Mateusz Bartnik

Piątek 15:15

Technologie sieciowe: sprawozdanie z laboratorium

Lista nr 1

1. Cele wykonanych ćwiczeń

Celem wykonanych ćwiczeń było przetestowanie programów:

1. *ping*:
   1.1. sprawdzenie za jego pomocą ilości węzłów na trasie pomiędzy dwoma urządzeniami w sieci
   1.2. zbadanie wpływu wielkości pakietu na długość przebytej trasy
   1.3. zbadanie wpływu wielkości pakietu na czas propagacji
   1.4. zbadanie wpływu fragmentacji pakietu na długość przebytej trasy oraz czas propagacji
2. *traceroute*
3. *Wireshark*

W następnych punktach przedstawiona została realizacja powyższych ćwiczeń.

2. Ping

2.1. Węzły na trasie

Program *ping* pozwala sprawdzić, czy istnieje sprawne połączenie pomiędzy dwoma węzłami w sieci komputerowej. Za pomocą tego programu można dowiedzieć się ile jest węzłów pomiędzy komputerem użytkownika a wybranym serwerem.

Do testów zostały wybrane dwa serwery: 213.180.141.140 (onet.pl) oraz 101.0.86.43 (serwer w Sydney) jako że znajdują się w dużej odległości geograficznej od siebie.

Liczba węzłów na trasie została sprawdzona manipulując wartością TTL dla poszczególnych sygnałów ping (podobnie jak odbywa się to w programie traceroute).

Poniższy skrypt służył do sprawdzania trasy:

```
for /l %i in (1,1,30) do @ping -i %i -n 1 onet.pl | find "TTL expired"
```

Powyższe wywołanie programu *ping* korzysta z następujących opcji:

-i: specyfikuje wartość TTL dla wysyłanych pakietów.

-n: specyfikuje ilość wysyłanych pakietów.

Skrypt wysyła sygnały ping do serwera za każdym razem zwiększając wartość TTL o jeden oraz wychwytuje wiadomość o tym, że pakiet „wygasł”. Tym sposobem wiadomo, że na pewnym etapie trasy znajduje się router obniżający TTL pakietu do wartości zero. Zliczając wiadomości o wygaśnięciu pakietu można dowiedzieć się, ile węzłów jest na trasie.

Dla onet.pl: 7 węzłów.

Dla serwera w Sydney: 18 węzłów.

Liczbie węzłów na trasie powrotnej również została obliczona na podstawie wartości TTL pokazanych jako wynik działania programu ping.

Dla serwisu onet.pl wartość TTL wynosiła 58, czyli zakładając początkową wartość 64 daje nam to 8 węzłów.

Dla serwera w Sydney wartość TTL wynosiła 111, czyli zakładając początkową wartość 128 daje nam to 17 węzłów.

Jak widać po powyższych wynikach wraz ze wzrostem odległości między węzłami sieci liczba węzłów również wzrasta. Widać także, że trasa do serwera i z powrotem może być inna.

2.2. Wpływ wielkości pakietu na czas propagacji

Największa wielkość pakietu jaką udało się wysłać do serwisu onet.pl wynosiła 14792 bajtów. Zwiększona wielkość nie wpłynęła na długość trasy.

Największa wielkość pakietu jaką udało się wysłać do serwera w Sydney wynosiła 1472 bajtów. Zwiększona wielkość nie wpłynęła na długość trasy.

2.3. Wielkość pakietu a czas

Wywołanie programu *ping* dla pakietu o małym rozmiarze oraz o dużym rozmiarze:

```
Pinging onet.pl [213.180.141.140] with 32 bytes of data:
Reply from 213.180.141.140: bytes=32 time=18ms TTL=58


Pinging onet.pl [213.180.141.140] with 14792 bytes of data:
Reply from 213.180.141.140: bytes=14792 time=41ms TTL=58
```

Jak widać po powyższych wynikach działania programu *ping*, wielkość pakietu ma wpływ na czas jaki zajmuje jego dostarczenie. Im większa ilość danych tym więcej czasu potrzeba na dostarczenie pakietu. Czas rośnie stosunkowo powoli wraz ze wzrostem rozmiaru danych.

2.4. Wpływ fragmentacji pakietów na czas propagacji i długość trasy

Największy niefragmentowany pakiet jaki udało się wysłać do serwisu onet.pl miał rozmiar 1432 bajtów.

