is the answer to the question: “What is the zing network utility?” or “What is zing?”

The zing network utility is used to send and receive a packet, to a network device so that the zing network utility determines whether the network device is reachable and the time for the packet takes to reach the device.

The idea and concept of the network utility was from an initial conversation, and then later doing an initial implementation of a prototype to see if the idea was viable in software.

Yet the hypothesis, the original premise of the zing network utility, was to do a "better" ping in that it was simpler, lightweight in terms of network use, and more general than ping—but achieving the same result.

3.1 Introduction

The author has written this text about the zing network utility. The author expects the reader to be familiar with concepts of TCP/IP networking, IP addresses, ICMP packets, the Open System Interconnect (OSI) model, Domain Name Systems (DNS) systems, networking, ports, services on ports, sockets, and many other computer network concepts.

The author expects the reader to have used and know how to use both the ping network utility, and even better the `traceroute`, `tracert`, or `tracepath` network utility. If these utilities are unknown, then this book about the zing network utility will not be clear or comprehendible to the reader.

The author's focus in writing this text is on the zing network utility, not networking theory or network concepts. Many great articles, books, podcasts, and videos exist about computer networks, network theory, et cetera.

3.2 Ping Utility

The zing network utility is similar to the result of the ping network utility. Thus the prequel to the zing network utility, ping, will be examined, discussed, and covered.

3.2.1 History of Ping

The ping network utility was invented in December 1983 by Mike Muss at the United States Army Research Laboratory. Muss explains,

> *"My original impetus for writing PING for 4.2a BSD UNIX came from an offhand remark in July 1983 by Dr. Dave Mills while we were attending a DARPA meeting in Norway, in which he described some work that he had done on his 'Fuzzball' LSI-11 systems to measure path latency using timed ICMP Echo packets."*

Muss named the ping network utility after the sound an active sonar makes, the "ping" noise. The backronym Packet InterNet Groper or PING, has been used to describe the ping network utility name meaning.

The ping network utility was released into the public domain, and was included in the Berkeley version of UNIX, the Berkeley Software Distribution or Berkeley Standard Distribution or (BSD), now discontinued.

The ping network utility is available in every major operating system and has become a ubiquitous network utility.

3.2.2 Purpose of Ping

The purpose and function of the ping network utility is to determine the reachability of a host system on a network with the time to reach the network host, and the efficacy of the network to reach the host system. Network congestion, delays, and other performance and timing issues are detected and reported by ping.

3.2.3 Ping Operation

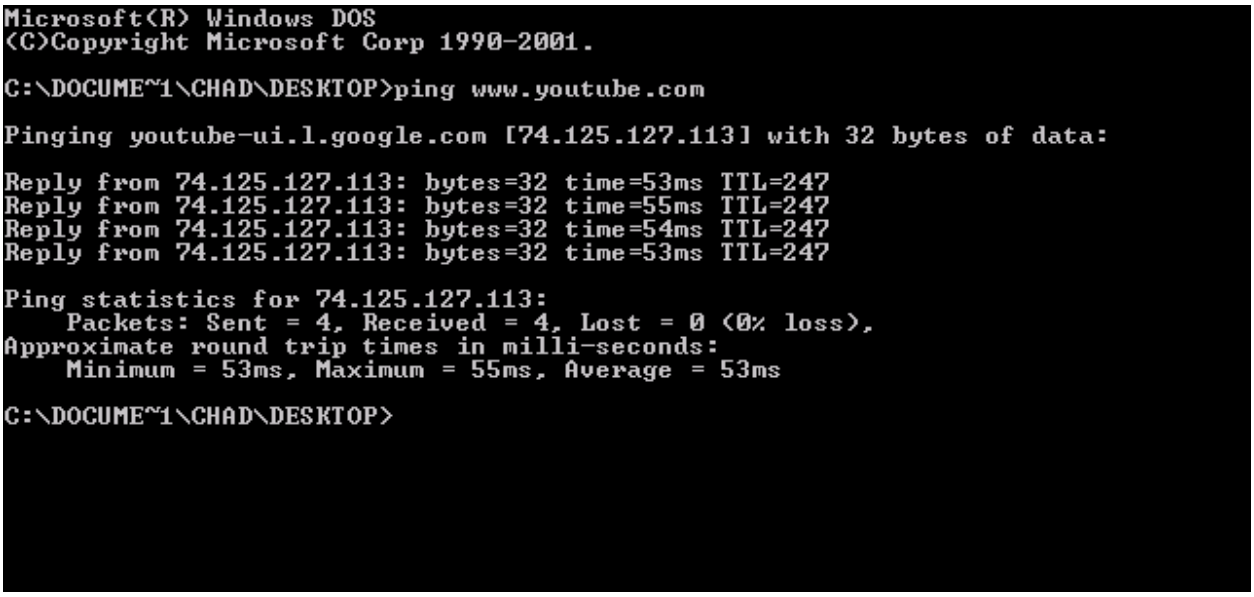
The ping network utility works by circumventing the TCP and Universal Datagram Protocol (UDP) protocols and packets. The host-to-host layer is unused and instead more "raw" sockets are utilized.

```
kousekip@ako-kaede-mirai ~ $ ping -6 -c 12 anilist.co
PING anilist.co(2606:4700:20::681a:e47 (2606:4700:20::681a:e47)) 56 data bytes
64 bytes from 2606:4700:20::681a:e47 (2606:4700:20::681a:e47): icmp_seq=1 ttl=64 time=26.7 ms
64 bytes from 2606:4700:20::681a:e47 (2606:4700:20::681a:e47): icmp_seq=2 ttl=64 time=22.8 ms
64 bytes from 2606:4700:20::681a:e47 (2606:4700:20::681a:e47): icmp_seq=3 ttl=64 time=24.6 ms
64 bytes from 2606:4700:20::681a:e47 (2606:4700:20::681a:e47): icmp_seq=4 ttl=64 time=23.0 ms
64 bytes from 2606:4700:20::681a:e47 (2606:4700:20::681a:e47): icmp_seq=5 ttl=64 time=27.4 ms
64 bytes from 2606:4700:20::681a:e47 (2606:4700:20::681a:e47): icmp_seq=6 ttl=64 time=22.0 ms
64 bytes from 2606:4700:20::681a:e47 (2606:4700:20::681a:e47): icmp_seq=7 ttl=64 time=23.0 ms
64 bytes from 2606:4700:20::681a:e47 (2606:4700:20::681a:e47): icmp_seq=8 ttl=64 time=23.5 ms
64 bytes from 2606:4700:20::681a:e47 (2606:4700:20::681a:e47): icmp_seq=9 ttl=64 time=23.4 ms
64 bytes from 2606:4700:20::681a:e47 (2606:4700:20::681a:e47): icmp_seq=10 ttl=64 time=22.0 ms
64 bytes from 2606:4700:20::681a:e47 (2606:4700:20::681a:e47): icmp_seq=11 ttl=64 time=24.3 ms
64 bytes from 2606:4700:20::681a:e47 (2606:4700:20::681a:e47): icmp_seq=12 ttl=64 time=25.2 ms

── anilist.co ping statistics ──
12 packets transmitted, 12 received, 0% packet loss, time 11016ms
rtt min/avg/max/mdev = 22.017/23.992/27.432/1.654 ms
kousekip@ako-kaede-mirai ~ $ ping -V
ping from iputils 20211215
```

Example Linux Ping

Instead, the ping network sends the ICMP packets, that are also utilized by the traceroute network utility.



Example DOS Ping

The ICMP packet is used to send and receive information about the reachability and timing of a host system on a network. ICMP packets are sent back to the sender from the receiver on the network. The ICMP packet transmission allows ping to know the round-trip time between two network systems, any errors, the packet loss, and the statistics of the results.

3.2.4 Problems with Ping

While the ping network utility has been around for nearly a half-century, it has been used maliciously to attack network host systems. By sending a payload of ICMP packets, and then the host system echos the ICMP packets back, there is potential to misuse the ping network utility.

3.2.5 Ping Attacks

One kind of ping attack is a distributed denial of service (DDoS) with multiple computer systems sending a flood of ping ICMP packets to a targeted computer system host on a network, usually the Internet.

A variation on the ping flood is the "Ping of Death" attack on a network system host on a network. A flood of malformed or oversized ICMP packets that exceed the maximum IPv4 packet size of 65,535 bytes are sent. This flood of oversized packets crashes or freezes the targeted network computer system as it is overloaded trying to process the oversized ICMP packets.

