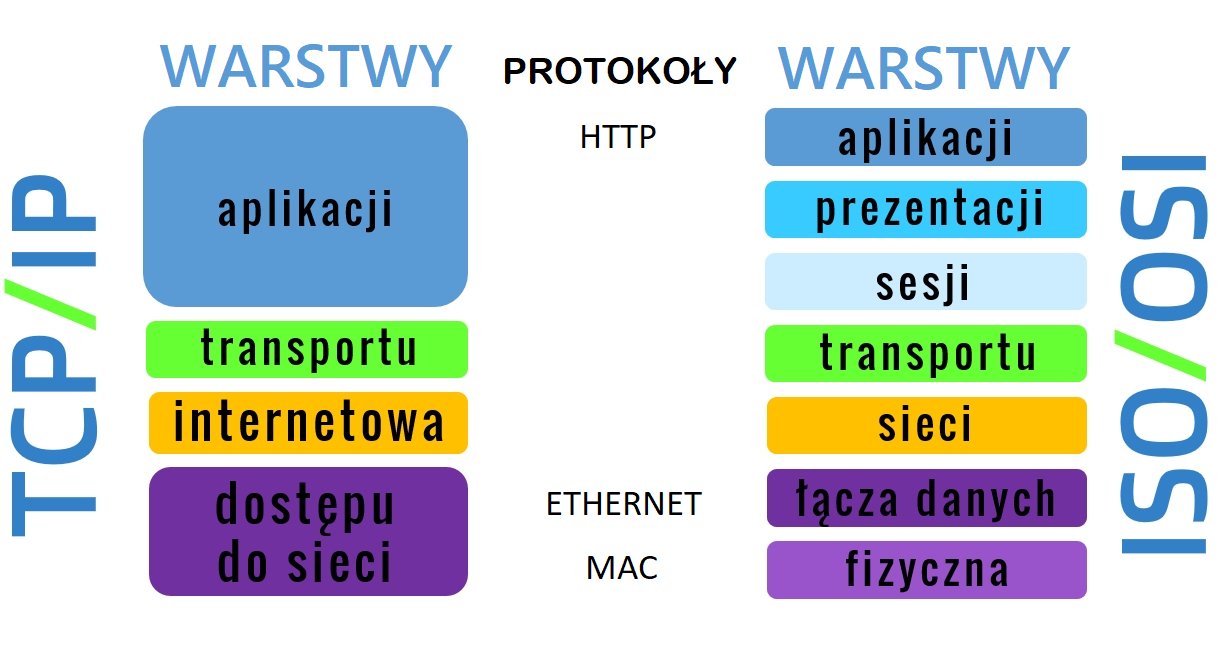
**Sprawozdanie z Listy 5 (Technologie Sieciowe)**

*Jakub Omieljaniuk (250090)*

W liście 3 omówiłem proces komunikacji między urządzeniami w sieci na najniższych warstwach modelów TCP/IP i ISO/OSI. Korzystaliśmy tam z protokołu Ethernet (ramkowanie danych) oraz protokołu CSMA/CD rozszerzającego protokół MAC (dostęp do medium transmisyjnego). Dane na tym poziomie były ciągami zer i jedynek, przekształcanymi na sygnał (prąd elektryczny lub fale), który jest rozprowadzany w medium transmisyjnym. Pokazałem również jak ciąg zer i jedynek jest zamieniany na czytelny dla programisty kod.

W liście 5 opiszę komunikację między urządzeniami (klientem a serwerem), odbywającą się na najwyższej warstwie modelów, czyli **warstwie aplikacji**. W tej warstwie operujemy na zrozumiałym dla użytkownika kodzie, korzystając m.in. z **protokołu HTTP** – ***Hypertext Transfer Protocol***.



1. Warstwy modeli i wykorzystywane protokoły

Protokół HTTP określa zasady komunikacji między urządzeniami (programami) posługując się zdefiniowaną formą **żądań** wysyłanych do serwera i **odpowiedzi** odsyłanych do klienta. Przedmiotem żądania jest konkretny zasób, którym może być np. obrazek, strona HTML czy plik z kodem JavaScript. Każdy zasób jest identyfikowany poprzez nadany mu unikalny łańcuch znaków zwany **URI *(Uniform Resource Identifier)***. Przykład URI:

