NAME

nping − Network packet generation tool / ping utility

SYNOPSIS

nping [*Options*] {*targets*}

DESCRIPTION

Nping is an open−source tool for network packet generation, response analysis and response time measurement. Nping allows users to generate network packets of a wide range of protocols, letting them tune virtually any field of the protocol headers. While Nping can be used as a simple ping utility to detect active hosts, it can also be used as a raw packet generator for network stack stress tests, ARP poisoning, Denial of Service attacks, route tracing, and other purposes.

Additionally, Nping offers a special mode of operation called the "Echo Mode", that lets users see how the generated probes change in transit, revealing the differences between the transmitted packets and the packets received at the other end. See section "Echo Mode" for details.

The output from Nping is a list of the packets that are being sent and received. The level of detail depends on the options used.

A typical Nping execution is shown in Example 1. The only Nping arguments used in this example are **−c**, to specify the number of times to target each host, **−−tcp** to specify TCP Probe Mode, **−p 80,433** to specify the target ports; and then the two target hostnames.

Example 1. A representative Nping execution

nping −c 1 −−tcp −p 80,433 scanme.nmap.org google.com

```
Starting Nping ( https://nmap.org/nping )
SENT (0.0120s) TCP 96.16.226.135:50091 > 64.13.134.52:80 S ttl=64 id=52072 iplen=40  seq=1077657388 win=1480
RCVD (0.1810s) TCP 64.13.134.52:80 > 96.16.226.135:50091 SA ttl=53 id=0 iplen=44  seq=4158134847 win=5840 <m
SENT (1.0140s) TCP 96.16.226.135:50091 > 74.125.45.100:80 S ttl=64 id=13932 iplen=40  seq=1077657388 win=1480
RCVD (1.1370s) TCP 74.125.45.100:80 > 96.16.226.135:50091 SA ttl=52 id=52913 iplen=44  seq=2650443864 win=57
SENT (2.0140s) TCP 96.16.226.135:50091 > 64.13.134.52:433 S ttl=64 id=8373 iplen=40  seq=1077657388 win=1480
SENT (3.0140s) TCP 96.16.226.135:50091 > 74.125.45.100:433 S ttl=64 id=23624 iplen=40  seq=1077657388 win=148

Statistics for host scanme.nmap.org (64.13.134.52):
|  Probes Sent: 2 | Rcvd: 1 | Lost: 1  (50.00%)
|_ Max rtt: 169.720ms | Min rtt: 169.720ms | Avg rtt: 169.720ms
Statistics for host google.com (74.125.45.100):
|  Probes Sent: 2 | Rcvd: 1 | Lost: 1  (50.00%)
|_ Max rtt: 122.686ms | Min rtt: 122.686ms | Avg rtt: 122.686ms
Raw packets sent: 4 (160B) | Rcvd: 2 (92B) | Lost: 2 (50.00%)
Tx time: 3.00296s | Tx bytes/s: 53.28 | Tx pkts/s: 1.33
Rx time: 3.00296s | Rx bytes/s: 30.64 | Rx pkts/s: 0.67
Nping done: 2 IP addresses pinged in 4.01 seconds
```

The newest version of Nping can be obtained with Nmap at **https://nmap.org**. The newest version of this man page is available at **https://nmap.org/book/nping-man.html**.

−−>
.SH "OPTIONS SUMMARY"

This options summary is printed when Nping is run with no arguments. It helps people remember the most common options, but is no substitute for the in−depth documentation in the rest of this manual. Some obscure options aren't even included here.

```
Nping 0.5.59BETA1 ( https://nmap.org/nping )
Usage: nping [Probe mode] [Options] {target specification}
```

TARGET SPECIFICATION:
Targets may be specified as hostnames, IP addresses, networks, etc.
Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0−255.1−254
PROBE MODES:
−−tcp−connect : Unprivileged TCP connect probe mode.
−−tcp : TCP probe mode.
−−udp : UDP probe mode.
−−icmp : ICMP probe mode.
−−arp : ARP/RARP probe mode.
−−tr, −−traceroute : Traceroute mode (can only be used with TCP/UDP/ICMP modes).
TCP CONNECT MODE:
−p, −−dest−port <port spec> : Set destination port(s).
−g, −−source−port <portnumber> : Try to use a custom source port.
TCP PROBE MODE:
−g, −−source−port <portnumber> : Set source port.
−p, −−dest−port <port spec> : Set destination port(s).
−−seq <seqnumber> : Set sequence number.
−−flags <flag list> : Set TCP flags (ACK,PSH,RST,SYN,FIN...)
−−ack <acknumber> : Set ACK number.
−−win <size> : Set window size.
−−badsum : Use a random invalid checksum.
UDP PROBE MODE:
−g, −−source−port <portnumber> : Set source port.
−p, −−dest−port <port spec> : Set destination port(s).
−−badsum : Use a random invalid checksum.
ICMP PROBE MODE:
−−icmp−type <type> : ICMP type.
−−icmp−code <code> : ICMP code.
−−icmp−id <id> : Set identifier.
−−icmp−seq <n> : Set sequence number.
−−icmp−redirect−addr <addr> : Set redirect address.
−−icmp−param−pointer <pnt> : Set parameter problem pointer.
−−icmp−advert−lifetime <time> : Set router advertisement lifetime.
−−icmp−advert−entry <IP,pref> : Add router advertisement entry.
−−icmp−orig−time <timestamp> : Set originate timestamp.
−−icmp−recv−time <timestamp> : Set receive timestamp.
−−icmp−trans−time <timestamp> : Set transmit timestamp.
ARP/RARP PROBE MODE:
−−arp−type <type> : Type: ARP, ARP−reply, RARP, RARP−reply.
−−arp−sender−mac <mac> : Set sender MAC address.
−−arp−sender−ip <addr> : Set sender IP address.
−−arp−target−mac <mac> : Set target MAC address.
−−arp−target−ip <addr> : Set target IP address.
IPv4 OPTIONS:
−S, −−source−ip : Set source IP address.
−−dest−ip <addr> : Set destination IP address (used as an alternative to {target specification}).
−−tos <tos> : Set type of service field (8bits).
−−id <id> : Set identification field (16 bits).
−−df : Set Don't Fragment flag.
−−mf : Set More Fragments flag.
−−ttl <hops> : Set time to live [0−255].
−−badsum−ip : Use a random invalid checksum.

```
  --ip-options <S|R [route]|L [route]|T|U ...> : Set IP options
  --ip-options <hex string>                    : Set IP options
  --mtu <size>                    : Set MTU. Packets get fragmented if MTU is
                                    small enough.
IPv6 OPTIONS:
  -6, --IPv6                      : Use IP version 6.
  --dest-ip                       : Set destination IP address (used as an
                                    alternative to {target specification}).
  --hop-limit                     : Set hop limit (same as IPv4 TTL).
  --traffic-class <class> :       : Set traffic class.
  --flow <label>                  : Set flow label.
ETHERNET OPTIONS:
  --dest-mac <mac>                : Set destination mac address. (Disables
                                    ARP resolution)
  --source-mac <mac>              : Set source MAC address.
  --ether-type <type>             : Set EtherType value.
PAYLOAD OPTIONS:
  --data <hex string>             : Include a custom payload.
  --data-string <text>            : Include a custom ASCII text.
  --data-length <len>             : Include len random bytes as payload.
ECHO CLIENT/SERVER:
  --echo-client <passphrase>      : Run Nping in client mode.
  --echo-server <passphrase>      : Run Nping in server mode.
  --echo-port <port>              : Use custom <port> to listen or connect.
  --no-crypto                     : Disable encryption and authentication.
  --once                          : Stop the server after one connection.
  --safe-payloads                 : Erase application data in echoed packets.
TIMING AND PERFORMANCE:
  Options which take <time> are in seconds, or append 'ms' (milliseconds),
  's' (seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m, 0.25h).
  --delay <time>                  : Adjust delay between probes.
  --rate  <rate>                  : Send num packets per second.
MISC:
  -h, --help                      : Display help information.
  -V, --version                   : Display current version number.
  -c, --count <n>                 : Stop after <n> rounds.
  -e, --interface <name>          : Use supplied network interface.
  -H, --hide-sent                 : Do not display sent packets.
  -N, --no-capture                : Do not try to capture replies.
  --privileged                    : Assume user is fully privileged.
  --unprivileged                  : Assume user lacks raw socket privileges.
  --send-eth                      : Send packets at the raw ethernet layer.
  --send-ip                       : Send packets using raw IP sockets.
  --bpf-filter <filter spec>      : Specify custom BPF filter.
OUTPUT:
  -v                              : Increment verbosity level by one.
  -v[level]                       : Set verbosity level. E.g: -v4
  -d                              : Increment debugging level by one.
  -d[level]                       : Set debugging level. E.g: -d3
  -q                              : Decrease verbosity level by one.
  -q[N]                           : Decrease verbosity level N times
  --quiet                         : Set verbosity and debug level to minimum.
  --debug                         : Set verbosity and debug to the max level.
EXAMPLES:
```

```
nping scanme.nmap.org
nping --tcp -p 80 --flags rst --ttl 2 192.168.1.1
nping --icmp --icmp-type time --delay 500ms 192.168.254.254
nping --echo-server "public" -e wlan0 -vvv
nping --echo-client "public" echo.nmap.org --tcp -p1-1024 --flags ack
```

SEE THE MAN PAGE FOR MANY MORE OPTIONS, DESCRIPTIONS, AND EXAMPLES

TARGET SPECIFICATION

Everything on the Nping command line that isn't an option or an option argument is treated as a target host specification. Nping uses the same syntax for target specifications that Nmap does. The simplest case is a single target given by IP address or hostname.

