Jakub Omieljaniuk (250090) – sprawozdanie z Listy 1 (Technologie Sieciowe)

Program **ping**, którego używam do poniższych zadań jest wbudowanym programem dostępnym w konsoli poleceń Windows, a także powłoce Bash w systemach Linux, dzięki któremu możemy testować połączenie między naszym urządzeniem a danym serwerem.

Podstawowe zapytanie (polecenie) programu ma następującą składnie:

```
> ping <nazwa/ip domeny>
```

```
C:\Users\Jakbuczyk> ping interia.pl

Pinging interia.pl [217.74.65.23] with 32 bytes of data:
Reply from 217.74.65.23: bytes=32 time=20ms TTL=56
Reply from 217.74.65.23: bytes=32 time=25ms TTL=56
Reply from 217.74.65.23: bytes=32 time=17ms TTL=56
Reply from 217.74.65.23: bytes=32 time=17ms TTL=56

Ping statistics for 217.74.65.23:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 17ms, Maximum = 25ms, Average = 19ms
```

1: Wywołanie polecenia >ping interia.pl

Program odpalony w systemie Windows zamienia nazwę domeny na jego IP za pomocą **protokołu DNS**[1], a następnie korzystając z **protokołu ICMP**[2] wysyła (domyślnie) 4 pakiety (***ICMP Echo Request***) i oczekuje na odebranie pakietów zwrotnych (***ICMP Echo Reply***). W tym wypadku test połączenia ukończył się pomyślnie – otrzymano odpowiedź na każdy z wysłanych pakietów co oznacza, że między moim urządzeniem a podanym adresem można nawiązać połączenie.

[1]*DNS – **Domain Name System**, system baz danych działający w architekturze klient – serwer realizujący zapytania o adresy IP zadanych domen lub serwerów poczty elektronicznej. Żaden serwer DNS nie posiada rekordów ze wszystkimi adresami występującymi w sieci, dlatego w przypadku gdy jeden serwer nie znajdzie IP dla żądanej domeny, wysyła zapytanie do innego serwera DNS. Wyróżnia się 13 głównych serwerów DNS rozmieszczonych na wielu kontynentach oraz niezliczona ilość podrzędnych serwerów. Każda domena powinna mieć co najmniej 2 serwery DNS obsługujące ją.*

[2]*ICMP - **Internet Control Message Protocol**, typ protokołu przeznaczony dla komunikatów kontrolnych. Dzięki niemu dwa hosty mogą zdiagnozować problem połączenia między sobą (np. gdy jeden z nich jest przeciążony lub nieosiągalny dla drugiego), a także wyznaczyć trasę, którą będą się komunikować. Pierwsze 8 bitów (z 32) w pakiecie ICMP determinują jaki typ komunikatu jest wysyłany do odbiorcy. Wartość 8 (zapisana jako 00001000) przeznaczona jest dla Echo Request, natomiast 0 (00000000) oznacza Echo Reply.*

Z podstawowego zapytania możemy uzyskać informację o: adresie IP serwera, do którego wysyłamy pakiety, domyślną wielkość tych pakietów, **czas propagacji** (czyli czas oczekiwania na odpowiedź od momentu wysłania pakietu ***Echo Request***), **wartość TTL**[3] dostarczonego pakietu zwrotnego, statystyki dotyczące ilości utraconych pakietów po drodze, a także maksymalny, minimalny i średni czas propagacji wysłanych pakietów.

W kontekście np. gier online, parametru czasu propagacji (w uproszczeniu używa się pojęcia ping) używa się do określenia stabilności, komfortu prowadzonej rozgrywki – „ping" wynoszący mniej niż 30 ms uznaje się za bardzo dobry, natomiast wartości powyżej 100 będą powodowały zauważalne opóźnienia w grach.

[3]*TTL – **Time to Live**, wartość określająca okres ważności (życia) pakietu danych wysyłanych w sieci. Zazwyczaj TTL określa liczbę przeskoków między węzłami, które może wykonać na swojej trasie w sieci. Każdy router IP zmniejsza wartość TTL o jeden, a gdy wartość ta wyniesie zero – pakiet jest odrzucony i usunięty z sieci, a do nadawcy wysyłany jest komunikat ICMP Time Exceeded. TTL pomaga unikać przeciążenia sieci spowodowanego „błądzącymi" pakietami.*

