

Sprawozdanie

Laboratorium 1
Valerii Bahrov

Ping:

ping służy do wysyłania sygnału echo do serwera docelowego. Generuje on pakiet o określonych 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 powinien 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 możemy 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 fragmentacji:

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. Zobaczmy, 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 węzłom.

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:

Файл Редактирование Просмотр Запуск Захват Анализ Статистика Телефония Беспроводной Инструменты Помощь									
Применить дисплейный фильтр ... <Ctrl/>									
No.	Time	Source	Destination	Protocol	Length	Info			
1	0.000000000	192.168.0.100	172.217.20.206	UDP	65	56928 → 443 [Len=23]			
2	0.005949359	172.217.20.206	192.168.0.100	UDP	62	443 → 56928 [Len=20]			
3	0.172289456	192.168.0.100	140.82.114.26	TCP	66	34260 → 443 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2169453681 TSecr=1202364235			
4	0.299374834	140.82.114.26	192.168.0.100	TCP	66	[TCP ACKed unseen segment] 443 → 34260 [ACK] Seq=1 Ack=2 Win=31 Len=0 TSval=1202375574 TSecr=2169408337			
5	0.428237426	192.168.0.100	172.217.16.46	TCP	66	33286 → 443 [ACK] Seq=1 Ack=1 Win=1721 Len=0 TSval=3062827180 TSecr=184181782			
6	0.428275917	192.168.0.100	151.101.114.49	TCP	66	46292 → 443 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=1174489794 TSecr=2044694868			
7	0.439765456	172.217.16.46	192.168.0.100	TCP	66	[TCP ACKed unseen segment] 443 → 33286 [ACK] Seq=1 Ack=2 Win=291 Len=0 TSval=184227131 TSecr=3062781626			
8	0.452984941	151.101.114.49	192.168.0.100	TCP	66	[TCP ACKed unseen segment] 443 → 46292 [ACK] Seq=1 Ack=2 Win=61 Len=0 TSval=2044706151 TSecr=1174217979			
9	2.488230871	192.168.0.100	151.101.192.134	TCP	66	52648 → 443 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2898050035 TSecr=1936410566			
10	2.488260511	192.168.0.100	104.16.77.166	TCP	54	49186 → 443 [ACK] Seq=1 Ack=1 Win=2682 Len=0			
11	2.488300316	104.16.77.166	192.168.0.100	TCP	66	[TCP ACKed unseen segment] 443 → 52648 [ACK] Seq=1 Ack=2 Win=68 Len=0			
12	2.502562251	151.101.192.134	192.168.0.100	TCP	66	[TCP ACKed unseen segment] 443 → 52648 [ACK] Seq=1 Ack=2 Win=74 Len=0 TSval=1936421831 TSecr=2897778870			
13	4.524298459	192.168.0.100	87.250.251.119	TCP	66	08028 → 443 [ACK] Seq=1 Ack=1 Win=843 Len=0 TSval=104638123 TSecr=35916389			
14	5.928948029	192.168.0.100	31.13.72.8	TLSv1.2	98	Application Data			
15	5.968949095	31.13.72.8	192.168.0.100	TCP	66	443 → 55546 [ACK] Seq=1 Ack=33 Win=142 Len=0 TSval=3231040406 TSecr=3732409401			
16	6.012677493	31.13.72.8	192.168.0.100	TLSv1.2	94	Application Data			
17	6.012706582	192.168.0.100	31.13.72.8	TCP	66	55546 → 443 [ACK] Seq=33 Ack=29 Win=501 Len=0 TSval=3732409485 TSecr=3231040460			
18	6.028303745	192.168.0.100	151.101.112.134	TCP	66	43884 → 443 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=3522955691 TSecr=1933678018			
19	6.028341951	192.168.0.100	93.184.220.29	TCP	66	50844 → 80 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=3157880803 TSecr=2081758193			
20	6.028352873	192.168.0.100	151.101.113.42	TCP	66	40188 → 443 [ACK] Seq=1 Ack=1 Win=12296 Len=0 TSval=344074403 TSecr=3598426797			
21	8.644159191	151.101.112.134	192.168.0.100	TCP	66	[TCP ACKed unseen segment] 443 → 43884 [ACK] Seq=1 Ack=2 Win=65 Len=0 TSval=1933689282 TSecr=3522683770			
22	8.644177676	151.101.113.42	192.168.0.100	TCP	66	[TCP ACKed unseen segment] 443 → 40188 [ACK] Seq=1 Ack=2 Win=76 Len=0 TSval=3599430060 TSecr=343802318			
23	8.628180356	140.82.114.26	192.168.0.100	TLSv1.2	90	[TCP ACKed unseen segment] , Application Data			
24	8.628398244	192.168.0.100	140.82.114.26	TLSv1.2	94	[TCP Previous segment not captured] , Application Data			
25	8.940547887	140.82.114.26	192.168.0.100	TCP	66	[TCP ACKed unseen segment] 443 → 34260 [ACK] Seq=25 Ack=30 Win=31 Len=0 TSval=1202377735 TSecr=2169462337			
26	10.608239848	192.168.0.100	151.101.113.42	TCP	66	40180 → 443 [ACK] Seq=1 Ack=1 Win=4095 Len=0 TSval=344076451 TSecr=2995529371			
27	10.608271789	192.168.0.100	105.199.108.154	TCP	66	55066 → 443 [ACK] Seq=1 Ack=1 Win=3751 Len=0 TSval=445509532 TSecr=3727324358			
28	10.690609752	105.199.108.154	192.168.0.100	TCP	66	[TCP ACKed unseen segment] 443 → 55066 [ACK] Seq=1 Ack=2 Win=61 Len=0 TSval=3727335619 TSecr=445297278			
29	10.692165770	151.101.113.42	192.168.0.100	TCP	66	[TCP ACKed unseen segment] 443 → 40180 [ACK] Seq=1 Ack=2 Win=179 Len=0 TSval=2995540632 TSecr=343804870			
30	11.729380514	fe80::c58d:dc56:32e...ff02::fb		MDNS	180	Standard query 0x0000 PTR _tcp._local. "OM" question PTR _nfs._tcp._local. "OM" question PTR _afpovertcp._tcp._local. "OM" q...			