Nping supports CIDR-style addressing. You can append /*numbits* to an IPv4 address or hostname and Nping will send probes to every IP address for which the first *numbits* are the same as for the reference IP or hostname given. For example, 192.168.10.0/24 would send probes to the 256 hosts between 192.168.10.0 (binary: 11000000 10101000 00001010 00000000) and 192.168.10.255 (binary: 11000000 10101000 00001010 11111111), inclusive. 192.168.10.40/24 would ping exactly the same targets. Given that the host scanme.nmap.org is at the IP address 64.13.134.52, the specification scanme.nmap.org/16 would send probes to the 65,536 IP addresses between 64.13.0.0 and 64.13.255.255. The smallest allowed value is /0, which targets the whole Internet. The largest value is /32, which targets just the named host or IP address because all address bits are fixed.

CIDR notation is short but not always flexible enough. For example, you might want to send probes to 192.168.0.0/16 but skip any IPs ending with .0 or .255 because they may be used as subnet network and broadcast addresses. Nping supports this through octet range addressing. Rather than specify a normal IP address, you can specify a comma-separated list of numbers or ranges for each octet. For example, 192.168.0-255.1-254 will skip all addresses in the range that end in .0 or .255, and 192.168.3-5,7.1 will target the four addresses 192.168.3.1, 192.168.4.1, 192.168.5.1, and 192.168.7.1. Either side of a range may be omitted; the default values are 0 on the left and 255 on the right. Using - by itself is the same as 0-255, but remember to use 0- in the first octet so the target specification doesn't look like a command-line option. Ranges need not be limited to the final octets: the specifier 0-.-.13.37 will send probes to all IP addresses on the Internet ending in .13.37. This sort of broad sampling can be useful for Internet surveys and research.

IPv6 addresses can only be specified by their fully qualified IPv6 address or hostname. CIDR and octet ranges aren't supported for IPv6 because they are rarely useful.

Nping accepts multiple host specifications on the command line, and they don't need to be the same type. The command **nping scanme.nmap.org 192.168.0.0/8 10.0.0,1,3-7.-** does what you would expect.

OPTION SPECIFICATION

Nping is designed to be very flexible and fit a wide variety of needs. As with most command-line tools, its behavior can be adjusted using command-line options. These general principles apply to option arguments, unless stated otherwise.

Options that take integer numbers can accept values specified in decimal, octal or hexadecimal base. When a number starts with 0x, it will be treated as hexadecimal; when it simply starts with 0, it will be treated as octal. Otherwise, Nping will assume the number has been specified in base 10. Virtually all numbers that can be supplied from the command line are unsigned so, as a general rule, the minimum value is zero. Users may also specify the word random or rand to make Nping generate a random value within the expected range.

IP addresses may be given as IPv4 addresses (e.g. 192.168.1.1), IPv6 addresses (e.g. 2001:db8:85a3::8e4c:760:7146), or hostnames, which will be resolved using the default DNS server configured in the host system.

Options that take MAC addresses accept the usual colon-separated 6 hex byte format (e.g. 00:50:56:d4:01:98). Hyphens may also be used instead of colons (e.g. 00-50-56-c0-00-08). The special

word random or rand sets a random address and the word broadcast or bcast sets ff:ff:ff:ff:ff:ff.

GENERAL OPERATION

Unlike other ping and packet generation tools, Nping supports multiple target host and port specifications. While this provides great flexibility, it is not obvious how Nping handles situations where there is more than one host and/or more than one port to send probes to. This section explains how Nping behaves in these cases.

When multiple target hosts are specified, Nping rotates among them in round−robin fashion. This gives slow hosts more time to send their responses before another probe is sent to them. Ports are also scheduled using round robin. So, unless only one port is specified, Nping never sends two probes to the same target host and port consecutively.

The loop around targets is the “inner loop” and the loop around ports is the “outer loop”. All targets will be sent a probe for a given port before moving on to the next port. Between probes, Nping waits a configurable amount of time called the “inter−probe delay”, which is controlled by the **−−delay** option. These examples show how it works.

nping −−tcp −c 2 1.1.1.1 −p 100−102

```
Starting Nping ( https://nmap.org/nping )
SENT (0.0210s) TCP 192.168.1.77 > 1.1.1.1:100
SENT (1.0230s) TCP 192.168.1.77 > 1.1.1.1:101
SENT (2.0250s) TCP 192.168.1.77 > 1.1.1.1:102
SENT (3.0280s) TCP 192.168.1.77 > 1.1.1.1:100
SENT (4.0300s) TCP 192.168.1.77 > 1.1.1.1:101
SENT (5.0320s) TCP 192.168.1.77 > 1.1.1.1:102
```

nping −−tcp −c 2 1.1.1.1 2.2.2.2 3.3.3.3 −p 8080

```
Starting Nping ( https://nmap.org/nping )
SENT (0.0230s) TCP 192.168.0.21 > 1.1.1.1:8080
SENT (1.0240s) TCP 192.168.0.21 > 2.2.2.2:8080
SENT (2.0260s) TCP 192.168.0.21 > 3.3.3.3:8080
SENT (3.0270s) TCP 192.168.0.21 > 1.1.1.1:8080
SENT (4.0290s) TCP 192.168.0.21 > 2.2.2.2:8080
SENT (5.0310s) TCP 192.168.0.21 > 3.3.3.3:8080
```

nping −−tcp −c 1 −−delay 500ms 1.1.1.1 2.2.2.2 3.3.3.3 −p 137−139

```
Starting Nping ( https://nmap.org/nping )
SENT (0.0230s) TCP 192.168.0.21 > 1.1.1.1:137
SENT (0.5250s) TCP 192.168.0.21 > 2.2.2.2:137
SENT (1.0250s) TCP 192.168.0.21 > 3.3.3.3:137
SENT (1.5280s) TCP 192.168.0.21 > 1.1.1.1:138
SENT (2.0280s) TCP 192.168.0.21 > 2.2.2.2:138
SENT (2.5310s) TCP 192.168.0.21 > 3.3.3.3:138
SENT (3.0300s) TCP 192.168.0.21 > 1.1.1.1:139
SENT (3.5330s) TCP 192.168.0.21 > 2.2.2.2:139
SENT (4.0330s) TCP 192.168.0.21 > 3.3.3.3:139
```

PROBE MODES

Nping supports a wide variety of protocols. Although in some cases Nping can automatically determine the mode from the options used, it is generally a good idea to specify it explicitly.

−−tcp−connect (TCP Connect mode)

TCP connect mode is the default mode when a user does not have raw packet privileges. Instead of writing raw packets as most other modes do, Nping asks the underlying operating system to establish a connection with the target machine and port by issuing the connect system call. This is the same

high−level system call that web browsers, P2P clients, and most other network−enabled applications use to establish a connection. It is part of a programming interface known as the Berkeley Sockets API. Rather than read raw packet responses off the wire, Nping uses this API to obtain status information on each connection attempt. For this reason, you will not be able to see the contents of the packets that are sent or received but only status information about the TCP connection establishment taking place.

−−tcp (TCP mode)
TCP is the mode that lets users create and send any kind of TCP packet. TCP packets are sent embedded in IP packets that can also be tuned. This mode can be used for many different purposes. For example you could try to discover open ports by sending TCP SYN messages without completing the three−way handshake. This technique is often referred to as half−open scanning, because you don't open a full TCP connection. You send a SYN packet, as if you are going to open a real connection and then wait for a response. A SYN/ACK indicates the port is open, while a RST indicates it's closed. If no response is received one could assume that some intermediate network device is filtering the responses. Another use could be to see how a remote TCP/IP stack behaves when it receives a non−RFC−compliant packet, like one with both SYN and RST flags set. One could also do some evil by creating custom RST packets using an spoofed IP address with the intent of closing an active TCP connection.

