Kraków 15.06.2015

Analiza protokołu TCP w oparciu o sieci Petriego

Prowadzący
Jakub Porzycki

Skład grupy:
Anna Ślusarczyk
Mikołaj Nowak
Kacper Furmański
Michał Ślusarczyk

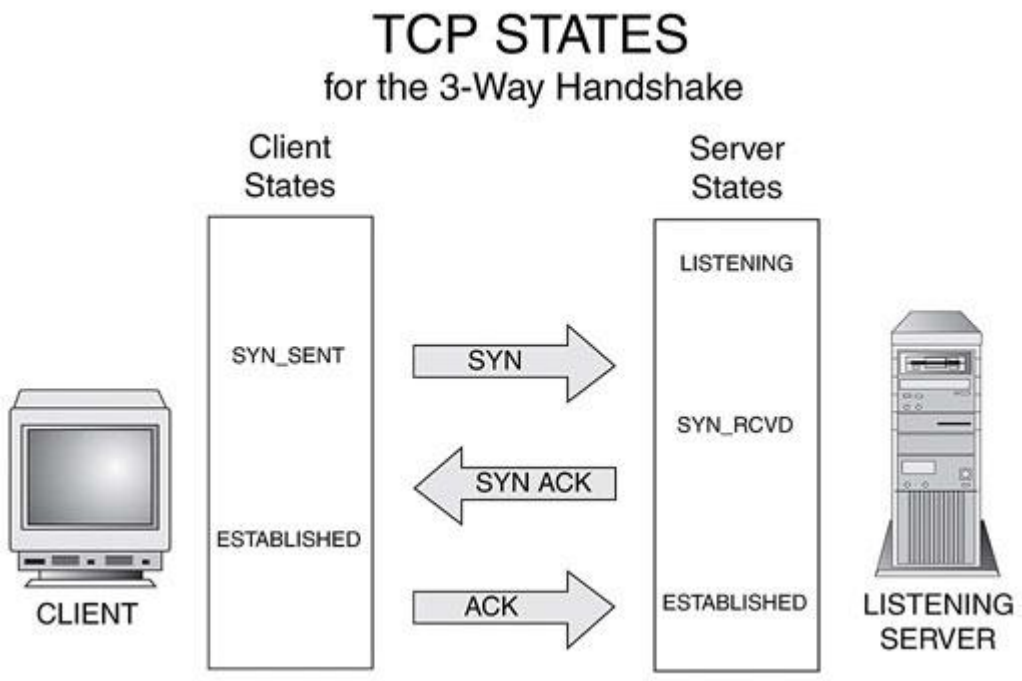
1. Opis systemu

1.1 Czym jest TCP?

Protokół TCP działa w czwartej warstwie modelu OSI wraz z protokołem UDP. Odpowiadają one za nawiązanie połączenia, a także przesył danych pomiędzy hostem, a serwerem. Dodatkowo nadzorują proces zamykania połączenia i powrotu komputera w stan początkowy (nasłuchu). Podstawową różnicą pomiędzy tymi protokołami jest kontrola czy dany pakiet został poprawnie dostarczony do adresata. W przypadku niedostarczenia stacja docelowa korzystająca z protokołu TCP może zlecić ponowne wysłanie informacji, natomiast w przypadku protokołu UDP nie następuje żadna kontrola poprawności odebranych danych.

1.2 Jak działa?

Host chcący nawiązać połączenie wysyła do serwera segment z ustawionym bitem SYN (synchronize – synchronizacja) i przechodzi w stan SYN_SENT. W tym momencie odbiorca czyli serwer jest informowany, że nadawca komunikatu chce nawiązać połączenie. Po odebraniu pakietu od hosta serwer zmienia swój stan na SYN_RECV i odsyła pakiet z ustawionymi bitami SYN oraz ACK (acknowledegement - potwierdzenie). Po otrzymaniu odpowiedzi od serwera host jeszcze raz wysyła pakiet z ustawionym bitem ACK, co pozwala odbiorcy zrozumieć, że jest on gotowy na odbiór danych. Razem z odpowiedzią host zaczyna przesyłać już pierwsze dane do urządzenia odbierającego. Po wymianie komunikatów synchronizujących oba urządzenia przechodzą w stan ESTABLISHED podczas którego następuje wymiana danych.

