POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY

Instytut Elektrotechniki Teoretycznej
i Systemów Informacyjno-Pomiarowych
Zakład Elektrotechniki Teoretycznej
i Informatyki Stosowanej

PRACA DYPLOMOWA INŻYNIERSKA

na kierunku Informatyka
w specjalności: Inżynieria oprogramowania

Wykorzystanie protokołu HTTP/2 do budowy szybkiej aplikacji internetowej

Piotr Szklanko
nr albumu 244145

promotor
mgr inż. Bartosz Chaber

Warszawa, 2017

Wykorzystanie protokołu HTTP/2.0 do budowy szybkiej aplikacji internetowej

Streszczenie

Praca składa się z krótkiego wstępu jasno i wyczerpująco opisującego oraz uzasadniającego cel pracy, trzech rozdziałów (2-4) zawierających opis istniejących podobnych rozwiązań, komponentów rozpatrywanychjako kandydaci do tworzonego systemu i wreszcie zagadnień wydajności wirtualnych rozwiązań. Piąty rozdział to opis środowiska obejmujący opis konfiguracji środowiska oraz przykładowe ćwiczenia laboratoryjne. Ostatni rozdział pracy to opis możliwości dalszego rozwoju projektu.

Słowa kluczowe: praca dyplomowa, LaTeX, jakość

THESIS TITLE

Abstract

This thesis presents a novel way of using a novel algorithm to solve complex problems of filter design. In the first chapter the fundamentals of filter design are presented. The second chapter describes an original algorithm invented by the authors. Is is based on evolution strategy, but uses an original method of filter description similar to artificial neural network. In the third chapter the implementation of the algorithm in C programming language is presented. The fifth chapter contains results of tests which prove high efficiency and enormous accuracy of the program. Finally some posibilities of further development of the invented algoriths are proposed.

Keywords: thesis, LaTeX, quality

Warszawa, 1 lutego 2017

POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY

OŚWIADCZENIE

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa inżynierska pt. Wykorzystanie protokołu HTTP/2 do budowy szybkiej aplikacji internetowej:

- została napisana przeze mnie samodzielnie,
- nie narusza niczyich praw autorskich,
- nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam, że przedłożona do obrony praca dyplomowa nie była wcześniej podstawą postępowania związanego z uzyskaniem dyplomu lub tytułu zawodowego w uczelni wyższej. Jestem świadom, że praca zawiera również rezultaty stanowiące własności intelektualne Politechniki Warszawskiej, które nie mogą być udostępniane innym osobom i instytucjom bez zgody Władz Wydziału Elektrycznego.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Piotr Szklanko.....

Spis treści

Rozdział 1

Wstęp

Moim celem jest przeprowadzenie testów protokołu HTTP w najnowszej wersji 2.0. Obecnie powszechnie stosowana jest wersja 1.1, która została wprowadzona w roku 1999. Jednakże szybki rozwój technologii internetowych sprawia, że wprowadzony osiemnaście lat temu protokół przestaje powoli spełniać swoje zadanie. Obecnie bez wykorzystania serwerów CDN czy cache przeglądarki oraz innych sposobów nie jest możliwe stworzenie płynnie działającej strony internetowej. Za pomocą własnoręcznie stworzonej aplikacji chcę przekonać się, czy wprowadzone funkcje faktycznie mają tak ogromny wpływ na szybkość komunikacji pomiędzy klientem i serwerem.

Swoją aplikację stworzyłem wykorzystując zestaw oprogramowania MEAN – MongoDB, Express.js, Angular i Node.js.

- MongoDB – baza danych NoSQL (cos o mongoose? dodać dokumentację do wszystkich punktów),
- Express.js – framework Node.js do tworzenia aplikacji sieciowych od strony serwera,
- Angular – framework JavaScript służący do budowy dynamicznej aplikacji internetowej od strony użytkownika,
- Node.js – środowisko uruchumieniowe języka JavaScript, które pozwala wystartować serwer.

Zdecydowałem się na to rozwiązanie z kilku powodów:

- po przejrzeniu dostępnych w sieci informacji doszedłem do wniosku, że implementacja protokołu HTTP/2 jest najlepiej opisana oraz wspierana przez środowisko związane z JavaScriptem,
- dobra znajomość języka JavaScript oraz jednoczesna chęć rozwoju umiejętności tworzenia aplikacji w tym języku,

- chęć poszerzenia wiedzy dotyczącej budowania aplikacji internetowych za pomocą technologii javascriptowych,
- nie ukrywam, że znaczący wpływ na moją decyzję miała również popularność języka JavaScript na rynku pracy.

Rozdział 2

HTTP/2

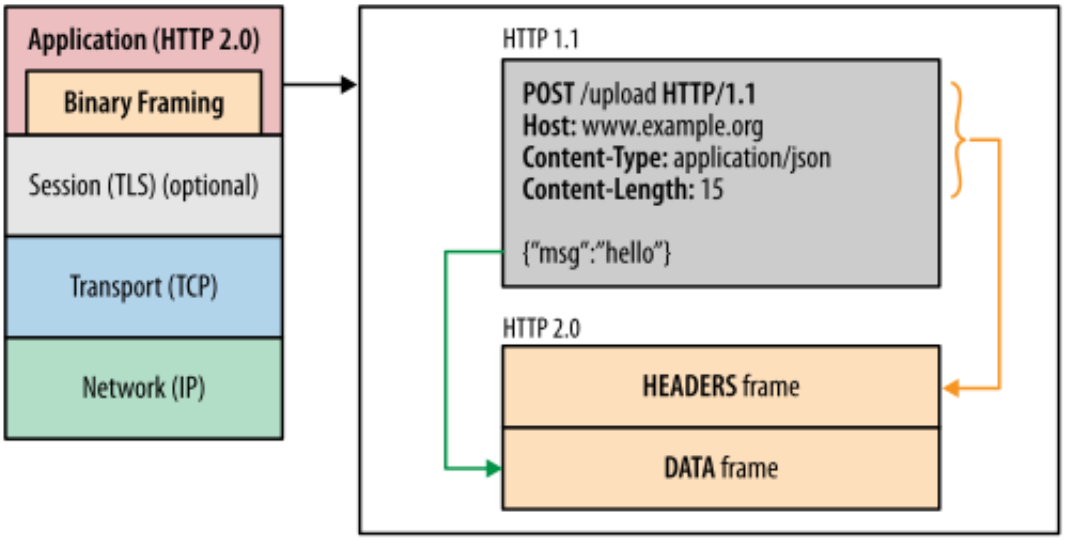
2.1 Historia

Pracę nad zmianami w protokole zapoczątkowała w 2009 roku firma Google ze swoim projektem SPDY. Zdecydowali się oni na stworzenie protokołu, który miał usprawnić działanie aplikacji oraz stron internetowych rozwiązując ograniczenia nałożone przez HTTP/1.1. Z biegiem czasu coraz więcej przeglądarek oraz stron internetowych, zarówno tych dużych jak i tych małych, zaczęło wspierać SPDY, co zainteresowało osoby pracujące nad protokołem HTTP. Zdecydowali się oni wykorzystać dokumentację protokołu SPDY jako początek prac nad własnym protokołem – HTTP/2. Od tego momentu aż do roku 2015, kiedy to standard HTTP/2 został oficjalnie zaakceptowany (**TODO**odnośnik RFC 7540 i może 7541), projekty były rozwijane równolegle. SPDY było wykorzystywane do testów nowych funkcjonalności, które miały zostać wprowadzone do nowego protokołu HTTP. Niedługo po oficjalnym zaakceptowaniu HTTP/2 ogłoszono, że SPDY nie będzie dalej wspierane.

W kilku poniższych akapitach postaram się przybliżyć zmiany, które zostały wprowadzone do protokołu HTTP.

2.2 Protokół binarny

Kluczową zmianą, która determinuje brak wstecznej kompatybilności z HTTP/1.1, jest przejście na kodowanie binarne przesyłanych wiadomości. Przykładowa ramka widoczna jest na rysunku 2.1 Jest to rozwiązanie dużo bardziej kompaktowe i łatwiejsze w implementacji, niż przesyłanie zwykłego tekstu. Dzięki temu zabiegowi w ramach jednego połączenia TCP z serwerem może zostać utworzonych wiele dwukierunkowych strumieni danych przesy-



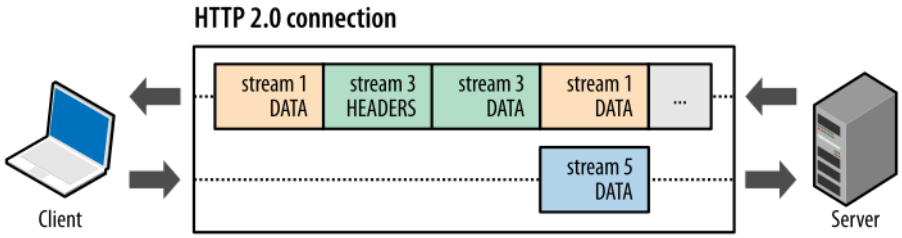
Rysunek 2.1: Schemat ramki protokołu HTTP/2

łających wiadomości HTTP. Taka wiadomość to w rzeczywistości zapytanie od klienta lub odpowiedź serwera składające się z ramek. Każda ramka natomiast musi przynajmniej posiadać nagłówek z informacją, do którego strumienia danych należy. Kodowanie binarne nie ma wpływu na składnie zawartości ramki – wszystkie nagłówki czy zapytania HTTP/1.1 pozostawiono bez zmian.

2.3 Multiplexing

W poprzedniej wersji protokołu, pomimo, że istniała możliwość przesyłania wielu zapytań w ramach jednego połączenia, nie można było wykonywać ich równolegle. Każde zapytanie musiało być rozpatrywane i odesłane przez serwer do klienta zgodnie z kolejnością nadania, co powodowało efekt HOL (head-of-line blocking ODNOSNIK DO JAKIEGOŚ ŹRÓDŁA?). Aby wykonywać zapytania równolegle należało utworzyć kilka zapytań TCP, co obciąża serwer oraz jest czasochłonne. Protokuł HTTP/2 umożlwia przesyłanie oraz odbieranie wielu wiadomości jednocześnie, co pokazuje schemat na rysunku 2.2. Są one rozbijane na pojedyncze ramki, przesyłane, a następnie odczytywane i składane z powrotem w całość po stronie odbiorcy. Dzięki temu nie jest już konieczne uciekanie się do takich zabiegów jak:

- scalanie plików (WEBPACK),



Rysunek 2.2: Schemat wykorzystania multiplexingu w HTTP/2

- wykorzystywanie spritów,
- domain sharding (DOCZYTAC).

To wszystko sprawia, że aplikacje stają się szybsze oraz prostsze.

2.4 Prioretyzacja

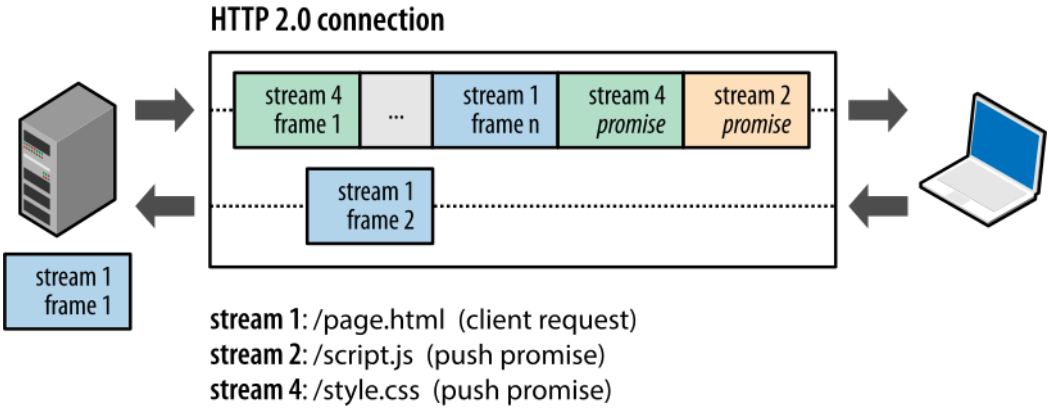
2.5 Server push

Wykorzystując protokół HTTP/1.1 nie mamy możliwości otrzymania zasobu, o który nie poprosiliśmy wysyłając zapytanie. Powoduje to opóźnienia na przykład podczas ładowania strony internetowej. Zanim otrzymamy skrypty czy arkusze styli, które wykorzystuje nasza strona musi ona o nie poprosić. Zapytanie do serwera wysyłane jest gdy w kodzie pliku html napotkamy na taki kod (przykład z mojego projektu):

```
<!-- CSS -->
<link rel="stylesheet"
        href="libs/bootstrap/dist/css/bootstrap.min.css">
<link rel="stylesheet"
      href="libs/font-awesome/css/font-awesome.min.css">

<!-- JS -->
<script src="libs/angular/angular.min.js"></script>
```

Takie rozwiązanie, chociaż w wielu przypadkach jest pożądane, tutaj jedynie spowalnia działanie aplikacji. Jeżeli mamy pewność, że użytkownik będzie potrzebował danych zasobów 2.3 możemy mu je od razu udostępnić, co zdecydowanie skraca czas ładowania aplikacji i dzięki temu unikam niechcianego efektu, gdy strona się załaduje, ale na przykład bez pliku zawierającego style, który jest dopiero przesyłany.



Rysunek 2.3: Schemat Server push HTTP/2

2.6 Kompresja nagłówków

Rozdział 3

Budowa aplikacji

3.1 Struktura aplikacji

3.2 Aplikacja od strony serwera

Rozdział 4

Testy

4.1 Chrome DevTools

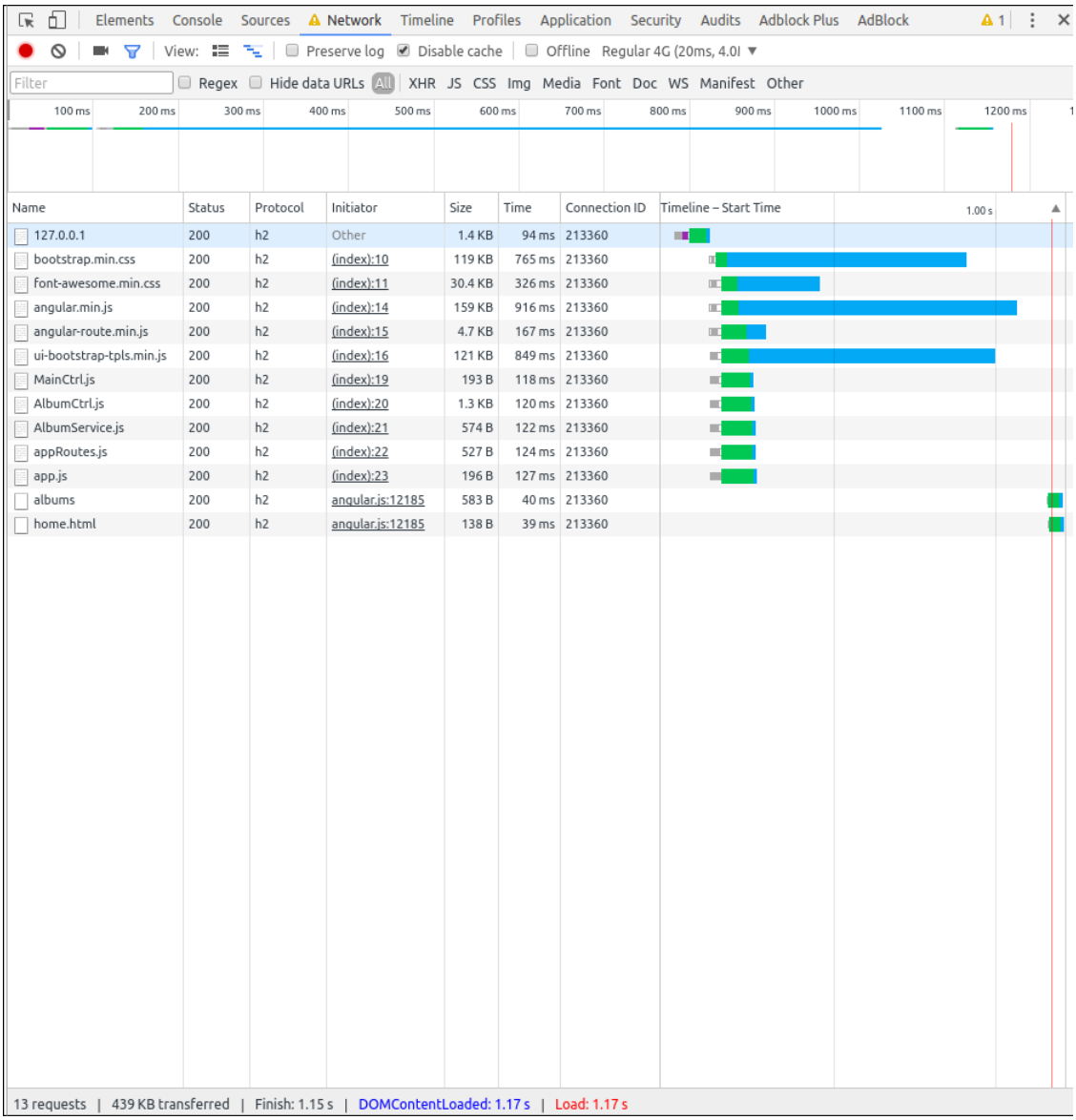
Do pomiaru prędkości oraz uzyskania innych ważnych informacji wykorzystałem narzędzie Chrome DevTools. Opiszę tutaj po krótce co i jak mierzyłem za pomocą tego oprogramowania.

Po uruchomieniu konsoli przeglądarki przechodzimy do zakładki Network i naszym oczom ukazuje się okno jak na rysunku 4.1.

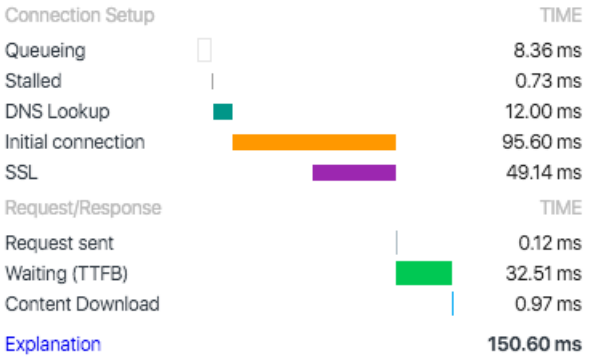
Widoczne okno składa się z pięciu głównych elementów:

1. paska kontroli – umożliwia on między innymi edycję wyglądu panelu sieciowego,
2. paska filtrów – pozwala na stworzenie reguł i wybór tylko tych pakietów, które nas interesują,
3. paska przeglądu – ukazuje nam oś czasu, która daje nam obraz tego, jak przesyłane były pakiety danych,
4. tabeli zapytań – zawiera szczegółowe informacje na temat każdego zapytania
5. podsumowania – zawiera informacje o łącznej liczbie zapytań, przesłanych danych oraz czasie trwania.

W moich badaniach najczęściej korzystałem z informacji zawartych w tabeli zapytań, a szczególnie z przedstawionej w niej osi czasu. Po najechaniu kursorem na którykolwiek pasek na osi otrzymujemy szczegółowe informacje o czasie każdego z etapów zapytania jak na rysunku 4.2.



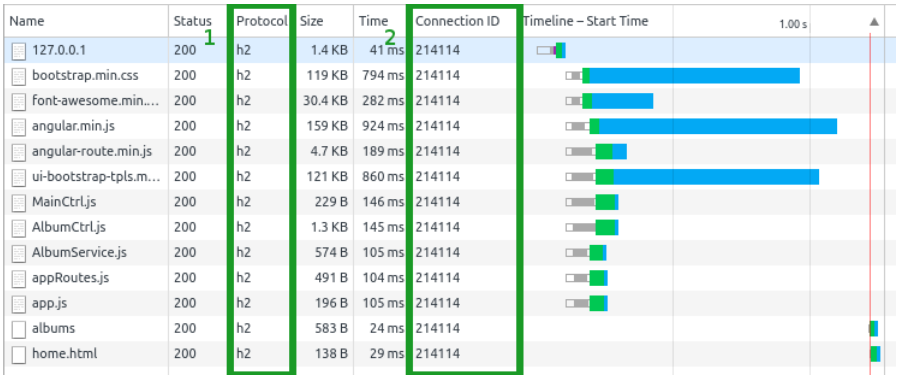
Rysunek 4.1: Wygląd okna Chrome DevTools



Rysunek 4.2: Szczegółowe informacje na temat czasu zapytania

4.2 Testy wstępne

Przed przejściem do porównania prędkości działania obu wersji protokołu chciałem zaprezentować pierwsze efekty implementacji protokołu HTTP/2, które przedstawia rysunek 4.3.



Rysunek 4.3: Dowód działania protokołu HTTP/2

Widzimy tutaj dwie rzeczy, które powinny nas zainteresować. W sekcji 'Protocol' oznaczonej na rysunku 4.3 numerem 1 widnieje napis h2 przy każdym zapytaniu. Jest to informacja, że do komunikacji z serwerem wykorzystana została najnowsza wersja protokołu HTTP. Dodatkowo w sekcji 'Connection ID' (na rysunku 4.3 jest to numer 2) widzimy, że wszystkie zapytania zostały wykonane z wykorzystaniem tego samego połączenia TCP. Nie mogłoby to mieć miejsca, gdybyśmy wykorzystali HTTP/1.1, co pokazuje rysunek ref.