−−udp (UDP mode)
UDP mode can have two different behaviours. Under normal circumstances, it lets users create custom IP/UDP packets. However, if Nping is run by a user without raw packet privileges and no changes to the default protocol headers are requested, then Nping enters the unprivileged UDP mode which basically sends UDP packets to the specified target hosts and ports using the sendto system call. Note that in this unprivileged mode it is not possible to see low−level header information of the packets on the wire but only status information about the amount of bytes that are being transmitted and received. UDP mode can be used to interact with any UDP−based server. Examples are DNS servers, streaming servers, online gaming servers, and port knocking/single−packet authorization daemons.

−−icmp (ICMP mode)
ICMP mode is the default mode when the user runs Nping with raw packet privileges. Any kind of ICMP message can be created. The default ICMP type is Echo, i.e., ping. ICMP mode can be used for many different purposes, from a simple request for a timestamp or a netmask to the transmission of fake destination unreachable messages, custom redirects, and router advertisements.

−−arp (ARP/RARP mode)
ARP lets you create and send a few different ARP−related packets. These include ARP, RARP, DRARP, and InARP requests and replies. This mode can ban be used to perform low−level host discovery, and conduct ARP−cache poisoning attacks.

−−traceroute (Traceroute mode)
Traceroute is not a mode by itself but a complement to TCP, UDP, and ICMP modes. When this option is specified Nping will set the IP TTL value of the first probe to 1. When the next router receives the packet it will drop it due to the expiration of the TTL and it will generate an ICMP destination unreachable message. The next probe will have a TTL of 2 so now the first router will forward the packet while the second router will be the one that drops the packet and generates the ICMP message. The third probe will have a TTL value of 3 and so on. By examining the source addresses of all those ICMP Destination Unreachable messages it is possible to determine the path that the probes take until they reach their final destination.

TCP CONNECT MODE

−p *port_spec*, **−−dest−port** *port_spec* (Target ports)
This option specifies which ports you want to try to connect to. It can be a single port, a comma−separated list of ports (e.g. 80,443,8080), a range (e.g. 1−1023), and any combination of those (e.g. 21−25,80,443,1024−2048). The beginning and/or end values of a range may be omitted, causing Nping to use 1 and 65535, respectively. So you can specify −p− to target ports from 1 through 65535. Using port zero is allowed if you specify it explicitly.

−g *portnumber*, **−−source−port** *portnumber* (Spoof source port)
This option asks Nping to use the specified port as source port for the TCP connections. Note that this might not work on all systems or may require root privileges. Specified value must be an integer in the range [0–65535].

TCP MODE

−p *port_spec*, **−−dest−port** *port_spec* (Target ports)
This option specifies which destination ports you want to send probes to. It can be a single port, a comma−separated list of ports (e.g. 80,443,8080), a range (e.g. 1−1023), and any combination of those (e.g. 21−25,80,443,1024−2048). The beginning and/or end values of a range may be omitted, causing Nping to use 1 and 65535, respectively. So you can specify −p− to target ports from 1 through 65535. Using port zero is allowed if you specify it explicitly.

−g *portnumber*, **−−source−port** *portnumber* (Spoof source port)
This option asks Nping to use the specified port as source port for the TCP connections. Note that this might not work on all systems or may require root privileges. Specified value must be an integer in the range [0–65535].

−−seq *seqnumber* (Sequence Number)
Specifies the TCP sequence number. In SYN packets this is the initial sequence number (ISN). In a normal transmission this corresponds to the sequence number of the first byte of data in the segment. *seqnumber* must be a number in the range [0–4294967295].

−−flags *flags* (TCP Flags)
This option specifies which flags should be set in the TCP packet. *flags* may be specified in three different ways:

1. As a comma−separated list of flags, e.g. **−−flags syn,ack,rst**
2. As a list of one−character flag initials, e.g. **−−flags SAR** tells Nping to set flags SYN, ACK, and RST.
3. As an 8−bit hexadecimal number, where the supplied number is the exact value that will be placed in the flags field of the TCP header. The number should start with the prefix 0x and should be in the range [0x00–0xFF], e.g. −−flags 0x20 sets the URG flag as 0x20 corresponds to binary 00100000 and the URG flag is represented by the third bit.

There are 8 possible flags to set: CWR, ECN, URG, ACK, PSH, RST, SYN, and FIN. The special value ALL means to set all flags. NONE means to set no flags. It is important that if you don't want any flag to be set, you request it explicitly because in some cases the SYN flag may be set by default. Here is a brief description of the meaning of each flag:

CWR (Congestion Window Reduced)
Set by an ECN−Capable sender when it reduces its congestion window (due to a retransmit timeout, a fast retransmit or in response to an ECN notification.

ECN (Explicit Congestion Notification)
During the three−way handshake it indicates that sender is capable of performing explicit congestion notification. Normally it means that a packet with the IP Congestion Experienced flag set was received during normal transmission. See RFC 3168 for more information.

URG (Urgent)
Segment is urgent and the urgent pointer field carries valid information.

ACK (Acknowledgement)
The segment carries an acknowledgement and the value of the acknowledgement number field is valid and contains the next sequence number that is expected from the receiver.

PSH (Push)
The data in this segment should be immediately pushed to the application layer on arrival.

RST (Reset)

There was some problem and the sender wants to abort the connection.

SYN (Synchronize)
The segment is a request to synchronize sequence numbers and establish a connection. The sequence number field contains the sender's initial sequence number.

FIN (Finish)
The sender wants to close the connection.

−−win *size* (Window Size)
Specifies the TCP window size, this is, the number of octets the sender of the segment is willing to accept from the receiver at one time. This is usually the size of the reception buffer that the OS allocates for a given connection. *size* must be a number in the range [0–65535].

−−badsum (Invalid Checksum)
Asks Nping to use an invalid TCP checksum for the packets sent to target hosts. Since virtually all host IP stacks properly drop these packets, any responses received are likely coming from a firewall or an IDS that didn't bother to verify the checksum. For more details on this technique, see **https://nmap.org/p60-12.html**.

UDP MODE

−p *port_spec*, **−−dest−port** *port_spec* (Target ports)
This option specifies which ports you want UDP datagrams to be sent to. It can be a single port, a comma−separated list of ports (e.g. 80,443,8080), a range (e.g. 1−1023), and any combination of those (e.g. 21−25,80,443,1024−2048). The beginning and/or end values of a range may be omitted, causing Nping to use 1 and 65535, respectively. So you can specify −p− to target ports from 1 through 65535. Using port zero is allowed if you specify it explicitly.

−g *portnumber*, **−−source−port** *portnumber* (Spoof source port)
This option asks Nping to use the specified port as source port for the transmitted datagrams. Note that this might not work on all systems or may require root privileges. Specified value must be an integer in the range [0–65535].

−−badsum (Invalid Checksum)
Asks Nping to use an invalid UDP checksum for the packets sent to target hosts. Since virtually all host IP stacks properly drop these packets, any responses received are likely coming from a firewall or an IDS that didn't bother to verify the checksum. For more details on this technique, see **https://nmap.org/p60-12.html**.

ICMP MODE

−−icmp−type *type* (ICMP type)
This option specifies which type of ICMP messages should be generated. *type* can be supplied in two different ways. You can use the **official type numbers assigned by IANA**[1] (e.g. **−−icmp−type 8** for ICMP Echo Request), or you can use any of the mnemonics listed in the section called “ICMP Types”.

−−icmp−code *code* (ICMP code)
This option specifies which ICMP code should be included in the generated ICMP messages. *code* can be supplied in two different ways. You can use the **official code numbers assigned by IANA**[1] (e.g. **−−icmp−code 1** for Fragment Reassembly Time Exceeded), or you can use any of the mnemonics listed in the section called “ICMP Codes”.

−−icmp−id *id* (ICMP identifier)
This option specifies the value of the identifier used in some of the ICMP messages. In general it is used to match request and reply messages. *id* must be a number in the range [0–65535].

−−icmp−seq *seq* (ICMP sequence)
This option specifies the value of the sequence number field used in some ICMP messages. In general it is used to match request and reply messages. *id* must be a number in the range [0–65535].