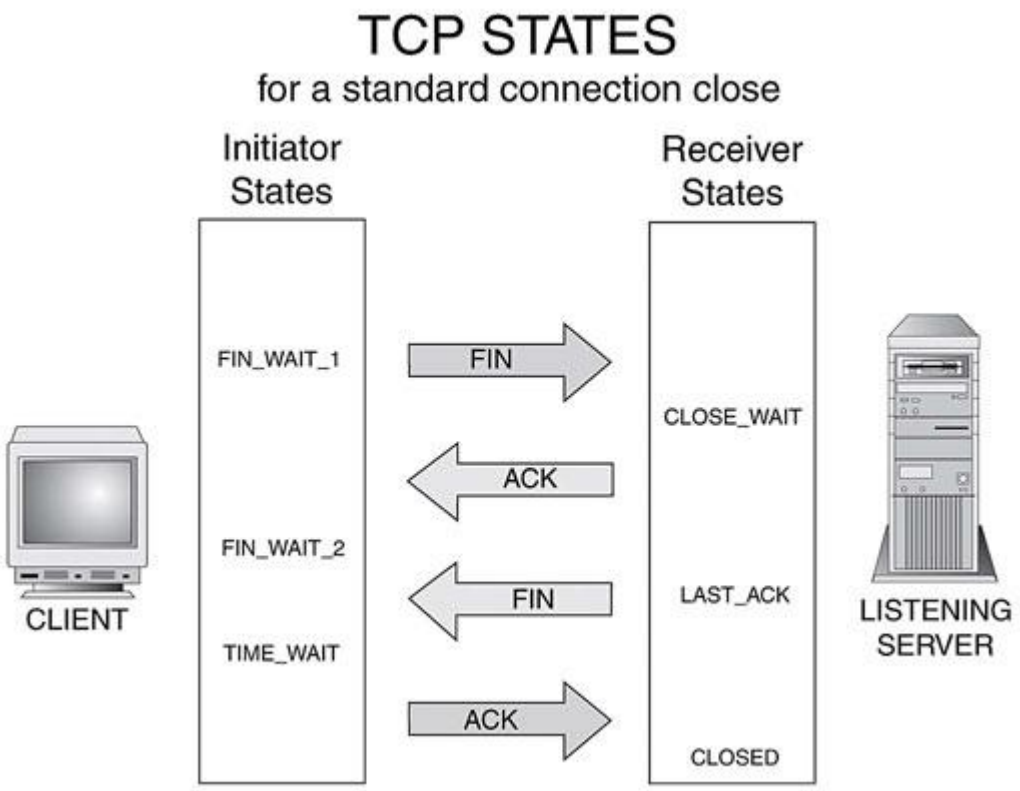


Źródło: http://www.itmanage.info/technology/linux/statefull_firewall/Concept_of_State.htm

Analogicznie przebiega proces zamykania połączenia. Urządzenie chcące zakończyć wymianę danych wysyła pakiet z ustawioną flagą FIN (finish – koniec) i przechodzi w stan FIN-WAIT-1. W stanie tym możliwe jest jeszcze odbieranie danych, jednakże zablokowana jest

możliwość wysyłania. Odbiorca po otrzymaniu informacji o chęci zakończenia połączenia zmienia swój stan CLOSE_WAIT i odsyła segment z ustawionymi bitami FIN oraz ACK.

Urządzenie inicjujące po odebraniu FIN oraz ACK od serwera przestawia się w stan FIN-WAIT-2 i przesyła swój segment z ustawionym ACK, a następnie zmienia stan na TIME_WAIT w celu upewnienia się ze odbiorca otrzymał komunikat. W stanie tym urządzenie może trwać maksymalnie 4 minuty. Po odebraniu komunikatów z obu stron połączenie może zostać bezpiecznie zamknięte.

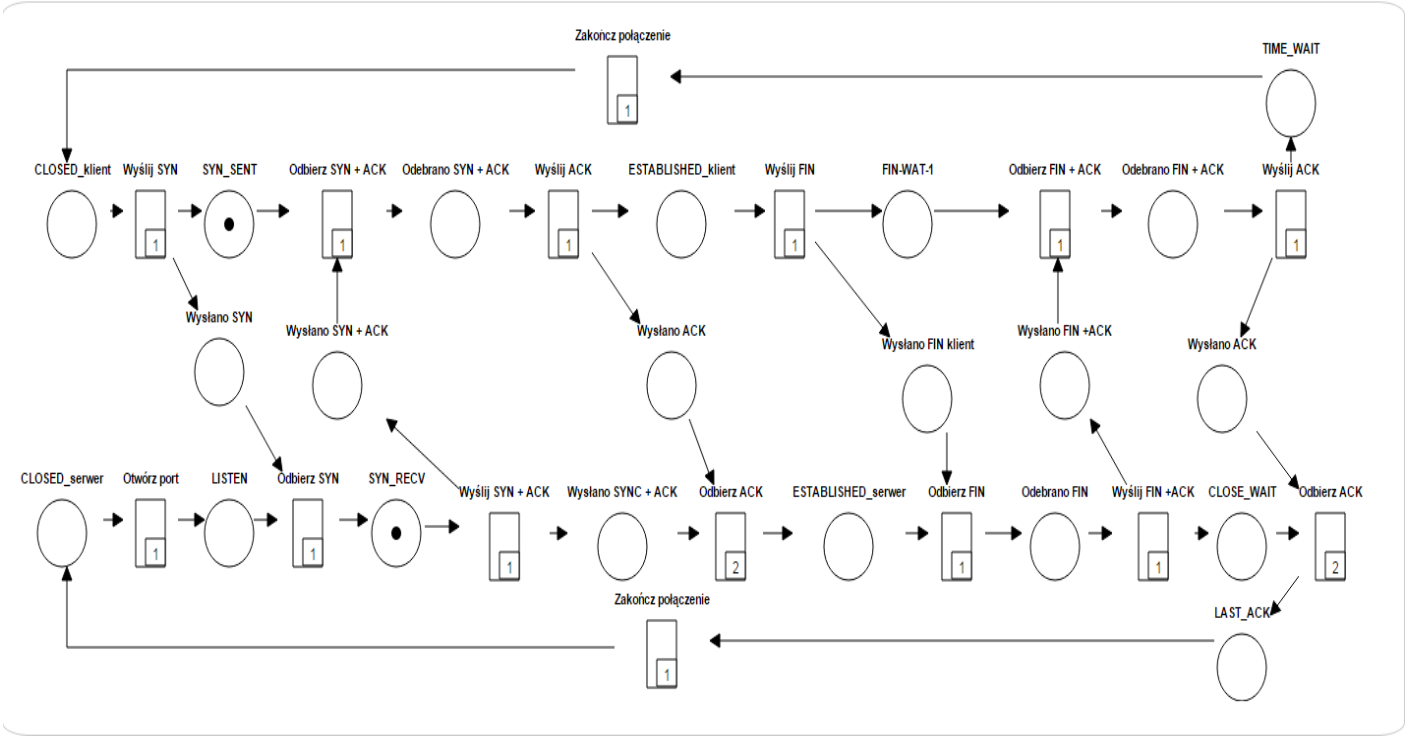


Źródło: http://www.itmanage.info/technology/linux/statefull_firewall/Concept_of_State.htm

2. Model systemu

2.1 Sieć Petriego

Sieć Petriego modelująca zachowanie wyżej opisanego protokołu po opracowaniu przyjęła następujący wygląd:



2.1 Opis stanów i przejść

Każde miejsce przedstawia stan w jakim znajduje się urządzenie w danej chwili po wysłaniu/odebraniu komunikatu. Większość stanów została nazwana dokładnie tak jak w oficjalnej dokumentacji. Stanami dodatkowymi są stany pomocnicze, które modelują sytuacje, w której pakiet został wysłany i jest na etapie doręczania do adresata lub urządzenie odebrało pakiet. Zazwyczaj po takich stanach następuje akcja szczegółowo opisana i przedstawiona na powyższych rysunkach poglądowych i system kontynuuje swoje działanie.

Dodatkowo każde przejście zostało opisane w sposób jasny i nie budzący żadnych wątpliwości. W przypadkach gdzie nazwy mogły być podobne zarówno dla serwera jak i dla klienta został dodany sufiks „_klient” lub „_serwer” w celu doprecyzowania, które urządzenie podejmuje zamodelowaną akcję.

Sieć została tak zsynchronizowana, aby żaden stan ani przejście nie zostało odwiedzone ani wykonane w kolejności niezgodnej z działaniem algorytmu. Dzięki temu sieć osiąga stany TIME_WAIT oraz LAST_ACK w bardzo zbliżonym momencie wykonania symulacji.

3. Właściwości sieci

3.1 Zgodność z opisem protokołu

Sieć została zamodelowana zgodnie z opisem protokołu dostępnym na stronach CISCO, a także źródeł dostępnych w internecie. Nawiązywanie połączenia – algorytm three ways handshake – został odwzorowany krok po kroku z dostępnej dokumentacji. Wszystkie stany, jak wspomniano powyżej zostały przedstawione dokładnie pod nazwą dostępną w opracowaniach, wyjątkiem są tutaj jedynie stany przejściowe, w których model oczekuje na wykonanie jakiejś akcji. W tym przypadku stany te zostały nazwane w sposób umożliwiający śledzenie co dokładnie ma miejsce po stronie urządzenia.

3.2 Analiza pod kątem modelu

W celu poprawnego wykonanie modelu należało rozważyć wszystkich „uczestników" biorących czynny udział w działaniu protokołu TCP. Aby tego dokonać konieczne było przeanalizowanie dokumentacji i źródeł dostępnych w internecie. Po wstępnej analizie i zidentyfikowaniu aktorów i ich zachowań podczas pracy algorytmu kolejnym krokiem było takie ustalenie stanów i przejść, aby sieć powstała w wyniku modelu nie tylko była poprawna merytorycznie, ale także spełniała warunki grafu, który można nazwać terminem Sieć Petriego. W tym celu należało ustalić stany w których może znajdować się model, a także tranzycje wpływające na ilość i rozmieszczenie tokenów w grafie.

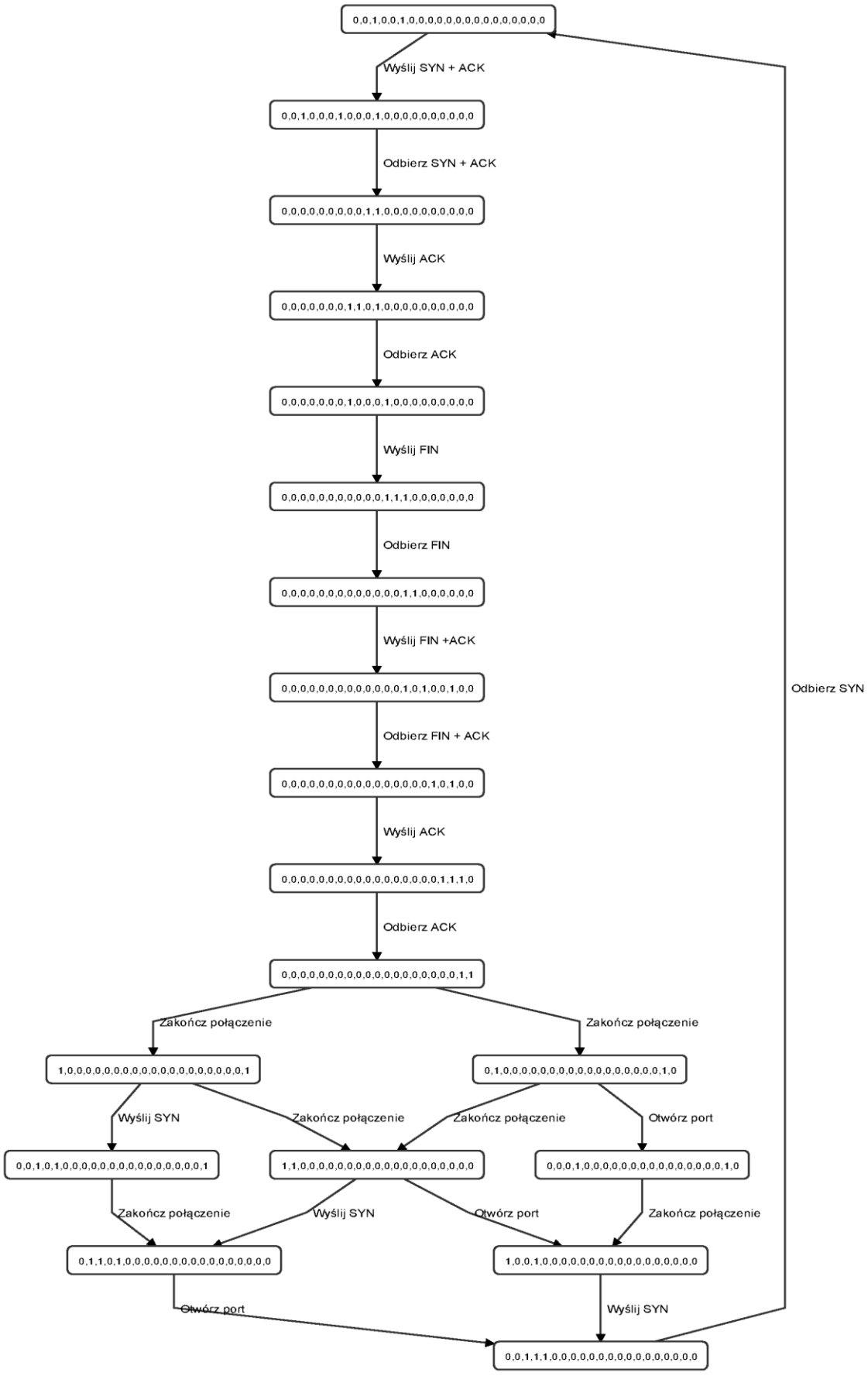
W przypadku naszej sieci, czyli grafu, w którym opisane jest zarówno zachowanie serwera jak i klienta założyliśmy, że stan początkowy sieci to sytuacja w której zarówno klient jak i serwer posiadają po jednym tokenie w momencie, gdy porty obu urządzeń są zamknięte. Następnie serwer otwiera swój port i oczekuje na sygnał od nadawcy, który chce rozpocząć wymianę danych. Po otwarciu portów nadawca wysyła pakiet synchronizacji i następuje nawiązywanie połączenie zgodnie ze scenariuszem opisanym w punkcie 1.2. Aby wykonać poprawnie wymianę komunikatów tokeny muszą „wędrować" pomiędzy maszynami, a równocześnie inicjować zmiany stanów po stronach samych urządzeń. W tych miejscach następuje rozdzielenie tokenów za pomocą tranzycji na dwie osobne ścieżki. Dodatkowo, aby spełnić wymagania Sieci Petriego, która nie dopuszcza sytuacji łączenia tranzycja-tranzycja oraz miejsce-miejsce wprowadziliśmy dodatkowe stany przejściowe, opisujące akcje takie jak „odebrano ACK" lub „wysłano FIN". Co więcej, aby zsynchronizować sieć, tak aby wszystkie akcje miały miejsce w odpowiednim momencie użyliśmy priorytetów ustawionych na dwóch tranzycjach (Odbierz ACK). Dzięki temu nie ma sytuacji, w której na przykład klient wysyła komunikat o zakończeniu połączenia w momencie gdy serwer nie przeszedł jeszcze w stan ESTABLISHED.

3.3 Poprawność sieci

W procesie tworzenia grafu braliśmy pod uwagę wszystkie właściwości jakie powinien on spełniać, aby móc zostać nazwany Siecią Petriego. Dzięki temu model otrzymany w rezultacie jest w pełni możliwy do analizy, co pozwala nam wyznaczyć wiele właściwości opisujących wyżej wspomnianą Sieć.

W modelu nie występuje sytuacja łączenia typu tranzycja-tranzycja, ani stan-stan. Sytuacja taka umożliwia przeprowadzenie pełnej symulacji działania protokołu krok po kroku, z możliwością przeanalizowania zachowania poszczególnych zachowań urządzeń w określonym czasie.