Rozdział 5

Wnioski

Bibliografia

[1] W. R. Stevens, G. R. Wright, „Biblia TCP/IP tom 1", RM, 1998.

[2] U. S. Department Of Defense, „Trusted Computer System Evaluation Criteria", 1985.

[3] B. W. Lampson, „A note on the confinment problem", w „Proc. of the Communications of the ACM", październik 1973, numer 16:10, strony 613-615.

[4] G. J. Simmons, „The prisoners' problem and the subliminal channel", w „Advances in Cryptology: Proceedings of Crypto 83 (D. Chaum, ed.)", strony 51-67, Plenum Press, 1984.

[5] A. Kerckhoffs, „La Cryptographie Militaire (Military Cryptography)", J. Sciences Militaires, luty 1883.

[6] A. Havill, „The Spy Who Stayed Out In The Cold: The Secret Life of Double Agent Robert Hanssen", St. Martin's Press, 2001.

[7] C.Cachin, „An Information-Theoretic Model for Steganography", w „Information and Computation", 4 marzec 2004.

[8] S.Chauhan, „Embedding Covert Channels into TCP/IP", 7th Information Hiding Workshop, czerwiec 2005.

[9] Information Sciences Institute, University of Southern California, „Transmission Control Protocol", RFC793, wrzesień 1981.

[10] V. Jacobson, R. Braden, D. Borman, „TCP extensions for high performance", RFC1323, maj 1992.

[11] S. Bellovin, „Defending against sequence number attacks.", RFC1948, IETF, 1996.

[12] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson, „Stream Control Transmission Protocol", RFC2960, Network Working Group, październik 2000.

[13] C. H. Rowland, „Covert Channels in the TCP/IP Protocol Suite", First Monday, 1997.
http://www.firstmonday.dk/issues/issue2_5/rowland/

[14] Alhambra, daemon9, „Project Loki: ICMP Tunneling", Phrack Magazine, Issue 49. http://phrack.org

[15] daemon9, „LOKI2", Phrack Magazine, Issue 51. http://phrack.org

[16] van Hauser, Reverse WWW Shell, THC, The Hacker's Choice.
www.thc.org

[17] T. Sohn, J. Seo, J. Moon, „A Study on the Covert Channel Detection of TCP/IP Header Using Support Vector Machine", Volume 2836 of Lecture Notes in Computer Science., Springer-Verlag (2003) 313-324.

[18] T. Sohn, T. Noh, J. Moon, „Support Vector Machine Based ICMP Covert Channel Attack Detection", Volume 2836 of Lecture Notes in Computer Science., Springer-Verlag, 2003, strony 461-464.

[19] J. Giffin, R. Greenstadt, P. Litwack, R. Tibbetts, „Covert messaging in TCP", w Dingledine, Privacy Enhancing Technologies. Volume 2482 of Lecture Notes in Computer Science., Springer-Verlag (2002) 194-208.
http://www.mit.edu/~gif/covert-channel/

[20] G. Fisk, M. Fisk, Ch. Papadopoulos, J. Neil, „Eliminating Steganography in Internet Traffic with Active Wardens", 5th International Workshop on Information Hiding, październik 2002.

[21] J. Rutkowska, „The Implementation of Passive Covert Channels in Linux Kernel", Chaos Communication Congress, grudzień 2004.

[22] Ch. Benvenuti, „Understanding Linux Network Internals", O'Reilly, grudzień 2005.

[23] kossak, „Building Into The Linux Network Layer", Phrack Magazine, Issue 55. http://phrack.org

[24] Steven J.Murdoch and Stephen Lewis, „Embedding Covert Channels into TCP/IP", University of Cambridge, Computer Laboratory, 29 lipec 2005.

[25] Eugene Tumoian, Maxim Anikeev, „Detecting NUSHU Covert Channels Using Neural Networks", Taganrog State University of Radio Engineering, Department of Information Security.

[26] mayhem, „IA32 Advanced Function Hooking", Phrack Magazine, Issue 58. http://phrack.org

[27] bioforge, „Hacking the Linux Kernel Network Stack", Phrack Magazine, Issue 61. http://phrack.org

[28] Robert Love, „Kernel Korner - Allocating Memory in the Kernel", 1 grudzień 2003.