Sprawozdanie z laboratorium kursu „Technologie sieciowe”.

Lista 1.

Bartosz Chodorowski <chomzee@ethernet.pl>
Numer indeksu: 166142

Wrocław, 12 marca 2009

1 Cele

Celem zajęć laboratoryjnych było przetestowanie sposobu działania podstawowych programów służących do monitorowania sieci komputerowych.

2 Realizacja

Do przeprowadzenia testów wybrano programy: `ping`, `arping`, `traceroute`, `tcpdump` oraz `wireshark`.

2.1 `ping`

Podstawowym narzędziem do sprawdzania łączności pomiędzy dwoma urządzeniami w sieci jest program `ping`. Z ustaloną częstotliwością wysyła on komunikaty ECHO_REQUEST oczekując jednocześnie odpowiedzi w postaci pakietu ECHO_RESPONSE.

2.1.1 Zasada działania

Testowane narzędzie korzysta z protokołu ICMP (ang. *Internet Control Message Protocol*, internetowy protokół komunikatów kontrolnych)[1]. Część pakietu odpowiadająca temu protokołowi zawiera takie pola jak: typ komunikatu (ECHO_REQUEST), sumę kontrolną, identyfikator (unikalny dla każdego uruchomienia programu `ping`) i numer sekwencyjny (unikalny dla każdego żądania echa). Za nagłówkiem ICMP następują dane (domyślnie 56 bajtów).

Tak zbudowane dane zostają następnie enkapsulowane zgodnie z protokołem warstwy sieciowej IP (ang. *Internet Protocol*, protokół sieciowy)[2]. Pakiet jest adresowany (adresowanie logiczne), ponownie liczona jest suma kontrolna.

Następnie system operacyjny używa protokołu zależnego od budowy sieci, do której dane zostaną skierowane (warstwa „łącza danych" modelu ISO/OSI). W tym przypadku użyty jest protokół Ethernet[3]. Na tym etapie następuje adresowanie fizyczne (kierujemy ramkę ethernetową do najbliższego urządzenia w sieci lokalnej).

Tak przygotowane dane zostają przekazane sterownikowi karty sieciowej, który zajmuje się ich wysłaniem do sieci (warstwa fizyczna modelu ISO/OSI).

Jeśli programowi `ping` przekazano adres docelowy w postaci nazwy domenowej, wówczas przed opisaną powyżej procedurą musi nastąpić rozwiązanie podanej nazwy (np. za pomocą pliku `/etc/hosts` bądź systemu DNS).

Gdy docelowy system operacyjny odbierze wysłany komunikat ECHO_REQUEST, odsyła przekazane dane, zmieniając typ ICMP pakietu na ECHO_RESPONSE. Następnie dane zostają enkapsulowane w protokoły wyższych warstw, podobnie jak to miało miejsce podczas wysyłania żądania.

Protokół ICMP jest uzupełnieniem IP, pełniącym głównie funkcje diagnostyczne. Korzystają z niego

[1] Protokół opisany w RFC 792: `http://tools.ietf.org/html/rfc792`.

[2] Obecnie powszechnie używa się starszą wersję protokołu IP, znaną jako IPv4. Wprowadzanie nowej wersji (IPv6) następuje powoli, dlatego wersja szósta IP została pominięta w tym sprawozdaniu. W ramach IPv6 używa się protokołu ICMPv6, który różni się od obecnego ICMP.

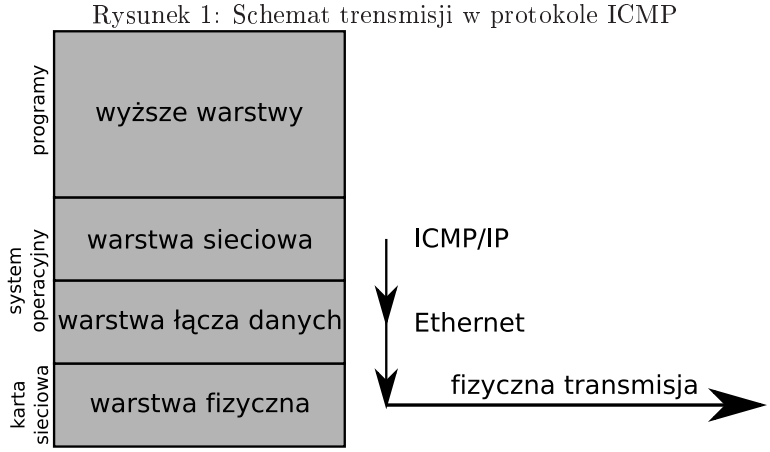
[3] IEEE 802.3: `http://standards.ieee.org/getieee802/802.3.html`.