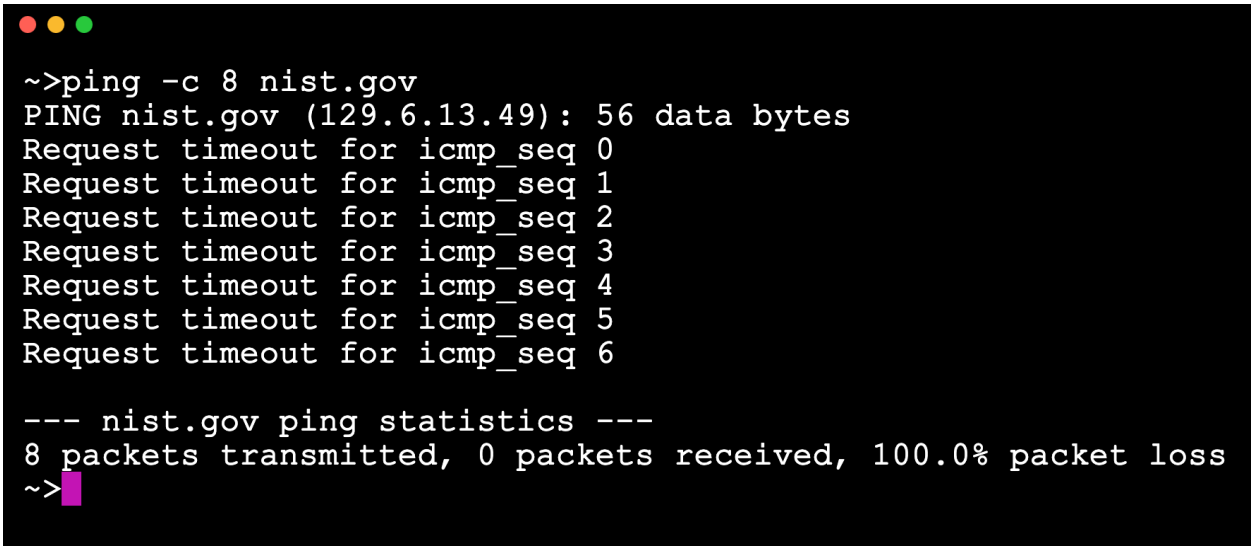
The ping attacks work by sending a payload of data to the targeted network computer system host; as zing has no payload this kind of attack is obviated.

3.2.6 Ping through Kernel

The ping network utility is not port-oriented; when a network host system is “pinged” it works through the operating system kernel, and the ICMP packets are echoed back to the ping network utility client. Thus a ping network service on the host system responds with ICMP packets, a classic send-receive echo client and server.

3.2.6 Problems with Ping

Given that the ping network utility can be used maliciously to attack host systems on a network, some host systems on a network do not have a ping capability. One such host system is for the United States National Institute of Standards and Technology (NIST) or the “nist.gov” domain.



ping nist.gov

3.3 Zing is a Zero-packet Ping

The zing network utility is conceptually a "zero-packet" ping. The reachability and time for a host system on a network are tested, checked, and reported like with the ping network utility.

3.3.1 Zing as Ping Utility

A zero-packet ping does what ping does, but using zero packets, no payload—hence the zero-packet internet groper (ZING) network utility. The primary difference is that no payload of packets is sent to a host on a network and then received back as an echo.

3.3.2 How Zing Works

The zing network utility works, or operates by simply connecting to a socket on a port, and then disconnecting from that socket on a host system on a network. The only packets sent are to connect and then disconnect.

The "zero" in the zing network utility is that zero packets as a payload are not sent to the remote host on the network. There are no sending packets as a payload to a host on a network, received

by the host, and then the packets repeatedly echo back to the zing network utility.

3.3.2.1 Zing TCP/IP Protocol Operation

The zing operation using the TCP/IP operation is a connect and then immediately disconnect from a socket on a host on the network.

Thus there are two operations on a socket to a host from the zing network utility:

1. Connect using TCP/IP protocol.
2. Disconnect using TCP/IP protocol.

Note that between the connect and disconnect there is no payload—no packets sent or received, hence the “zero” packet nature of the zing network utility.

3.3.2.2 Connect with TCP/IP Protocol

Connect via TCP/IP protocol is a three-way handshake, thus three packets are sent from the zing network client to the host system. The TCP/IP connect process is:

1. zing-client sends a "SYN" packet to the host.
2. host system sends a "SYN" or "ACK" packet to the zing client.
3. zing-client sends "ACK" to the host.

The overall connection process with a TCP/IP socket uses three packets.

At this point, a TCP/IP connection is open on the socket with a port. The diagram illustrates the TCP/IP connect process:

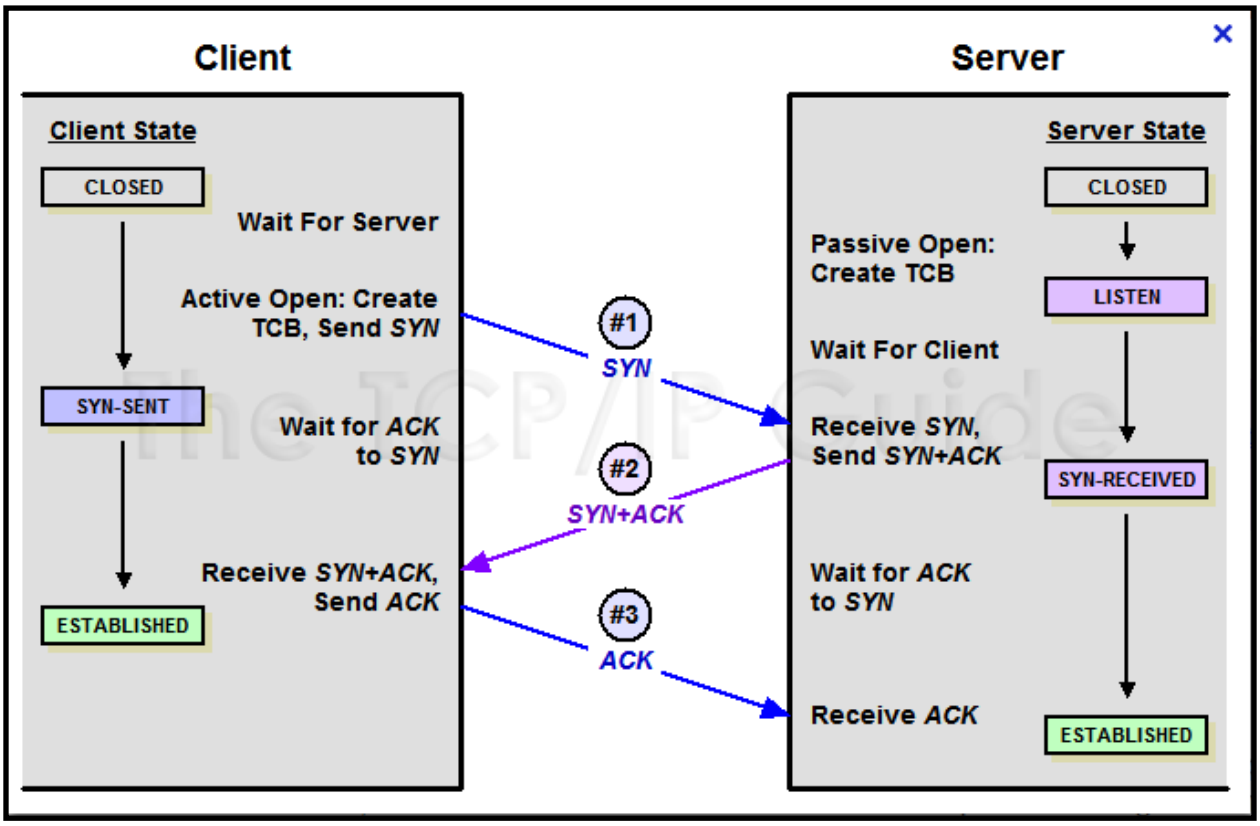


Diagram TCP/IP Connect between Client to Server

3.3.2.3 Disconnect with TCP/IP Protocol

Zing does a "graceful" clean disconnect, not abruptly by intention and design. Although ultimately it depends on the operating system of the platform, and the implementation of the programming language.

The TCP/IP disconnect is a two-way handshake twice, so four packets are sent. The TCP/IP disconnect process is:

1. zing-client sends a "FIN" packet to the host.
2. host system sends an "ACK" packet to the zing client.
3. host system sends a "FIN" packet to the zing client.
4. zing-client sends an "ACK" packet to the host.

At this point, a TCP/IP connection is closed on the socket with a port. The overall disconnect process with a TCP/IP socket uses four packets.