Aby sprawdzić ile jest węzłów na trasie do serwera za pomocą programu ping, należy wywoływać komendy z parametrem **-i <wartość>,** który ustawia TLL na podaną <wartość>. Jeśli znajdziemy wartość x dla której otrzymujemy komunikaty „TTL expired in transit" (jest to również komunikat protokołu ***ICMP*** – typ 11: ***Time Exceeded***), a dla x+1 pakiet będzie dochodził do żądanego serwera, to liczba węzłów pomiędzy naszym urządzeniem a podanym serwerem wynosi x+1. Liczba węzłów między moim urządzeniem a serwerem interii.pl wynosi 9:

```
C:\Users\Jakbuczyk>ping -i 8 interia.pl

Pinging interia.pl [217.74.65.23] with 32 bytes of data:
Reply from 217.74.64.190: TTL expired in transit.
Reply from 217.74.64.190: TTL expired in transit.
Reply from 217.74.64.190: TTL expired in transit.
Reply from 217.74.64.190: TTL expired in transit.

Ping statistics for 217.74.65.23:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

3: wysłanie pakietów z TTL = 8

```
C:\Users\Jakbuczyk>ping -i 9 interia.pl

Pinging interia.pl [217.74.65.23] with 32 bytes of data:
Reply from 217.74.65.23: bytes=32 time=18ms TTL=56
Reply from 217.74.65.23: bytes=32 time=17ms TTL=56
Reply from 217.74.65.23: bytes=32 time=22ms TTL=56
Reply from 217.74.65.23: bytes=32 time=20ms TTL=56

Ping statistics for 217.74.65.23:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 17ms, Maximum = 22ms, Average = 19ms
```

2: wysłanie pakietów z TTL = 9

W powyższym przykładzie (3) widzimy, że komunikat ***ICMP Time Exceeded*** dla pakietu wysłanego z TTL równym 8 został wysłany z serwera innego niż serwer interii.pl. W ten sposób możemy sprawdzać jakie serwery pośredniczą w wysyłaniu naszego pakietu do miejsca docelowego. W tym wypadku widzimy, że 8 węzłem jest serwer o adresie IP 217.74.64.190, w którym to TLL osiągnął wartość 0 co spowodowało wysłanie komunikatu ***ICMP Time Exceeded***.

Ustalenie ilość węzłów jakie pokonuje pakiet zwrotny jest trudniejsze, ze względu na brak informacji z jaką wartością TTL wysyłany jest komunikat ***Echo Reply***. Możemy jednak założyć, że jest to wartość 32, 64, 128 lub 255, bo takie domyślne ustawienia ma większość routerów (jest to związane z systemem dwójkowym). Dodatkowo liczba węzłów w dzisiejszych realiach raczej nie osiąga wartości większych od 30, więc jeśli w przykładzie *(2)* otrzymywaliśmy pakiety z komunikatem ***Echo Reply*** z TTL o wartości 56 to możemy przypuścić, że ilość węzłów, które pokonała wynosi 64 - 56 = **8 węzłów**.

Inny przykład dla serwera położonego we Francji:

```
C:\Users\Jakbuczyk>ping -i 13 www.kerguelen-voyages.com

Pinging clusterweb1.groupefiga.com [212.78.32.201] with 32 bytes of data:
Reply from 212.78.32.201: bytes=32 time=68ms TTL=118
Reply from 212.78.32.201: bytes=32 time=73ms TTL=118
Reply from 212.78.32.201: bytes=32 time=67ms TTL=118
Reply from 212.78.32.201: bytes=32 time=64ms TTL=118

Ping statistics for 212.78.32.201:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 64ms, Maximum = 73ms, Average = 68ms

C:\Users\Jakbuczyk>ping -i 12 www.kerguelen-voyages.com

Pinging clusterweb1.groupefiga.com [212.78.32.201] with 32 bytes of data:
Reply from 212.78.33.31: TTL expired in transit.
Reply from 212.78.33.31: TTL expired in transit.
Reply from 212.78.33.31: TTL expired in transit.
Reply from 212.78.33.31: TTL expired in transit.

Ping statistics for 212.78.32.201:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

4: Wywołanie ping dla TTL=13 i TTL=12

Wysłanie pakietu do serwera, który jest położony geograficznie dalej od naszego urządzenia zwiększa prawdopodobieństwo, że przejdzie on przez większą liczbę węzłów, ale nie jest regułą. Niektóre strony pomimo dużej odległości geograficznej mogą mieć dobre połączenia (np. strony rządowe) o małej liczbie węzłów, natomiast strony prywatne mogą przechodzić przez wiele węzłów.

Powyższy przykład pokazuje jednak, że liczba węzłów do serwera strony **kerguelen-voyages.com** jest większa niż dla interii.pl i wynosi **13 węzłów** z i prawdopodobnie **10 węzłów** (zakładane początkowe TTL 128 minus otrzymane 118 w odpowiedzi) od serwera docelowego.

Za pomocą parametrów **-l <size>** oraz **-f** można ustawić odpowiednio wielkość pojedynczego wysyłanego pakietu na *<size>* bajtów oraz wymusić blokowanie **fragmentowania pakietów**[4].