poszczególne systemy operacyjne, a nie programy (jak to ma miejsce w przypadku protokołów TCP, UDP). W związku z tym wysłanie komunikatu ICMP musi wiązać się z uzyskaniem praw administratora systemu. Dlatego właśnie plik binarny programu `ping` musi mieć ustawiony bit SUID, dzięki czemu możliwe jest używanie go przez zwykłych użytkowników systemu. Tę sytuację obrazuje listing 1. Schemat transmisji danych z uwzględnieniem poszczególnych warstw modelu ISO/OSI przedstawia rysunek 1.

Listing 1: Bit SUID ustawiony dla programu `ping`

```
chomzee@dropsik ~ $ ls -l `which ping`
-rws--x--x 1 root root 30532 VII  6  2008 /bin/ping
```

Rysunek 1: Schemat trensmisji w protokole ICMP



Program `ping` mierzy czas, jaki upłynął od każdego wysłanego żądania do otrzymanej odpowiedzi, dzięki czemu można mierzyć opóźnienia w sieci.

2.1.2 Przykładowe uruchomienia

Listing 2: Sprawdzanie łączności z hostem `www.im.pwr.wroc.pl`

```
chomzee@dropsik ~ $ ping -c 5 www.im.pwr.wroc.pl
PING alfa.im.pwr.wroc.pl (156.17.7.2) 56(84) bytes of data.
64 bytes from alfa.im.pwr.wroc.pl (156.17.7.2): icmp_seq=1 ttl=56 time=70.6 ms
64 bytes from alfa.im.pwr.wroc.pl (156.17.7.2): icmp_seq=2 ttl=56 time=70.7 ms
64 bytes from alfa.im.pwr.wroc.pl (156.17.7.2): icmp_seq=3 ttl=56 time=68.7 ms
64 bytes from alfa.im.pwr.wroc.pl (156.17.7.2): icmp_seq=4 ttl=56 time=69.8 ms
64 bytes from alfa.im.pwr.wroc.pl (156.17.7.2): icmp_seq=5 ttl=56 time=70.8 ms

--- alfa.im.pwr.wroc.pl ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4016ms
rtt min/avg/max/mdev = 68.793/70.161/70.826/0.844 ms
chomzee@dropsik ~ $ ping -s 1000 www.im.pwr.wroc.pl
PING alfa.im.pwr.wroc.pl (156.17.7.2) 1000(1028) bytes of data.
1008 bytes from alfa.im.pwr.wroc.pl (156.17.7.2): icmp_seq=1 ttl=56 time=146 ms
1008 bytes from alfa.im.pwr.wroc.pl (156.17.7.2): icmp_seq=2 ttl=56 time=147 ms
1008 bytes from alfa.im.pwr.wroc.pl (156.17.7.2): icmp_seq=3 ttl=56 time=146 ms
1008 bytes from alfa.im.pwr.wroc.pl (156.17.7.2): icmp_seq=4 ttl=56 time=147 ms
^C
--- alfa.im.pwr.wroc.pl ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3013ms
rtt min/avg/max/mdev = 146.217/146.777/147.240/0.640 ms
```

Listing 2 przedstawia udaną wymianę komunikatów ICMP ECHO z komputerem o adresie `www.im.pwr.wroc.pl`. Za pierwszym razem użyto opcji `-c`, dzięki której ustawiono ilość wysłanych pakietów. Drugie użycie programu

prezentuje użycie opcji `-s`, dzięki której można ustawić rozmiar wysyłanych danych (rozmiar zawsze powiększa się o 8 bajtów ze względu na rozmiar nagłówka ICMP). W linii 18 spowodowano przerwanie wykonywania programu wysyłając sygnał SIGINT (wciśnięto kombinację klawiszy control-c). Ciekawy jest fakt, że 1008 bajtowe pakiety IP potrzebowały ponad dwa razy więcej czasu na podróż niż pakiety 64 bajtowe. Linie 10-11 oraz 20-21 przedstawiają ogólne statystyki, między innymi minimalny, średni i maksymalny czas oczekiwania na odpowiedź.