However, the actual process is determined by the operating system on the zing client that closes the TCP/IP socket. The diagram illustrates the TCP/IP disconnect process:

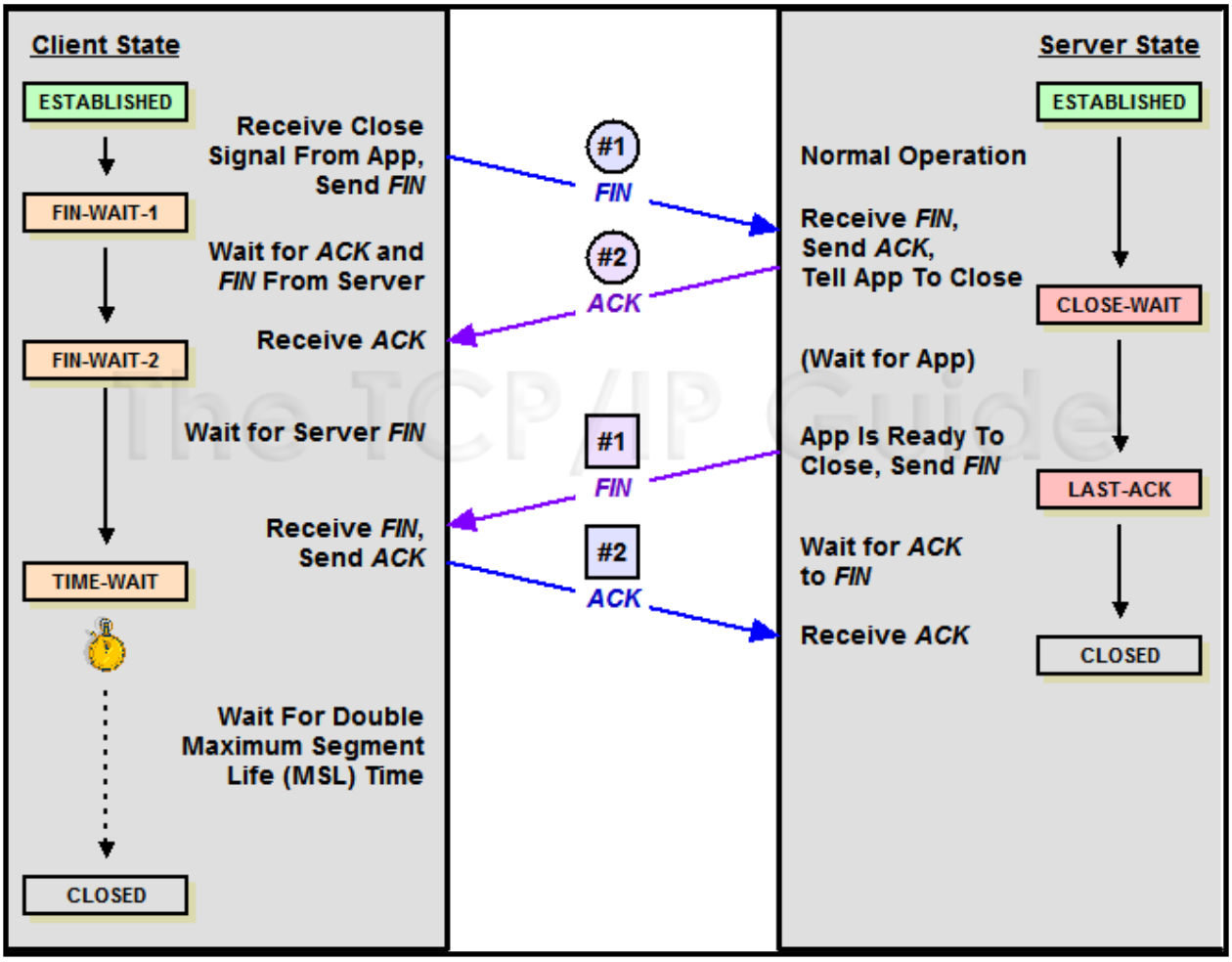


Diagram TCP/IP Disconnect between Client to Server

3.3.2.4 Zing TCP/IP Operation Synopsis

The zing operation that "zings" a host system on a network requires both a connect and disconnect operation in tandem. This overall operation sends eight TCP/IP packets between the zing client and the remote host system.

Thus the zing network utility does send packets but it is the bare minimum required with TCP/IP protocol to simply connect and disconnect from a network host system. There is no payload, so zero-packets sent and received by the zing network utility.

3.3.3 Functionality

The zing network utility does not work through the operating system or the operating system kernel, but on a service on a port for a host on a network. The two default ports that the zing network utility uses are port 80 for HTTP and port 443 for HTTPS. Both ports are for an insecure and secure hypertext transfer protocol service—a web server.

However other ports for a host on a network can be used by the zing network utility. For example, on a host for a DNS server, port 53 can be used. One common service on a port but not anymore for security reasons is the character generator service on port 19.

The zing network utility does not require any kind of special functionality through the operating system kernel, or on a port for a host on a network. An existing service on a host is sufficient to zing the host on a network.

3.3.4 Three Questions

The zing network utility uses a design philosophy or approach of three questions to do a zing operation on a port for a host on a network. These three questions are:

1. Exists — does the host exist on the network
2. Present — is the host present on the network
3. Active — is the host active on the network port

The design philosophy of the zing network utility is to answer the three questions and fail early. The three questions for the zing network utility are: “EPA” for Exist, Present, and Active. By failing early if any question is negative, then the zing network utility will report this, halt, and exit.

3.3.4.1 Exists

The host exists is the simplest question. For a host of a given domain (such as the author's domain "`wfgilreath.xyz`" if there is no Internet protocol (IP) address for the given host, then the host cannot exist on the network. There is no method to reach the host if it does not have an active, valid IP address.

Active IP address means for the canonical hostname, the zing network utility can get the address. For valid, given the choice of IP version 4 (the default) or IP version 6 does the address exist in that type of IP address? If negative, then again the zing network utility fails early. Thus the requirement or the answer to the question of "Does the host system exist?" is whether there is a valid IP address for that host system on the network.

3.3.4.2 Present

If the host exists and therefore has a valid IP address, the next question is if the host is present at the IP address. A host system on a network is present if the zing network utility can reach the host on the network.

The host system on a network might have an IP address that is valid but is unavailable at that IP address. Again if the answer is negative, the zing network utility reports this, halts, and exits.

Thus the answer to the requirement for the host system or the question “Is the host system present on the network?” or to be present on the network is that it is reachable.

3.3.4.3 Active

If the host exists with a valid IP address and is present on the network, then the last question to answer is if the host is active on the port or ports. Again the host system on a network is active on a port if there is a service running on that port. The service itself is immaterial, the service need only be active so that the zing network utility can connect and disconnect to do the zing operation on the host at the port specified. The zing network utility does not require a specific “zing” service on a specific port.

If the answer is negative, then the zing network utility will report, halt, and exit. The host might be present and reachable at a valid IP address, but if not active on the port or ports, then the zing network utility will give up. The answer to the question “Is the host active on a port?” is if a service is running and is connectable by the zing network utility.

3.4 Zing Synopsis

The three questions for the zing network utility are: “EPA” for Exist, Present, and Active.

The three questions are answered by the zing network utility, and if all are affirmative, then the zing network utility can then time the network performance.

If any of the three questions are answered in the negative, then information about the host on a network is given from the three questions, and the zing operation is not possible for the host IP address and type, the host itself, and the port or ports.

For security, some hosts on a network do not have ping capability. Attempting to ping them will result in total failure of sending the ICMP packets. For the zing network utility, this is equivalent to a negative answer for the three questions of existing, active, and present.

3.5 Default Host Ports

One question is the design decision to use two default ports: HTTP, and HTTPS. There are many other ports on systems with active services. The table lists some (but not all) of the ports on a system with the service, and a brief description.

Port	Service	Description
19	CHARGEN	Character Generator
21	FTP	File Transfer Protocol
22	SSH	Secure Shell Access
25	SMTP	Simple Mail Transport Protocol
53	DNS	Domain Name Service
80	HTTP	HyperText Transfer Protocol
110	POP3	Post Office Protocol
143	IMAP	Internet Message Access Protocol
389	LDAP	Lightweight Directory Access Protocol
443	HTTPS	HTTP Secure Service

Table of Network Ports and Services

These ports on a host system on a network are port 80 for HTTP and port 443 for HTTPS. Both are for hypertext transfer protocol, port 80 is unsecured, and port 443 is secure.

Yet why these ports? As there are 65536 or 2^{16} ports to choose from on a host computer system on a network.

The design decision for HTTP and HTTPS is simple, both services are the most prevalent and ubiquitous on the Internet.

A user for a specific port, such as a local printer, can specify that requirement in the command-line arguments.

The majority of network activity and traffic on the Internet is related to web, HTTP, or HTTPS services. Molte states, “The vast majority of API traffic is the result of POST or GET requests (98% of all requests).” [Molte 2022]

Hence the most popular, most widely used service is for web service or web APIs, so those are the two defaults in the zing network utility.

3.6 Operation Synopsis

The zing network utility operates using the TCP/IP over a network for a host on a port. Once the host is confirmed to exist, is present on the network, and is active on the port or ports on the host.

The zing operation is to open a TCP socket locally and then connect via TCP/IP over the network to a remote host on a port or ports. Once connected, the zing operation is to disconnect from the remote host on the port or ports and close the local TCP socket. A simple connect and disconnect over TCP/IP with sockets is a zing operation.

