

Ping

Opis programu:

Służy do badania połączenia między hostami testującym i testowanym (między naszym komputerem a testowanym serwerem). Umożliwia sprawdzenie istnienia połączenia między hostami, liczbę przebytych w tym czasie węzłów, opóźnienia w transmisji oraz liczbę zgubionych pakietów.

Przykładowe wywołanie programu *ping* dla strony „cs.pwr.edu.pl”:

```
dominik@dominik-Nitro-AN515-55:~$ ping cs.pwr.edu.pl
PING cs.pwr.edu.pl (156.17.7.22) 56(84) bytes of data.
64 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=1 ttl=53 time=46.7 ms
64 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=2 ttl=53 time=50.4 ms
64 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=3 ttl=53 time=51.8 ms
64 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=4 ttl=53 time=53.2 ms
^C
--- cs.pwr.edu.pl ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 46.720/50.509/53.184/2.406 ms
dominik@dominik-Nitro-AN515-55:~$
```

Rozmiar wysłanego pakietu

Adres IP testowanego serwera

„Który z kolei wysłany pakiet”

Pozostały czas życia pakietu

Czas po którym wrócił do nas wcześniej wysłany sygnał

Przydatne flagi, które wykorzystamy do wykonania zadanych testów:

- **-t [ilość]** – ustalenie TTL wysyłanych pakietów (ilość węzłów po których chcemy przejść)
- **-c [ilość]** – ustalenie liczby pakietów do wysłania
- **-s [ilość]** – ustalenie wielkości pakietów
- **-M do** – bez fragmentacji

Zadania do wykonania:

1. **Sprawdź za jego pomocą ile jest węzłów na trasie do (i od) wybranego, odległego geograficznie, serwera.**

Wykorzystując flagę **-t** podczas uruchamiania programu *ping* możemy ustalić **tty (time to live)** wysyłanych pakietów, czyli **ilość węzłów** po których pakiety przejdą, próbując dostać się do docelowego serwera. Przy przejściach pakietów przez węzły wartość **tty** ulega dekrementacji. Na tej podstawie jesteśmy w stanie ustalić ilość węzłów na trasie do wybranego serwera.

```
dominik@dominik-Nitro-AN515-55:~$ ping -t 19 -c 3 www.apn.net.au
PING www.apn.net.au (202.171.100.139) 56(84) bytes of data.
From 202.171.100.5 (202.171.100.5) icmp_seq=1 Time to live exceeded
From 202.171.100.5 (202.171.100.5) icmp_seq=2 Time to live exceeded
From 202.171.100.5 (202.171.100.5) icmp_seq=3 Time to live exceeded

--- www.apn.net.au ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2003ms
```

Jak widzimy na powyższym zrzucie ekranu, po przypisaniu pakietom 19 **tty** wyświetla się komunikat „Time to live exceeded”, oraz „100% packet loss”. Oznacza to, że pakiety nie są w stanie dotrzeć do podanego adresu przechodząc tylko przez 19 węzłów. W takich przypadkach pakiety zostają skasowane przez ostatni węzeł (router) do którego udało im się dotrzeć (na którym **tty** = 0). Celem tego jest ograniczenie niepotrzebnego ruchu w sieci.

Gdy zwiększymy początkowe **tty** do 20, pakiety z powodzeniem dotrą do docelowego serwera. Świadczy o tym komunikat „... 3 received, 0% packet loss, ...”, oraz wyświetlenie przez program danych takich jak *icmp_seq*, *ttl*, *time*.

Wynika z tego, że na trasie do wybranego serwera znajduje się 20 węzłów.

```
dominik@dominik-Nitro-AN515-55:~$ ping -t 20 -c 3 www.apn.net.au
PING www.apn.net.au (202.171.100.139) 56(84) bytes of data.
64 bytes from www.apn.net.au (202.171.100.139): icmp_seq=1 ttl=47 time=514 ms
64 bytes from www.apn.net.au (202.171.100.139): icmp_seq=2 ttl=47 time=440 ms
64 bytes from www.apn.net.au (202.171.100.139): icmp_seq=3 ttl=47 time=360 ms

--- www.apn.net.au ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 360.496/438.167/513.546/62.503 ms
```

Chcąc obliczyć liczbę węzłów przebytych przez pakiety w drodze powrotnej należy odjąć wartość **ttl** od najbliższej (większej od **ttl**) z popularnych wartości startowych (32, 64, 128, 255).