−−icmp−redirect−addr *addr* (ICMP Redirect address)
This option sets the address field in ICMP Redirect messages. In other words, it sets the IP address of the router that should be used when sending IP datagrams to the original destination. *addr* can be

either an IPv4 address or a hostname.

−−icmp−param−pointer *pointer* (ICMP Parameter Problem pointer)
This option specifies the pointer that indicates the location of the problem in ICMP Parameter Problem messages. *pointer* should be a number in the range [0–255]. Normally this option is only used when ICMP code is set to 0 ("Pointer indicates the error").

−−icmp−advert−lifetime *ttl* (ICMP Router Advertisement Lifetime)
This option specifies the router advertisement lifetime, this is, the number of seconds the information carried in an ICMP Router Advertisement can be considered valid for. *ttl* must be a positive integer in the range [0–65535].

−−icmp−advert−entry *addr,pref* (ICMP Router Advertisement Entry)
This option adds a Router Advertisement entry to an ICMP Router Advertisement message. The parameter must be two values separated by a comma. *addr* is the router's IP and can be specified either as an IP address in dot−decimal notation or as a hostname. *pref* is the preference level for the specified IP. It must be a number in the range [0–4294967295]. An example is **−−icmp−advert−entry 192.168.128.1,3**.

−−icmp−orig−time *timestamp* (ICMP Originate Timestamp)
This option sets the Originate Timestamp in ICMP Timestamp messages. The Originate Timestamp is expressed as the number of milliseconds since midnight UTC and it corresponds to the time the sender last touched the Timestamp message before its transmission. *timestamp* can be specified as a regular time (e.g. 10s, 3h, 1000ms), or the special string now. You can add or subtract values from now, for example **−−icmp−orig−time now−2s**, **−−icmp−orig−time now+1h**, **−−icmp−orig−time now+200ms**.

−−icmp−recv−time *timestamp* (ICMP Receive Timestamp)
This option sets the Receive Timestamp in ICMP Timestamp messages. The Receive Timestamp is expressed as the number of milliseconds since midnight UTC and it corresponds to the time the echoer first touched the Timestamp message on receipt. *timestamp* is as with **−−icmp−orig−time**.

−−icmp−trans−time *timestamp* (ICMP Transmit Timestamp)
This option sets the Transmit Timestamp in ICMP Timestamp messages. The Transmit Timestamp is expressed as the number of milliseconds since midnight UTC and it corresponds to the time the echoer last touched the Timestamp message before its transmission. *timestamp* is as with **−−icmp−orig−time**.

ICMP Types

These identifiers may be used as mnemonics for the ICMP type numbers given to the **−−icmp−type** option. In general there are three forms of each identifier: the full name (e.g. destination−unreachable), the short name (e.g. dest−unr), or the initials (e.g. du). In ICMP types that request something, the word "request" is omitted.

echo−reply, echo−rep, er
Echo Reply (type 0). This message is sent in response to an Echo Request message.

destination−unreachable, dest−unr, du
Destination Unreachable (type 3). This message indicates that a datagram could not be delivered to its destination.

source−quench, sour−que, sq
Source Quench (type 4). This message is used by a congested IP device to tell other device that is sending packets too fast and that it should slow down.

redirect, redi, r
Redirect (type 5). This message is normally used by routers to inform a host that there is a better route to use for sending datagrams. See also the **−−icmp−redirect−addr** option.

echo−request, echo, e
Echo Request (type 8). This message is used to test the connectivity of another device on a network.

router−advertisement, rout−adv, ra
: Router Advertisement (type 9). This message is used by routers to let hosts know of their existence and capabilities. See also the **−−icmp−advert−lifetime** option.

router−solicitation, rout−sol, rs
: Router Solicitation (type 10). This message is used by hosts to request Router Advertisement messages from any listening routers.

time−exceeded, time−exc, te
: Time Exceeded (type 11). This message is generated by some intermediate device (normally a router) to indicate that a datagram has been discarded before reaching its destination because the IP TTL expired.

parameter−problem, member−pro, pp
: Parameter Problem (type 12). This message is used when a device finds a problem with a parameter in an IP header and it cannot continue processing it. See also the **−−icmp−param−pointer** option.

timestamp, time, tm
: Timestamp Request (type 13). This message is used to request a device to send a timestamp value for propagation time calculation and clock synchronization. See also the **−−icmp−orig−time**, **−−icmp−recv−time**, and **−−icmp−trans−time**.

timestamp−reply, time−rep, tr
: Timestamp Reply (type 14). This message is sent in response to a Timestamp Request message.

information, info, i
: Information Request (type 15). This message is now obsolete but it was originally used to request configuration information from another device.

information−reply, info−rep, ir
: Information Reply (type 16). This message is now obsolete but it was originally sent in response to an Information Request message to provide configuration information.

mask−request, mask, m
: Address Mask Request (type 17). This message is used to ask a device to send its subnet mask.

mask−reply, mask−rep, mr
: Address Mask Reply (type 18). This message contains a subnet mask and is sent in response to a Address Mask Request message.

traceroute, trace, tc
: Traceroute (type 30). This message is normally sent by an intermediate device when it receives an IP datagram with a traceroute option. ICMP Traceroute messages are still experimental, see RFC 1393 for more information.

ICMP Codes

These identifiers may be used as mnemonics for the ICMP code numbers given to the **−−icmp−code** option. They are listed by the ICMP type they correspond to.

Destination Unreachable

network−unreachable, netw−unr, net
: Code 0. Datagram could not be delivered to its destination network (probably due to some routing problem).

host−unreachable, host−unr, host
: Code 1. Datagram was delivered to the destination network but it was impossible to reach the specified host (probably due to some routing problem).

protocol−unreachable, prot−unr, proto
: Code 2. The protocol specified in the Protocol field of the IP datagram is not supported by the host to which the datagram was delivered.

port−unreachable, port−unr, port

Code 3. The TCP/UDP destination port was invalid.

needs−fragmentation, need−fra, frag
: Code 4. Datagram had the DF bit set but it was too large for the MTU of the next physical network so it had to be dropped.

source−route−failed, sour−rou, routefail
: Code 5. IP datagram had a Source Route option but a router couldn't pass it to the next hop.

network−unknown, netw−unk, net?
: Code 6. Destination network is unknown. This code is never used. Instead, Network Unreachable is used.

host−unknown, host−unk, host?
: Code 7. Specified host is unknown. Usually generated by a router local to the destination host to inform of a bad address.

host−isolated, host−iso, isolated
: Code 8. Source Host Isolated. Not used.

network−prohibited, netw−pro, !net
: Code 9. Communication with destination network is administratively prohibited (source device is not allowed to send packets to the destination network).

host−prohibited, host−pro, !host
: Code 10. Communication with destination host is administratively prohibited. (The source device is allowed to send packets to the destination network but not to the destination device.)

network−tos, unreachable−network−tos, netw−tos, tosnet
: Code 11. Destination network unreachable because it cannot provide the type of service specified in the IP TOS field.

host−tos, unreachable−host−tos, toshost
: Code 12. Destination host unreachable because it cannot provide the type of service specified in the IP TOS field.

communication−prohibited, comm−pro, !comm
: Code 13. Datagram could not be forwarded due to filtering that blocks the message based on its contents.

host−precedence−violation, precedence−violation, prec−vio, violation
: Code 14. Precedence value in the IP TOS field is not permitted.

precedence−cutoff, prec−cut, cutoff
: Code 15. Precedence value in the IP TOS field is lower than the minimum allowed for the network.

Redirect

redirect−network, redi−net, net
: Code 0. Redirect all future datagrams with the same destination network as the original datagram, to the router specified in the Address field. The use of this code is prohibited by RFC 1812.

redirect−host, redi−host, host
: Code 1. Redirect all future datagrams with the same destination host as the original datagram, to the router specified in the Address field.

redirect−network−tos, redi−ntos, redir−ntos
: Code 2. Redirect all future datagrams with the same destination network and IP TOS value as the original datagram, to the router specified in the Address field. The use of this code is prohibited by RFC 1812.

redirect−host−tos, redi−htos, redir−htos
: Code 3. Redirect all future datagrams with the same destination host and IP TOS value as the

original datagram, to the router specified in the Address field.

Router Advertisement

normal−advertisement, norm−adv, normal, zero, default, def
: Code 0. Normal router advertisement. In Mobile IP: Mobility agent can act as a router for IP datagrams not related to mobile nodes.

not−route−common−traffic, not−rou, mobile−ip, !route, !commontraffic
: Code 16. Used for Mobile IP. The mobility agent does not route common traffic. All foreign agents must forward to a default router any datagrams received from a registered mobile node

Time Exceeded

ttl−exceeded−in−transit, ttl−exc, ttl−transit
: Code 0. IP Time To Live expired during transit.

fragment−reassembly−time−exceeded, frag−exc, frag−time
: Code 1. Fragment reassembly time has been exceeded.