[4]*fragmentacja pakietów – proces zachodzący podczas przesyłania pakietów o rozmiarze większym niż dopuszczalny maksymalny rozmiar pakietu (**MTU** – Maximum Transmission Unit) ustawiony na routerze znajdującym się w danej sieci. W takim przypadku router dokonuje podziału (fragmentacji) pakietu na mniejsze i wysyła je dalej wraz z informacją o podziale pakietu (m. in. poprzez flagę MF – More Fragments). Następny lub docelowy router „składa" fragmenty pakietów z powrotem w jeden. Procesy te zabierają dodatkowy czas propagacji, a większa liczba pakietów oznacza większe niebezpieczeństwo ich zagubienia w sieci, dlatego systemy operacyjne starają się unikać fragmentacji.*

W celu zbadania wpływu wielkości pakietu na czas propagacji wysłałem po 10 pakietów (za pomocą parametru **-n 10**) o różnych wielkościach na jeden serwer. Oto jedno z zapytań:

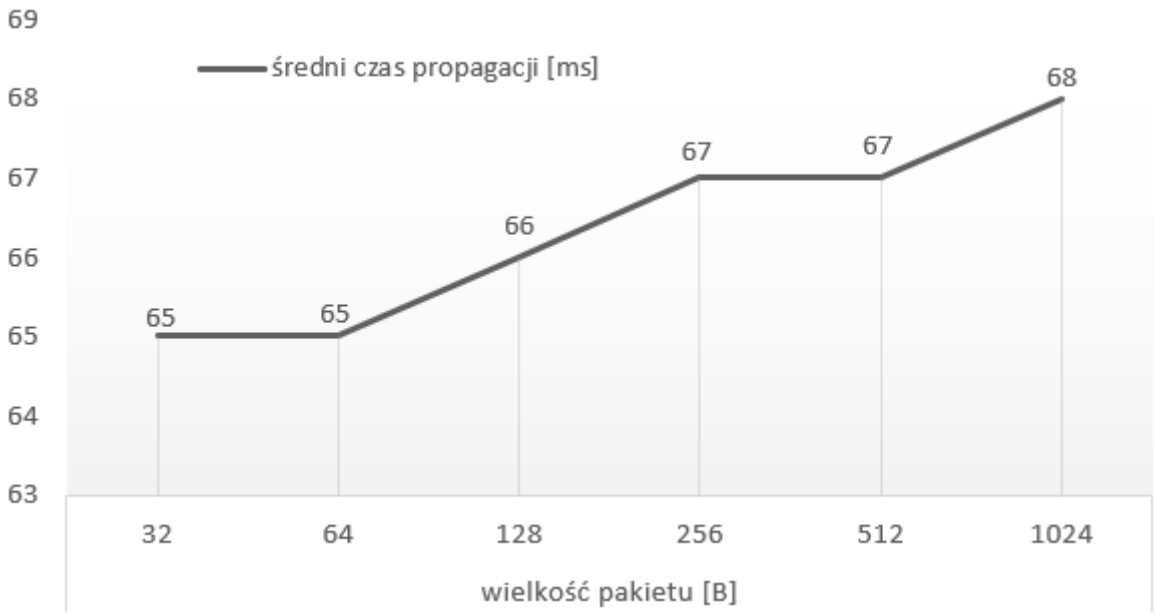
```
C:\Users\Jakbuczyk>ping -n 10 -l 1024 www.kerguelen-voyages.com

Pinging clusterweb1.groupefiga.com [212.78.32.201] with 1024 bytes of data:
Reply from 212.78.32.201: bytes=1024 time=76ms TTL=118
Reply from 212.78.32.201: bytes=1024 time=66ms TTL=118
Reply from 212.78.32.201: bytes=1024 time=65ms TTL=118
Reply from 212.78.32.201: bytes=1024 time=66ms TTL=118
Reply from 212.78.32.201: bytes=1024 time=68ms TTL=118
Reply from 212.78.32.201: bytes=1024 time=68ms TTL=118
Reply from 212.78.32.201: bytes=1024 time=71ms TTL=118
Reply from 212.78.32.201: bytes=1024 time=70ms TTL=118
Reply from 212.78.32.201: bytes=1024 time=66ms TTL=118
Reply from 212.78.32.201: bytes=1024 time=66ms TTL=118

Ping statistics for 212.78.32.201:
    Packets: Sent = 10, Received = 10, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 65ms, Maximum = 76ms, Average = 68ms
```

5: Wysłanie pakietów o wielkości 1024 bajtów

A tutaj wyniki przedstawione w formie tabeli:



6: Zależność czasu propagacji od wielkości wysyłanego pakietu

Pakiety o rozmiarze 2048 bajtów były gubione w sieci i moje urządzenie nie uzyskiwało już odpowiedzi. Wiązało się to prawdopodobnie z potrzebą fragmentacji tych pakietów co zmniejszyło stabilność połączenia. Wartość TTL dla każdego zapytania była taka sama, więc można założyć, że długość trasy nie zmieniała się. Można natomiast zauważyć zależność **wzrostu czasu propagacji wraz ze wzrostem wielkości pakietu**.

```
C:\Users\Jakbuczyk>ping -f -l 1432 google.com

Pinging google.com [172.217.16.46] with 1432 bytes of data:
Reply from 172.217.16.46: bytes=68 (sent 1432) time=22ms TTL=55
Reply from 172.217.16.46: bytes=68 (sent 1432) time=26ms TTL=55

Ping statistics for 172.217.16.46:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 22ms, Maximum = 26ms, Average = 24ms
Control-C
^C
C:\Users\Jakbuczyk>ping -f -l 1433 google.com

Pinging google.com [172.217.16.46] with 1433 bytes of data:
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.

Ping statistics for 172.217.16.46:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

7: Szukanie największego niefragmentowanego pakietu możliwego do wysłania

Domyślna wartość MTU dla protokołu Ethernet to **1500 bajtów** - taka też jest ustawiona na moim urządzeniu (aby to sprawdzić należy wpisać polecenie: *netsh interface ipv4 show subinterfaces*). Pomimo tego **największy niefragmentowany pakiet** jaki można wysłać ma rozmiar **1433 bajtów** (*obrazek (7)*). Jest to spowodowane tym, że na dopuszczalne MTU składa się także rozmiar nagłówków, które łącznie z rozmiarem danych muszą zmieścić się w wartości MTU. Najmniejszy pakiet, który będzie wymagał fragmentacji w danym połączeniu jest wyznaczony przez router (biorący udział w tej sieci) o najmniejszej wartości MTU.

```
C:\Users\Jakbuczyk>netsh interface ipv4 show subinterfaces

   MTU  MediaSenseState   Bytes In  Bytes Out  Interface
------  ---------------  ---------  ---------  -------------
4294967295                1     933228     150062  Loopback Pseudo-Interface 1
  1500                1  2571237008   81658152  Wi-Fi
  1500                5          0          0  Ethernet
  1500                5          0          0  Połączenie lokalne* 1
  1500                5          0          0  Połączenie sieciowe Bluetooth
  1500                5          0          0  Połączenie lokalne* 2
  1500                1          0     987420  VirtualBox Host-Only Network
```

8: Wyszukanie wartości MTU na urządzeniu z Windows

Konieczność fragmentacji pakietów wpływa negatywnie na połączenie, ponieważ konieczność składania i odtwarzania pakietów oraz potrzeba ponownego wysyłania całych pakietów, gdy zaginie tylko jeden z nich destabilizuje nasze połączenie.

Program **tracert** jest również wbudowanym programem w systemie Windows, którego możemy obsługiwać z poziomu wierszu poleceń.

```
C:\Users\Jakbuczyk>tracert

Usage: tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout]
               [-R] [-S srcaddr] [-4] [-6] target_name

Options:
    -d                 Do not resolve addresses to hostnames.
    -h maximum_hops    Maximum number of hops to search for target.
    -j host-list       Loose source route along host-list (IPv4-only).
    -w timeout         Wait timeout milliseconds for each reply.
    -R                 Trace round-trip path (IPv6-only).
    -S srcaddr         Source address to use (IPv6-only).
    -4                 Force using IPv4.
    -6                 Force using IPv6.
```

9: Składnia poleceń w programie tracert

Program realizuje funkcjonalność wyznaczania trasy do zadanego serwera w sposób bardziej zautomatyzowany niż mogliśmy to zrobić programem ping. Oznacza to, że nie musimy badać każdego węzła manipulując wartością TTL, ponieważ program robi to za nas i wyświetla gotową ścieżkę:

```
C:\Users\Jakbuczyk>tracert interia.pl

Tracing route to interia.pl [217.74.65.23]
over a maximum of 30 hops:

  1     3 ms     3 ms     2 ms  192.168.0.1
  2     *        *        *     Request timed out.
  3    18 ms    16 ms    18 ms  pl-ktw01a-rc1-ae-18-0.aorta.net [84.116.253.129]
  4    19 ms    20 ms    18 ms  pl-krk07a-ra1-ae-7-1400.aorta.net [84.116.193.25]
  5    18 ms    20 ms    26 ms  195.150.6.9
  6    19 ms    16 ms    17 ms  ae-uzg.cyfro.net [195.150.0.114]
  7    21 ms    22 ms    27 ms  195.150.7.133
  8    18 ms    16 ms    19 ms  217.74.64.190
  9    21 ms    21 ms    18 ms  star.interia.pl [217.74.65.23]