3.6.1 Socket Connect

The TCP/IP connect operation between two systems on a network is illustrated in the diagram:

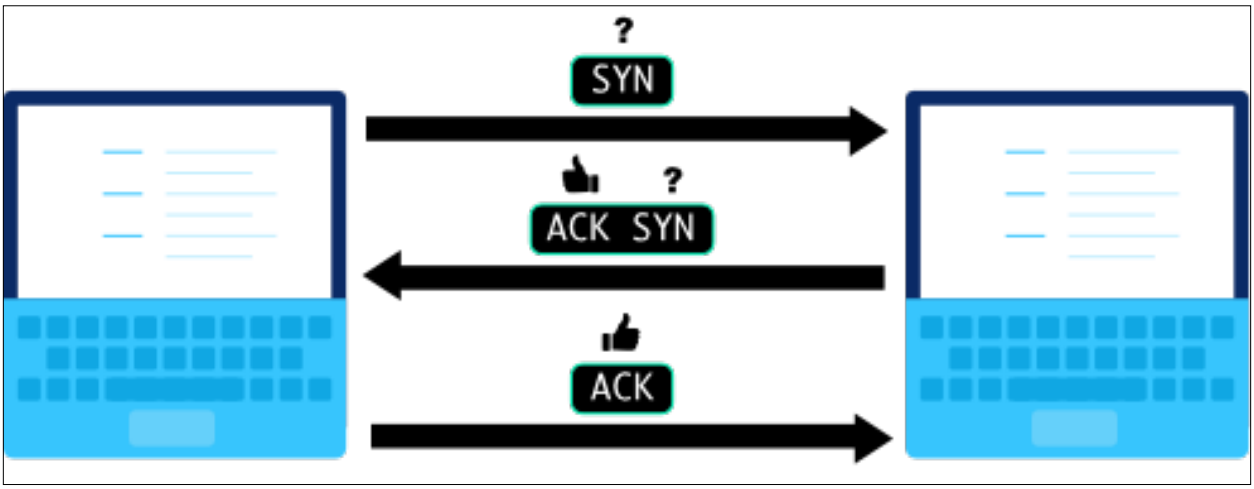


Diagram TCP/IP Connect Operation with Packets

This is the socket connect operation over TCP/IP. Overall the operation requires about four TCP/IP packets.

3.6.2 Socket Disconnect

The TCP/IP disconnect operation between two systems on a network is illustrated in the diagram:

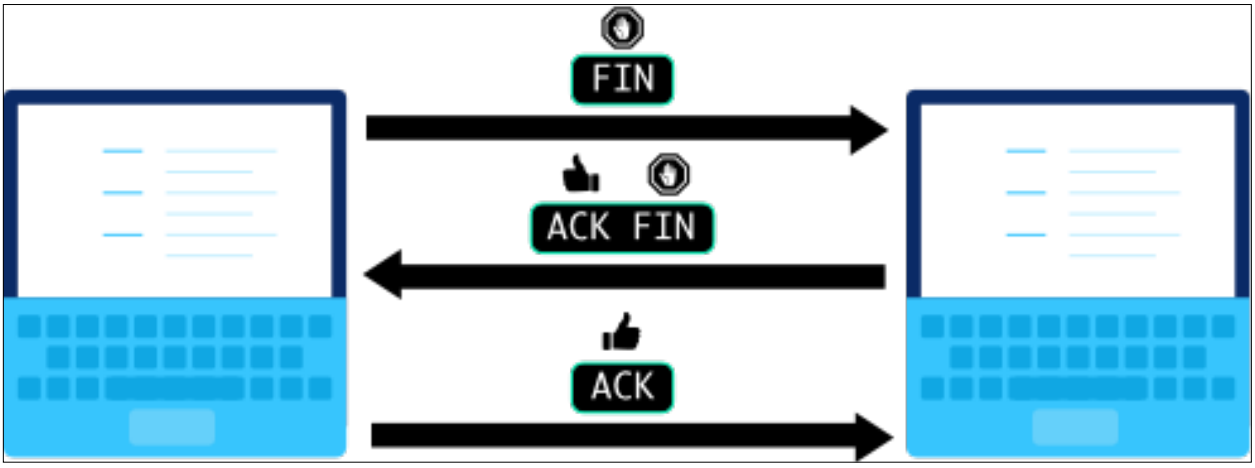


Diagram TCP/IP Disconnect Operation with Packets

This is the socket disconnect operation over TCP/IP. Overall the operation requires about four TCP/IP packets.

3.6.3 Operation Timing

The zing operation is timed by elapsed wall clock time (not process or thread time) in milliseconds, so that the time performance statistics are collected, processed, and reported along with the overall time for the zing network utility to zing the host on the port or ports on the network.

3.7 Performance Statistics

The zing network utility does a zing operation on a host for the port or ports and gathers timing information. The basic zing operation is repeated multiple times, the number of operations or ops per cycle. This repetition of zing operations avoids any extreme values. Then the number of cycles of zing operations is done, and the performance time is stored and reported.

The zing network utility then calculates the basic statistics as the ping network utility does. The timing statistics are:

- minimum — the extreme value that is the least, smallest value for time
- maximum — the extreme value that is the greatest, largest value for time
- average — the overall mean, or average value of time
- standard deviation — the dispersion or variance of the timing values

From the statistics, the overall performance of reaching a host for the port or ports is given for the user based upon the command-line interface (CLI) parameters to the zing network utility.

The zing network utility is like the ping network utility, achieving the same end, yet the zing network utility is not the ping network utility as it does not send any packets as a payload.

Yet there are some significant differences between zing and ping, much more beyond the first letter of the name of each utility. There are three distinctions between the zing and ping network utilities.

3.8 Zing in Contrast to Ping

The zing network utility is like or akin the the ping network utility. A simile is that ping is like an internal combustion engine automobile, whereas zing is like an electric automobile. The same entity is a conveyance that is driven between two points, yet very different in terms of motive power.

Both network utilities achieve the same result with the network time to reach a network host system. Yet while similar, they are very different. The important idea is that the zing network utility is not a replacement for ping (as zing is not the same as ping) but a complement for the ping network utility. Replacement implies that both zing and ping are interchangeable and both are certainly not.

3.8.1 Packet Payload

The ping network utility sends an ICMP packet, which is data, a payload sent from one computer system to another computer system. Thus the ping network utility sends a payload of packets over a network.

However, the zing network utility does not send packets or a payload. The zing network utility connects and disconnects by sending a series of IP packets using the TCP protocol through a

network socket in the software. There is a connection and disconnect, but beyond that process, no data payload is sent to the host system on the network.

Hence the “zero” in the zero packet internet groper, there is zero datum sent or received—zero payload in contrast to the ping network utility.

3.8.2 Zing Safety

The zing network utility, by the feature that no data payload is sent to the remote host on the network, has greater safety in contrast to the ping network utility that sends ICMP packets.

The ping network utility has been used to do an "ICMP packet" flood or "ping flood" on various hosts in a denial-of-service attack (DoS) or with many systems a distributed denial-of-service attack (DDoS) on a host system. ICMP packets are sent en masse a massive flood, without waiting for a response.

The infamous "ping of death" has been used to overload a network system, causing the host to crash, and freeze by sending ICMP packets that are malformed. The targeted host system on the network struggles to process the mangled, corrupted packets by an attacker.

The zing network utility only connects and disconnects to a host on a network. There is no data payload both sent and received. Yet DNS servers on DNS port 53, and such search services as Google, and Bing on ports HTTP port 80, and HTTPS port 443 are designed to serve massive numbers of connections that then disconnect.

Thus the zing network utility is inherently safer as it simply does not send data to and from the host system on a network.

However, too many zing operations on a port on a host system on a network would be the equivalent of an online system being overloaded. With too many connection requests for the service present on the port of the host system cannot handle the load of the connection requests. Many online services have this transpire when the service cannot handle the volume of connections sent from many different systems.

3.8.3 No Zing Service

Most network services require an actual service to run for a network client to access the service on the host computer system. The ping service is not different, although ping does not run on a specific port for the host computer system.

The zing network utility does not require an explicit zing service, unlike ping which requires ping to be enabled for the ping utility to function for that host computer system. Thus the zing network utility can connect to any TCP/IP service on a port on the host computer system on a network.

An example is nist.gov which does not have a ping service available. Yet the domain and network host system “nist.gov" does have a web service running on ports 80 for HTTP and 443 for HTTPS.

```
~>zing -c 4 -op 2 -p 80  -t 1200 nist.gov

ZING: nist.gov (129.6.13.49): 1 ports used, 2 ops per cycle.

ZING: Port: 80    nist.gov [129.6.13.49] Time: 96.008-ms.
ZING: Port: 80    nist.gov [129.6.13.49] Time: 95.293-ms.
ZING: Port: 80    nist.gov [129.6.13.49] Time: 95.691-ms.
ZING: Port: 80    nist.gov [129.6.13.49] Time: 96.346-ms.
ZING: Port: 80    nist.gov [129.6.13.49] Time: 93.716-ms.
ZING: Port: 80    nist.gov [129.6.13.49] Time: 91.181-ms.
ZING: Port: 80    nist.gov [129.6.13.49] Time: 93.545-ms.
ZING: Port: 80    nist.gov [129.6.13.49] Time: 96.796-ms.

--- zing summary for nist.gov/129.6.13.49 ---
8 total ops used; total time: 764 ms
total-time min/avg/max/stddev = 184.897/141.819/192.037/47.907 ms

~>
```

zing nist.gov HTTP port

Thus the zing network utility can zing the domain nist.gov without the domain and the host computer system having a zing network service on the host computer system.