W naszym przypadku ilość węzłów przebytych przez pakiet w drodze powrotnej będzie równa:

$$64 - 47 = 17$$

Uwaga:

Możemy zaobserwować, że ilość węzłów na trasie (komputer → domena) może różnić się od ilości przebytych w drodze powrotnej tj. (domena → komputer).

- 2. Zbadaj jaki wpływ ma na to wielkość pakietu. Zbadaj jak wielkość pakietu wpływa na obserwowane czasy propagacji. Wyniki sprawdź dla różnych adresów DNS. Jeśli potrzeba, sporządź wykres.**

Przykładowy test dla strony www.jeju-utd.com

```
dominik@dominik-Nitro-AN515-55:~$ ping -s 19992 -c 10 www.jeju-utd.com
PING www.jeju-utd.com (169.56.111.164) 19992(20020) bytes of data.
20000 bytes from a4.6f.38a9.ip4.static.sl-reverse.com (169.56.111.164): icmp_seq=2 ttl=112 time=363 ms
20000 bytes from a4.6f.38a9.ip4.static.sl-reverse.com (169.56.111.164): icmp_seq=3 ttl=112 time=389 ms
20000 bytes from a4.6f.38a9.ip4.static.sl-reverse.com (169.56.111.164): icmp_seq=4 ttl=112 time=839 ms
20000 bytes from a4.6f.38a9.ip4.static.sl-reverse.com (169.56.111.164): icmp_seq=5 ttl=112 time=458 ms
20000 bytes from a4.6f.38a9.ip4.static.sl-reverse.com (169.56.111.164): icmp_seq=6 ttl=112 time=466 ms
20000 bytes from a4.6f.38a9.ip4.static.sl-reverse.com (169.56.111.164): icmp_seq=7 ttl=112 time=377 ms
20000 bytes from a4.6f.38a9.ip4.static.sl-reverse.com (169.56.111.164): icmp_seq=9 ttl=112 time=434 ms
20000 bytes from a4.6f.38a9.ip4.static.sl-reverse.com (169.56.111.164): icmp_seq=10 ttl=112 time=459 ms

--- www.jeju-utd.com ping statistics ---
10 packets transmitted, 8 received, 20% packet loss, time 9012ms
rtt min/avg/max/mdev = 363.004/473.170/839.301/143.388 ms
dominik@dominik-Nitro-AN515-55:~$
```

Tabela 1: Wartość **tty** w zależności od wielkości pakietów

Lokalizacja	Wielkość pakietu ->	32 B	64 B	1000 B	20 000 B
	Adres				
Polska	cs.pwr.edu.pl	53	53	53	53
Włochy	www.gazzetta.it	54	54	54	54
Australia	www.sydney.edu.au	56	56	56	56
Korea Płd.	www.jeju-utd.com	112	112	112	112

Wniosek:

Wielkość pakietów nie wpływa na ilość przebytych węzłów.

Maksymalna możliwa wielkość wysyłanego pakietu dla programu *ping*

Tabela 2: Średni czas propagacji w zależności od wielkości pakietów

Wielkość pakietu ->	32 B	64 B	1000 B	20 000 B	65 515 B
Adres					
cs.pwr.edu.pl	55.425	54.526	55.834	85.332	-
www.gazzetta.it	37.660	58.919	31.768	-	-
www.sydney.edu.au	112.399	60.732	23.264	74.774	221.957
www.spiegel.de	100.980	71.974	99.113	127.530	334.081
www.pgenarodowy.pl	66.089	81.923	36.798	150.392	401.074
www.chicago.gov	22.054	24.434	38.621	125.204	266.338
www.jeju-utd.com	322.165	319.203	365.809	473.170	361.531

Wartości w tabeli to średni czas z 30 pomiarów.

Wnioski:

- **Zwykle czas propagacji rośnie wraz ze wzrostem rozmiaru pakietów**, jednak dla niektórych testowanych serwerów nie widzimy takiego zachowania.
- Wykonywałem kilka pomiarów dla tych samych argumentów i czasy potrafiły się znacząco od siebie różnić. (Jest dużo czynników, które wpływają na czas propagacji np. ruch w sieci)
- Do niektórych serwerów nie docierają pakiety o większej wielkości.

```
dominik@dominik-Nitro-AN515-55:~$ ping -s 19992 -c 30 www.gazzetta.it
PING na-eu-gazzetta.map.fastly.net (151.101.245.50) 19992(20020) bytes of data.

--- na-eu-gazzetta.map.fastly.net ping statistics ---
30 packets transmitted, 0 received, 100% packet loss, time 29706ms
```

3. Zbadaj jaki wpływ na powyższe ma konieczność fragmentacji pakietów. Jaki największy niefragmentowany pakiet uda się przesłać. Dlaczego i od czego to zależy?