4. Graf pokrycia sieci



5. Analiza systemu

Po stworzeniu modelu i umieszczeniu go w naszej aplikacji symulującej działanie Sieci Petriego udało nam się przeprowadzić analizę poszczególnych właściwości. Jako pierwsze sprawdziliśmy ograniczenia miejsc w naszej sieci. Następnie korzystając z tablicy przechowującej ograniczenia danych miejsc sprawdziliśmy czy nasza sieć jest bezpieczna. Kolejnymi analizowanymi przez naszą aplikację własnościami były zachowawczość sieci oraz sprawdzenie czy nie występują w niej zakleszczenia. Wszystkie te własności zostały przedstawione w naszym programie.

Place boundedness:

Place name	Bound
CLOSED_klient	1
CLOSED_serwer	1
SYN_SENT	1
LISTEN	1
Wysłano SYN	1
SYN_RECV	1
Wysłano SYN + ACK	1
ESTABLISHED_klient	1
Wysłano ACK	1
Odebrano SYN + ACK	1
Wysłano SYNC + ACK	1
ESTABLISHED_serwer	1
Wysłano FIN klient	1
FIN-WAT-1	1
Odebrano FIN	1
Wysłano FIN +ACK	1
Odebrano FIN + ACK	1
Wysłano ACK	1
CLOSE_WAIT	1
TIME_WAIT	1
LAST_ACK	1

Net bound: 1

Net safe: true

Net conservative: false

Net dead lock free: true

Wyniki przeprowadzonej analizy w aplikacji.

Dodatkowo zostały wygenerowane macierze wejście oraz wyjścia, na podstawie których można obliczyć macierz incydencji.

N+

0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

N-

0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

N

$$\begin{bmatrix}
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1
\end{bmatrix}$$

Po pełnej analizie problemu stwierdziliśmy, że sieć została zamodelowana w sposób prawidłowy. Fakt, że sieć jest siecią bezpieczną gwarantują brak sytuacji, w której znaczniki mnożyłyby się w nieskończoność. Sytuacja taka byłaby równoznaczna z błędem popełnionym przy modelowaniu tranzycji w sieci.

Dodatkowo sieć nie jest zachowawcza, co także potwierdza poprawność modelu, ponieważ tokeny oznaczające zasób lub akcje wykonywaną przez urządzenia muszą wędrować nie tylko wzdłuż linii działania serwera lub hosta, ale także krążyć pomiędzy dwoma komputerami gwarantując komunikację i przesył danych.

Kolejną własnością jest brak zakleszczeń. Według naszej analizy nie ma możliwości zaistnienia sytuacji, w której sieć osiąga taki stan, że niemożliwe jest wykonanie którejś tranzycji. Sytuacja taka jasno wskazywałaby na to, że podczas procesu planowania sieci rozminęliśmy się z opisem działania protokołu TCP.

6. Podsumowanie i wnioski

Naszym zadaniem było zamodelowanie oraz przeanalizowanie właściwości protokołu TCP za pomocą aplikacji symulującej działanie Sieci Petriego. Po zapoznaniu się z dokumentacją opisującą działanie protokołu udało nam się stworzyć model, a następnie przeprowadzić symulację działania protokołu za pomocą Sieci Petriego. W modelu wyraźnie zaznaczono podział na cześć nadawcy-hosta, oraz adresata komunikatów czyli serwera. Dodatkowo dzięki dodaniu stanów przejściowych można zaobserwować w jakim stanie znajduje się poszczególne urządzenie w zależności od fazy połączenia.

Podczas pracy nad aplikacją oraz modelem mieliśmy szansę na zapoznanie się bliżej ze sposobem działania jednego z najważniejszych protokołów w modelu OSI, oraz dowiedzieć się krok po kroku jak odbywa się nawiązywanie oraz zrywanie połączenia. Dzięki podwójnym potwierdzeniom przesyłanym przez obie maszyny protokół ten jest idealnym rozwiązaniem przy komunikacji sieciowej, w której ważne jest, aby całość informacji została dostarczona do adresata.

Wydaje nam się że dzięki jasnej dokumentacji i wyraźnie zaznaczonym krokom oraz zadaniom, które musi wykonać każda z maszyn protokół TCP jest bardzo dobrym materiałem na analizę za pomocą Sieci Petriego. Co więcej nazwy poszczególnych stanów w dużej mierze ułatwiły modelowanie problemu oraz stworzenie sieci która w bardzo dużym stopniu pokrywa się z rzeczywistym działaniem protokołu.