Przed faktycznym wysłaniem komunikatów ICMP program `ping` dokonał rozwiązania nazwy `www.im.pwr.wroc.pl`, stwierdzając, że nazwa jest aliasem nazwy domenowej `alpha.im.pwr.wroc.pl`, a adres IP komputera którego szukamy to `156.17.7.2` (linie 2 oraz 13 listingu 2).

Listing 3: Brak odpowiedzi

```
1 chomzee@dropsik ~ $ ping ethernet.pl
2 PING ethernet.pl (85.219.166.156) 56(84) bytes of data.
3 ^C
4 --- ethernet.pl ping statistics ---
5 8 packets transmitted, 0 received, 100% packet loss, time 7013ms
```

Brak odpowiedzi ze strony pingowanego hosta przedstawia listing 3. W linii 2 zauważamy, że poprawnie udało się rozwiązać nazwę domenową, uzyskując szukany adres IP. Po około siedmiosekundowym oczekiwaniu, przerwano działanie programu sygnałem SIGINT.

Listing 4: Nieudane rozwiązanie nazwy

```
1 chomzee@dropsik ~ $ ping foo.bar
2 ping: unknown host foo.bar
```

Program `ping` zwraca komunikat o błędzie, gdy rozwiązanie nazwy domenowej zawodzi. Taką sytuację przedstawia listing 4.

2.2 `arping`

Niektóre komputery z różnych względów blokują komunikaty ICMP ECHO_REQUEST. Wówczas niemożliwe jest sprawdzenie, czy dany komputer działa w sieci czy nie. Jeśli jednak ukrywający się komputer znajduje się w tej samej sieci lokalnej, możliwe jest sprawdzenie jego aktywności przez program `arping`.

2.2.1 Zasada działania

Aby wysłać dowolne dane do sieci lokalnej, należy posiadać adres fizyczny odbiorcy (adres MAC). Maszyny działające w sieci muszą więc posiadać mechanizm translacji adresów logicznych z warstwy sieciowej (IP) do adresów fizycznych warstwy łącza danych (Ethernet). Do tego celu służy protokół ARP (ang. *Address Resolution Protocol*)[4].

Aby otrzymać adres fizyczny mając dany adres logiczny, maszyna wysyła na adres rozgłoszeniowy zapytanie, do kogo należy dany adres. Pakiet dociera do wszystkich użytkowników sieci lokalnej, powinien być zignorowany przez każdego za wyjątkiem właściciela szukanego adresu IP. Kieruje on bowiem do pytającego odpowiedź, która pozwala skojarzyć adres IP z odpowiednim adresem MAC.

Pomimo tego, że firewall pewnego komputera w sieci może blokować komunikaty ICMP, powinien odpowiedzieć na zapytanie ARP. Ten fakt wykorzystuje program `arping`. Jego działanie przypomina działanie programu `ping`, jednak zamiast komunikatów ICMP ECHO_REQUEST, wysyłane są zapytania ARP.

[4] Zdefiniowany w RFC 826: `http://tools.ietf.org/html/rfc826`.

Program `arping` wymaga oczywiście uprawnień administratora systemu i, w przeciwieństwie do programu `ping`, nie jest domyślnie udostępniany pozostałym użytkownikom systemu.

2.2.2 Przykładowe uruchomienia

Listing 5: Zdalny komputer blokuje komunikaty ECHO_REQUEST, jednak odpowiada na zapytania ARP

```
1  chomzee@dropsik ~ # ping 192.168.1.1
2  PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
3  ^C
4  --- 192.168.1.1 ping statistics ---
5  4 packets transmitted, 0 received, 100% packet loss, time 2999ms
6
7  chomzee@dropsik ~ # arping -I wlan0 192.168.1.1
8  ARPING 192.168.1.1 from 192.168.1.5 wlan0
9  Unicast reply from 192.168.1.1 [00:1C:F0:C9:7D:51]  3.187ms
10 Unicast reply from 192.168.1.1 [00:1C:F0:C9:7D:51]  2.396ms
11 Unicast reply from 192.168.1.1 [00:1C:F0:C9:7D:51]  2.422ms
12 Unicast reply from 192.168.1.1 [00:1C:F0:C9:7D:51]  2.399ms
13 ^CSent 4 probes (1 broadcast(s))
14 Received 4 response(s)
```

Listing 5 pokazuje przykład praktycznego zastosowania programu `arping`. Komputer o adresie `192.168.1.1` ignoruje zapytania programu `ping`, reaguje jednak na zapytania ARP. Używamy argumentu `-I`, aby explicite wskazać interfejs sieciowy, przez który zapytania mają zostać wysłane.

2.3 `traceroute`

Do diagnozowania problemów leżących poza siecią lokalną bardzo przydatny okazuje się program `traceroute`. Służy on do badania trasy wysyłanych pakietów IP.

2.3.1 Zasada działania

Do badania trasy wykorzystuje pole TTL (ang. *Time To Live*, czas życia) protokołu IP. Podczas normalnej komunikacji ustawia się to pole na odpowiednią dużą wartość (najczęściej 64, 128 lub 255). Każdy router, który przekazuje ten pakiet dalej powinien zmniejszyć wartość tego pola o jeden. Jeśli router otrzyma pakiet z TTL równym 1, odrzuca pakiet wysyłając do nadawcy stosowny komunikat ICMP (time-to-live exceeded).

Program `traceroute` wysyła do danego komputera datagramy UDP/IP z kolejnymi wartościami TTL poczynając od 1. Jednocześnie trwa nasłuchiwanie odpowiedzi, które ustala jakie routery po kolei biorą udział w komunikacji. Mierzony jest również czas pomiędzy wysłaniem pakietu a uzyskaniem odpowiedzi.

2.3.2 Przykładowe uruchomienia

Listing 6: Przykład działania programu `traceroute`

```
1 chomzee@dropsik ~ $ traceroute -n www.im.pwr.wroc.pl
2 traceroute to www.im.pwr.wroc.pl (156.17.7.2), 30 hops max, 40 byte packets
3 1  192.168.2.1  17.587 ms  18.240 ms  18.657 ms
4 2  10.146.0.1  48.997 ms  49.227 ms  56.052 ms
5 3  192.168.144.1  56.353 ms  57.934 ms  58.576 ms
6 4  217.172.225.245  68.273 ms  68.898 ms  70.184 ms
7 5  153.19.102.233  73.468 ms  66.245 ms  67.252 ms
```

```
6  153.19.102.93  76.273 ms  60.642 ms  60.736 ms
7  156.17.255.155  64.650 ms  57.685 ms  62.061 ms
8  156.17.18.253  58.712 ms  63.104 ms  62.118 ms
9  156.17.18.221  62.334 ms  57.139 ms  61.228 ms
10  156.17.7.2  59.764 ms  62.599 ms  63.273 ms
```

Listing 6 pokazuje przykładowe wyjście programu `traceroute`. Użyto również opcji `-n`, która powoduje przedstawienie wszystkich adresów w postaci liczbowej (domyślnie następuje próba translacji adresu na postać domenową). W każdej z 10 ponumerowanych linii widzimy kolejno: adres routera, od którego otrzymano pakiet „time-to-live exceeded" oraz trzy wartości czasu, jakie upłynęły od momentu wysłania danego datagramu – każdy pomiar powtórzono bowiem 3 razy.

Z otrzymanego wyniku można wywnioskować, że komputer o adresie `www.im.pwr.wroc.pl (156.17.7.2)` znajduje się w odległości 10 routerów od komputera z którego wykonano pomiar, a czas odpowiedzi wynosi około 60 ms.

Listing 7: Bardziej rozbudowanywynik działania programu `traceroute`

```
chomzee@dropsik ~ $ traceroute -q 2 mit.edu
traceroute to mit.edu (18.7.22.69), 30 hops max, 40 byte packets
1  loki (192.168.2.1)  17.470 ms  18.246 ms
2  10.146.0.1 (10.146.0.1)  18.771 ms  21.965 ms
3  192.168.144.1 (192.168.144.1)  27.597 ms  27.889 ms
4  89.228.2.174 (89.228.2.174)  46.685 ms  47.245 ms
5  212.73.253.169 (212.73.253.169)  54.330 ms  55.308 ms
6  ae-5-5.ebr1.Frankfurt1.Level3.net (4.69.134.14)  69.868 ms  70.414 ms
7  ae-81-81.csw3.Frankfurt1.Level3.net (4.69.140.10)  73.830 ms ae-61-61.csw1.
    Frankfurt1.Level3.net (4.69.140.2)  80.128 ms
8  ae-82-82.ebr2.Frankfurt1.Level3.net (4.69.140.25)  77.812 ms ae-72-72.ebr2.
    Frankfurt1.Level3.net (4.69.140.21)  78.385 ms
9  ae-42-42.ebr2.Washington1.Level3.net (4.69.137.54)  149.770 ms ae-41-41.ebr2.
    Washington1.Level3.net (4.69.137.50)  153.647 ms
10  ae-92-92.csw4.Washington1.Level3.net (4.69.134.158)  153.466 ms ae-3-3.ebr3.
    NewYork1.Level3.net (4.69.132.90)  156.297 ms
11  ae-84-84.ebr4.Washington1.Level3.net (4.69.134.185)  151.175 ms ae-64-64.
    ebr4.Washington1.Level3.net (4.69.134.177)  171.230 ms
12  ae-3-3.ebr1.NewYork1.Level3.net (4.69.132.94)  169.748 ms  167.491 ms
13  * *
14  ae-0-11.bar1.Boston1.Level3.net (4.69.140.89)  208.748 ms  171.610 ms
15  ae-7-7.car1.Boston1.Level3.net (4.69.132.241)  174.142 ms  173.530 ms
16  MASSACHUSET.car1.Boston1.Level3.net (4.53.48.98)  170.381 ms  149.622 ms
17  W92-RTR-1-BACKBONE.MIT.EDU (18.168.0.25)  153.551 ms  161.701 ms
18  WEB.MIT.EDU (18.7.22.69)  161.079 ms  154.854 ms
```

O wiele ciekawszy wynik działania programu prezentuje listing 7. Nie użyto opcji `-n`, więc pojawiły się adresy w postaci domenowej. Argumentem `-q` ustalono ilość wykonywanych prób.

Linia 9 (siódmy pomiar) zawiera dwa adresy. Wynika z tego, że w obu przeprowadzonych próbach otrzymano odpowiedzi od różnych routerów (tak więc trasa przekazywanych pakietów nie jest statyczna). Podobna sytuacja zaszła przypoczątkowych wartościach TTL 7, 8, 9, 10, 11.

Dwie gwiazdki w pomiarze 13 oznaczają brak odpowiedzi. Prawdopodobnie odpowiedź została zablokowana przez firewall maszyny, która powinna wygenerować stosowny pakiet ICMP.

2.4 `tcpdump`

Niezastąpionym narzędziem do diagnozowania problemów w sieciach komputerowych są narzędzia określane mianem „sniffer". Programy tego typu służą do zbierania, filtrowania i analizowania ruchu sieciowego.

Program `tcpdump` jest najbardziej znanym i sprawdzonym snifferem. Używany jest często do analizowania problemów w sieci, wykrywania intruzów, monitorowania przepływu itp.

2.4.1 Zasada działania

`tcpdump` do pozyskiwania pakietów z sieci używa biblioteki `libpcap`, która notabene została napisana przez twórców `tcpdump`a. Jest to najpopularniejsza biblioteka tego typu. Największą wadą programu `tcpdump` jest dość niewygodny, tekstowy interfejs użytkownika. Jest on jednocześnie jego największą zaletą – wymagania sprzętowe są nikłe, program można uruchomić na każdym komputerze, bez systemu okienkowego. Zazwyczaj dane przechwycone przez `tcpdump` zapisuje się w pliku, aby później je przeanalizować i obejrzeć w innym programie (np. `wireshark`).

2.4.2 Przykładowe uruchomienia

Listing 8: Zbieranie ruchu sieciowego programem `tcpdump`

```
1 chomzee@dropsik ~ # tcpdump -i wlan0 -s 0 -w out.dump
2 tcpdump: listening on wlan0, link-type EN10MB (Ethernet), capture size 65535
    bytes
3 ^C5142 packets captured
4 5142 packets received by filter
5 0 packets dropped by kernel
6 (~) # du -h out.dump
7 5,0M    out.dump
8 (~) # file out.dump
9 out.dump: tcpdump capture file (little-endian) - version 2.4 (Ethernet, capture
    length 65535)
```

Listing 8 prezentuje niektóre najważniejsze opcje programu `tcpdump`. Parametrem `-i` określono, że pakiety mają być pobierane z interfejsu sieciowego `wlan0`. Gdyby opcja `-i` nie była ustawiona, użyty zostałby domyślny interfejs. Następnie opcją `-s` (ang. *snap-len*) definiujemy maksymalny rozmiar pakietów, jakie mają być przechwytywane. Jeśli pakiet będzie większy niż narzucone ograniczenie, zostanie on „ucięty", przechwyconych zostanie tylko określona liczba bajtów. Podanie wartości `0` spowoduje ustawienie najmniej restrykcyjnego ograniczenia (w linii 2 listingu widzimy, że w tym przypadku ta wartość wynosi 65535 bajtów). Za pomocą opcji `-w` nakazujemy odłożenie wyników w pliku, który wyśmienicie nadaje się do dalszej analizy.

Po „złapaniu" odpowiedniej liczby pakietów działanie programu zostało przerwane z terminala (linia 3), program `tcpdump` wypisał ilość przechwyconych pakietów (5142). W linii 6 sprawdzamy rozmiar powstałego pliku, a w linii 8 sprawdzamy jego typ.

Listing 9: Oczekiwanie na pierwsze połączenie przychodzące

```
1  chomzee@dropsik ~ # tcpdump -i tun0 -s 0 -c 1 -XX -n \
2  > 'tcp[tcpflags] == tcp-syn and dst 192.168.2.5'
3  tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
4  listening on tun0, link-type RAW (Raw IP), capture size 65535 bytes
5  21:23:24.215239 IP 192.168.2.1.52925 > 192.168.2.5.22: S
       1037092264:1037092264(0) win 5840 <sackOK,timestamp 11024174 0,mss 1368,nop,
       wscale 1>
6  0x0000:  4500 003c f286 4000 4006 c2de c0a8 0201  E..<..@.@.......
7  0x0010:  c0a8 0205 cebd 0016 3dd0 c5a8 0000 0000  ........=.......
8  0x0020:  a002 16d0 a218 0000 0402 080a 00a8 372e  ..............7.
9  0x0030:  0000 0000 0204 0558 0103 0301            .......X....
10 1 packets captured
11 1 packets received by filter
12 0 packets dropped by kernel
```

Ciekawszy wynik działania programu pokazuje listing 9. Oprócz opcji znanych z poprzedniego listingu, użyto parametru `-XX`, który powoduje wypisanie zawartości pakietu w postaci heksadecymalnej i testowej wraz ze wszystkimi nagłówkami. Charakter użytego interfejsu sieciowego (tunel IP) powoduje, że przechwycone pakiety zaczynają się od nagłówka protokołu IP. Parametr `-n` wymusza, aby `tcpdump` nie próbował rozwiązywać nazw domenowych komputerów biorących udział w komunikacji. Opcja `-c 1` powoduje, że program zakończy pracę, gdy tylko otrzyma jeden pakiet pasujący do filtra.

Na końcu 1 linii użyto odwróconego ukośnika, aby kontynuować wprowadzanie polecenia w linii 2. Następnie użyto apostrofu (operatora cytowania dosłownego), aby wprowadzić główny argument polecenia `tcpdump`, który jest interpretowany jako filtr. Podany filtr jest czuły jedynie na pakiety protokołu TCP, które mają ustawioną wyłącznie flagę SYN oraz przychodzą na `192.168.2.5` (adres komputera, na którym polecenie zostało wykonane).