Tabela 3: Fragmentacja a czas propagacji

Wielkość pakietu ->	256 B	256 B (bez fragmentacji)	1000 B	1000 B (bez fragmentacji)
Adres				
cs.pwr.edu.pl	112.093	128.943	170.677	149.887
www.pgenarodowy.pl	367.991	119.152	257.537	101.808
www.spiegel.de	153.586	104.241	247.268	109.640
www.gazzetta.it	325.411	218.010	136.243	386.722
www.sydney.edu.au	37.209	114.940	61.858	46.907
www.jeju-utd.com	961.513	941.085	965.220	968.908

Przykładowy test dla strony cs.pwr.edu.pl (bez fragmentacji, 30 pakietów, każdy o wielkości 256 bajtów)

```
dominik@dominik-Nitro-AN515-55:~$ ping -M do -c 30 -s 248 cs.pwr.edu.pl
PING cs.pwr.edu.pl (156.17.7.22) 248(276) bytes of data.
256 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=1 ttl=53 time=200 ms
256 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=2 ttl=53 time=67.8 ms
256 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=3 ttl=53 time=55.7 ms
256 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=4 ttl=53 time=216 ms
^C
--- cs.pwr.edu.pl ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 55.701/134.756/215.784/73.370 ms
dominik@dominik-Nitro-AN515-55:~$
```

Wnioski:

- **Z przeprowadzonych testów wynika, że fragmentacja nie ma wpływu na czas propagacji.** W niektórych przypadkach wartości czasu bez fragmentacji są większe od czasów z użytą fragmentacją, a w innych zupełnie na odwrót. Ponadto odpalając program dla jednakowych argumentów, wyniki testów potrafiły znacząco się od siebie różnić
- **1480 bajtów** było największym rozmiarem pakietu jaki udało mi się wysłać bez użycia fragmentacji. Nie jesteśmy w stanie wysłać tym sposobem większych pakietów, ponieważ ogranicza nas **MTU** (Maximum Transmission Unit). Jego wartość wynosi 1500 bajtów.
- Fragmentacja polega na podzieleniu wysyłanego pakietu na kilka części i wysłaniu ich osobno. Utrata nawet jednej części skutkuje koniecznością wysłania całego pakietu jeszcze raz.

Całkowity rozmiar wysłanego pakietu (datagramu) ← nagłówek + ładunek

Rozmiar pakietu, który możemy podać używając fagi -s, to wyłącznie rozmiar ładunku

```
dominik@dominik-Nitro-AN515-55:~$ ping -M do -c 1 -s 1472 cs.pwr.edu.pl
PING cs.pwr.edu.pl (156.17.7.22) 1472(1500) bytes of data.
1480 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=1 ttl=53 time=53.3 ms

--- cs.pwr.edu.pl ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 53.323/53.323/53.323/0.000 ms
dominik@dominik-Nitro-AN515-55:~$ ping -M do -c 1 -s 1473 cs.pwr.edu.pl
PING cs.pwr.edu.pl (156.17.7.22) 1473(1501) bytes of data.
ping: local error: message too long, mtu=1500

--- cs.pwr.edu.pl ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

Przy próbie wysłania pakietu bez fragmentacji o rozmiarze przekraczającym 1500 bajtów wyskakuje błąd z informacją o przekroczeniu MTU.

Uwaga:

- Nagłówek datagramu zawiera wszystkie niezbędne informacje do transportu pierwotnego zestawu do miejsca przeznaczenia. Na załączonych zrzutach ekranu możemy zauważyć, że składa się z 28 bajtów. Program ping używa pakietu ICMP.
- Ładunek datagramu to dane, które mają zostać przetransportowane.

4. Określ "średnicę" internetu (najdłuższą ścieżkę, którą uda się wyszukać).

Sprawdzenie długości węzła dla strony www.jeju-utd.com z serwerem w Korei Pld.