3.9 Advantages of Zing

The zing network utility has five distinct advantages:

1. Host does not have a "zing" service, and does not know of zing operation explicitly.

2. Zing cannot "packet swarm" as it only connects, disconnects, and does not send any payload.

3. Zing is a “lightweight ping" in that no payload is sent or received from the zing network utility client and host.

4. Zing network utility can function where there is no ping service.

5. Zing network utility uses TCP/IP which is connection-oriented and thus reliable in terms of operation.

3.10 Disadvantages of Zing

The disadvantage of the zing network utility is that if there is no service active on the port of the host, then the zing network utility cannot do a zing operation.

The zing network utility cannot zing “www.caamp.info” on port 53, the DNS server port with the default timeout parameter value.

zing DNS port www.caamp.info

Yet the zing network utility can zing “www.caamp.info” on port 80.

```
~>zing -c 4 -op 4 -p 80 www.caamp.info

ZING: www.caamp.info (52.33.207.7): 1 ports used, 4 ops per cycle.

ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 28.461-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 23.936-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 27.517-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 29.386-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 32.134-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 29.476-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 30.175-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 27.670-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 30.079-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 28.359-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 33.214-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 27.545-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 32.747-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 28.697-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 33.508-ms.
ZING: Port: 80    www.caamp.info [52.33.207.7] Time: 30.872-ms.

--- zing summary for www.caamp.info/52.33.207.7 ---
16 total ops used; total time: 534 ms
total-time min/avg/max/stddev = 109.300/91.119/125.824/27.957 ms

~>
```

zing HTTP port www.caamp.info

There is a service present on the host “www.caamp.info” on port 80, whereas on port 53 there is no service present.

3.11 Zing Synopsis

While the are important distinctions and differences between zing and ping, the most significant idea is that zing is a complement or accompaniment to ping. The zing network utility is not a replacement, successor, or substitute for the ping network utility.

Two important contradistinctions between the zing network utility and ping are:

- The zing network utility does not require a service to zing a network host system, whereas ping does.
- The zing network utility sends no packet payload to a network host system, whereas ping does.
- The zing network utility is a simple connect and disconnect, not a specific “zing” operation, whereas ping is very specific.

Chapter 4. Utilizing

The zing network utility is utilized or used for a zing operation by the input and output. The input supplies the parameters of the zing operation, and the output quantifies the results in terms of network time, and some simple statistics.

4.1 Zing Input

Using the zing network utility at the CLI is very simple. The six parameters are passed at the CLI to the zing network utility. These parameters relate to the zing operation, and then the network configuration.

CLI Argument	Parameter
-4	use TCP/IP v4 (32-bit address)
-6	use TCP/IP v6 (128-bit address)
-c	set count of zing operations per port
-op	set limit of zing operations per cycle
-p	set list of ports to use on host
-t	set the network timeout in milliseconds
<host>	set host address or name to zing

Table of Command Line Interface Arguments and Zing Parameter

4.1.1 Operation

The zing operation, to zing a host on a port for a system requires the bare minimum of parameters:

- host — the canonical name of the host on the network, although an IP address can be used. The default host is “localhost” or “127.0.0.1” the computer system that the zing network utility is running on when utilized.

- port list — the list of ports to zing, delimited by commas as separators in the list. The port list is optional, as the zing network utility has the default ports as port 80 for HTTP and port 443 for HTTPS.

4.1.2 Network Settings

The network configuration parameters or settings relate to how zing will operate on a network for the zing operation. If not provided, the zing network utility will use the default value for each parameter.

The four network parameters are:

- address type - the type of TCP/IP address version 4 or version 6 to use by zing on the host. While TCP/IP version 6 is a superset of version 4, not all hosts are reachable or present on the network in TCP/IP version 6. The default is TCP/IP address version 4.

- count - the count of the number of zing operations to do on a host for a port. The default count is 4 zing operations.

- limit - this is the operational limit, the number of operations or cycles of zing operations per overall zing operation. Multiple operations or cycles of zing operations are used to reduce the impact of extreme values of zing operation time on the overall timing of the zing network utility. The default limit is 4 operations or cycles per zing operation.

- timeout - this is the network timeout, the waiting period to connect to a remote host system. The default value for a network time is 4000 milliseconds or 4 seconds.

4.2 Zing Output

For the output, the output is a report, of the time to zing a host on a port, the same format and structure of the datum of the ping utility is used for familiarity and comprehension.

```
~>ping -c 4 vmware.com
PING vmware.com (208.91.0.132): 56 data bytes
64 bytes from 208.91.0.132: icmp_seq=0 ttl=54 time=55.360 ms
64 bytes from 208.91.0.132: icmp_seq=1 ttl=54 time=50.326 ms
64 bytes from 208.91.0.132: icmp_seq=2 ttl=54 time=53.518 ms
64 bytes from 208.91.0.132: icmp_seq=3 ttl=54 time=52.153 ms

--- vmware.com ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 50.326/52.839/55.360/1.844 ms
~>
```

ping vmware.com

```
~>zing -c 4 -op 1 -p 443 vmware.com

ZING: vmware.com (208.91.0.132): 1 ports used, 1 ops per cycle.

ZING: Port: 443   vmware.com [208.91.0.132] Time: 52.348-ms.
ZING: Port: 443   vmware.com [208.91.0.132] Time: 44.711-ms.
ZING: Port: 443   vmware.com [208.91.0.132] Time: 54.876-ms.
ZING: Port: 443   vmware.com [208.91.0.132] Time: 46.071-ms.

--- zing summary for vmware.com/208.91.0.132 ---
4 total ops used; total time: 203 ms
total-time min/avg/max/stddev = 44.711/36.414/54.876/13.755 ms

~>
```

zing vmware.com

The zing network utility does not use a different style or structure for the output report. The zing network utility output is similar to the ping network utility by design.

Thus the zing network utility does not require any re-learning to comprehend, interpret, and understand the output information.

4.2.1 Output Report

The author's host is zinged on ports 80 for HTTP and 443 for HTTPS. Consider this example of using the zing network utility:

```
ZING: wfgilreath.xyz (44.230.85.241): 2 ports used, 4 ops per cycle.

ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 31.591-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 27.607-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 28.201-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 26.039-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 30.636-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 27.205-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 27.407-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 26.422-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 25.794-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 28.083-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 27.393-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 29.200-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 30.637-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 31.378-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 25.567-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 32.740-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 29.093-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 30.164-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 26.616-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 27.622-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 27.823-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 26.837-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 28.174-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 25.971-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 31.129-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 28.157-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 28.539-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 31.373-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 29.379-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 25.538-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 28.727-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 31.384-ms.

--- zing summary for wfgilreath.xyz/44.230.85.241 ---
32 total ops used; total time: 1009 ms
total-time min/avg/max/stddev = 53.829/49.588/62.015/7.890 ms

~>
```

zing wfgilreath.xyz output report

4.2.2 Report Format

The zing network utility creates a zing output report with three sections:

- The header of the report output, which is a summary.
- Each zing operation with IP address, port, and timing.
- A final summary of the zing operations with some basic statistics.

4.2.2.1 Report Header

The zing network utility summary of the host, the host IP address in the specified TCP/IP version, and then the number of ports used, the size of the port list, and the number of operations per cycle.

```
ZING: wfgilreath.xyz (44.230.85.241): 2 ports used, 4 ops per cycle.
```

zing wfgilreath.xyz report header

4.2.2.2 Report Operations

The zing operation on each port of the host, for the number or count of zing operations used to zing the host. Each operation consists of the port, the hostname, the host IP address in the TCP/IP version, and then the time to zing the host on the port.

```
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 31.591-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 27.607-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 28.201-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 26.039-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 30.636-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 27.205-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 27.407-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 26.422-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 25.794-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 28.083-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 27.393-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 29.200-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 30.637-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 31.378-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 25.567-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 32.740-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 29.093-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 30.164-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 26.616-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 27.622-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 27.823-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 26.837-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 28.174-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 25.971-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 31.129-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 28.157-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 28.539-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 31.373-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 29.379-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 25.538-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 28.727-ms.
ZING: Port: 443   wfgilreath.xyz [44.230.85.241] Time: 31.384-ms.
```

zing wfgilreath.xyz report operations

4.2.2.3 Report Summary

The summary of the zing operation on the host system on the ports.

```
--- zing summary for wfgilreath.xyz/44.230.85.241 ---
32 total ops used; total time: 1009 ms
total-time min/avg/max/stddev = 53.829/49.588/62.015/7.890 ms

~>
```

zing wfgilreath.xyz report summary

The zing report is total number of zing operations (the product of the count of operations and the number of ports zinged) and the total time. The statistics of minimum time, average time, maximum time, and the standard deviation or the variance in the time values.