Trace complete.
```

10: Zbadanie węzłów połączenia do serwera interia.pl

Tracert, podobnie jak program ping, wysyła pakiety ***Echo Request*** ale z tą różnicą, że rozpoczyna od wysłania pakietu z ustawionym TTL = 1 (aby pozyskać adres IP pierwszego węzła). W przypadku gdy nasz komputer otrzyma w odpowiedzi pakiet ***ICMP Time Exceeded*** program wie, że dana wartość TTL nie wystarczyła, aby dotrzeć do żądanego serwera. Wysyła on zatem kolejny pakiet ***Echo Request***, ale z zwiększonym TTL o 1. Gdy program otrzyma w odpowiedzi pakiet ***Echo Reply*** – wie, że jego pakiet dotarł do celu i może zakończyć wyznaczanie trasy.

Najdłuższą ścieżkę, którą udało mi się wyszukać za pomocą programu tracert posiadała **26 węzłów** do i (prawdopodobnie) 20 węzłów od serwera ***Uniwersytetu w Papui Nowej Gwinei***:

```
C:\Users\Jakbuczyk>tracert -h 50 -w 50000 www.upng.ac.pg

Tracing route to www.upng.ac.pg [27.122.21.181]
over a maximum of 50 hops:

  1     4 ms     4 ms     3 ms  192.168.0.1
  2     *     7606 ms  8101 ms  84.116.254.140
  3    22 ms    28 ms    22 ms  pl-ktw01a-rc1-ae-18-0.aorta.net [84.116.253.129]
  4    22 ms    20 ms    24 ms  pl-waw26b-rc1-ae-40-0.aorta.net [84.116.133.29]
  5    24 ms    34 ms    20 ms  pl-waw26b-ri1-ae-24-0.aorta.net [84.116.138.73]
  6    25 ms    23 ms    23 ms  be6830.rcr21.b016833-0.waw01.atlas.cogentco.com [130.117.14.125]

  7    22 ms    28 ms    21 ms  be2882.ccr21.waw01.atlas.cogentco.com [154.54.59.37]
  8    43 ms    48 ms    44 ms  be2252.ccr41.ham01.atlas.cogentco.com [154.54.60.253]
  9    44 ms    46 ms    44 ms  be2815.ccr41.ams03.atlas.cogentco.com [154.54.38.205]
 10   149 ms   232 ms   153 ms  be12194.ccr41.lon13.atlas.cogentco.com [154.54.56.93]
 11   153 ms   153 ms   153 ms  be2099.ccr31.bos01.atlas.cogentco.com [154.54.82.34]
 12   256 ms   150 ms   149 ms  be3599.ccr21.alb02.atlas.cogentco.com [66.28.4.237]
 13   152 ms   152 ms   151 ms  be2878.ccr21.cle04.atlas.cogentco.com [154.54.26.129]
 14   150 ms   151 ms   158 ms  be2717.ccr41.ord01.atlas.cogentco.com [154.54.6.221]
 15   164 ms   164 ms   163 ms  be2831.ccr21.mci01.atlas.cogentco.com [154.54.42.165]
 16   179 ms   173 ms   173 ms  be3035.ccr21.den01.atlas.cogentco.com [154.54.5.89]
 17   186 ms   183 ms   183 ms  be3037.ccr21.slc01.atlas.cogentco.com [154.54.41.145]
 18   204 ms   207 ms   212 ms  be2085.ccr21.sea02.atlas.cogentco.com [154.54.2.197]
 19   213 ms   306 ms   209 ms  be2671.ccr21.pdx01.atlas.cogentco.com [154.54.31.78]
 20   210 ms   210 ms   208 ms  be2216.ccr51.pdx02.atlas.cogentco.com [154.54.31.158]
 21   341 ms   340 ms   342 ms  be2237.ccr51.syd01.atlas.cogentco.com [154.54.45.122]
 22   354 ms   346 ms   341 ms  154.18.97.170
 23   376 ms   379 ms   376 ms  202.58.129.21
 24   380 ms   378 ms   376 ms  202.165.194.4
 25   431 ms   429 ms   426 ms  202.95.202.144
 26     *        *      379 ms  27.122.21.181

Trace complete.
```

11: Wyznaczanie trasy za pomocą tracert

```
Reply from 27.122.21.181: bytes=32 time=377ms TTL=44
Reply from 27.122.21.181: bytes=32 time=384ms TTL=44
Reply from 27.122.21.181: bytes=32 time=377ms TTL=44
Reply from 27.122.21.181: bytes=32 time=401ms TTL=44
Reply from 27.122.21.181: bytes=32 time=384ms TTL=44