```
dominik@dominik-Nitro-AN515-55:~$ ping -c 1 -t 19 www.jeju-utd.com
PING www.jeju-utd.com (169.56.111.164) 56(84) bytes of data.
From 6a.48.38a9.ip4.static.sl-reverse.com (169.56.72.106) icmp_seq=1 Time to live exceeded

--- www.jeju-utd.com ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

dominik@dominik-Nitro-AN515-55:~$ ping -c 1 -t 20 www.jeju-utd.com
PING www.jeju-utd.com (169.56.111.164) 56(84) bytes of data.
64 bytes from a4.6f.38a9.ip4.static.sl-reverse.com (169.56.111.164): icmp_seq=1 ttl=112 time=742 ms

--- www.jeju-utd.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 742.489/742.489/742.489/0.000 ms
dominik@dominik-Nitro-AN515-55:~$
```

Sprawdzenie długości węzła dla strony dnc.org.nz z serwerem w Nowej Zelandii.

```
dominik@dominik-Nitro-AN515-55:~$ ping -c 1 -t 19 dnc.org.nz
PING dnc.org.nz (104.21.54.171) 56(84) bytes of data.
64 bytes from 104.21.54.171 (104.21.54.171): icmp_seq=1 ttl=55 time=180 ms

--- dnc.org.nz ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 180.315/180.315/180.315/0.000 ms
```

W poszukiwaniu najdłuższego węzła próbowałem „pingować” serwery odległe geograficznie. W moim przypadku był to serwer w Korei Południowej, którego trasa składała się z 20 węzłów.

5. Czy potrafisz wyszukać trasy przebiegające przez sieci wirtualne (zdalne platformy "cloud computing"). Ile węzłów mają ścieżki w tym przypadku?

Żeby wyszukać trasy przebiegające przez sieci wirtualne należy „pingować” potencjalny serwer. Jeśli wartość TTL będzie ulegała zmianie mimo podawania tego samego adresu to jest możliwość że trasa serwera przebiega właśnie przez serwery wirtualne.

Wydaje mi się, że udało mi się znaleźć trasę prowadzącą przez wirtualne sieci. Wysłałem sygnały do chińskiej strony taobao.com i wartości ttl różniły się od siebie.


```

dominik@dominik-Nitro-AN515-55:~$ ping -c 3 taobao.com
PING taobao.com (140.205.220.96) 56(84) bytes of data.
64 bytes from 140.205.220.96 (140.205.220.96): icmp_seq=1 ttl=21 time=1054 ms
64 bytes from 140.205.220.96 (140.205.220.96): icmp_seq=2 ttl=21 time=403 ms
64 bytes from 140.205.220.96 (140.205.220.96): icmp_seq=3 ttl=21 time=788 ms

--- taobao.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 403.441/748.754/1054.468/267.251 ms, pipe 2
dominik@dominik-Nitro-AN515-55:~$ ping -c 3 -t 50 taobao.com
PING taobao.com (140.205.220.96) 56(84) bytes of data.
64 bytes from 140.205.220.96 (140.205.220.96): icmp_seq=1 ttl=7 time=1105 ms
64 bytes from 140.205.220.96 (140.205.220.96): icmp_seq=2 ttl=7 time=486 ms
64 bytes from 140.205.220.96 (140.205.220.96): icmp_seq=3 ttl=7 time=1545 ms

--- taobao.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 486.391/1045.367/1544.949/434.190 ms, pipe 2
dominik@dominik-Nitro-AN515-55:~$ ping -c 3 taobao.com
PING taobao.com (140.205.94.189) 56(84) bytes of data.
64 bytes from 140.205.94.189 (140.205.94.189): icmp_seq=2 ttl=22 time=285 ms
64 bytes from 140.205.94.189 (140.205.94.189): icmp_seq=3 ttl=22 time=921 ms

--- taobao.com ping statistics ---
3 packets transmitted, 2 received, 33,3333% packet loss, time 2005ms
rtt min/avg/max/mdev = 284.885/602.921/920.958/318.036 ms

```

Traceroute

Opis programu:

Za jego pomocą jesteśmy sprawdzić po jakiej trasie poruszają się pakiety aby dotrzeć do docelowego serwera.

Przykładowe użycie programu traceroute.