4.2.2 Error Reporting

For an error, the error is reported, and the zing network utility then terminates. For an error the user must retry, the zing network utility will not make presumptions.

An example is connecting to the author’s domain “caamp.info" on port 53 which is the DNS protocol port. Consider the error report from the zing network utility:

zing www.caamp.info DNS port time out

Since a DNS server is not running on the author’s domain on port 53, the attempt to connect will time out.

4.2.3 Error Types

There are five known types of errors using the zing network utility, but overall there are six possible error types, although there is the possibility of spurious errors from the computer system and network, these are unpredictable random errors.

4.2.3.1 Known Error Types

The five possible errors when using the zing network utility are given in the table, along with the zing network utility question.

Error Type	Zing Question
Connection refused by host on the port	Active
Host domain not found in DNS	Exist
The host is unreachable on the network	Present
IP address version not supported	Exist
Time out connecting to host on port	Present

Table of Zing Network Utility Errors

There is one error type that cannot be specifically named, or the zing network utility question specified. This is the unknown, or a transient, or random error.

4.2.3.2 Unknown Error Types

Unknown, or random errors are those unexpected, improbable situations.

For example, the UART (Universal Asynchronous Receiver Transmitter) [Campb 2024] chip in the network modem is damaged, or there is a power failure on the router—a hardware failure.

Yet another example is that a network cable is unplugged or flexed while the network is in operation. Thus the cable no longer carries any network packets, hence a network failure that is random, unexpected, and unknown.

This is the category of error that is unknown and random beyond the zing network utility. When this kind of error type occurs, the user must manually redo the zing network utility operation to the host and port on the network.

4.3 Zing Examples

Some examples of using the zing network utility are given and discussed. The primary variation is on the ports, with the default port 80 for HTTP and port 443 for HTTPS. The other variation is on the TCP/IP version, either version 4 or 6. The count of the number of zing operations, the number of operations per cycle, and the timeout varies ad infinitum.

The important idea is that if the zing operation fails, try again with a variation of the parameters. Each host system on the network has its own particular idiosyncrasies based upon the service, and the host platform system. More simply as the old maxim goes, if at first you do not succeed with your zing operation, try again.

4.3.1 Nothing Parameters Example

The example is having nothing parameters—no parameters passed to the zing network utility.

```
~>zing
usage: zing ( [-4|-6] [-c <count>] | [-op <limit>] | [-t <timeout>] <host> | -h )
zing -p 80,443 1.1.1.1
zing -4 -c 6 -op 4 -t 3000 -p 80,443 google.com
~>
```

zing CLI No Parameters

This is the same as using the help -h parameter with the zing network utility.

```
~>zing -h
usage: zing ( [-4|-6] [-c <count>] | [-op <limit>] | [-t <timeout>] <host> | -h )
zing -p 80,443 1.1.1.1
zing -4 -c 6 -op 4 -t 3000 -p 80,443 google.com
~>
```

zing -h Help Parameter

A no parameters or nothing parameters causes the zing network utility to show the help parameter response information.

4.3.2 Implicit Parameters with Host Name

The example is to use the implicit or default parameters with a host name. Only the host name ‘wfgilreath.xyz’ is passed to the zing network utility.

```
~>zing wfgilreath.xyz

ZING: wfgilreath.xyz (52.33.207.7): 2 ports used, 10 ops per cycle.

ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 167.859-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 31.143-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 29.812-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 29.138-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 27.496-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 28.458-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 29.919-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 27.567-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 28.598-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 29.493-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 29.150-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 23.945-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 26.244-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 28.281-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 31.377-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 26.249-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 27.080-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 28.345-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 31.844-ms.
ZING: Port: 80    wfgilreath.xyz [52.33.207.7] Time: 25.014-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 26.422-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 29.988-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 36.461-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 27.597-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 28.945-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 31.223-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 29.502-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 33.561-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 28.071-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 29.614-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 32.934-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 29.123-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 28.281-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 33.284-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 30.741-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 31.634-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 29.771-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 31.495-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 27.553-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 30.580-ms.

--- zing summary for wfgilreath.xyz/52.33.207.7 ---
40 total ops used; total time: 1323 ms
total-time min/avg/max/stddev = 138.532/105.391/285.448/95.055 ms

~>
```

zing Host Name with Implicit Parameters

4.3.3 Explicit Parameters

The example is to use the explicit parameters with a host name. The host name ‘wfgilreath.xyz’ is passed to the zing network utility with the default parameters given explicitly.

```
~>zing -4 -c 5 -op 4 -p 80,443 wfgilreath.xyz

ZING: wfgilreath.xyz (44.230.85.241): 2 ports used, 8 ops per cycle.

ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 28.207-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 39.968-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 31.581-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 27.815-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 30.371-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 30.013-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 31.664-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 31.202-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 31.029-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 31.922-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 30.309-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 28.925-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 27.926-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 27.631-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 27.702-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 28.997-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 30.320-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 27.865-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 30.243-ms.
ZING: Port: 80    wfgilreath.xyz [44.230.85.241] Time: 30.176-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 29.014-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 30.176-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 26.193-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 31.071-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 31.201-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 28.930-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 27.323-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 30.966-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 30.282-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 29.243-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 29.868-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 34.394-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 31.939-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 27.393-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 29.914-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 29.988-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 29.560-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 46.713-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 28.831-ms.
ZING: Port: 443   wfgilreath.xyz [52.33.207.7] Time: 29.083-ms.

--- zing summary for wfgilreath.xyz/52.33.207.7 ---
40 total ops used; total time: 1339 ms
total-time min/avg/max/stddev = 112.256/95.259/127.571/26.025 ms

~>
```

zing Host Name Default Parameters Explicit

The example is to use the explicit parameters with a host name, but using TCP/IP version 6. The host name ‘wfgilreath.xyz’ is passed to the zing network utility with the other default parameters given explicitly.

```
~>zing -6 -c 5 -op 4 -p 80,443 wfgilreath.xyz

ZING: wfgilreath.xyz (::ffff:52.33.207.7): 2 ports used, 8 ops per cycle.

ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 23.041-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 25.664-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 27.410-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 26.317-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 25.061-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 29.341-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 27.936-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 28.358-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 38.592-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 31.913-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 27.330-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 29.632-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 28.142-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 27.913-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 30.206-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 26.720-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 29.918-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 32.008-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 27.435-ms.
ZING: Port: 80    wfgilreath.xyz [::ffff:52.33.207.7] Time: 32.087-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 27.314-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 27.790-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 27.555-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 26.170-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 28.343-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 25.724-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 28.271-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 25.303-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 29.824-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 30.267-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 27.557-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 31.792-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 29.417-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 27.977-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 25.419-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 26.101-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 27.233-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 23.529-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 27.717-ms.
ZING: Port: 443   wfgilreath.xyz [::ffff:52.33.207.7] Time: 29.780-ms.

--- zing summary for wfgilreath.xyz/::ffff:52.33.207.7 ---
40 total ops used; total time: 1136 ms
total-time min/avg/max/stddev = 102.432/94.518/127.467/22.253 ms

~>
```

zing Host Name TCP/IP Version 6 with Default Parameters Explicit

4.3.3 Other Host Examples

Another host example with the zing network utility, is a local device on a local network. This example is from [Gilre 2023] zing the local device, a laser printer on the local network.

```
~>zing.bash -c 4 -op 2 -p 80,443 10.0.0.23

ZING 10.0.0.23/80 on 80 is: Active. Continue.

Port: 80: op 1.1. 10.0.0.23 80 Time: 31 ms.
Port: 80: op 1.2. 10.0.0.23 80 Time: 63 ms.
Port: 80: op 2.1. 10.0.0.23 80 Time: 29 ms.
Port: 80: op 2.2. 10.0.0.23 80 Time: 56 ms.
Port: 80: op 3.1. 10.0.0.23 80 Time: 28 ms.
Port: 80: op 3.2. 10.0.0.23 80 Time: 52 ms.
Port: 80: op 4.1. 10.0.0.23 80 Time: 129 ms.
Port: 80: op 4.2. 10.0.0.23 80 Time: 155 ms.

ZING 10.0.0.23/80 on 443 is: Active. Continue.

Port: 443: op 1.1. 10.0.0.23 443 Time: 28 ms.
Port: 443: op 1.2. 10.0.0.23 443 Time: 58 ms.
Port: 443: op 2.1. 10.0.0.23 443 Time: 33 ms.
Port: 443: op 2.2. 10.0.0.23 443 Time: 61 ms.
Port: 443: op 3.1. 10.0.0.23 443 Time: 36 ms.
Port: 443: op 3.2. 10.0.0.23 443 Time: 68 ms.
Port: 443: op 4.1. 10.0.0.23 443 Time: 29 ms.
Port: 443: op 4.2. 10.0.0.23 443 Time: 141 ms.

--- ZING 80/10.0.0.23 4 ops at 2 per op statistics ---
Time to 80/10.0.0.23: min/avg/max/stddev=26/81/77/19.18 ms

~>
```

zing device on local network

Chapter 5. Design Implementation

The design and implementation of the zing network utility is the software architecture of zing. At its simplest, the zing network utility software architecture consists of several functions that perform specific tasks for the operation of the zing network utility. Thus the software architecture is more functional, and depending on the platform, operating system, and programming language—simple to implement.

