Sprawozdanie

Laboratorium 1
Valerii Bahrov

Ping:

ping służy do wysyłania sygnału echo do serwera docelowego. Generuje on pakiet o okreslonych parametrach i wysyła pod zadany adres, a kiedy (i jeżeli) otrzyma odpowiedź, zwraca jej parametry tekstowo w konsoli. Przykładowe wywołanie:

```
valera@valera:~$ ping cs.pwr.edu.pl
PING cs.pwr.edu.pl (156.17.7.22) 56(84) bytes of data.
64 bytes from 156.17.7.22: icmp_seq=1 ttl=57 time=14.0 ms
64 bytes from 156.17.7.22: icmp_seq=2 ttl=57 time=14.7 ms
64 bytes from 156.17.7.22: icmp_seq=3 ttl=57 time=7.33 ms
64 bytes from 156.17.7.22: icmp_seq=4 ttl=57 time=6.11 ms
64 bytes from 156.17.7.22: icmp_seq=5 ttl=57 time=19.7 ms
64 bytes from 156.17.7.22: icmp_seq=6 ttl=57 time=7.27 ms
^C
--- cs.pwr.edu.pl ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5009ms
rtt min/avg/max/mdev = 6.112/11.513/19.711/4.965 ms
```

Otrzymujemy więc dane o adresie, czasie w którym dane przechodziły oraz parametrze TTL pakietów zwrotnych. TTL określa maksymalną ilość węzłów, przez którą może przejść jeszcze pakiet. Podstawowymi wartościami są tu zwykle 64, 128 i 255. Dlatego żeby sprawdzić liczbę węzłów jakich pakiet powinny pokonać żeby dotrzeć do serwera, trzeba znaleźć różnicę między wartością TTL ustaloną przez serwer a wartością pola z jaką pakiet do nas dociera. Za pomocą parametru -t możemu ustalić TTL, czyli liczbę przeskoków jaką może wykonać na swojej trasie ping. Gdy liczba przeskoków jest za mała, wtedy pakiet nie dotrze i 100% danych zostanie utraconych. Dzięki temu możemy wykorzystać to dla zmierzenia liczby węzłów dzielących nadawcę od serwera sprawdzając minimalną wartość TTL dla jakiej dane dotrą do hosta.

Adres	Liczba węzłów do	Liczba węzłów z	Lokalizacja	Średni czas
cs.pwr.edu.pl	9	7	Wrocław	9 ms
put.poznan.pl	10	9	Poznań	15 ms
harvard.edu	11	10	Massachusetts	25 ms
ox.ac.uk	22	20	Oxford	48 ms
ameblo.jp	12	10	Tokyo	27 ms

Czy trasy tam i z powrotem mogą być różne?

Jak widać z tabelki, ilość węzłów do hosta i z powrotem nie muszą być takie same.

Jaki wpływ ma wielkość pakietu na czas propagacji?

Adres	60b	300b	500b	1000b
cs.pwr.edu.pl	4 ms	4 ms	5 ms	13 ms
put.poznan.pl	9 ms	11 ms	13 ms	17 ms
harvard.edu	20 ms	25 ms	26 ms	75 ms
ox.ac.uk	45 ms	54 ms	94 ms	106 ms
ameblo.jp	24 ms	59 ms	59 ms	50 ms

Można zauważyć, że rozmiar pakietu ma bardzo mały wpływ na oczekiwania.

Badanie wpływu fragmentcji:

Adres	TTL	Czas	rozmiar
cs.pwr.edu.pl	57	115 ms	30 000 bajtów
put.poznan.pl	55	75 ms	20 000 bajtów
harvard.edu	54	100 ms	5000 bajtów
ox.ac.uk	44	96 ms	1 000 bajtów
ameblo.jp	54	33 ms	1 000 bajtów

Jak widzimy, za pomocą fragmentacji my możemy przesłać jakąkolwiek liczbą bajtów do serwera. Zobaczymy, ile maksymalnie bajtów możemy przesłać na serwer bez fragmentacji:

Bez fragmentacji:

Adres	TTL	Czas	max_rozmiar
cs.pwr.edu.pl	57	23 ms	1472 bajty
put.poznan.pl	55	28 ms	1472 bajty
harvard.edu	54	44 ms	1472 bajty
ox.ac.uk	44	46 ms	1472 bajty
ameblo.jp	54	49 ms	1472 bajty

Robimy wniosek, że ilość bajtów jest ograniczona z góry i równa się 1500 dla pakietu, więc wysłać możemy tylko 1472.