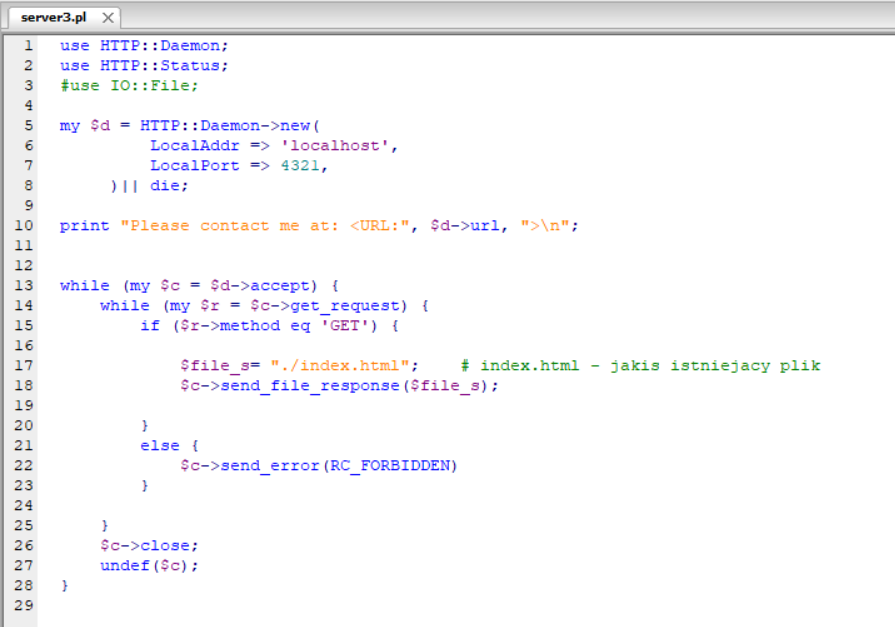
*/articles/sport/23hj42.html*

**Adres URL *(Uniform Resource Locator)*** jest bardziej szczegółową formą identyfikacji zasobów, ponieważ musi zawierać dodatkowo ścieżkę do niego i podaje protokół, za pomocą którego można go ściągnąć:

[*http://www.sport.com/articles/sport/23hj42.html*](http://www.sport.com/articles/sport/23hj42.html)

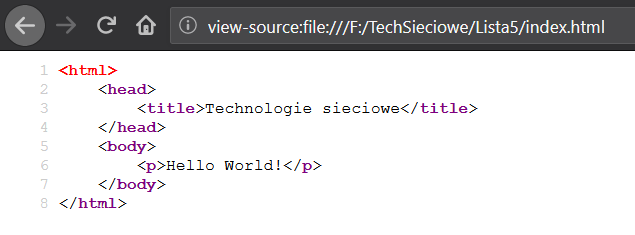
Adres URL jest jednocześnie URI, natomiast nie każdy URI musi być adresem URL (tak samo jak każdy kwadrat jest prostokątem, ale nie każdy prostokąt jest kwadratem).

Interpretacja żądań serwera może być realizowana przez napisany program (skrypt) na serwerze. Przykład takiego programu został nam załączony do tej listy:



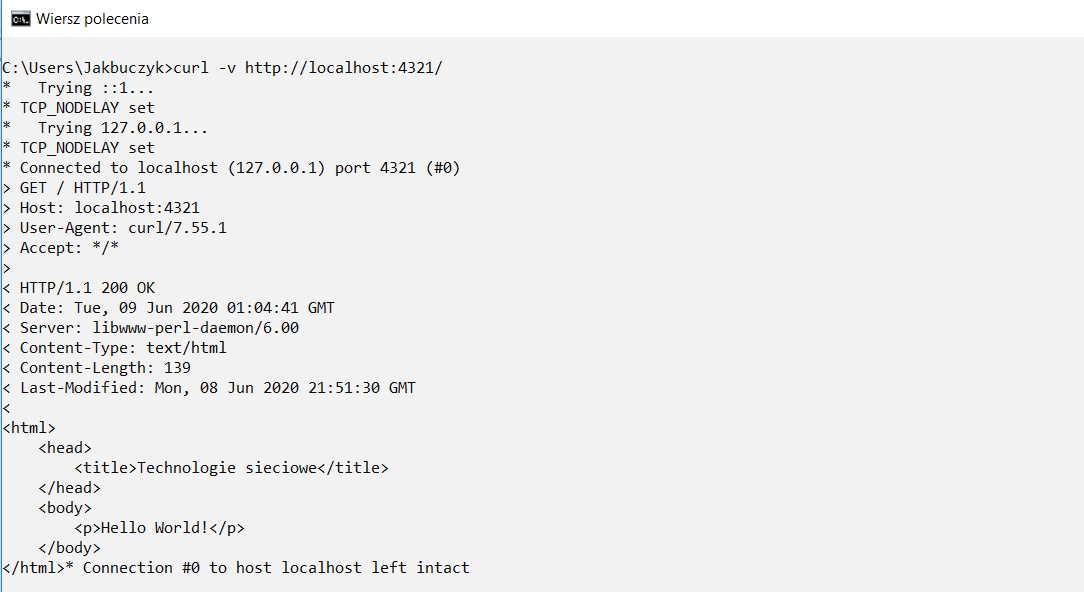
2. Skrypt serwera obsługujący żądania klienta

W kodzie tym zmieniłem 6 linijkę, zamieniając lokalny adres na ‘localhost’ i odpaliłem go przez program Padre (Perl IDE). Stworzyłem również plik index.html w folderze, z którego odpalany jest skrypt server3.pl:



3. Plik index.html

Do przeanalizowania wysyłanych informacji między klientem a serwerem posłużę się programem **curl** obsługiwanym z linii komend. Flaga -v pozwala nam na uzyskanie szczegółowych informacji. Dane wysyłane przez klienta poprzedzone są znakiem >, natomiast odbierane - znakiem <:



4. Wysłanie żądania do serwera

Poleceniem *curl -v* [*http://localhost:4321/*](http://localhost:4321/)nawiązałem połączenie z serwerem, poprzez wysłanie **żądania GET1** zasobu /, czyli domyślnego (głównego) zasobu serwera, a także wysłałem informację o wersji protokołu HTTP, którym się komunikuję (HTTP/1.1). Ogólny format żądania zawiera:

* typ żądania1, zasób do którego się odonosi i wersję protokołu,
* nagłówki
* pusta linia oznaczająca koniec nagłówków,
* ciało wiadomości (w przypadku żądań GET jest puste)

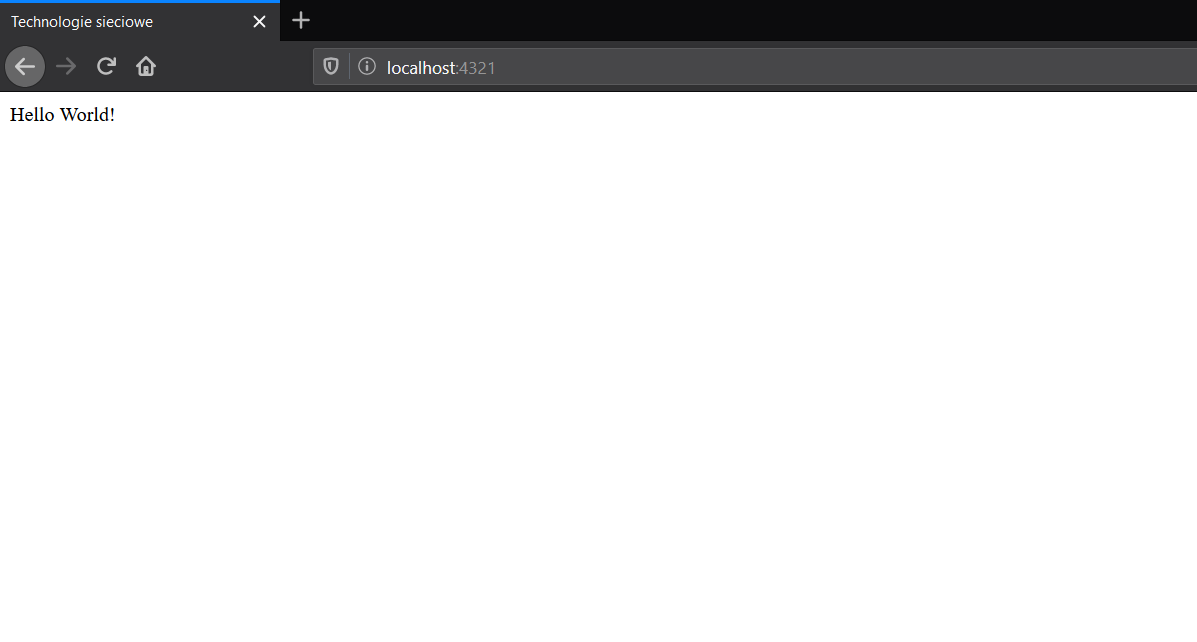
Od serwera otrzymałem najpierw **status odpowiedzi2** (200 OK), a także kod HTML domyślnej strony (w tym wypadku index.html). Ogólny format odpowiedzi zawiera:

* wersję protokołu i status odpowiedzi
* nagłówki
* pusta linia oznaczająca koniec nagłówków,
* ciało wiadomości (w tym przypadku kod HTML)

***1czasownik HTTP*** *(typ żądania) – kluczowe słowo określające typ żądania. Najczęściej używane czasowniki to:* ***GET*** *(pobieranie zasobów z nagłówkami),* ***HEAD*** *(pobieranie jedynie nagłówków),* ***POST*** *(przesyłanie danych do serwera),* ***PUT*** *(aktualizacja danych na serwerze),* ***DELETE*** *(usuwanie danych z serwera).*

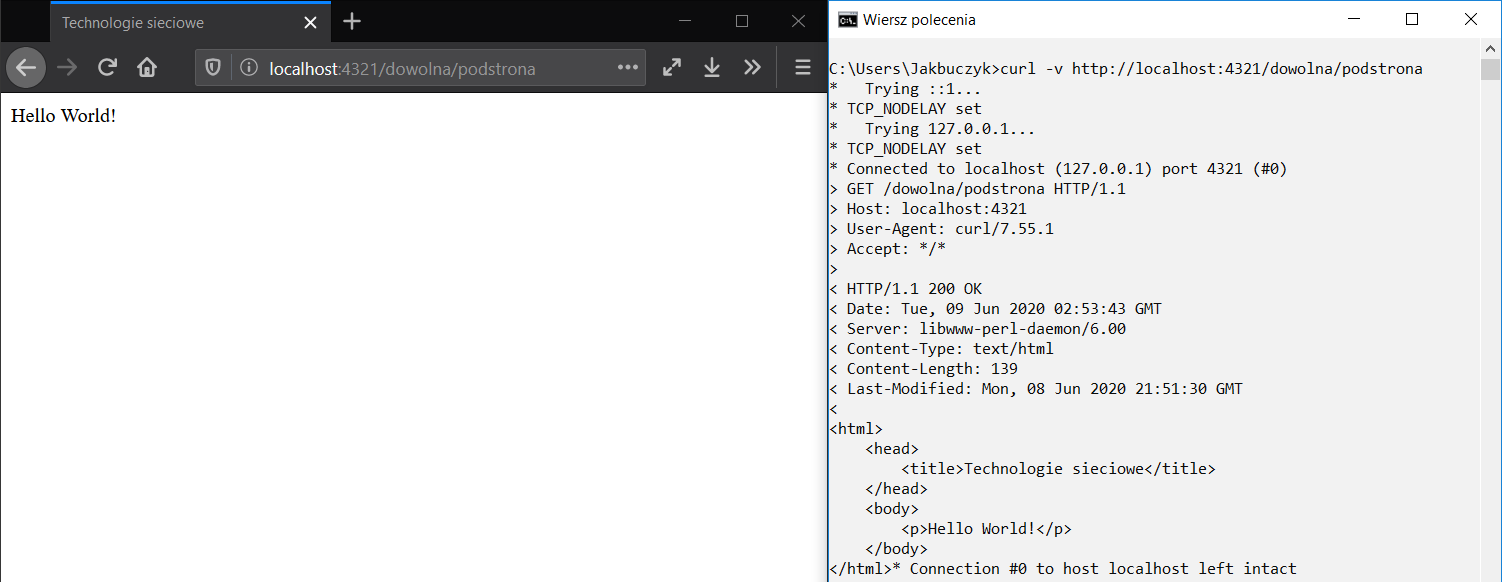
***2status odpowiedzi*** *– 3-cyfrowy kod z tytułem będący podstawową informacją czy dane żądanie się powiodło. Każdy status określa inną sytuację, np.* ***404 Not Found*** *mówi o tym, że żądanego zasób nie znaleziono na serwerze, a* ***200 OK****, że dane żądanie zostało poprawnie przetworzone. Dokładne opisy wszystkich statusów można znaleźć w Internecie.*

Programem klienckim może być również przeglądarka internetowa, która samodzielnie wykona żądanie do serwera, podanego przez nas w adresie URL:

**

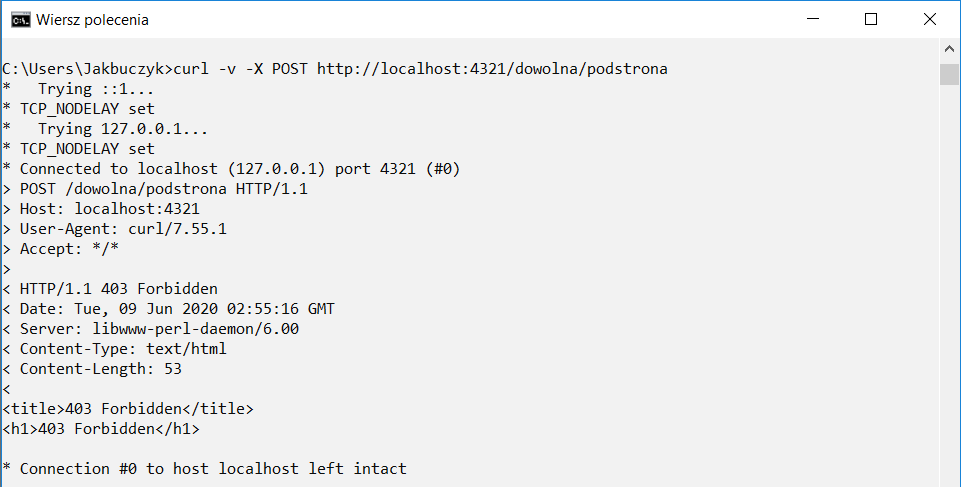
5. Strona serwera

Przy bardziej rozbudowanych stronach, przeglądarka może wysłać wiele żądań pobrania zasobów do serwera, aby załadować pojedynczą podstronę wraz ze wszystkimi obrazkami i skryptami. W naszym skrypcie, serwer czeka na dowolne żądanie typu GET (linijka 15 w server3.pl) i w odpowiedzi wysyła kod pliku index.html (linijka 18). Możemy zatem wpisać dowolną, nawet nieistniejącą, podstronę, a w odpowiedzi dostaniemy stronę index.html:



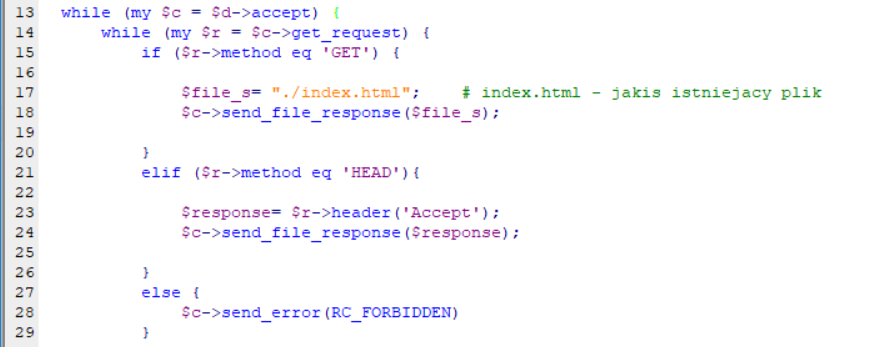
6. Strona serwera dla dowolnego zapytania GET

W przypadku zapytania innego niż GET (np. POST), otrzymamy w odpowiedzi status **403 Forbidden** (co jest zdefiniowane w naszym skrypcie server3.pl w 22 linijce):



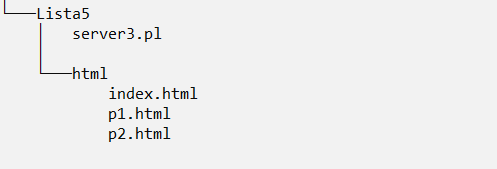
7. Żądanie POST wysłane do serwera

Aby serwer w odpowiedzi na zapytanie HEAD przesyłał nagłówek naszego zapytania, należy dodać obsługę zapytań HEAD w server3.pl w pętli while (linijki 21-26):

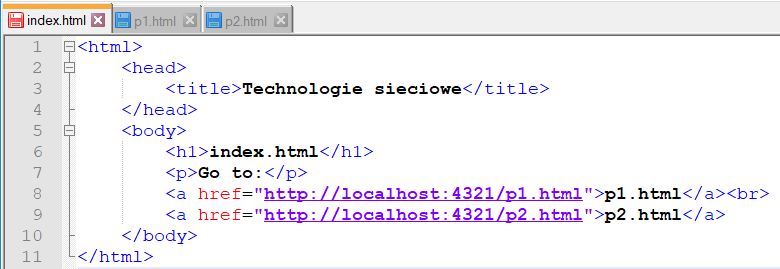


8. Dodanie obsługi żądania HEAD

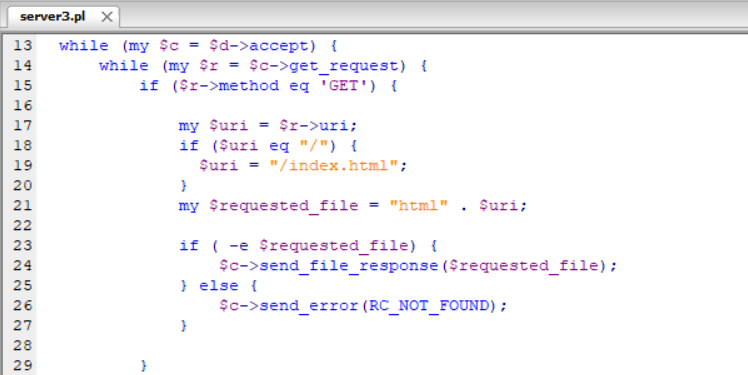
W celu rozszerzenia funkcjonalności serwera do obsługi prostej struktury serwisu WWW, stworzyłem folder o nazwie *html* w katalogu, gdzie znajduje się skrypt. Zawarłem w nim 3 podstrony składające się na mój serwis: index.html, p1.html, p2.html. Następnie w skrypcie server3.pl należy zmienić instrukcje obsługujące zapytania typu GET:



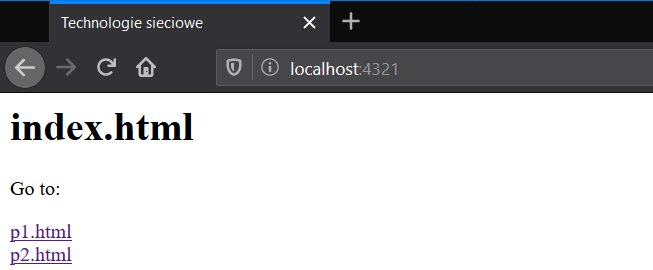
9. Struktura folderu serwera

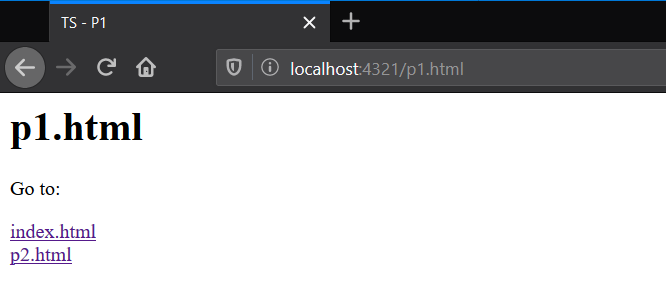


10. Nowy kod index.html zawierający linki do pozostałych podstron



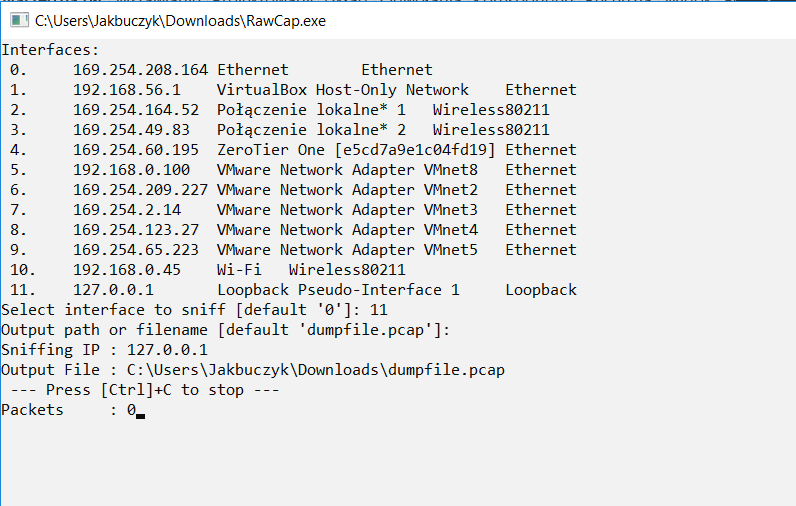
11. Zmodyfikowany fragment server.pl





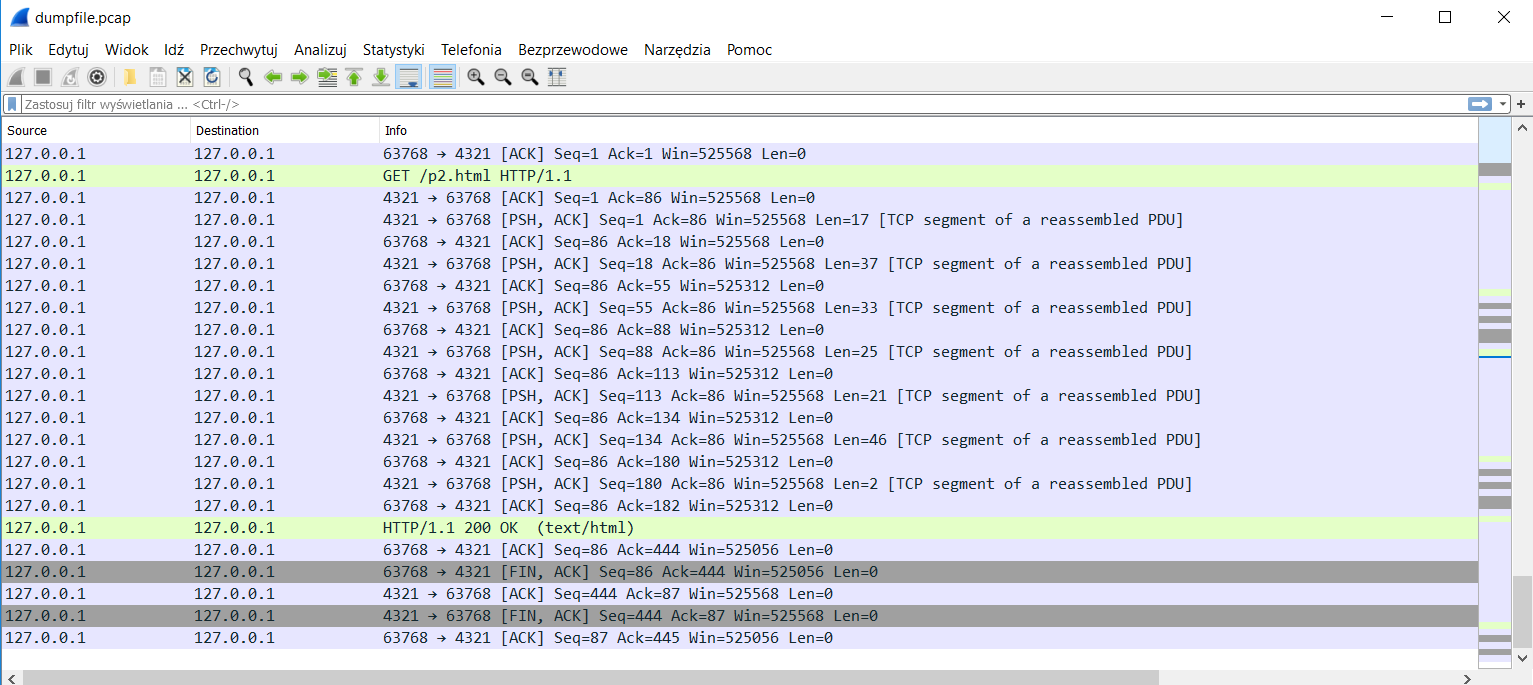
12. Serwis WWW w przeglądarce internetowej

Do przechwytywania komunikatów do/od serwera posłużę się programem **RawCap**:



13. Włączenie przechwytywania komunikatów w programie RawCap

Historia przechwytywania komunikatów zapisze się w pilku dumpfile.pcap, który otworzę za pomocą programu **Wireshark**:



14. Analiza przechwyconych komunikatów w programie Wireshark

Na załączonym obrazku widać komunikaty związane z żądaniem wysłania strony *p2.html* przez klienta i pomyślnym jego przetworzeniem przez serwer. Pakiety z flagą ACK są używane do weryfikacji wysyłki i odbioru. Flaga FIN informuje o zakończeniu połączenia – po otrzymaniu pakietu z flagą FIN, program czeka na ostatni pakiet ACK i kończy połączenie. Flaga PSH wymusza określony priorytet przetworzenia pakietu.