Parameter Problem

pointer−indicates−error, poin−ind, pointer
: Code 0. The pointer field indicates the location of the problem. See the **−−icmp−param−pointer** option.

missing−required−option, miss−option, option−missing
: Code 1. IP datagram was expected to have an option that is not present.

bad−length, bad−len, badlen
: Code 2. The length of the IP datagram is incorrect.

ARP MODE

−−arp−type *type* (ICMP Type)
: This option specifies which type of ARP messages should be generated. *type* can be supplied in two different ways. You can use the **official numbers assigned by IANA**[2] (e.g. **−−arp−type 1** for ARP Request), or you can use one of the mnemonics from the section called “ARP Types”.

−−arp−sender−mac *mac* (Sender MAC address)
: This option sets the Sender Hardware Address field of the ARP header. Although ARP supports many types of link layer addresses, currently Nping only supports MAC addresses. *mac* must be specified using the traditional MAC notation (e.g. 00:0a:8a:32:f4:ae). You can also use hyphens as separators (e.g. 00−0a−8a−32−f4−ae).

−−arp−sender−ip *addr* (Sender IP address)
: This option sets the Sender IP field of the ARP header. *addr* can be given as an IPv4 address or a hostname.

−−arp−target−mac *mac* (target MAC address)
: This option sets the Target Hardware Address field of the ARP header.

−−arp−target−ip *addr* (target ip address)
: This option sets the Target IP field of the ARP header.

ARP Types

These identifiers may be used as mnemonics for the ARP type numbers given to the **−−arp−type** option.

arp−request, arp, a
: ARP Request (type 1). ARP requests are used to translate network layer addresses (normally IP addresses) to link layer addresses (usually MAC addresses). Basically, and ARP request is a broadcasted message that asks the host in the same network segment that has a given IP address to provide its MAC address.

arp−reply, arp−rep, ar
: ARP Reply (type 2). An ARP reply is a message that a host sends in response to an ARP request to

provide its link layer address.

rarp−request, rarp, r
: RARP Requests (type 3). RARP requests are used to translate a link layer address (normally a MAC address) to a network layer address (usually an IP address). Basically a RARP request is a broadcasted message sent by a host that wants to know his own IP address because it doesn't have any. It was the first protocol designed to solve the bootstrapping problem. However, RARP is now obsolete and DHCP is used instead. For more information about RARP see RFC 903.

rarp−reply, rarp−rep, rr
: RARP Reply (type 4). A RARP reply is a message sent in response to a RARP request to provide an IP address to the host that sent the RARP request in the first place.

drarp−request, drarp, d
: Dynamic RARP Request (type 5). Dynamic RARP is an extension to RARP used to obtain or assign a network layer address from a fixed link layer address. DRARP was used mainly in Sun Microsystems platforms in the late 90's but now it's no longer used. See RFC 1931 for more information.

drarp−reply, drarp−rep, dr
: Dynamic RARP Reply (type 6). A DRARP reply is a message sent in response to a RARP request to provide network layer address.

drarp−error, drarp−err, de
: DRARP Error (type 7). DRARP Error messages are usually sent in response to DRARP requests to inform of some error. In DRARP Error messages, the Target Protocol Address field is used to carry an error code (usually in the first byte). The error code is intended to tell why no target protocol address is being returned. For more information see RFC 1931.

inarp−request, inarp, i
: Inverse ARP Request (type 8). InARP requests are used to translate a link layer address to a network layer address. It is similar to RARP request but in this case, the sender of the InARP request wants to know the network layer address of another node, not its own address. InARP is mainly used in Frame Relay and ATM networks. For more information see RFC 2390.

inarp−reply, inarp−rep, ir
: Inverse ARP Reply (type 9). InARP reply messages are sent in response to InARP requests to provide the network layer address associated with the host that has a given link layer address.

arp−nak, an
: ARP NAK (type 10). ARP NAK messages are an extension to the ATMARP protocol and they are used to improve the robustness of the ATMARP server mechanism. With ARP NAK, a client can determine the difference between a catastrophic server failure and an ATMARP table lookup failure. See RFC 1577 for more information.

IPV4 OPTIONS

−S *addr*, **−−source−ip** *addr* (Source IP Address)
: Sets the source IP address. This option lets you specify a custom IP address to be used as source IP address in sent packets. This allows spoofing the sender of the packets. *addr* can be an IPv4 address or a hostname.

−−dest−ip *addr* (Destination IP Address)
: Adds a target to Nping's target list. This option is provided for consistency but its use is deprecated in favor of plain target specifications. See the section called “TARGET SPECIFICATION”.

−−tos *tos* (Type of Service)
: Sets the IP TOS field. The TOS field is used to carry information to provide quality of service features. It is normally used to support a technique called Differentiated Services. See RFC 2474 for more information. *tos* must be a number in the range [0–255].

−−id *id* (Identification)
: Sets the IPv4 Identification field. The Identification field is a 16−bit value that is common to all fragments belonging to a particular message. The value is used by the receiver to reassemble the

original message from the fragments received. *id* must be a number in the range [0–65535].

−−df (Don't Fragment)
Sets the Don't Fragment bit in sent packets. When an IP datagram has its DF flag set, intermediate devices are not allowed to fragment it so if it needs to travel across a network with a MTU smaller that datagram length the datagram will have to be dropped. Normally an ICMP Destination Unreachable message is generated and sent back to the sender.

−−mf (More Fragments)
Sets the More Fragments bit in sent packets. The MF flag is set to indicate the receiver that the current datagram is a fragment of some larger datagram. When set to zero it indicates that the current datagram is either the last fragment in the set or that it is the only fragment.

−−ttl *hops* (Time To Live)
Sets the IPv4 Time−To−Live (TTL) field in sent packets to the given value. The TTL field specifies how long the datagram is allowed to exist on the network. It was originally intended to represent a number of seconds but it actually represents the number of hops a packet can traverse before being dropped. The TTL tries to avoid a situation in which undeliverable datagrams keep being forwarded from one router to another endlessly. *hops* must be a number in the range [0–255].

−−badsum−ip (Invalid IP checksum)
Asks Nping to use an invalid IP checksum for packets sent to target hosts. Note that some systems (like most Linux kernels), may fix the checksum before placing the packet on the wire, so even if Nping shows the incorrect checksum in its output, the packets may be transparently corrected by the kernel.

−−ip−options *S|R [route]|L [route]|T|U ...*, **−−ip−options** *hex string* (IP Options)
The IP protocol offers several options which may be placed in packet headers. Unlike the ubiquitous TCP options, IP options are rarely seen due to practicality and security concerns. In fact, many Internet routers block the most dangerous options such as source routing. Yet options can still be useful in some cases for determining and manipulating the network route to target machines. For example, you may be able to use the record route option to determine a path to a target even when more traditional traceroute−style approaches fail. Or if your packets are being dropped by a certain firewall, you may be able to specify a different route with the strict or loose source routing options.

The most powerful way to specify IP options is to simply pass in hexadecimal data as the argument to **−−ip−options**. Precede each hex byte value with \x. You may repeat certain characters by following them with an asterisk and then the number of times you wish them to repeat. For example, \x01\x07\x04\x00*4 is the same as \x01\x07\x04\x00\x00\x00\x00.

Note that if you specify a number of bytes that is not a multiple of four, an incorrect IP header length will be set in the IP packet. The reason for this is that the IP header length field can only express multiples of four. In those cases, the length is computed by dividing the header length by 4 and rounding down. This will affect the way the header that follows the IP header is interpreted, showing bogus information in Nping or in the output of any sniffer. Although this kind of situation might be useful for some stack stress tests, users would normally want to specify explicit padding, so the correct header length is set.

Nping also offers a shortcut mechanism for specifying options. Simply pass the letter R, T, or U to request record−route, record−timestamp, or both options together, respectively. Loose or strict source routing may be specified with an L or S followed by a space and then a space−separated list of IP addresses.

For more information and examples of using IP options with Nping, see the mailing list post at **http://seclists.org/nmap-dev/2006/q3/0052.html**.

−−mtu *size* (Maximum Transmission Unit)
This option sets a fictional MTU in Nping so IP datagrams larger than *size* are fragmented before

transmission. *size* must be specified in bytes and corresponds to the number of octets that can be carried on a single link−layer frame.