5.1 Software Architecture

One approach to implementing or porting the zing network utility is functional decomposition—by the critical, core functions in the source code for zing.

Take the zing network utility, and break it down or decompose it into function units that are then scaled. Essentially the zing network utility is implemented as a collection of functions that use other functions.

5.1.1 Functions of Zing Network Utility

The seven functions of the zing network utility are:

- condist()
- isIPAddr()
- process_args()
- zing_fun()
- zing_op()
- zing_stats()
- zing_util()

Each function is an element of operation in the zing network utility. There are two call chains of functions within the overall function call sequence.

These two function call series reflect the main sequence of function calls for the zing network utility. The secondary sequence is the function that calls for a zing function, operation, and then the atomic, basic connect-disconnect timing function.

5.1.2 Function condist()

The condist() function is the “CONnect-DIsconnect Socket Timing” function. The condist() function creates a socket on the host and port, connects, and disconnects.

The time to do this is returned as a real value. The timeout is passed to give some upper limit or least upper bound on the time limitation to connect to a host on a port.

This is the individual zing network utility “zing host on the port.” This answers the two questions of “Is the host present?”, and “Is the host active on the port?”

If both questions are affirmative, then the time to reach the host, connect, and disconnect on the port is then determined and returned. For an error, a sentinel value of -1 is returned, and within the condist() function an error is reported and/or logged.

The host and port operation, to get the IP address from the DNS server, answers the question “Does the host exist?” This can be done within the condist() function, or externally with the information passed as parameters to the condist() function.

The simplest is to pass the IP address and port, but some programming languages' network library requires the socket to be created on the domain name and port. Thus the condist() function is more formally specified with condist(host, port, time) with the time the timeout to connect to a host on a port.

5.1.3 Function process_args()

The process_args() function takes the CLI arguments that are parameters to the zing network utility and then determines each parameter, and the value. The parameter and value then override the default values for the various zing network utility parameters.

For example, if "-p 53" is given as the port for the host on the network, then the default ports of HTTP port 80 and HTTPS port 443 are overridden.

5.1.4 Function zing_op()

Once the functionality to create, connect, and disconnect on a host for a port is implemented in the condist() function, then the condist() function is used within the zing network utility parameter limit. The limit is the number of atomic operations to create a single “ZING OPeration. Hence the function is the zing_op() function.

The zing_op() function uses the condist() function but also takes the control parameter of the limit, or the number of operations to create an overall zing_op() function. A series of calls to the consist() function is needed to get an accumulated average time for a zing_op() to the host for the port.

The zing_op() function takes the same parameters as the condist() function, host, port, and time. Yet the zing_op() function takes the parameter limit. Thus the formal specification of the zing_op() function is zing_op(host, port, time, limit).

The limit is simply the upper bound on an internal loop. In the loop, the condist() function is called, and the time accumulated in the timer accumulator variable. After the loop completes in the zing_op() function, the value of the accumulated time divided by the limit is returned.

If the sentinel value of -1 is returned by the condist() function, then the zing_op() function returns immediately as some error has occurred, thus the zing_op() function cannot finish.

Pseudo-code for the zing_op() function, composing with the condist() function is:

```
fun real zing_op(host, port, time, limit)

  var real time  = 0.0
  var int  count = 0

  while count < limit do
    let count = count + 1
    let time = time + condist(host, port, time)
    if time == -1 then
      return -1.0
    fi
  done

  return time / limit

end fun
```

5.1.5 Function zing_fun()

The zing_fun() for “ZING FUNction” is congruent to the zing_op() function, only the count of the zing_op() is used. A table of count size is created, and the time information is stored. If the returned time from a call to the zing_op() function is -1 then the zing_fun() exits.

The zing_fun() takes the parameters of the zing_op() function, only with the count of zing operations to perform. Hence the zing_fun() formally specified is:

```
zing_fun(host, port, time, limit, count)
```

The implementation simply used a loop to call zing_op() and store the result in a time table, or array the size of the count.

```
fun zing_fun(host, port, time, limit, count)

  var int index = 0
  var int[] time_tbl = new int[count]

  while index < count do
    let time_tbl[index] = zing_op(host, time, port, limit)
    if time_tbl[index] == - 1 then
      nil
    fi
    let index = index + 1
  done
  return time_tbl

end fun
```

The zing_fun() function can either return the time table and then statistics are calculated on the table, or nil on an error.

5.1.6 Function zing_stat()

The zing_stat() function takes the timetable of zing_op values and then calculates some simple statistics such as the minimum value, maximum value, average value, and then the standard deviation of the timetable.

These values are reported at the end of the overall zing operation to a host on a port.

5.1.7 Function zing_util()

The zing utility function processes the command-line interface (CLI) arguments. Call process_args() function, which sets up the global variables that are initialized to default values. An error in the CLI arguments is reported, and the zing utility exits.

The zing utility function processes the CLI arguments and then calls the zing_fun() function for each port in the port list. The pseudo-code for the zing_util() function is:

```
fun zing_util(text[] args)

  process_args(args)

  var int time_table = nil
  for port in port_list do
    let time_table = zing_fun(host,port,iptv,time,limit,count)
    if time_table == nil then
      return
    fi
    zing_stat(time_table)
  done

end fun
```

The zing utility function is the upper-level function that calls the zing_fun() function, and that calls the zing_op() function which then calls the condist() function. This series of function calls is the zing operation on a host at a port.

Errors are reported, the function returns or the overall call of functions is successful, and the time to zing a host on a port is reported for the value of count iterations.

The zing utility function contains the starting function to process the CLI arguments process_args() and the closing function to report the statistics of the zing operation on the host zing_stats(). The loop over the ports in the port list, then calls the zing_fun() function to zing the host on a specific port.

5.1.8 Function isIPAddr()

The isIPAddr() function checks that for a given host name the IP address exists but also in the specific version or type of IP address, 32-bit (Internet address protocol version 4 or IPv4) address, or 128-bit (Internet address protocol version 6 IPv6) address.

The isIPAddr() function has the function parameter list of:

```
isIPAddr(host,ipV4Flag)
```

In the first argument, the "host" parameter is the canonical hostname of the host system on the network, such as:

- "`www.caamp.info`"
- "`www.wfgilreath.xyz`"
- "`www.zepton.xyz`"

(Note: As a disclaimer these example canonical domain names are owned by the author.)

The second argument, the "ipV4Flag" parameter is a logical or boolean type indicating if the address type specified is IP address IPv4 or if not true, then the address specified is the IP address IPv6. The second argument could be an enumerated type or other type; a simple true or false is the simplest form or parameter to the isIPAddr() function.

The isIPAddr() function can return a boolean true or false, or report an error and then halt the zing network utility. The implementation specifics depend on the programming language, the runtime environment, and the platform.

The zing network utility does not make any assumptions or presumptions about the user CLI arguments. The default IP address version or type is the 32-bit IPv4 address, should the user not specify the IP address type in the CLI parameters.

When the IP address does not exist for the given IP address type specified by the user, then an error is reported to the user, and the zing network utility exits with an error result.

Otherwise, when the IP address does exist for the specified version or type, and the IP address is used in the condist() function. A socket is created at the IP address for the specific type by the condist() function.

5.19 Other Functions

The seven functions given and explained are not the only functional approach to implementing the zing network utility. The software architecture is influenced by the platform and programming language used to implement the zing network utility.

An implementation in a more functional language might use a different approach. A cloud-based version of the zing network utility would be very different in terms of functions and functionality.

5.2 Zing Source Code

The zing network utility is implemented in seven programming languages for three different operating systems platforms Linux, MacOS, and Windows.

The zing network utility also runs on runtime platforms of the Java Virtual Machine (JVM) and the .NET Common Language Runtime (CLR).

The zing network utility is implemented in the nine programming languages:

- bash - for Linux, MacOS, Windows (POSIX shell)
- C - for Linux, MacOS
- C# - cross-platform .Net CLR
- Golang
- Java - cross-platform JVM
- Kotlin
- Python
- Ruby
- Rust

The Java version of the zing network utility was discussed and described in a Linux magazine article [Gilre 2022], and the bash version of the zing network utility was described in an Admin magazine article [Gilre 2023].

5.3 Design and Implementation

The zing network utility has been implemented or “ported” to nine programming languages on three operating systems, and two runtime platforms as of this writing February 2024.