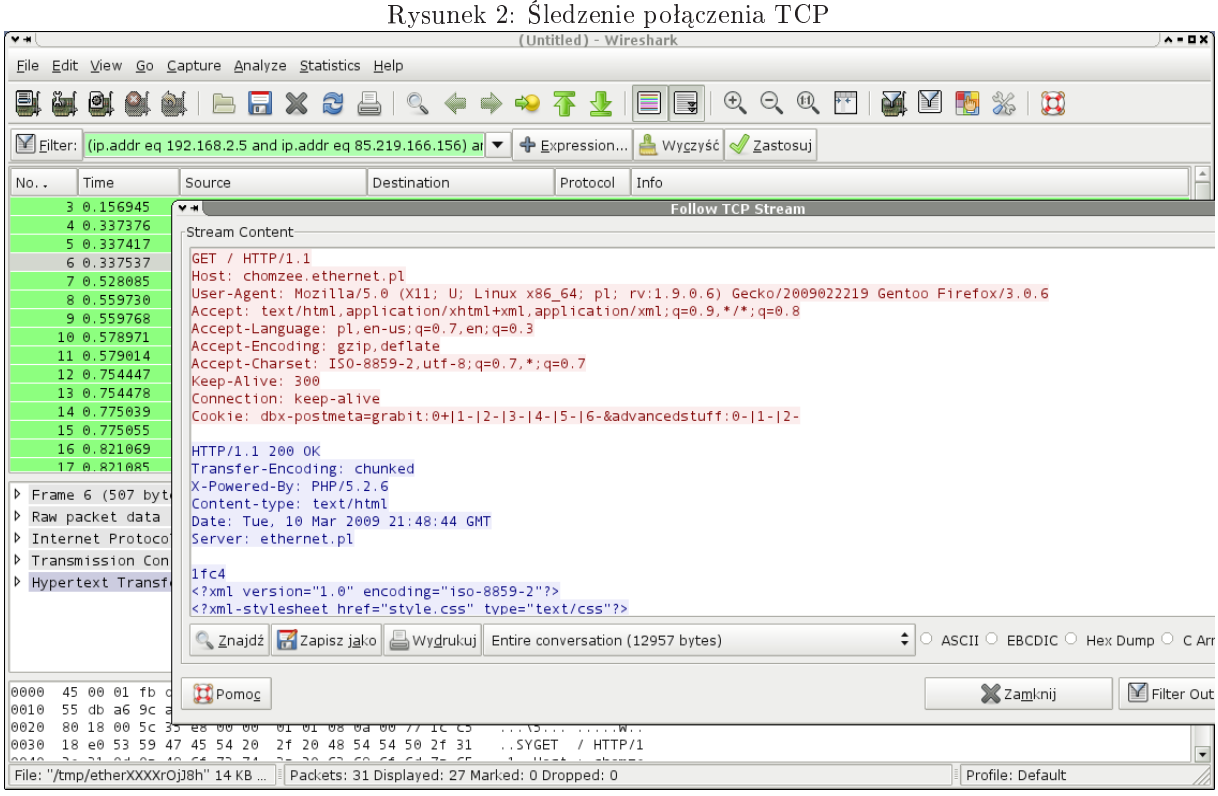
Innymi słowy, tak skonstruowane polecenie zakończyło się przy pierwszej próbie nawiązania połączenia TCP/IP przez inną maszynę z sieci (`192.168.2.1`), po czym szczegółowo wydrukowało zawartość pakietu. Linia 5 opisuje w skrócie najistotniejsze pola protokołów IP oraz TCP.

2.5 `wireshark`

`wireshark` jest graficznym programem służącym do przechwytywania i analizowania ruchu sieciowego. Dane do analizy może samodzielnie pobierać, tak samo jak `tcpdump` (za pomocą biblioteki `libpcap`), jak również może wczytywać pliki zapisane za pomocą opcji `-w` programu `tcpdump`.

Największą zaletą tego programu jest jego wygodny, graficzny interfejs użytkownika. Na uwagę zasługują też liczne funkcje statystyczne i filtrujące.

Rysunek 2: Śledzenie połączenia TCP



2.5.1 Przykładowe wykorzystanie

Opcja „follow TCP stream" (przedstawiona na rysunku 2) pozwala przedstawienie komunikacji w ramach pojedynczej sesji TCP/IP w bardzo przystępny sposób. Jest ona szczególnie pomocna przy badaniu tekstowych protokołów warstwy aplikacji takich jak HTTP, SMTP.

3 Wnioski

Testy wyżej wymienionych programów jednoznacznie potwierdziły ich użyteczność. Programy `ping` oraz `arping` znajdują zastosowanie w sprawdzaniu łączności pomiędzy dwoma komputerami. `traceroute` jest narzędziem niezbędnym do diagnozowania problemów związanych z routingiem. Programy `tcpdump` i `wireshark` są snifferami, które odpowiednio użyte dostarczają wielu użytecznych informacji.