IPV6 OPTIONS

−6, **−−ipv6** (Use IPv6)

Tells Nping to use IP version 6 instead of the default IPv4. It is generally a good idea to specify this option as early as possible in the command line so Nping can parse it soon and know in advance that the rest of the parameters refer to IPv6. The command syntax is the same as usual except that you also add the **−6** option. Of course, you must use IPv6 syntax if you specify an address rather than a hostname. An address might look like **3ffe:7501:4819:2000:210:f3ff:fe03:14d0**, so hostnames are recommended.

While IPv6 hasn't exactly taken the world by storm, it gets significant use in some (usually Asian) countries and most modern operating systems support it. To use Nping with IPv6, both the source and target of your packets must be configured for IPv6. If your ISP (like most of them) does not allocate IPv6 addresses to you, free tunnel brokers are widely available and work fine with Nping. You can use the free IPv6 tunnel broker service at **http://www.tunnelbroker.net**.

Please note that IPv6 support is still highly experimental and many modes and options may not work with it.

−S *addr*, **−−source−ip** *addr* (Source IP Address)

Sets the source IP address. This option lets you specify a custom IP address to be used as source IP address in sent packets. This allows spoofing the sender of the packets. *addr* can be an IPv6 address or a hostname.

−−dest−ip *addr* (Destination IP Address)

Adds a target to Nping's target list. This option is provided for consistency but its use is deprecated in favor of plain target specifications. See the section called “TARGET SPECIFICATION”.

−−flow *label* (Flow Label)

Sets the IPv6 Flow Label. The Flow Label field is 20 bits long and is intended to provide certain quality−of−service properties for real−time datagram delivery. However, it has not been widely adopted, and not all routers or endpoints support it. Check RFC 2460 for more information. *label* must be an integer in the range [0–1048575].

−−traffic−class *class* (Traffic Class)

Sets the IPv6 Traffic Class. This field is similar to the TOS field in IPv4, and is intended to provide the Differentiated Services method, enabling scalable service discrimination in the Internet without the need for per−flow state and signaling at every hop. Check RFC 2474 for more information. *class* must be an integer in the range [0–255].

−−hop−limit *hops* (Hop Limit)

Sets the IPv6 Hop Limit field in sent packets to the given value. The Hop Limit field specifies how long the datagram is allowed to exist on the network. It represents the number of hops a packet can traverse before being dropped. As with the TTL in IPv4, IPv6 Hop Limit tries to avoid a situation in which undeliverable datagrams keep being forwarded from one router to another endlessly. *hops* must be a number in the range [0–255].

ETHERNET OPTIONS

In most cases Nping sends packets at the raw IP level. This means that Nping creates its own IP packets and transmits them through a raw socket. However, in some cases it may be necessary to send packets at the raw Ethernet level. This happens, for example, when Nping is run under Windows (as Microsoft has disabled raw socket support since Windows XP SP2), or when Nping is asked to send ARP packets. Since in some cases it is necessary to construct ethernet frames, Nping offers some options to manipulate the different fields.

−−dest−mac *mac* (Ethernet Destination MAC Address)

This option sets the destination MAC address that should be set in outgoing Ethernet frames. This is useful in case Nping can't determine the next hop's MAC address or when you want to route probes through a router other than the configured default gateway. The MAC address should have the usual format of six colon−separated bytes, e.g. 00:50:56:d4:01:98. Alternatively, hyphens may be used instead of colons. Use the word random or rand to generate a random address, and broadcast or bcast to use ff:ff:ff:ff:ff:ff. If you set up a bogus destination MAC address your probes may not reach the intended targets.

−−**source−mac** *mac* (Ethernet Source MAC Address)
This option sets the source MAC address that should be set in outgoing Ethernet frames. This is useful in case Nping can't determine your network interface MAC address or when you want to inject traffic into the network while hiding your network card's real address. The syntax is the same as for −−dest−mac. If you set up a bogus source MAC address you may not receive probe replies.

−−**ether−type** *type* (Ethertype)
This option sets the Ethertype field of the ethernet frame. The Ethertype is used to indicate which protocol is encapsulated in the payload. *type* can be supplied in two different ways. You can use the **official numbers listed by the IEEE**[3] (e.g. −−**ether−type 0x0800** for IP version 4), or one of the mnemonics from the section called “Ethernet Types”.

Ethernet Types
These identifiers may be used as mnemonics for the Ethertype numbers given to the −−**arp−type** option.

ipv4, ip, 4
Internet Protocol version 4 (type 0x0800).

ipv6, 6
Internet Protocol version 6 (type 0x86DD).

arp
Address Resolution Protocol (type 0x0806).

rarp
Reverse Address Resolution Protocol (type 0x8035).

frame−relay, frelay, fr
Frame Relay (type 0x0808).

ppp
Point−to−Point Protocol (type 0x880B).

gsmp
General Switch Management Protocol (type 0x880C).

mpls
Multiprotocol Label Switching (type 0x8847).

mps−ual, mps
Multiprotocol Label Switching with Upstream−assigned Label (type 0x8848).

mcap
Multicast Channel Allocation Protocol (type 0x8861).

pppoe−discovery, pppoe−d
PPP over Ethernet Discovery Stage (type 0x8863).

pppoe−session, pppoe−s
PPP over Ethernet Session Stage (type 0x8864).

ctag
Customer VLAN Tag Type (type 0x8100).

epon
Ethernet Passive Optical Network (type 0x8808).

pbnac
Port−based network access control (type 0x888E).

stag
Service VLAN tag identifier (type 0x88A8).

ethexp1
Local Experimental Ethertype 1 (type 0x88B5).

ethexp2
Local Experimental Ethertype 2 (type 0x88B6).

ethoui
OUI Extended Ethertype (type 0x88B7).

preauth
Pre−Authentication (type 0x88C7).

lldp
Link Layer Discovery Protocol (type 0x88CC).

mac−security, mac−sec, macsec
Media Access Control Security (type 0x88E5).

mvrp
Multiple VLAN Registration Protocol (type 0x88F5).

mmrp
Multiple Multicast Registration Protocol (type 0x88F6).

frrr
Fast Roaming Remote Request (type 0x890D).

PAYLOAD OPTIONS

−−data *hex string* (Append custom binary data to sent packets)
This option lets you include binary data as payload in sent packets. *hex string* may be specified in any of the following formats: 0xAABBCCDDEEFF..., AABBCCDDEEFF... or \xAA\xBB\xCC\xDD\xEE\xFF.... Examples of use are **−−data 0xdeadbeef** and **−−data \xCA\xFE\x09**. Note that if you specify a number like 0x00ff no byte−order conversion is performed. Make sure you specify the information in the byte order expected by the receiver.

−−data−string *string* (Append custom string to sent packets)
This option lets you include a regular string as payload in sent packets. *string* can contain any string. However, note that some characters may depend on your system's locale and the receiver may not see the same information. Also, make sure you enclose the string in double quotes and escape any special characters from the shell. Example: **−−data−string "Jimmy Jazz..."**.

−−data−length *len* (Append random data to sent packets)
This option lets you include *len* random bytes of data as payload in sent packets. *len* must be an integer in the range [0–65400]. However, values higher than 1400 are not recommended because it may not be possible to transmit packets due to network MTU limitations.

ECHO MODE

The "Echo Mode" is a novel technique implemented by Nping which lets users see how network packets change in transit, from the host where they originated to the target machine. Basically, the Echo mode turns Nping into two different pieces: the Echo server and the Echo client. The Echo server is a network service that has the ability to capture packets from the network and send a copy ("echo them") to the originating client through a side TCP channel. The Echo client is the part that generates such network packets, transmits them to the server, and receives their echoed version through a side TCP channel that it has previously established with the Echo server.

This scheme lets the client see the differences between the packets that it sends and what is actually received by the server. By having the server send back copies of the received packets through the side

channel, things like NAT devices become immediately apparent to the client because it notices the changes in the source IP address (and maybe even source port). Other devices like those that perform traffic shaping, changing TCP window sizes or adding TCP options transparently between hosts, turn up too.

The Echo mode is also useful for troubleshooting routing and firewall issues. Among other things, it can be used to determine if the traffic generated by the Nping client is being dropped in transit and never gets to its destination or if the responses are the ones that don't get back to it.

Internally, client and server communicate over an encrypted and authenticated channel, using the Nping Echo Protocol (NEP), whose technical specification can be found in **https://nmap.org/svn/nping/docs/EchoProtoRFC.txt**

The following paragraphs describe the different options available in Nping's Echo mode.