Still, other programming languages could implement the zing network utility. The author knows there are smarter, more resourceful, clever persons out there, and would welcome a port of the zing network utility to other languages.

To support that end, porting the zing network utility to another programming language, the author will describe and explain the major elements of the zing network utility.

5.4 Implement Your Own

For an implementation of the zing network utility in your favorite programming language, there are several features or functionality needed. These features are:

- Timing functionality is wall clock timing, not system timing.
- Network functionality for access via TCP/IP sockets.
- Network functionality for addressing via DNS, and IP addresses.
- Data structures functionality to store, and retrieve timing information.

The zing network utility has been implemented by the author in bash, a shell scripting language, and also in Java, a portable runtime high-level language, among other programming languages.

The core functionality that is required is network functionality to access a network or the Internet and to address query addresses. The zing network utility is a network utility. Thus network functionality is imperative to implement the zing network utility in your favorite language.

There is a set of functions that provide the functional operation for the zing network utility. These functions are:

- CLI arguments function to access the options at the CLI by the user and then use these parameters to configure the zing operation.

- Function to check by DNS for the host, IP address, and type - this is to get the host address as an IP4 or IP6 form.

- Function to connect and disconnect with the time - this does the core zing operation and returns the amount of wall-clock time it took to zing the host on the network.

- Function to take the timetable of the zing operation timing, and compute the various statistics reported by the zing network utility.

- Time table - a data structure that holds the time for each cycle of the zing operation on a host.

- Internally three loops iterate over the count of cycles, the operation limit, and the ports given for the host system on the network. The three loops are:

- Loop for limit - perform a zing operation for an op limit and this is one zing cycle.

- Loop for the count - perform the zing operation for a count of cycles, and the time is stored in the timetable.

- Loop for port list - for each port, do a zing operation on the number of cycles on the host, store, and report the time.

- The last phase of the zing network utility is to take the timetable, compute the statistics, report them, and then exit.

Chapter 6. References

- [Campb 2024] Campbell, Scott. “Basics of UART Communication,” DIY Electronics, https://www.circuitbasics.com/basics-uart-communication/, 2024. Accessed March 17, 2024.

- [Gilrea 2022] Gilreath, William F. “Zing - the Zero Packet PING Network Utility,” GitHub repository, https://github.com/wgilreath/zing, 2022.

- [Gilre 2022] Gilreath, William F. “Zing Me,” *Linux Magazine*, December 2022, pp. 54-58. https://www.linux-magazine.com/Issues/2022/265/Zing

- [Gilre 2023] Gilreath, William F. “Have a Bash with Zinger,” *Admin Magazine*, October 2023, pp. 42-46.

- [Inter 2024] Internet Engineering Task Force. “RFC 9000, QUIC: A UDP-Based Multiplexed and Secure Transport,” IETF. doi:10.17487/RFC9000. RFC 9000. https://datatracker.ietf.org/doc/html/rfc9000, Accessed January 2024.

- [Molte 2022] Molteni, Daniele. "Landscape of API Traffic," 2022. https://blog.cloudflare.com/landscape-of-api-traffic/ Accessed February 12, 2024.

- [Muuss 2019] Muuss, Mike. “The Story of the PING Program,” https:// ftp.arl.army.mil/~mike/ping.html, Accessed August 14, 2022.

- [Poste 1981] Postel, J., “RFC 792: Internet Control Message Protocol,” RFC Editor, 1981. https://www.rfc-editor.org/rfc/rfc792, Accessed August 14, 2022.

- [Viere 1929] Viereck, George Sylvester. “What Life Mans to Einstein: An Interview,” *The Saturday Evening Post*, October 26, 1929, Column 1, p. 117.

- [Törne 1938] Von Törne, Bengt. “Sibelius: A Close-up,” Houghton Mifflin Company, Boston, United States, 1938, p. 27.

- [Wikip 2022a] Wikipedia, “IP address,” https://en.wikipedia.org/wiki/ Ping_(networking_utility), Accessed August 14, 2022.

- [Wikip 2022b] Wikipedia, “ping (networking utility),” https://en.wikipedia.org/wiki/IP_address, Accessed August 14, 2022.

- [Wikip 2024a] Wikipedia, “Ping Flood,” https://en.wikipedia.org/wiki/Ping_flood, Accessed February 12, 2024.

- [Wikip 2024b] Wikipedia, “Ping of Death,” https://en.wikipedia.org/wiki/Ping_of_death, Accessed February 12, 2024.

- [Wikip 2022c] Wikipedia, “Standard deviation,” https://en.wikipedia.org/ wiki/Standard_deviation, Accessed September 3, 2022.

Chapter 7. Manual Page

The manual page, or the “man” page from Linux or MacOS is given here. The man page for zing is:

zing(8) - Man Page

SYNOPSIS

zing [-h | [-4|-6] [-c count] [-op ops] [-p ports] [-t timeout]] host

DESCRIPTION

The zing utility uses a socket to connect on multiple ports of a remote host that have an IP address either IPv4 or IPv6. The default ports are HTTP port 80 and HTTPS port 443 on a remote host. The zing utility will connect multiple times, determine if the remote host is present, and the minimum, maximum, average, and standard deviation of the time to reach the host.

OPTIONS

-4 Use IPv4 addresses for the remote host.

-6 Use IPv6 addresses for the remote host.

-c The count of the number of ops or operations to the remote host.

-h Print help with the zing command-line interface parameters.

-op The number of operations or ops for an operation to remote host.

-p The list of ports to zing on the remote host.

-t Specify the timeout in milliseconds before zing exits.

EXAMPLES

zing -4 -c 4 -op 4 -p 80,443 -t 4000 google.com

EXIT STATUS

The zing utility exits with one of the following values:

0 The zing operation was successful with the specified host.

1 The zing operation was unable to find or reach the specified host.

SEE ALSO

netstat(1), **ifconfig(8)**, **routed(8)**, **traceroute(8)**, **ping(8)**

HISTORY

The zing utility appeared in the November 2022 in Linux Magazine.

AUTHOR

The original zing utility is written by William F. Gilreath (will@wfgilreath.xyz) in Java, ported to C# and Python.

BUGS

I should think so. No known bugs initially.

COPYRIGHT

Chapter 8. Administrivia

Lastly and finally, the administrative trivia, or “administrivia” about the author, contacting the author, the source code for the zing network utility, and the license for the book and source code of the zing network utility.

8.1 About the Author

William F. Gilreath (he/him/his) is a senior software development/machine learning engineer with over 30+ years of software development experience in the field.

The Author—Yours Truly

The author is a senior software development/machine learning engineer, computer scientist, mathematician, and writer. He is a writer of code, equations, poems, text, and a lover of cats.

More about the author is given at his home site:

```
https://www.wfgilreath.xyz
```

8.2 Contact

The author welcomes comments, feedback, and questions—although he may not be able to respond to all e-mails. Please feel free to reach out and contact him at the e-mail address of: `will@wfgilreath.xyz` by e-mail via the Internet.

While the author cannot guarantee a response to every e-mail message, he does read them and responds, where and when possible.

8.3 Source Code Access

The source code for the zing network utility is online, and publicly available on a GitHub repository ([Gilrea 2022] for access and download) located online at:

https://github.com/wgilreath/zing/src

8.4 Licenses

The license for the e-book and the source code for the zing network utility are given.

8.4.1 License for Book

CC BY-ND 4.0
Attribution-NoDerivatives 4.0 International

This license requires that users give credit to the creator. It allows users to copy and distribute the material in any medium or format in unadapted form only, even for commercial purposes.

BY: Credit must be given to you, the creator.

ND: No derivatives or adaptations of your work are permitted.

https://creativecommons.org/licenses/by-nd/4.0/

8.4.2 License for Software

The software license for the zing network utility is the GNU General Public License version 3 or GPL v3.

> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.

https://www.gnu.org/licenses/gpl-3.0.html

Glossary of Acronyms

- CLI — command-line interface
- CLR — Common Language Runtime
- DDoS — distributed denial of service
- DNS — Domain Name System
- HTTP — Hyper Text Transfer Protocol
- HTTPS — Hyper Text Transfer Protocol Secure
- ICMP — Internet Control Message Protocol
- IP — Internet Protocol
- JVM — Java Virtual Machine
- QUIC — Quick UDP Internet Connection
- TCP — Transmission Control Protocol
- TCP/IP — Transmission Control Protocol/Internet Protocol
- UART — Universal Asynchronous Receiver-Transmitter
- UDP — Universal Data Protocol

“There has never been a statue erected to honor a critic.”

Jean Sibelius (1865-1957)

The book covers the origin, history, implementation, and software architecture of the Zing network utility. The reader will explore utilizing, operating, and the functionality of the Zing network utility that are examined and discussed in depth.

The text serves as a guide for using the Zing network utility and as a source for future reference of details and specifics. By the end of the book, the reader will learn and discover how the Zing network utility can be a valuable tool in your toolbox or a weapon in your arsenal.

ISBN: 9798224428151