Ping statistics for 27.122.21.181:
    Packets: Sent = 28, Received = 17, Lost = 11 (39% loss),
Approximate round trip times in milli-seconds:
    Minimum = 376ms, Maximum = 401ms, Average = 380ms
```

12: Badanie połączenia programem ping (polecenie ping -t upng.ac.pg)

Na podstawie zapytań dla innych stron mogę wysnuć tezę, iż „średnica Internetu” wynosi aktualnie nie więcej niż 30 węzłów.

W przypadku trafienia na **sieci wirtualne**[5] ilość węzłów może być większa. Często jednak trafienie na taką sieć powoduje „błądzenie” pakietu i w efekcie przekroczenie czasu na odpowiedź. Poniżej zapytanie dla serwera **Microsoft Azure** zajmującym się przetwarzaniem danych w chmurze:

```
C:\Users\Jakbuczyk>tracert -h 50 microsoftazure.com

Tracing route to microsoftazure.com [40.112.72.205]
over a maximum of 50 hops:

  1     4 ms     5 ms     4 ms  192.168.0.1
  2     *        *        *     Request timed out.
  3    23 ms    25 ms    22 ms  pl-ktw01a-rc1-ae-18-0.aorta.net [84.116.253.129]
  4    21 ms    22 ms    25 ms  pl-waw26b-rc1-ae-40-0.aorta.net [84.116.133.29]
  5    22 ms    22 ms    22 ms  pl-waw26b-ri1-ae-24-0.aorta.net [84.116.138.73]
  6    23 ms    22 ms    23 ms  ae60-0.waw01-96cbe-1a.ntwk.msn.net [104.44.12.180]
  7    30 ms    39 ms    31 ms  ae22-0.icr01.ber20.ntwk.msn.net [104.44.233.128]
  8    55 ms    55 ms    54 ms  be-100-0.ibr01.ber20.ntwk.msn.net [104.44.23.133]
  9    57 ms    58 ms    55 ms  be-7-0.ibr01.ham30.ntwk.msn.net [104.44.19.116]
 10    58 ms    57 ms    54 ms  be-4-0.ibr05.ams06.ntwk.msn.net [104.44.16.129]
 11    55 ms    56 ms    55 ms  be-2-0.ibr03.ams06.ntwk.msn.net [104.44.16.123]
 12    60 ms    54 ms    56 ms  be-4-0.ibr01.ams30.ntwk.msn.net [104.44.18.184]
 13    57 ms    56 ms    56 ms  be-7-0.ibr01.dub07.ntwk.msn.net [104.44.17.56]
 14    59 ms    55 ms    54 ms  ae102-0.icr02.dub07.ntwk.msn.net [104.44.11.56]
 15     *        *        *     Request timed out.
 16     *        *        *     Request timed out.
 17     *        *        *     Request timed out.