−−ec *passphrase*, **−−echo−client** *passphrase* (Run Echo client)
This option tells Nping to run as an Echo client. *passphrase* is a sequence of ASCII characters that is used used to generate the cryptographic keys needed for encryption and authentication in a given session. The passphrase should be a secret that is also known by the server, and it may contain any number of printable ASCII characters. Passphrases that contain whitespace or special characters must be enclosed in double quotes.

When running Nping as an Echo client, most options from the regular raw probe modes apply. The client may be configured to send specific probes using flags like **−−tcp**, **−−icmp** or **−−udp**. Protocol header fields may be manipulated normally using the appropriate options (e.g. **−−ttl**, **−−seq**, **−−icmp−type**, etc.). The only exceptions are ARP−related flags, which are not supported in Echo mode, as protocols like ARP are closely related to the data link layer and its probes can't pass through different network segments.

−−es *passphrase*, **−−echo−server** *passphrase* (Run Echo server)
This option tells Nping to run as an Echo server. *passphrase* is a sequence of ASCII characters that is used used to generate the cryptographic keys needed for encryption and authentication in a given session. The passphrase should be a secret that is also known by the clients, and it may contain any number of printable ASCII characters. Passphrases that contain whitespace or special characters must be enclosed in double quotes. Note that although it is not recommended, it is possible to use empty passphrases, supplying **−−echo−server** "". However, if what you want is to set up an open Echo server, it is better to use option **−−no−crypto**. See below for details.

−−ep *port*, **−−echo−port** *port* (Set Echo TCP port number)
This option asks Nping to use the specified TCP port number for the Echo side channel connection. If this option is used with **−−echo−server**, it specifies the port on which the server listens for connections. If it is used with **−−echo−client**, it specifies the port to connect to on the remote host. By default, port number 9929 is used.

−−nc, **−−no−crypto** (Disable encryption and authentication)
This option asks Nping not to use any cryptographic operations during an Echo session. In practical terms, this means that the Echo side channel session data will be transmitted in the clear, and no authentication will be performed by the server or client during the session establishment phase. When **−−no−crypto** is used, the passphrase supplied with **−−echo−server** or **−−echo−client** is ignored.

This option must be specified if Nping was compiled without openSSL support. Note that, for technical reasons, a passphrase still needs to be supplied after the −−echo−client or −−echo−server flags, even though it will be ignored.

The −−no−crypto flag might be useful when setting up a public Echo server, because it allows users to connect to the Echo server without the need for any passphrase or shared secret. However, it is strongly recommended to not use −−no−crypto unless absolutely necessary. Public Echo servers should be configured to use the passphrase "public" or the empty passphrase (−−echo−server "") as the use of cryptography does not only provide confidentiality and authentication but also message integrity.

−−once (Serve one client and quit)
This option asks the Echo server to quit after serving one client. This is useful when only a single Echo session wants to be established as it eliminates the need to access the remote host to shutdown the server.

−−safe−payloads (Zero application data before echoing a packet)
This option asks the Echo server to erase any application layer data found in client packets before echoing them. When the option is enabled, the Echo server parses the packets received from Echo clients and tries to determine if they contain data beyond the transport layer. If such data is found, it is overwritten with zeroes before transmitting the packets to the appropriate Echo client.

Echo servers can handle multiple simultaneous clients running multiple echo sessions in parallel. In order to determine which packet needs to be echoed to which client and through which session, the Echo server uses an heuristic algorithm. Although we have taken every security measure that we could think of to prevent that a client receives an echoed packet that it did not generate, there is always a risk that our algorithm makes a mistake and delivers a packet to the wrong client. The −−safe−payloads option is useful for public echo servers or critical deployments where that kind of mistake cannot be afforded.

The following examples illustrate how Nping's Echo mode can be used to discover intermediate devices.

Example 2. Discovering NAT devices

```
# nping --echo-client "public" echo.nmap.org --udp

Starting Nping ( https://nmap.org/nping )
SENT (1.0970s) UDP 10.1.20.128:53 > 178.79.165.17:40125 ttl=64 id=32523 iplen=28
CAPT (1.1270s) UDP 80.38.10.21:45657 > 178.79.165.17:40125 ttl=54 id=32523 iplen=28
RCVD (1.1570s) ICMP 178.79.165.17 > 10.1.20.128 Port unreachable (type=3/code=3) ttl=49 id=16619 iplen=56
[...]
SENT (5.1020s) UDP 10.1.20.128:53 > 178.79.165.17:40125 ttl=64 id=32523 iplen=28
CAPT (5.1335s) UDP 80.38.10.21:45657 > 178.79.165.17:40125 ttl=54 id=32523 iplen=28
RCVD (5.1600s) ICMP 178.79.165.17 > 10.1.20.128 Port unreachable (type=3/code=3) ttl=49 id=16623 iplen=56

Max rtt: 60.628ms | Min rtt: 58.378ms | Avg rtt: 59.389ms
Raw packets sent: 5 (140B) | Rcvd: 5 (280B) | Lost: 0 (0.00%)| Echoed: 5 (140B)
Tx time: 4.00459s | Tx bytes/s: 34.96 | Tx pkts/s: 1.25
Rx time: 5.00629s | Rx bytes/s: 55.93 | Rx pkts/s: 1.00
Nping done: 1 IP address pinged in 6.18 seconds
```

The output clearly shows the presence of a NAT device in the client's local network. Note how the captured packet (CAPT) differs from the SENT packet: the source address for the original packets is in the reserved 10.0.0.0/8 range, while the address seen by the server is 80.38.10.21, the Internet side address of the NAT device. The source port was also modified by the device. The line starting with RCVD corresponds to the responses generated by the TCP/IP stack of the machine where the Echo server is run.

Example 3. Discovering a transparent proxy

```
# nping --echo-client "public" echo.nmap.org --tcp -p80

Starting Nping ( https://nmap.org/nping )
SENT (1.2160s) TCP 10.0.1.77:41659 > 178.79.165.17:80 S ttl=64 id=3317 iplen=40  seq=567704200 win=1480
RCVD (1.2180s) TCP 178.79.165.17:80 > 10.0.1.77:41659 SA ttl=128 id=13177 iplen=44  seq=3647106954 win=163
SENT (2.2150s) TCP 10.0.1.77:41659 > 178.79.165.17:80 S ttl=64 id=3317 iplen=40  seq=567704200 win=1480
SENT (3.2180s) TCP 10.0.1.77:41659 > 178.79.165.17:80 S ttl=64 id=3317 iplen=40  seq=567704200 win=1480
SENT (4.2190s) TCP 10.0.1.77:41659 > 178.79.165.17:80 S ttl=64 id=3317 iplen=40  seq=567704200 win=1480
```

```
SENT (5.2200s) TCP 10.0.1.77:41659 > 178.79.165.17:80 S ttl=64 id=3317 iplen=40  seq=567704200 win=1480

Max rtt: 2.062ms | Min rtt: 2.062ms | Avg rtt: 2.062ms
Raw packets sent: 5 (200B) | Rcvd: 1 (46B) | Lost: 4 (80.00%)| Echoed: 0 (0B)
Tx time: 4.00504s | Tx bytes/s: 49.94 | Tx pkts/s: 1.25
Rx time: 5.00618s | Rx bytes/s: 9.19 | Rx pkts/s: 0.20
Nping done: 1 IP address pinged in 6.39 seconds
```

In this example, the output is a bit more tricky. The absence of error messages shows that the Echo client has successfully established an Echo session with the server. However, no CAPT packets can be seen in the output. This means that none of the transmitted packets reached the server. Interestingly, a TCP SYN−ACK packet was received in response to the first TCP−SYN packet (and also, it is known that the target host does not have port 80 open). This behavior reveals the presence of a transparent web proxy cache server (which in this case is an old MS ISA server).

TIMING AND PERFORMANCE OPTIONS

−−delay *time* (Delay between probes)

This option lets you control for how long will Nping wait before sending the next probe. Like in many other ping tools, the default delay is one second. *time* must be a positive integer or floating point number. By default it is specified in seconds, however you can give an explicit unit by appending ms for milliseconds, s for seconds, m for minutes, or h for hours (e.g. 2.5s, 45m, 2h).

−−rate *rate* (Send probes at a given rate)

This option specifies the number of probes that Nping should send per second. This option and **−−delay** are inverses; **−−rate 20** is the same as **−−delay 0.05**. If both options are used, only the last one in the parameter list counts.

MISCELLANEOUS OPTIONS

−h, **−−help** (Display help)

Displays help information and exits.