```

dominik@dominik-Nitro-AN515-55:~$ traceroute google.com
traceroute to google.com (172.217.16.14), 30 hops max, 60 byte packets
 1  gateway (192.168.0.1) 60.367 ms 60.424 ms 60.519 ms
 2  192.168.86.3 (192.168.86.3) 75.607 ms 110.620 ms 114.300 ms
 3  ws-zlot6-zlot4.siec.internetunion.pl (188.121.30.209) 156.790 ms ws-gaj01-zlot6.siec.internetunion.pl (188.121.30.221) 187.496 ms
 4  ws-gaj01-gorn46.siec.internetunion.pl (188.121.30.142) 109.221 ms 109.235 ms 109.396 ms
 5  142.250.166.32 (142.250.166.32) 187.710 ms ws-gaj01-gorn46.siec.internetunion.pl (188.121.30.142) 109.502 ms 109.521 ms
 6  108.170.250.209 (108.170.250.209) 230.387 ms 108.170.250.193 (108.170.250.193) 28.472 ms 142.250.166.32 (142.250.166.32) 45.418 ms
 7  216.239.40.213 (216.239.40.213) 16.979 ms 216.239.40.43 (216.239.40.43) 16.887 ms 17.592 ms
 8  216.239.40.213 (216.239.40.213) 23.462 ms 20.970 ms 23.397 ms
 9  mil02s06-in-f14.1e100.net (172.217.16.14) 18.357 ms 19.371 ms 16.678 ms
dominik@dominik-Nitro-AN515-55:~$

```

DNS (Domain Name System)

Używając strony <https://dnschecker.org/ip-location.php?ip=188.121.30.142> możemy po otrzymanych DNS sprawdzić lokalizację geograficzną serwerów znajdujących się na trasie.

IP2Location		IP2Location	
📍 IP: 188.121.30.142	🌐 Country: Poland	📍 IP: 172.217.16.14	🌐 Country: United States of America
🏠 State: Dolnoslaskie	🏠 City: Wroclaw	🏠 State: California	🏠 City: Mountain View
📍 Latitude: 51.0999	📍 Longitude: 17.0333	📍 Latitude: 37.4059	📍 Longitude: -122.0785
📶 ISP: Internet Union Spolka Akcyjna		📶 ISP: Google LLC	
IP Location Services by: IP2Location Updated: March 16 2022		IP Location Services by: IP2Location Updated: March 16 2022	

Z uruchomienia programu *tracerout* wynika, że długość trasy z mojego routera do docelowego serwera wynosi 9 węzłów. Możemy z ciekawości sprawdzić czy ta długość pokrywa się z danymi wyświetlanymi przez program *ping*.

Na zrzucie ekranu znajdującym się na następnej stronie dokumentu widzimy, że potrzebujemy co najmniej 9 węzłów, żeby dostać się do celu. Zatem oba programy dają jednoznaczny wynik.

```

dominik@dominik-Nitro-AN515-55:~$ ping -t 9 -c 3 google.com
PING google.com (172.217.16.14) 56(84) bytes of data.
64 bytes from mil02s06-in-f14.1e100.net (172.217.16.14): icmp_seq=1 ttl=119 time=9.62 ms
64 bytes from mil02s06-in-f14.1e100.net (172.217.16.14): icmp_seq=2 ttl=119 time=28.2 ms
64 bytes from mil02s06-in-f14.1e100.net (172.217.16.14): icmp_seq=3 ttl=119 time=9.47 ms

--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 9.468/15.766/28.214/8.802 ms
dominik@dominik-Nitro-AN515-55:~$ ping -t 8 -c 3 google.com
PING google.com (172.217.16.14) 56(84) bytes of data.
From 216.239.40.43 (216.239.40.43) icmp_seq=1 Time to live exceeded
From 216.239.40.43 (216.239.40.43) icmp_seq=2 Time to live exceeded
From 216.239.40.43 (216.239.40.43) icmp_seq=3 Time to live exceeded

--- google.com ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2002ms

```

Wireshark

Opis programu:

Jest to tzw. „sniffer”, czyli program służący do przechwytywania i zapisywania ruchu sieciowego. Rejestruje wszystkie pakiety wychodzące i przychodzące wraz z informacjami o nich (tj. czas, źródło, cel, protokół). Ponadto pozwala na użycie filtrów, dzięki czemu możemy przechwycić tylko te pakiety, które interesują nas w danym momencie.

Przykładowe wywołanie programu Wireshark

The screenshot shows the Wireshark interface with the following details:

- Filter Buttons Preferences...** dialog box is open, showing fields for **Opis:** (Enter a description for the filter button), **Filtr:** (Enter a filter expression to be applied), and **Comment:** (Enter a comment for the filter button). Buttons for **Anuluj** and **OK** are visible.
- Packet List:**

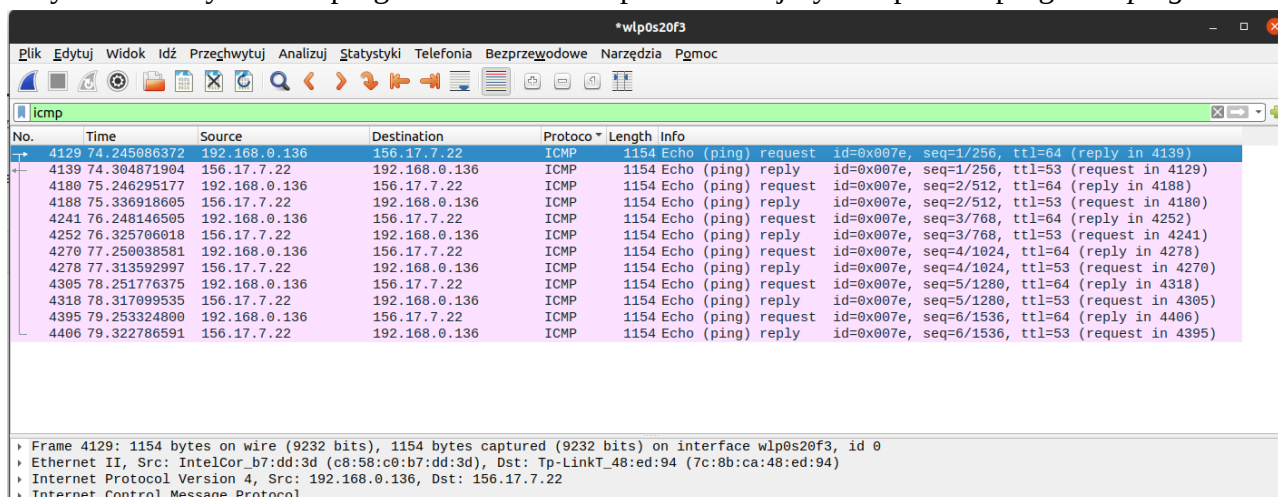
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	35.186.224.25	192.168.0.136	TCP	66	443 → 46100 [ACK] Seq=1 Ack=1 Win=1050 Len=0 TSval=4251913123
2	0.000448507	192.168.0.136	213.163.87.242	UDP	218	34975 → 50003 Len=176
3	0.010392988	35.186.224.25	192.168.0.136	TLSv1.2	153	Application Data
4	0.010404124	192.168.0.136	35.186.224.25	TCP	66	46100 → 443 [ACK] Seq=1 Ack=88 Win=2721 Len=0 TSval=402806701
5	0.010393085	35.186.224.25	192.168.0.136	TLSv1.2	550	Application Data
6	0.010417526	192.168.0.136	35.186.224.25	TCP	66	46100 → 443 [ACK] Seq=1 Ack=572 Win=2718 Len=0 TSval=402806701
7	0.010393105	35.186.224.25	192.168.0.136	TLSv1.2	339	Application Data
8	0.010420100	192.168.0.136	35.186.224.25	TCP	66	46100 → 443 [ACK] Seq=1 Ack=845 Win=2716 Len=0 TSval=402806701
9	0.010588086	192.168.0.136	213.163.87.242	UDP	231	34975 → 50003 Len=189
10	0.012539121	192.168.0.136	151.101.246.248	TLSv1.2	510	Application Data
11	0.012581460	213.163.87.242	192.168.0.136	RTCP	94	Receiver Report
12	0.015178767	35.186.224.25	192.168.0.136	TLSv1.2	105	Application Data
13	0.015186286	192.168.0.136	35.186.224.25	TCP	66	46100 → 443 [ACK] Seq=1 Ack=884 Win=2721 Len=0 TSval=402806701
14	0.015255156	192.168.0.136	35.186.224.25	TLSv1.2	105	Application Data
15	0.031936900	35.186.224.25	192.168.0.136	TCP	66	443 → 46100 [ACK] Seq=884 Ack=40 Win=1050 Len=0 TSval=4251913123
- Packet Details:**
 - Frame 2: 218 bytes on wire (1744 bits), 218 bytes captured (1744 bits) on interface wlp0s20f3, id 0
 - Ethernet II, Src: IntelCor_b7:dd:3d (c8:58:c0:b7:dd:3d), Dst: Tp-LinkT_48:ed:94 (7c:8b:ca:48:ed:94)
 - Internet Protocol Version 4, Src: 192.168.0.136, Dst: 213.163.87.242
 - User Datagram Protocol, Src Port: 34975, Dst Port: 50003
 - Data (176 bytes)
- Packet Bytes:**

```

0000  7c 8b ca 48 ed 94 c8 58 c0 b7 dd 3d 08 00 45 00  |...H...X...=.E.
0010  00 cc 9d 6f 40 00 40 11 ad eb c0 a8 00 88 d5 a3  |...o@.@...
0020  57 f2 88 9f c3 53 00 b8 ef 8f 90 78 71 d1 8a 13  |W...S...xq...
0030  fe ec 00 00 3a 58 be de 00 01 59 b5 ae ce 03 c1  |...:X...Y...
0040  db a2 c4 e0 b8 08 a8 70 03 9a ae 6d 28 57 dc c4  |...p...m(W...
0050  5b ee 1b 2b 18 c0 49 12 87 b5 fc c1 5d ab 48 21  |[...+...I...].H!
0060  b9 f5 a7 d7 27 8b 57 46 2a 5f c8 83 fd d1 a6 56  |...'WF*_...V
0070  e2 c0 52 34 74 a6 97 30 82 2e d1 25 23 a9 ac 71  |..R4t...0...%#...q
0080  22 4d 31 0c 79 a5 6c cb 47 a3 87 49 94 86 79 76  |"M1:y.l.G..I..yv
0090  c0 81 6a 71 f2 f0 a0 0b 84 8f b9 50 60 ae db 2f  |..jq....P`../
00a0  80 1a 91 c1 e2 16 f0 3d 3a d3 0b ff c6 df b2 a4  |..a...:=.....
00b0  12 b0 9f e2 be 22 ec 29 18 94 b6 cc 1b b7 b3 6b  |...".).....k

```

Przykładowe wywołanie programu Wireshark po wcześniejszym odpaleniu programu *ping*



The image shows a Wireshark packet capture window titled "wlp0s20f3". The filter bar shows "icmp". The packet list displays 12 packets, alternating between requests and replies. The packet details pane shows the structure of an ICMP Echo (ping) request and reply, including fields like ID, sequence number, and TTL.

No.	Time	Source	Destination	Protocol	Length	Info
4129	74.245086372	192.168.0.136	156.17.7.22	ICMP	1154	Echo (ping) request id=0x007e, seq=1/256, ttl=64 (reply in 4139)
4139	74.304871904	156.17.7.22	192.168.0.136	ICMP	1154	Echo (ping) reply id=0x007e, seq=1/256, ttl=53 (request in 4129)
4180	75.246295177	192.168.0.136	156.17.7.22	ICMP	1154	Echo (ping) request id=0x007e, seq=2/512, ttl=64 (reply in 4188)
4188	75.336918605	156.17.7.22	192.168.0.136	ICMP	1154	Echo (ping) reply id=0x007e, seq=2/512, ttl=53 (request in 4180)
4241	76.248146505	192.168.0.136	156.17.7.22	ICMP	1154	Echo (ping) request id=0x007e, seq=3/768, ttl=64 (reply in 4252)
4252	76.325706018	156.17.7.22	192.168.0.136	ICMP	1154	Echo (ping) reply id=0x007e, seq=3/768, ttl=53 (request in 4241)
4270	77.250038581	192.168.0.136	156.17.7.22	ICMP	1154	Echo (ping) request id=0x007e, seq=4/1024, ttl=64 (reply in 4278)
4278	77.313592997	156.17.7.22	192.168.0.136	ICMP	1154	Echo (ping) reply id=0x007e, seq=4/1024, ttl=53 (request in 4270)
4305	78.251776375	192.168.0.136	156.17.7.22	ICMP	1154	Echo (ping) request id=0x007e, seq=5/1280, ttl=64 (reply in 4318)
4318	78.317099535	156.17.7.22	192.168.0.136	ICMP	1154	Echo (ping) reply id=0x007e, seq=5/1280, ttl=53 (request in 4305)
4395	79.253324800	192.168.0.136	156.17.7.22	ICMP	1154	Echo (ping) request id=0x007e, seq=6/1536, ttl=64 (reply in 4406)
4406	79.322786591	156.17.7.22	192.168.0.136	ICMP	1154	Echo (ping) reply id=0x007e, seq=6/1536, ttl=53 (request in 4395)

```
dominik@dominik-Nitro-ANS15-55:~$ ping -s 9992 cs.pwr.edu.pl
PING cs.pwr.edu.pl (156.17.7.22) 9992(10020) bytes of data.
10000 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=1 ttl=53 time=59.8 ms
10000 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=2 ttl=53 time=90.7 ms
10000 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=3 ttl=53 time=77.6 ms
10000 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=4 ttl=53 time=63.6 ms
10000 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=5 ttl=53 time=65.4 ms
10000 bytes from informatyka.im.pwr.wroc.pl (156.17.7.22): icmp_seq=6 ttl=53 time=69.5 ms
^C
--- cs.pwr.edu.pl ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5008ms
rtt min/avg/max/mdev = 59.829/71.111/90.710/10.372 ms
```

Słownik

- **Round Trip Time** (Round-trip delay time, RTD, RTT,) – Czas propagacji .minimalny czas wymagany do przesłania sygnału w obu kierunkach, od nadawcy do odbiorcy, a następnie w drugą stronę, po którym możemy otrzymać potwierdzenie otrzymania wiadomości.
- **TTL** - (ang. Time To Live) to parametr określający maksymalny czas życia pakietów. TTL definiuje jak długo wysłany pakiet danych może krążyć w sieci przechodząc od jednego routera do drugiego. Po przejściu przez każdy sieciowy węzeł, wartość TTL zmniejszana jest o 1, i kiedy pakiet osiągnie w końcu wartość 0, jest po prostu kasowany przez ostatni router. Procedura taka stosowana jest po to, aby pakiety, których adres przeznaczenia jest nieprawdziwy lub nieosiągalny, nie błąkały się bez końca w Sieci i nie generowały niepotrzebnego ruchu. Dla systemów Windows 9x i ME, domyślna wartość TTL wynosi 32, natomiast w bardziej zaawansowanych wersjach tego systemu - NT/2000/XP - osiąga poziom 128.
- **MTU** (Maximum Transmission Unit) – rozmiar największego datagramu (w bajtach), który można przekazać przez warstwę protokołu komunikacyjnego.
- **Datagram** – podstawowa jednostka przekazu powiązana z siecią komutacyjną pakietów. Datagramy zwykle są zbudowane z sekcji nagłówka i ładunku. Każdy datagram składa się z dwóch składników: **nagłówka** i **ładunku danych**. Nagłówek zawiera wszystkie niezbędne informacje do transportu pierwotnego zestawu do miejsca przeznaczenia bez konieczności wcześniejszych wymian, między zestawem a siecią. Nagłówki mogą zawierać adresy źródła i miejsca docelowego, ale również typowane pola. Ładunek to dane, które mają zostać przetransportowane. Proces zagnieżdżania się ładunku danych w oznaczonym nagłówku nazywany jest kapsułkowaniem. Datagramy dostarczają możliwość bezpołączeniowej komunikacji w sieci komutacyjnej pakietów. Dostarczenie, czas dostarczenia i kolejność datagramów nie musi być gwarantowana przez sieć.
- **IPv4** (ang. Internet Protocol version 4) – czwarta wersja protokołu komunikacyjnego IP przeznaczonego dla Internetu. Identyfikacja hostów w IPv4 opiera się na adresach IP. Dane przesyłane są w postaci standardowych datagramów. Wykorzystanie IPv4 jest możliwe niezależnie od technologii łączącej urządzenia sieciowe – sieć telefoniczna, kablowa, radiowa itp. IPv4 znajduje się obecnie w powszechnym użyciu.

Źródła:

<https://pl.wikipedia.org>

<https://www.atel.com.pl>

ICMP - [https://en.wikipedia.org/wiki/Ping_\(networking_utility\)#ECHO-REQUEST](https://en.wikipedia.org/wiki/Ping_(networking_utility)#ECHO-REQUEST)