 18     *        *        *     Request timed out.
 19     *        *        *     Request timed out.
 20     *        *        *     Request timed out.
 21     *        *        *     Request timed out.
 22     *        *        *     Request timed out.
 23     *        *        *     Request timed out.
 24     *        *        *     Request timed out.
 25     *        *        *     Request timed out.
 26     *        *        *     Request timed out.
 27     *        *        *     Request timed out.
 28     *        *        *     Request timed out.
 29     *        *        *     Request timed out.
 30     *        *        *     Request timed out.
 31     *        *        *     Request timed out.
 32     *        *        *     Request timed out.
 33     *        *        *     Request timed out.
 34     *        *        *     Request timed out.
 35     *        *        *     Request timed out.
 36     *        *        *     Request timed out.
```

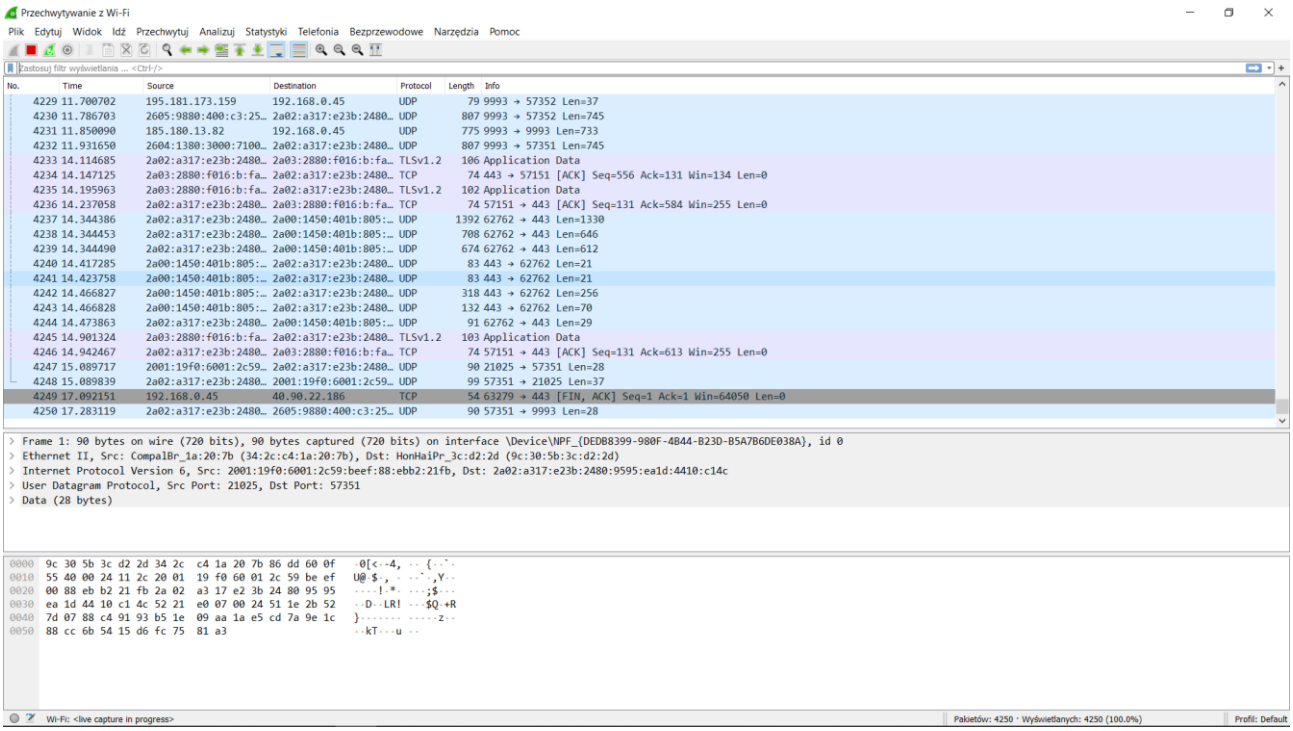
13: Prawdopodobne natrafienie na sieć wirtualną

[5]sieć wirtualna – struktura składająca się z wydzielonych logicznie mniejszych sieci powstałych z podziału jednej, większej sieci fizycznej. Sieci te mogą się między sobą komunikować za pomocą przełącznika, a połączenie z Internetem realizują poprzez jeden adres.

Program **Wireshark** wymaga instalacji, posiada własne GUI i największe możliwości pozyskania i analizowania informacji o ruchu w sieci generowanym przez nasze urządzenie (taki program nazywany jest **snifferem**).

Po włączeniu programu i wybraniu **interfejsu sieciowego**[5], który chcemy podsłuchiwać wyświetli nam się lista wszystkich pakietów, które przechodzą do i z danego interfejsu w czasie rzeczywistym.

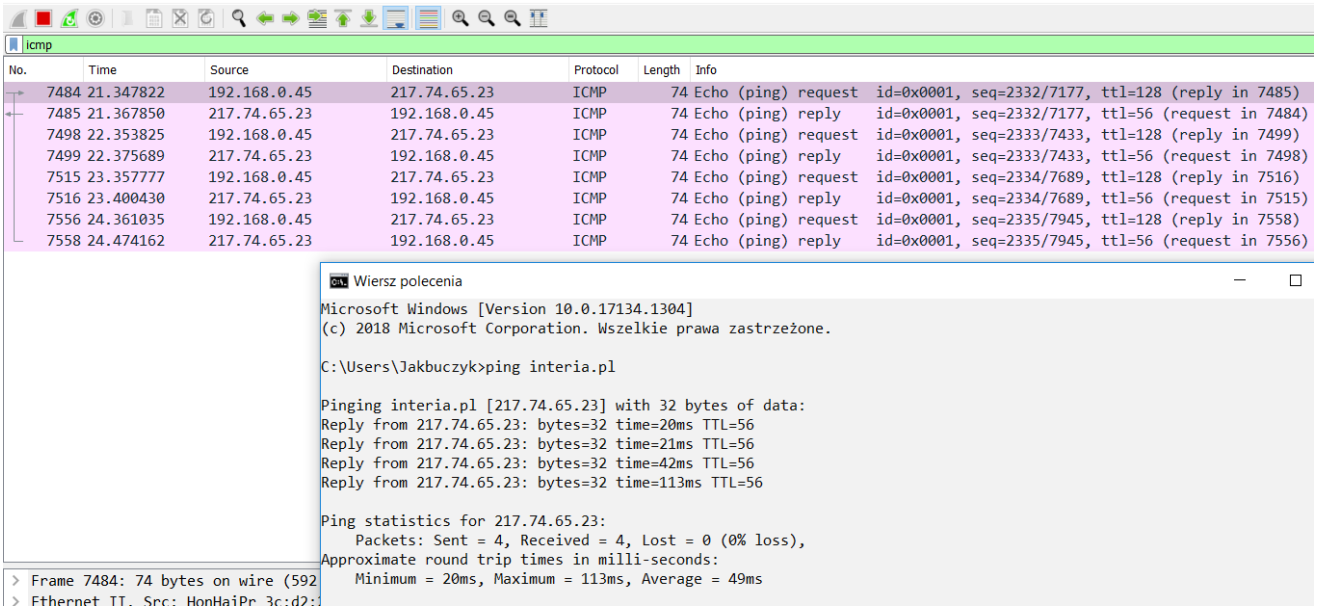
[5]*interfejs sieciowy – określany również jako port, jest to fizyczne urządzenie zapewniające przekształcanie sygnałów elektrycznych lub radiowych tak, aby różne urządzenia mogły się komunikować. Wyróżniamy zarówno interfejsy przewodowe (np. gniazda Ethernet) jaki i bezprzewodowe (Wi-Fi).*



14: Przechwytywanie wszystkich pakietów przechodzących przez sieć Wi-Fi

Program wypisał 4250 pakietów, które w krótkim czasie od włączenia Wiresharka zostały wysłane i odebrane poprzez moją sieć Wi-Fi. Aby wyszczególnić tylko te pakiety, które nas interesują należy posłużyć się **filtrowaniem**. W programie mamy dwa tryby filtrowania: wyświetlania i przeszukiwania. **Filtr wyświetlania** gromadzi informacje o wszystkie pakietach, ale wyświetla tylko te zgodne z filtrem, natomiast **filtr przeszukiwania** przechwytuje jedynie ruch zdefiniowany w filtrze.

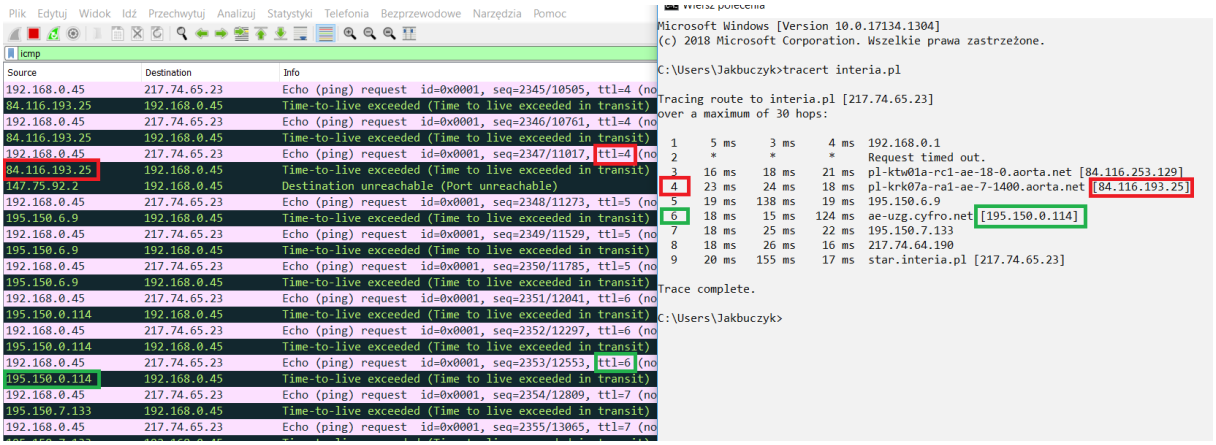
Aby prześledzić jedynie protokoły ICMP stwórzmy filtr wyświetlania *icmp*:



15: Filtr wyświetlania icmp

Jak widać dane z programu ping zgadzają się z danymi wyświetlonymi przez program Wireshark. Przez sieć Wi-Fi przeszły 4 pakiety ICMP **Echo Request** i 4 pakiety ICMP **Echo Reply**.

Za pomocą Wiresharka można dobrze zobaczyć zasadę działania programu tracert, którą opisałem wcześniej.



16: Analiza działania programu tracert za pomocą Wiresharka

W programie Wireshark można było zaobserwować wysyłane pakiety **Echo Request** ze zwiększającym się TTL i odbierane pakiety **Time Exceeded** z kolejnych węzłów na tym połączeniu. Na obrazku zaznaczyłem proces uzyskania adresu IP 4 i 6 węzła.

Podsumowując:

ping jest najprostszym programem najlepiej sprawdzającym się do uzyskania podstawowych informacji (np. sprawdzenia połączenia między naszym urządzeniem a serwerem, zbadanie stabilności takiego połączenia, uzyskanie adresu IP po domenie bądź odwrotnie).

tracert jest wciąż programem prostym w obsłudze, lepiej sprawdzi się do wyznaczenia węzłów występujących w danym połączeniu.

Wireshark jest narzędziem najbardziej rozbudowanym, wymagającym większej wiedzy dotyczącej obsługi programu, aby móc wyciągnąć z niego użyteczne informacje. W porównaniu do dwóch poprzednich programów, Wireshark obserwuje jedynie ruch w sieci, nie wysyła własnych pakietów. Dlatego też praca z tym programem najlepiej sprawdza się w połączeniu z pingiem/tracert.