−V, **−−version** (Display version)

Displays the program's version number and quits.

−c *rounds*, **−−count** *rounds* (Stop after a given number of rounds)

This option lets you specify the number of times that Nping should loop over target hosts (and in some cases target ports). Nping calls these “rounds”. In a basic execution with only one target (and only one target port in TCP/UDP modes), the number of rounds matches the number of probes sent to the target host. However, in more complex executions where Nping is run against multiple targets and multiple ports, the number of rounds is the number of times that Nping sends a complete set of probes that covers all target IPs and all target ports. For example, if Nping is asked to send TCP SYN packets to hosts 192.168.1.0−255 and ports 80 and 433, then $256 \times 2 = 512$ packets are sent in one round. So if you specify **−c 100**, Nping will loop over the different target hosts and ports 100 times, sending a total of $256 \times 2 \times 100 = 51200$ packets. By default Nping runs for 5 rounds. If a value of 0 is specified, Nping will run continuously.

−e *name*, **−−interface** *name* (Set the network interface to be used)

This option tells Nping what interface should be used to send and receive packets. Nping should be able to detect this automatically, but it will tell you if it cannot. *name* must be the name of an existing network interface with an assigned IP address.

−−privileged (Assume that the user is fully privileged)

Tells Nping to simply assume that it is privileged enough to perform raw socket sends, packet sniffing, and similar operations that usually require special privileges. By default Nping quits if such operations are requested by a user that has no root or administrator privileges. This option may be useful on Linux, BSD or similar systems that can be configured to allow unprivileged users to perform raw−packet transmissions. The **NPING_PRIVILEGED** environment variable may be set as an alternative to using **−−privileged**.

−−unprivileged (Assume that the user lacks raw socket privileges)
This option is the opposite of **−−privileged**. It tells Nping to treat the user as lacking network raw socket and sniffing privileges. This is useful for testing, debugging, or when the raw network functionality of your operating system is somehow broken. The **NPING_UNPRIVILEGED** environment variable may be set as an alternative to using **−−unprivileged**.

−−send−eth (Use raw ethernet sending)
Asks Nping to send packets at the raw ethernet (data link) layer rather than the higher IP (network) layer. By default, Nping chooses the one which is generally best for the platform it is running on. Raw sockets (IP layer) are generally most efficient for Unix machines, while ethernet frames are required for Windows operation since Microsoft disabled raw socket support. Nping still uses raw IP packets despite this option when there is no other choice (such as non−ethernet connections).

−−send−ip (Send at raw IP level)
Asks Nping to send packets via raw IP sockets rather than sending lower level ethernet frames. It is the complement to the **−−send−eth** option.

−−bpf−filter *filter spec* **−−filter** *filter spec* (Set custom BPF filter)
This option lets you use a custom BPF filter. By default Nping chooses a filter that is intended to capture most common responses to the particular probes that are sent. For example, when sending TCP packets, the filter is set to capture packets whose destination port matches the probe's source port or ICMP error messages that may be generated by the target or any intermediate device as a result of the probe. If for some reason you expect strange packets in response to sent probes or you just want to sniff a particular kind of traffic, you can specify a custom filter using the BPF syntax used by tools like tcpdump. See the documentation at**http://www .tcpdump.org/** for more information.

−H, **−−hide−sent** (Do not display sent packets)
This option tells Nping not to print information about sent packets. This can be useful when using very short inter−probe delays (i.e., when flooding), because printing information to the standard output has a computational cost and disabling it can probably speed things up a bit. Also, it may be useful when using Nping to detect active hosts or open ports (e.g. sending probes to all TCP ports in a /24 subnet). In that case, users may not want to see thousands of sent probes but just the replies generated by active hosts.

−N, **−−no−capture** (Do not attempt to capture replies)
This option tells Nping to skip packet capture. This means that packets in response to sent probes will not be processed or displayed. This can be useful when doing flooding and network stack stress tests. Note that when this option is specified, most of the statistics shown at the end of the execution will be useless. This option does not work with TCP Connect mode.

OUTPUT OPTIONS

−v[*level***]**, **−−verbose [***level***]** (Increase or set verbosity level)
Increases the verbosity level, causing Nping to print more information during its execution. There are 9 levels of verbosity (−4 to 4). Every instance of **−v** increments the verbosity level by one (from its default value, level 0). Every instance of option **−q** decrements the verbosity level by one. Alternatively you can specify the level directly, as in **−v3** or **−v−1**. These are the available levels:

Level −4
No output at all. In some circumstances you may not want Nping to produce any output (like when one of your work mates is watching over your shoulder). In that case level −4 can be useful because although you won't see any response packets, probes will still be sent.

Level −3
Like level −4 but displays fatal error messages so you can actually see if Nping is running or it failed due to some error.

Level −2
Like level −3 but also displays warnings and recoverable errors.

Level −1

Displays traditional run−time information (version, start time, statistics, etc.) but does not display sent or received packets.

Level 0
: This is the default verbosity level. It behaves like level −1 but also displays sent and received packets and some other important information.

Level 1
: Like level 0 but it displays detailed information about timing, flags, protocol details, etc.

Level 2
: Like level 1 but displays very detailed information about sent and received packets and other interesting information.

Level 3
: Like level 2 but also displays the raw hexadecimal dump of sent and received packets.

Level 4 and higher
: Same as level 3.

−q[*level***]**, **−−reduce−verbosity [***level***]** (Decrease verbosity level)
: Decreases the verbosity level, causing Nping to print less information during its execution.

−d[*level***]** (Increase or set debugging level)
: When even verbose mode doesn't provide sufficient data for you, debugging is available to flood you with much more! As with the **−v**, debugging is enabled with a command−line flag **−d** and the debug level can be increased by specifying it multiple times. There are 7 debugging levels (0 to 6). Every instance of **−d** increments debugging level by one. Provide an argument to **−d** to set the level directly; for example **−d4**.

Debugging output is useful when you suspect a bug in Nping, or if you are simply confused as to what Nping is doing and why. As this feature is mostly intended for developers, debug lines aren't always self−explanatory. You may get something like

NSOCK (1.0000s) Callback: TIMER SUCCESS for EID 12; tcpconnect_event_handler(): Received callback of type

If you don't understand a line, your only recourses are to ignore it, look it up in the source code, or request help from the development list (nmap−dev). Some lines are self−explanatory, but the messages become more obscure as the debug level is increased. These are the available levels:

Level 0
: Level 0. No debug information at all. This is the default level.

Level 1
: In this level, only very important or high−level debug information will be printed.

Level 2
: Like level 1 but also displays important or medium−level debug information

Level 3
: Like level 2 but also displays regular and low−level debug information.

Level 4
: Like level 3 but also displays messages only a real Nping freak would want to see.

Level 5
: Like level 4 but it enables basic debug information related to external libraries like Nsock.

Level 6
: Like level 5 but it enables full, very detailed, debug information related to external libraries like Nsock.

BUGS

Like its authors, Nping isn't perfect. But you can help make it better by sending bug reports or even writing patches. If Nping doesn't behave the way you expect, first upgrade to the latest version available from **https://nmap.org**. If the problem persists, do some research to determine whether it has already been discovered and addressed. Try searching for the problem or error message on Google since that aggregates so many forums. If nothing comes of this, create an Issue on our tracker (**http://issues.nmap.org**) and/or mail a bug report to <dev@nmap.org>. If you subscribe to the nmap−dev list before posting, your message will bypass moderation and get through more quickly. Subscribe at **https://nmap.org/mailman/listinfo/dev**. Please include everything you have learned about the problem, as well as what version of Nping you are using and what operating system version it is running on. Other suggestions for improving Nping may be sent to the Nmap dev mailing list as well.

If you are able to write a patch improving Nping or fixing a bug, that is even better! Instructions for submitting patches or git pull requests are available from **https://github.com/nmap/nmap/blob/master/CONTRIBUTING.md**

Particularly sensitive issues such as a security reports may be sent directly to Fyodor directly at <fyodor@nmap.org>. All other reports and comments should use the dev list or issue tracker instead because more people read, follow, and respond to those.

AUTHORS

Luis MartinGarcia <luis.mgarc@gmail.com> (**http://www.luismg.com**)

Fyodor <fyodor@nmap.org> (**http://insecure.org**)

NOTES

1. official type numbers assigned by IANA
   http://www.iana.org/assignments/icmp-parameters
2. official numbers assigned by IANA
   http://www.iana.org/assignments/arp-parameters/
3. official numbers listed by the IEEE
   http://standards.ieee.org/regauth/ethertype/eth.txt