Takie same wyniki zostały uzyskane dla serwera w Sydney.

Brak fragmentacji pakietu uzyskiwany jest poprzez dodanie opcji *-f* przy uruchamianiu programu *ping*.

```
ping -f -l 1432 -n 1 onet.pl
Pinging onet.pl [213.180.141.140] with 1432 bytes of data:
Reply from 213.180.141.140: bytes=1432 time=18ms TTL=58
Ping statistics for 213.180.141.140:
Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 18ms, Maximum = 18ms, Average = 18ms

ping -f -l 1433 -n 1 onet.pl
Pinging onet.pl [213.180.141.140] with 1433 bytes of data:
Packet needs to be fragmented but DF set.
Ping statistics for 213.180.141.140:
Packets: Sent = 1, Received = 0, Lost = 1 (100% loss)
```

Pakiety, które nie były fragmentowane były średnio nieznacznie szybciej dostarczane.

```
ping -l 1432 -n 100 onet.pl
Approximate round trip times in milli-seconds:
Minimum = 15ms, Maximum = 81ms, Average = 25ms



ping -f -l 1432 -n 100 onet.pl
Approximate round trip times in milli-seconds:
Minimum = 15ms, Maximum = 83ms, Average = 22ms
```

3. Tracert

Odpowiednikiem programu *traceroute* w systemie Windows jest program *tracert*. Pozwala on sprawdzić trasę jaką przemierza pakiet między urządzeniem użytkownika a serwerem. Za jego pomocą prześledzono trasę pakietu do serwera.

Przykładowe uruchomienie oraz wyniki zwracane przez program *tracert* dla serwisu filmweb.pl wraz z opcją *-d* (brak wysyłania zapytań do DNS co przyspiesza pracę programu):

```
tracert -d filmweb.pl


Tracing route to filmweb.pl [193.200.227.13]
over a maximum of 30 hops:

  1     3 ms     3 ms     3 ms 192.168.1.1
  2    38 ms    24 ms    26 ms 10.9.232.205
  3    49 ms    38 ms    44 ms 83.220.99.17
  4    43 ms    36 ms    39 ms 213.158.198.102
  5    51 ms    38 ms    27 ms 213.158.198.253
  6    43 ms    38 ms    33 ms 195.182.218.74
  7     *        *        *     Request timed out.
  8    56 ms    33 ms    40 ms 193.200.227.13
```

Program *Tracert* pokazuje informacje o etapach trasy jaką przebywa pakiet takie jak: numer przeskoku, czasy między poszczególnymi węzłami, nazwy routerów oraz ich adresy IP.

4. Wireshark

Program *Wireshark* służy do analizowania ruchu sieciowego, który został przechwycony na urządzeniu. Za jego pomocą sprawdzono jakie pakiety danych przechodzą przez komputer.

W górnej części interfejsu graficznego znajduje się filtr wyświetlania. Za jego pomocą można wyświetlić tylko niektóre pakiety, na przykład tylko te korzystające z protokołu HTTP. Interfejs składa się z trzech okien. W górnym wyświetlana jest lista przechwyconych pakietów, w drugim szczegóły pakietu natomiast w trzecim bajty danych. Pierwsze okno zawiera takie informacje jak czas, źródłowy adres pakietu oraz końcowy.

Program umożliwia również dodanie filtra przechwytywania co gwarantuje, że przechwytywane będą tylko wyszczególnione rodzaje pakietów.

W ramach testu w programie *Wireshark* dodano filtr przechwytywania związany z protokołem *icmp*. Następnie w konsoli uruchomiono program *ping* w skutek czego *Wireshark* przechwycił przesyłane pakiety.

Przykładowy przechwycony pakiet:

```
2     0.028833   213.180.141.140  192.168.0.214    ICMP  74

Echo (ping) reply    id=0x0001, seq=1/256, ttl=58 (request in 1)
```

Szczegóły przechwyconego pakietu:

```
Frame 6: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)

Ethernet II, Src: CompalBr_c1:9e:0f (54:67:51:c1:9e:0f), Dst:
IntelCor_a4:b7:ad (78:0c:b8:a4:b7:ad)

Internet Protocol Version 4, Src: 213.180.141.140, Dst: 192.168.0.214

Internet Control Message Protocol
```