«Średnica» internetu:

```
valera@valera:~$ ping uap.edu.pe
PING uap.edu.pe (146.66.100.132) 56(84) bytes of data.
64 bytes from 146.66.100.132: icmp_seq=1 ttl=33 time=208 ms
64 bytes from 146.66.100.132: icmp_seq=2 ttl=33 time=244 ms
64 bytes from 146.66.100.132: icmp_seq=3 ttl=33 time=252 ms
64 bytes from 146.66.100.132: icmp_seq=4 ttl=33 time=182 ms
64 bytes from 146.66.100.132: icmp_seq=5 ttl=33 time=301 ms
64 bytes from 146.66.100.132: icmp_seq=6 ttl=33 time=137 ms
64 bytes from 146.66.100.132: icmp_seq=7 ttl=33 time=148 ms
64 bytes from 146.66.100.132: icmp_seq=8 ttl=33 time=262 ms
64 bytes from 146.66.100.132: icmp_seq=9 ttl=33 time=137 ms
64 bytes from 146.66.100.132: icmp_seq=10 ttl=33 time=208 ms
64 bytes from 146.66.100.132: icmp_seq=12 ttl=33 time=156 ms
64 bytes from 146.66.100.132: icmp_seq=13 ttl=33 time=255 ms
64 bytes from 146.66.100.132: icmp_seq=14 ttl=33 time=172 ms
64 bytes from 146.66.100.132: icmp_seq=15 ttl=33 time=196 ms
^C
--- uap.edu.pe ping statistics ---
15 packets transmitted, 14 received, 6,66667% packet loss, time 14040ms
rtt min/avg/max/mdev = 137.017/204.187/300.944/50.160 ms
```

Jak widać, ścieżka równa się 27 węzłam.

Sieci wirtualne:

Sieci wirtualne modyfikują wartość wskaźnika TTL, przez co utrudnione jest śledzenie pakietów. To że nasz pakiet na swojej drodze przechodzi przez sieć wirtualną można rozpoznać po tym, że pingując kilka razy (z odstępami czasowymi) dostajemy znaczne różnice TTL, lub odpowiedź uzyskujemy z różnych adresów IP.

Traceroute:

pokazuje ścieżkę, jaką przebywają pakiety wysyłane od nas do docelowego serwera. W tym celu program wysyła pakiety z inkrementowanymi TTL, dzięki czemu kolejne routery na ścieżce pakietów odrzucają je w ten sposób dostajemy informacje o tym, jaki router odrzucił pakiet i wiemy, że normalnie musiałby przejść dalej. Przykładowe wywołanie:

```
valera@valera:~$ traceroute cs.pwr.edu.pl
traceroute to cs.pwr.edu.pl (156.17.7.22), 30 hops max, 60 byte packets
 1  _gateway (192.168.0.1)  4.790 ms  4.791 ms  10.086 ms
 2  156.17.230.254 (156.17.230.254)  11.426 ms  13.276 ms  13.314 ms
 3  234.ds.pwr.wroc.pl (156.17.229.234)  13.403 ms  16.739 ms  16.772 ms
 4  t15-wittiga2.ds.pwr.wroc.pl (156.17.229.241)  48.532 ms  49.261 ms  49.555 m
s
 5  gw.ha.pwr.wroc.pl (156.17.229.253)  39.311 ms 156.17.229.255 (156.17.229.255
)  43.085 ms gw.ha.pwr.wroc.pl (156.17.229.253)  43.688 ms
 6  156.17.254.40 (156.17.254.40)  44.309 ms 156.17.254.41 (156.17.254.41)  41.9
79 ms  42.230 ms
 7  156.17.254.41 (156.17.254.41)  38.691 ms  36.853 ms 156.17.18.244 (156.17.18
.244)  21.204 ms
 8  156.17.18.244 (156.17.18.244)  23.673 ms  22.988 ms  21.189 ms
 9  156.17.7.22 (156.17.7.22)  22.122 ms 156.17.33.1 (156.17.33.1)  5.659 ms  3.
535 ms
```

Wireshark:

Darmowe oprogramowanie open-source służące do analizowania pakietów. Działa w sposób pasywny, tzn. nie wysyła żadnych informacji, a tylko przechwytuje dane docierające do interfejsu sieciowego. Nie wpływa także w żaden sposób na działanie aplikacji przesyłających dane przez sieć**.** Przykładowe wywołanie: