CopperAnt

sprawozdanie

Symulacja komputerowa
Teleinformatyka
Grupa 1

1. Analiza problemu

Realizując projekt postawiono sobie za zadanie podjęcie próby wykonania symulacji sieci komputerowej złożonej z wielu stacji roboczych podzielonych na segmenty i połączonych za pomocą odpowiednich urządzeń m.in przełączniki oraz routery wraz z ich konfiguracją umożliwiającą poprawne działanie zgodne z podstawowymi standardami. Po zapoznaniu się z istotą zagadnienia oraz przeprowadzeniu analiz wymagań, które zostały przedstawione w dalszej części rozdziału podjęto decyzję o implementacji systemu symulującego uproszczony model warstwy ISO/OSI oraz dokonano wyboru rodzaju dziedziny zdarzeń sterujących w projekcie. Postanowiono, że program zostanie zaimplementowany w oparciu o dziedzinę zdarzeń dyskretnych, co pozwoli na sprawdzenie możliwości stworzenia i zasymulowania działania sieci komputerowej w niej wraz z zapoznaniem się ze sposobem korzystania z takiego modelu.

Zastosowany został uproszczony model ISO/OSI, gdyż celem systemu nie jest dokładne odwzorowanie tego modelu, lecz zapoznanie się z możliwymi metodami realizacji symulacji. Wymagało to, oprócz zadbania o odpowiednie skonfigurowanie urządzeń, także przeanalizowania występujących w sieci zdarzeń oraz odpowiednie ich zasymulowanie, przez co możliwe stało się dogłębne zrozumienie mechanizmów działania sieci komputerowej.

2. Opis problemu

2.1. Opis wymagań

- system posiada graficzny interfejs umożliwiający tworzenie oraz zarządzanie urządzeniami sieciowymi m.in przez łączenie portów, nadawanie adresów IP
- system umożliwia logowanie wydarzeń zachodzących w symulacji, zbierane są informacje o poszczególnych urządzeniach, jak i o całości symulacji
- system generuje ruch sieciowy będący odpowiedzią na zdarzenia inicjowane tylko i wyłącznie przez komputery
- ruch w sieci ma odbywać się zgodnie ze standardami modelu ISO/OSI oraz zasadami świata rzeczywistego, tzn. występują opóźnienia na kolejnych urządzeniach przez które przechodzi pakiet, występują kolizje które należy obsłużyć
- system umożliwia zestawienie połączeń i wygenerowanie ruchu w sieci tylko i wyłącznie pomiędzy urządzeniami poprawnie skonfigurowanymi
- system odpowiednio reaguje na błędną konfigurację sieci np. wprowadzenie dwóch identycznych adresów MAC

2.2. Model symulacji

2.2.1. Porównanie istniejących modeli symulacji

Przed przystąpieniem do pracy dokonano analizy dwóch modeli symulacji:

- w dziedzinie czasu (process oriented),
- w dziedzinie zdarzeń dyskretnych (event oriented).

Poniższa grafika przedstawia porównanie tych dwóch koncepcji.

Rys. 1 Porównanie modeli symulacji: w dziedzinie czasu (górna ilustracja) oraz w dziedzinie zdarzeń dyskretnych (dolna ilustracja)

Na powyższej ilustracji numery w kratkach symbolizują czas. Ramki białe to takie w trakcie których nie dzieje się nic istotnego, istotnego z punktu widzenia symulowanego systemu. Ramki szare symbolizują moment zawierający pewne zdarzenie. W pierwszym przypadku zegar symulacji działa w równoodległych odstępach - niezależnie czy w symulacji zachodzi coś istotnego. Drugie rozwiązanie wyposażone jest w nieregularnie działający zegar. Kolejne obserwowane momenty zawsze zawierają przynajmniej jedno zdarzenie istotne dla symulowanego procesu.

Ze względu na fakt, że wybór modelu symulacji narzuca stosowany model programowania rozpatrzono następujące cechy:

	Domena czasu	Domena zdarzeń dyskretnych
Wielowątkowość	Konieczne zastosowanie wielowątkowości. Należy dużo uwagi przyłożyć do mechanizmów synchronizacji.	Wielowątkowość nie jest wymagana ponieważ zdania ustawiane są w kolejce.
Symulacja nieregularnych zdarzeń	Złe rozwiązanie - jeżeli zdarzenia występują bardzo rzadko w czasie symulacja musi wykonać bardzo dużo bezczynnych cykli.	Dobre rozwiązanie. Symulacje zawierające zdarzenia rzadko rozłożone w czasie mogę być przeprowadzone szybko bez utraty

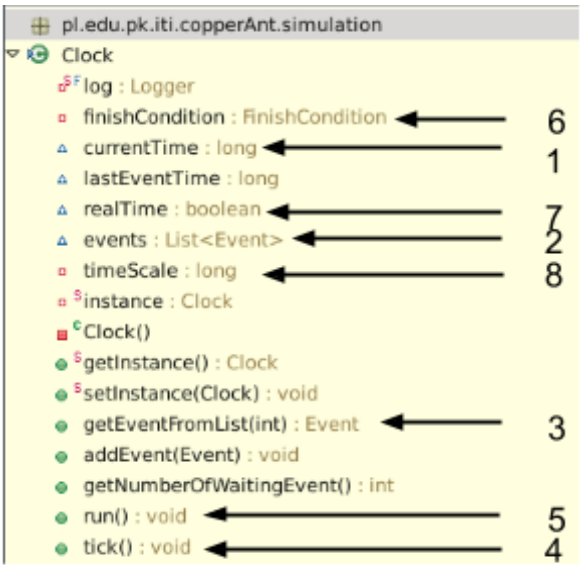
	Rozwiązaniem może być przeskalowanie czasu. Jest do dobra decyzja tylko, jeśli nie zależy nam na dużej dokładności w obszarach zawierających zdarzenia.	dokładności.
Przyjazne do obserwacji przez człowieka	Tak, jeżeli symulacja jest prezentowana użytkownikowi w czasie działania może on wygodnie obserwować jej przebieg	Nie - zdarzenia prezentowane użytkownikowi w nieproporcjonalnych odstępach czasu mogą być nieintuicyjne. W celu prezentacji danych w trakcie symulacji tego typu należy zastosować dodatkowe mechanizmy spowalniające symulacje co zaburza istotę tego rozwiązania.
Analiza przyczynowości	Trudniejsza - zdarzenia dzieją się równocześnie w wielu wątkach przez co trudniej jest przeanalizować z czego wynika uzyskany wynik	Łatwiejsza - każde zdarzenie które istnieje w symulacji znalazło się tam ponieważ zostało dodane przez inne zdarzenie. Analiza takie łańcucha zdarzeń może być pomocna przy analizie.

Tabela 1. Porównanie modeli symulacji

Mając na uwadze powyższe cechy zdecydowano się wybrać model sterowany zdarzeniami. Dodatkowym argumentem za wybraniem tego modelu był fakt, że dla grupy projektowej było to rozwiązanie nowe, nie prezentowane wcześnie na innych zajęciach.

2.2.2. Opis klasy Clock

Sercem symulacji w dziedzinie zdarzeń dyskretnych jest zegar symulacji tutaj reprezentowany przez klasę Clock. Poniżej przedstawiono wycinek pochodzący ze środowiska programistycznego Eclipse ukazujący kluczowe elementy tej klasy. W dalszej części zostaną opisane kluczowe cechy zastosowanego rozwiązania.



Rys 2. Kluczowe elementy klasy Clock

ad 1. Klasa przechowuje licznik reprezentujący dany moment czasowy. Ten licznik nie jest zwiększany o jeden, lecz zmienia wartość skokowo przyjmując wartość zapisane w kolejnych zdarzeniach w listy.

ad 2. Zdarzenia (obiekty typu Event) są umieszczane w liście - kolejce symulacji. W celu zapewnienia wydajności działania zdarzenia są sortowane według przypisanego im czasu zajścia w momencie dodawania, tak aby nie było konieczności przeglądania całej listy przy każdym cyklu zegara. Ze względu na fakt, że w symulacji istnieje częsta potrzeba wstawiania elementów w środek struktury zastosowano implementacje listy opartą o dwukierunkową listę referencyjną (LinkedList ze standardowej biblioteki Javy).

ad 3. Istnieje możliwość pobrania dowolnego zdarzenia z listy (przed jego egzekucją).

ad 4. W pojedynczym cykl zegara pobierane jest jedno zdarzenie z listy i następuje jego wykonanie.

ad 5. Metoda run wykonuje cykle zegara tak długo aż nie zostanie spełniony warunek końca symulacji.

ad 6. Klasa Clock posiada pole finishCondition reprezentujące warunek stopu. Mogą być wykorzystane różne warunki kończące symulację.

ad 7. Ze względu na wadę symulacji w dziedzinie zdarzeń losowych polegającą na trudności obserwacji jej przebiegu w czasie rzeczywistym wprowadzono opcjonalną możliwość zastosowania opóźnień. Gdy zegar symulacji jest przestawiony na tryb realTime proces symulacji będzie zamrażany przed wykonaniem każdego zadania na czas proporcjonalny do odstępu czasu jaki minął od ostatniego zdarzenia.

ad 8. Jeżeli tryb realTime jest włączony do dyspozycji użytkownika zegara jest skalowanie czasu zgodnie z preferencjami. Domyślna skala to 10 ms na każdą jednostkę czasu z licznika zegara.

2.2.3. Generatory zdarzeń losowych

W symulacji prawie wszystkie zdarzenia pojawiają się w zegarze symulacji w momencie wykonania innych zdarzeń (zdarzenia reprezentujące przyczynę dodają zdarzenia reprezentujące skutek).

Jedynym wyjątkiem od tej reguły są zdarzenia reprezentujące moment wysłania przez komputer pakietu (klasa ComputerInitializeTrafficEvent). Te zdarzenia pojawiają się co pewien czas. Może on być dobrany w różny sposób w zależności o zastosowanej implementacji interfejsu TimeIntervalGenerator. Istnieje możliwość skorzystania ze:

- zdarzeń pojawiających się co stały odstęp czasu,
- zdarzeń pojawiających się co losowy odstęp czasu. Do dyspozycji są rozkłady wykorzystujące bibliotekę Colt dostarczaną przez CERN poprzez repozytoria Maven[1]. Rozkłady te to:
    - poissonowski,
    - exponencjalny,
    - normalny.

[1] https://dst.lbl.gov/ACSSoftware/colt/

2.2.4. Możliwe warunki zakończenia symulacji

W symulacji istnieje możliwość definiowana różnych warunków zakończenia programu. W trakcie testowania aplikacji skorzystano z dwóch podstawowych:

- wyczerpanie zdarzeń w kolejce zegara,
- osiągniecie przez licznik zegara pewnej zadanej wartości.

Powyższe warunki były wystarczające dla prostych scenariuszy testowych. Pozostawiono jednak możliwość rozwinięcia warunków stopu na dowolne. W przypadku takiej potrzeby można dopisać warunek stopu czekający na zaistnienie pewnej sytuacji w symulacji (np. przekroczenie pewnego progu obciążenia na zadanym urządzeniu)

2.3. Model sieci i typy zdarzeń

W symulacji zbudowano model sieci będący uproszczeniem modelu warstwowego ISO/OSI. Poniżej opisano wszystkie elementy symulacji reprezentujące jej zachowanie:

2.3.1. Pakiety

Dla uproszczenia problemu zdecydowano się na porzucenie standardowego formatu ramek oraz koncepcji enkapsulacji. W symulacji istnieje jeden rodzaj pakietu niosący w sobie wszystkie dane potrzebne w trakcie transmisji.

Pakiety posiadają atrybut rozmiar, który może być wykorzystywany do szacowania czasu opóźnień. Zdecydowano nie wprowadzać segmentacji dużych pakietów (nie istnieje pojęcie MTU[2]).

Pakiety posiadają pole TTL[3] - wartość dekrementowaną przy każdym przejściu przez router. Dzięki temu, zapobiega się nieskończonemu krążeniu pakietów w sieci (zjawisko szczególnie widoczne przy niewłaściwej konfiguracji routerów)

Warto zauważyć, że w całej symulacji wykorzystano adresacje w formie znanej z IPv4. Zdecydowano ustalić stały rozmiar maski podsieci na 24 bity bez możliwości definiowania podsieci o innym rozmiarze.

Pakiety nie posiadają reprezentacji graficznej.

2.3.2. Kable

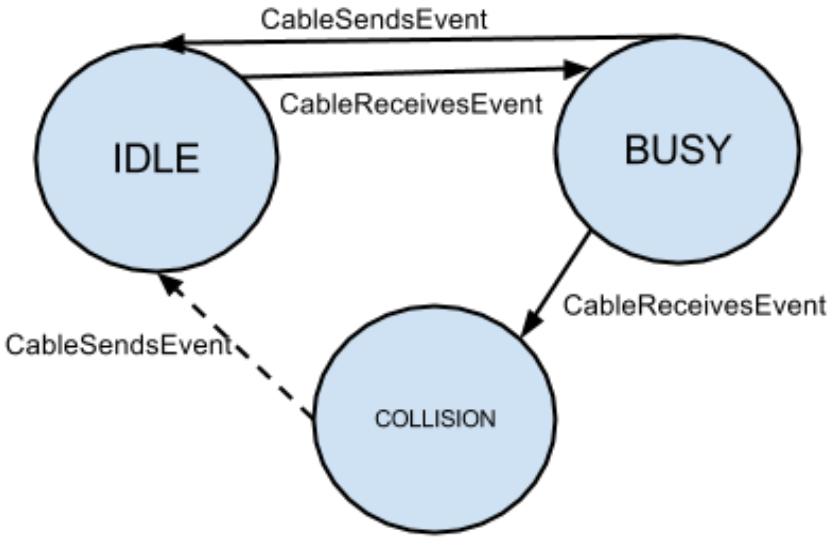
Kable reprezentowane są przez klasę Cable. Nie modyfikują one zawartości pakietów (przyjęto, że nie występują przekłamania w trakcie transmisji). Kable w modelu symulacji pełnią dwie funkcje:

[2] Maximal Transfer Unit
[3] Time To Life

- mogą nadawać opóźnienie między wysłaniem, a odebraniem pakietu - zależne od długości kabla i wielkości pakietu,
- kable mogę znajdować się w jednym z trzech stanów:
    - bezczynny (idle),
    - zajęty (busy),
    - kolizja (collision).

Dzięki temu możliwe jest symulowanie zdarzeń w których pakiety są gubione poprzez wystąpienie kolizji.

Kable zmieniają stan zgodnie z poniższym schematem



Rys.3 Zmiana stanu kabla

Jedna z krawędzi została oznaczona przerywaną linią ponieważ, aby nastąpiła taka zmiana stanu kabla konieczne jest spełnienie dodatkowych warunków związanych z czasem, który upłynął od wystąpienia kolizji.

2.3.3. Porty

Port jest elementem występującym w każdym urządzeniu sieciowym. Dzięki wielokrotnemu wykorzystaniu tego komponentu obsługa przesyłania

pakietów od i do kabla jest taka sama niezależnie od typu sprzętu sieciowego. Port odpowiada drugiej warstwie modelu ISO i działa tak samo niezależnie od warstwy w której operuje urządzenie sieciowe. Port posiada swój adres MAC wykorzystywany do komunikacji w obszarze jednej sieci.

Rys. 4 Symbol reprezentujący port

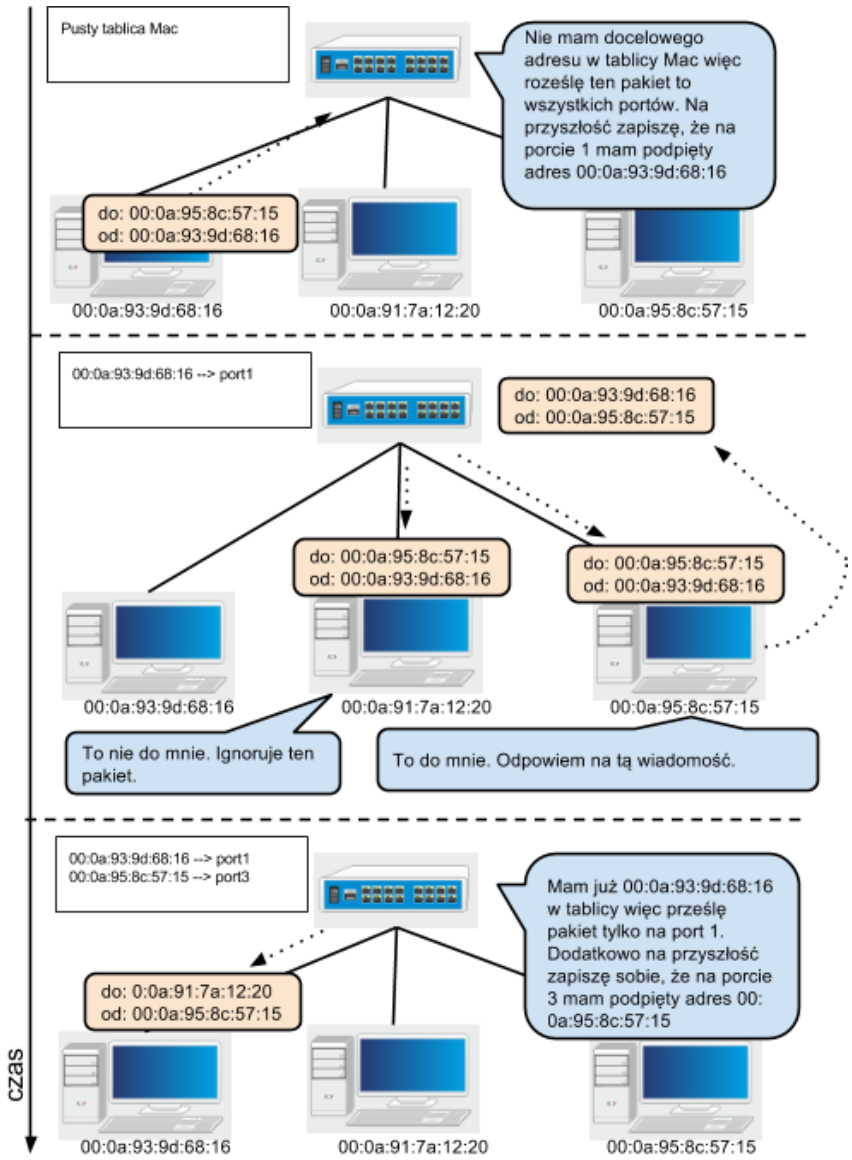
2.3.4. Hub

Hub jest najprostszym rozwiązaniem w symulacji wykorzystywanym do łączenia wielu urządzeń w obszarze jednej sieci. Działa w warstwie drugiej. Algorytm jego działania jest bardzo prosty: każdy otrzymany pakiet zostaje wysłany na wszystkie porty urządzenia. Jest to rozwiązanie praktycznie nie stosowane w obecnych sieciach komputerowych jednak został uwzględniony w symulacji dla celów porównawczych.

2.3.5. Switch

Switch działa jako Hub z udoskonalonym algorytmem przekazywania pakietów. Także działa w drugiej warstwie modelu ISO/OSI. Posiada wewnętrzną tablicę w której przechowuje informacje o tym, na który port powinny być kierowane pakiety do odpowiedniego adresata. Switch uczy się tych informacji w trakcie działania symulacji. Początkowo, gdy nie ma żadnych informacji o sieci zachowuje się jak Hub.

Rys. 5 Symbol reprezentujący Switch



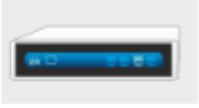
Rys. 6 Schemat działania switcha

2.3.6. Router

Router działa w trzeciej warstwie modelu ISO/OSI. Odpowiada za komunikację pomiędzy różnymi sieciami. Analizuje adresy IP i przekierowuje je na odpowiedni port zgodnie ze swoją tablicą routingu.

Router przy przekazywaniu pakietu modyfikuje jego dane sterujące: nadpisuje adresy MAC oraz dekrementuje pole TTL.

Dodatkowo w naszej symulacji routerom nadano role serwerów DHCP.



Rys. 7 Symbol reprezentujący Router

2.3.7. PC

Komputery w symulacji jako jedyne pełnią rolę urządzeń inicjujących ruch. Wszystkie pakiety krążące w sieci są konsekwencją ruchu zapoczątkowanego przez komputery. Poza nadawaniem, odbieraniem i odpowiadaniem na pakiety komputery nie pełnią żadnej dodatkowej roli.

2.3.8. Uproszczone odpowiedniki protokołów sieciowych

W symulacji zastosowano uproszczone protokoły znane z prawdziwych rozwiązań. Ich nazwy odnoszące się do istniejących protokołów symbolizują ich funkcję w sieci, jednak implementacja nie została w żaden sposób powiązana z oficjalnymi standardami.

2.3.8.1. ICMP (PING)

W aplikacji do symulowania ruchu w sieci używane jest wysyłanie tak zwanego żądania odpowiedzi (ang. ping). Każde urządzenie może wysłać żądanie odpowiedzi do dowolnego innego. Odbiorca jest zobowiązany odpowiedzieć na nie.

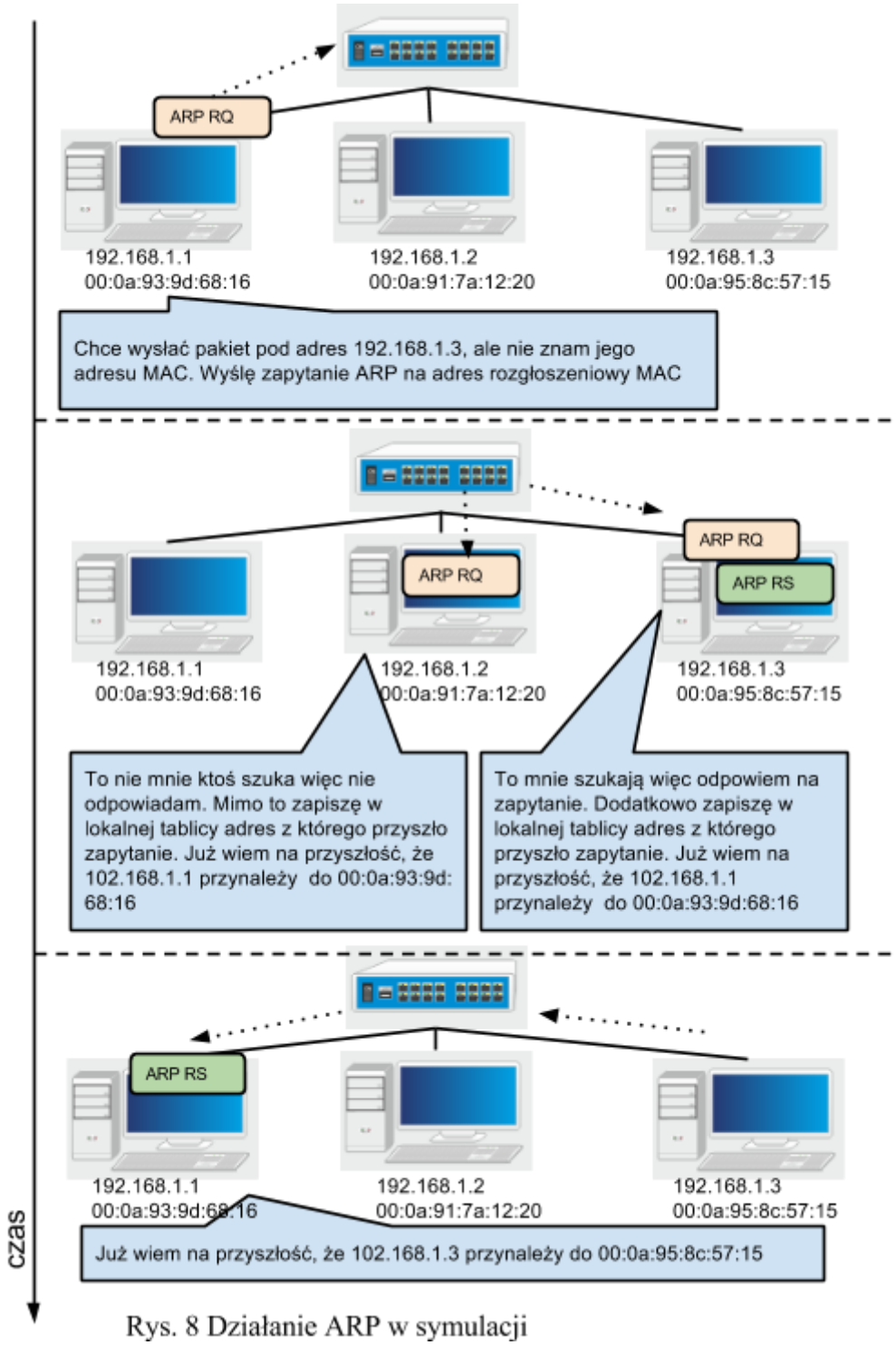
2.3.8.2. ARP

W symulacji użyto bardzo uproszczonej wersji protokołu ARP. W tej wersji ARP posiada dwa typy pakietów:

- żądanie adresu IP,
- odpowiedź na żądanie adresu IP.

W pakiecie z żądaniem przesyłany jest adres MAC jaki ma być przetłumaczony na IP, a w odpowiedzi zostaje wysłany odpowiadający temu adresowi MAC adres IP.

Poniżej przestawiono schemat działania protokołu APR na przykładzie prostej sieci (z pominięciem logiki działania przełącznika).



Rys. 8 Działanie ARP w symulacji

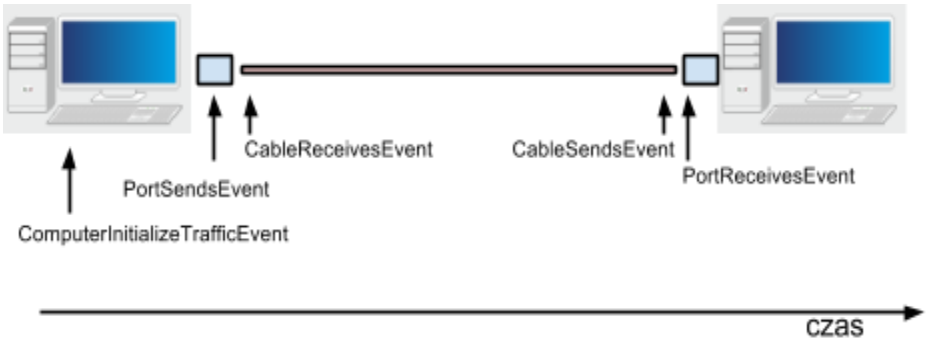
2.3.8.3. DHCP

Celem protokołu DHCP jest dynamiczne przydzielanie adresów IP urządzeniom. Jest to rozwiązanie opcjonalne w niniejszej symulacji, istnieje możliwość statycznego przypisywania adresów.

Dla uproszczenia przyjęto, że serwer usługi DHCP jest zaimplementowany zawsze na routerze.

2.3.8.4. Sekwencja zdarzeń występujących w symulacji

W symulacji zdarzenia są inicjowane w odpowiedzi na wystąpienie innych zdarzeń, z pominięciem zdarzenia inicjalizacji ruchu. Poniżej przestawiono sposób w jaki następuje kaskada zdarzeń symulująca transmisje pakietu pomiędzy dwoma komputerami połączonymi bezpośrednio.

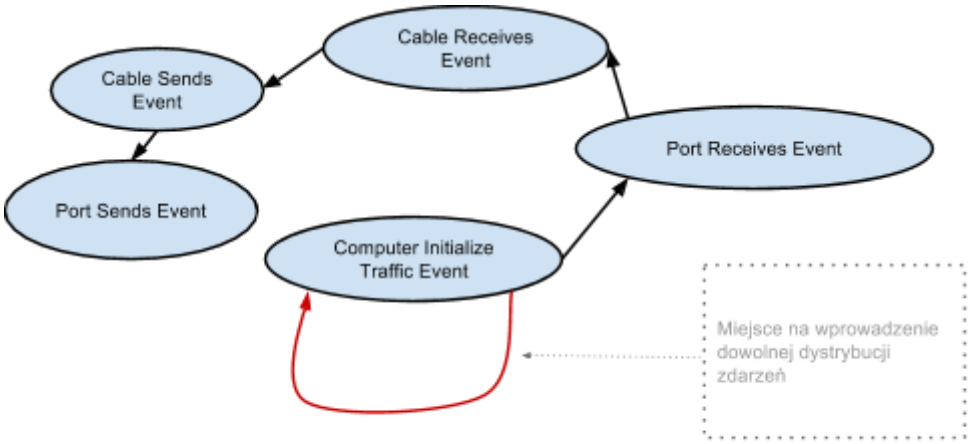


Rys. 9 Zdarzeń występujących pomiędzy 2 komputerami

Zdecydowano się na rozdzielenie procesu transmisja na zdarzenia tego typu, aby mieć kontrolę nad różnymi etapami procesu przesyłania danych. Dzięki temu możliwe jest wprowadzenie:

- opóźnień (np. na kablu),
- prób retransmisji (np. w przypadku wystąpienia kolizji),
- zagubienia pakietów (np. w przypadku błędnej konfiguracji komputera).

Poniżej przedstawiono graf prezentujący zależności między zdarzeniami. Krawędź skierowaną należy rozumieć jako relacje - “powoduje”.



Rys. 10 Graf prezentujący zależność między zdarzeniami

Szczególnie istotna jest tutaj krawędź zaznaczona na czerwono. Zdarzenie inicjalizacji ruchu jako jedyne nie jest powodowane przez zdarzenia innego typu. Tylko od zastosowanej implementacji zależy z jakim rozkładem będą pojawiały się nowe pakiety w sieci. Cały pozostały ruch jest konsekwencją tego typu zdarzeń.

3. Użyte technologie i metodyka pracy

3.1. Java 8

Do budowy projektu wykorzystano język programowania Java w wersji 8 - najnowszej oficjalnej. Zdecydowano się na wybór tej technologii ze względu na poniższe cechy:

- Jest to popularny język i większość członków grupy projektowej miała już z nim styczność.
- Ze względu na specyfikę działania maszyny wirtualnej istnieje znacznie mniejsze ryzyko niewłaściwego zarządzania pamięcią (w tym tzw. “wycieków”)
- Projekt nie wymagał cech będących atutem języków niższego poziomu (takich jak C/C++) - wysokiej wydajności, pełnej kontroli nad sprzętem.

3.2. JavaFx

Do graficznego przestawienia symulacji zastosowano bibliotekę JavaFX. Jest to nowe rozwiązanie - oficjalnie wydane przez firmę Oracle wraz z publikacją ósmej wersji Javy.

3.3. Maven

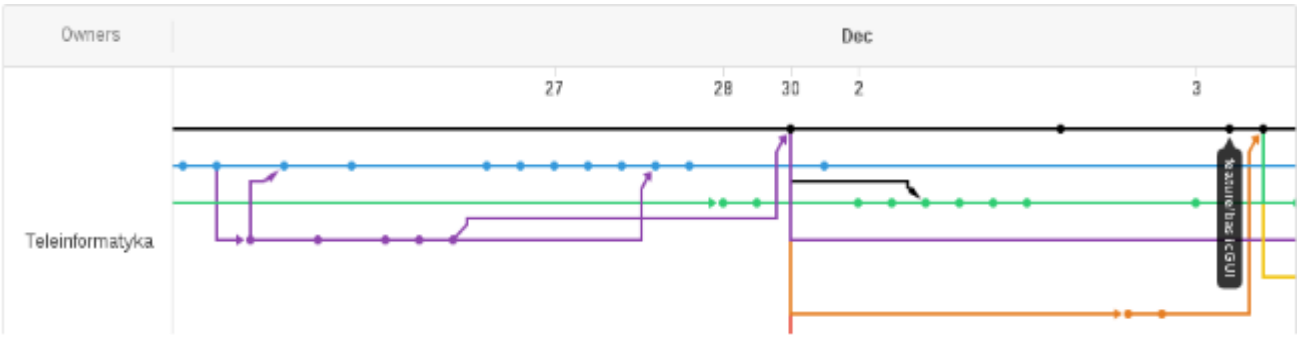
Narzędzie Maven odpowiedzialne jest za szereg czynności związanych z wytwarzaniem oprogramowania. W projekcie wykorzystano je do dwóch celów:

- Zarządzanie zależnościami aplikacji - dzięki niemu można bez wielkiego nakładu pracy użyć dodatkowy bibliotek dostarczanych przez innych autorów.
- Zunifikowany sposób testowania aplikacji - dzięki niemu po wszystkich zmianach w kodzie uruchamiane były wszystkie istniejące testy w celu uniknięcia wprowadzenia błędów do systemu.

3.4. Github

Projekt był tworzony w kilkunastoosobowej grupie, z tego powodu by ułatwić sobie pracę zdecydowano się na przetrzymywanie kodu źródłowego we wspólnym repozytorium systemu kontroli wersji. W tym celu wykorzystano jeden z najpopularniejszych obecnie serwisów tego typu - GitHub.

Zastosowano metodykę pracy opartą na gałęziach - każda nowa funkcjonalność rozwijana była w osobnej kopii projektu. Po zakończeniu pracy nad nowymi elementami zmiany były poddawane przeglądowi (ang. code review), a następnie włączane do głównej gałęzi. Poniżej przedstawiono wycinek schematu[4] reprezentującego prace tego typu.



Rys. 11 Sieć gałęzi projektu

3.5. Travis

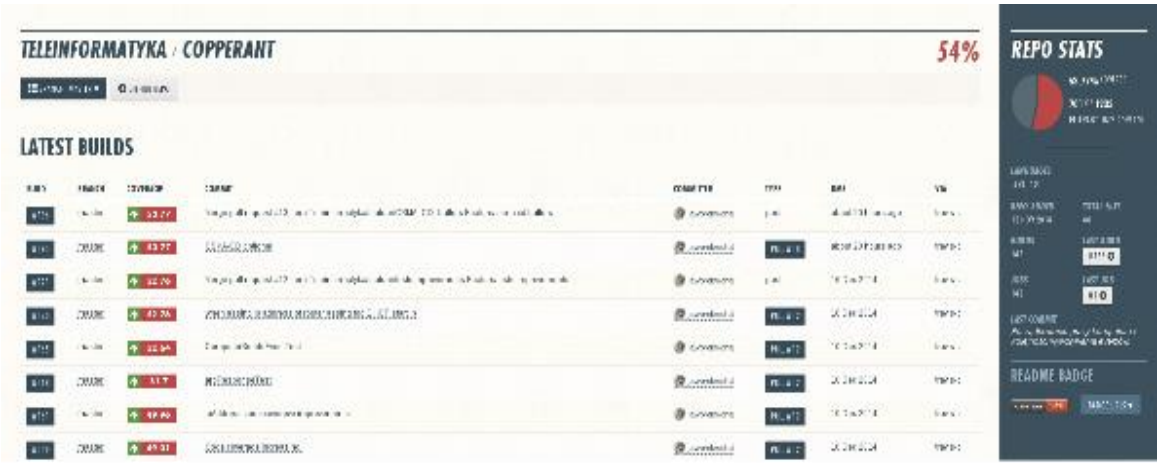
Dodatkowo zastosowano praktykę ciągłej integracji (ang. continuous integration). Polega ona na ciągłym, automatycznym testowaniu aplikacji po każdej zatwierdzonej zmianie. Dzięki temu, nie dopuszcza się do sytuacji w której kilku programistów rozwija

[4] Cały schemat dostępny pod adresem: https://github.com/Teleinformatyka/CopperAnt/network

projekt w różnych, niekompatybilnych kierunkach. Jako serwer ciągłej integracji została użyta usługa TravisCI[5].

3.6. Coveralls

Wraz z regularnym testowaniem aplikacji monitorowane było pokrycie kodu testami (ang. code coverage). Jest to metryka pomiaru jakości wytwarzanego oprogramowania mówiąca o tym jaka część linii programu jest wykonywana w trakcie przeprowadzania testów. W tym celu wykorzystano serwis coveralls.io[6]. Poniższej przestawiono fragment zrzutu ekranu zastosowanego narzędzia.

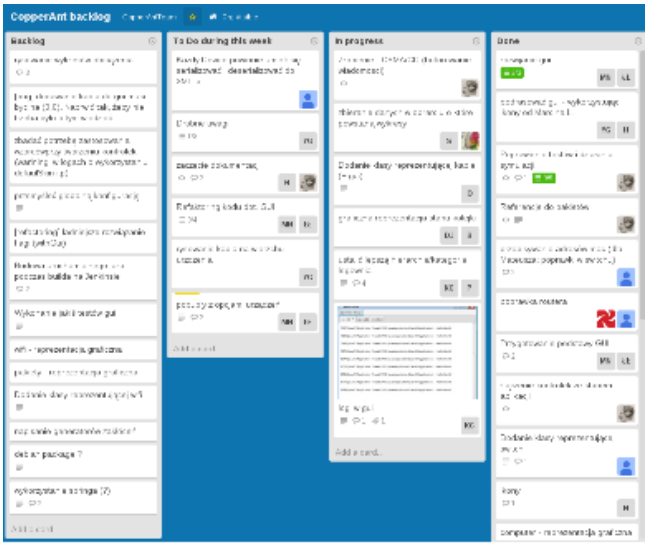


Rys. 12 Zrzut ekranu z serwisu coveralls.io

3.7. Trello

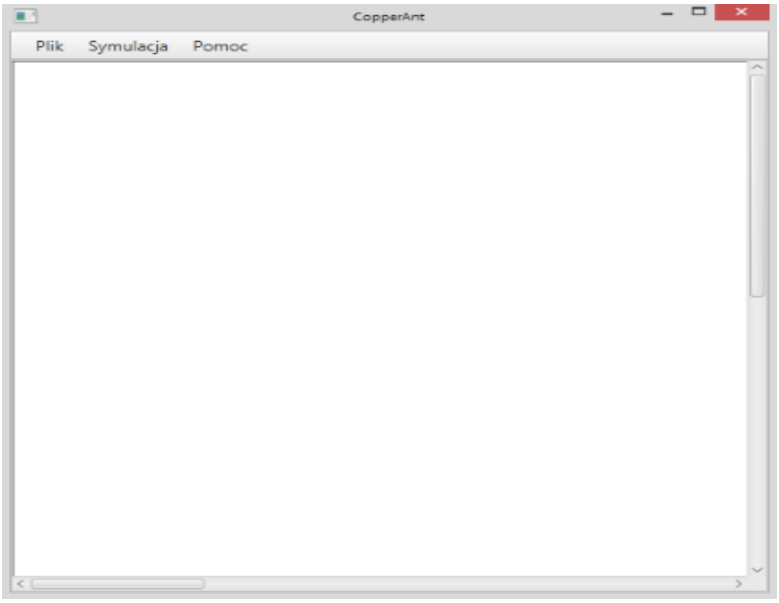
W celu synchronizacji pracy grupowej skorzystano z darmowego serwisu Trello.com. Wykorzystano go do rozdzielania zadań między uczestnikami projektu, komunikacji oraz kontroli postępów prac. Poniżej zamieszczono zrzut ekranu z wspomnianego narzędzia:

[5] Adres do projektu: https://travis-ci.org/Teleinformatyka/CopperAnt
[6] Adres do projektu: https://coveralls.io/r/Teleinformatyka/CopperAnt



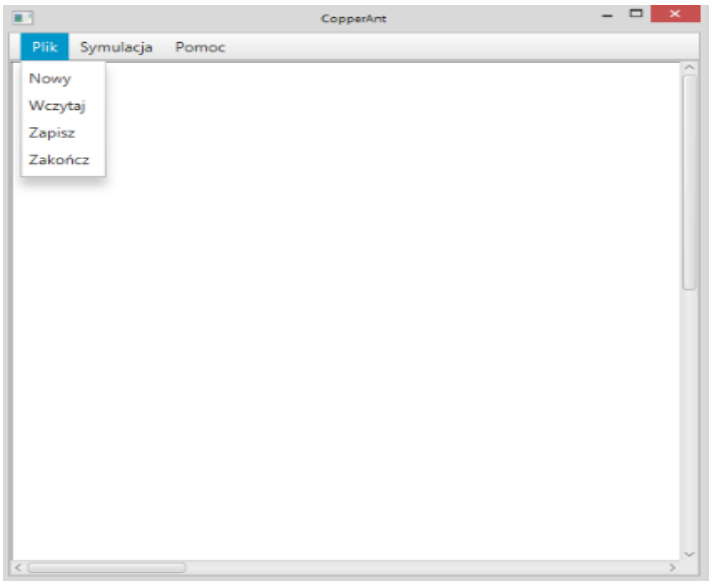
Rys. 13 Zrzut ekranu z serwisu trello.com

4. Konfiguracja, inicjalizacja, uruchomienie



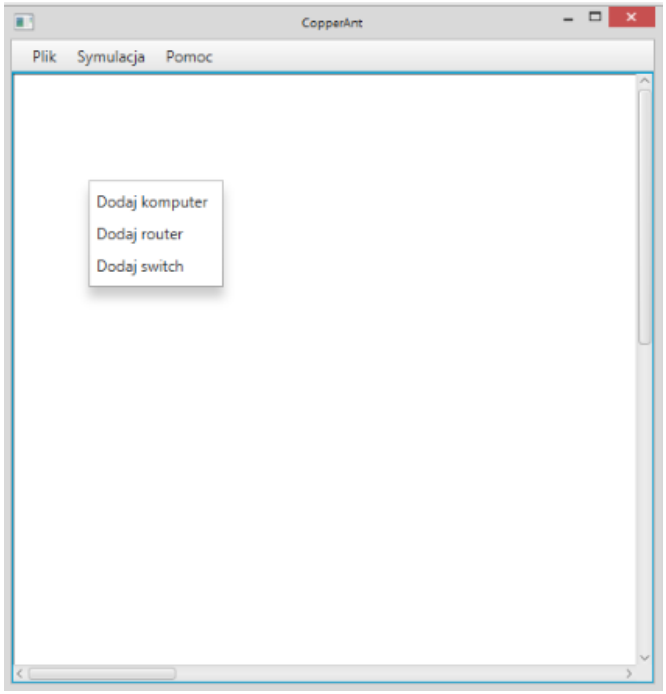
Rys. 14 Okno startowe aplikacji

Po uruchomieniu aplikacji wyświetlone zostaje okno startowe będące sceną dla tworzonej symulacji. Pierwszym sposobem na utworzenie nowej symulacji jest wczytanie przykładowej – predefiniowanej sieci przez wywołanie opcji Ładuj w menu rozwijanym Plik.



Rys. 15 Menu rozwijane Plik

Drugim sposobem jest ręczne dodanie poszczególnych urządzeń w sieci oraz następnie połączenie ich w segmenty wraz z odpowiednią konfiguracją adresów IP. Operacje te kolejno wykonujemy przez kliknięcie na scenie prawym przyciskiem myszy i wybranie żądanego urządzenia sieciowego.



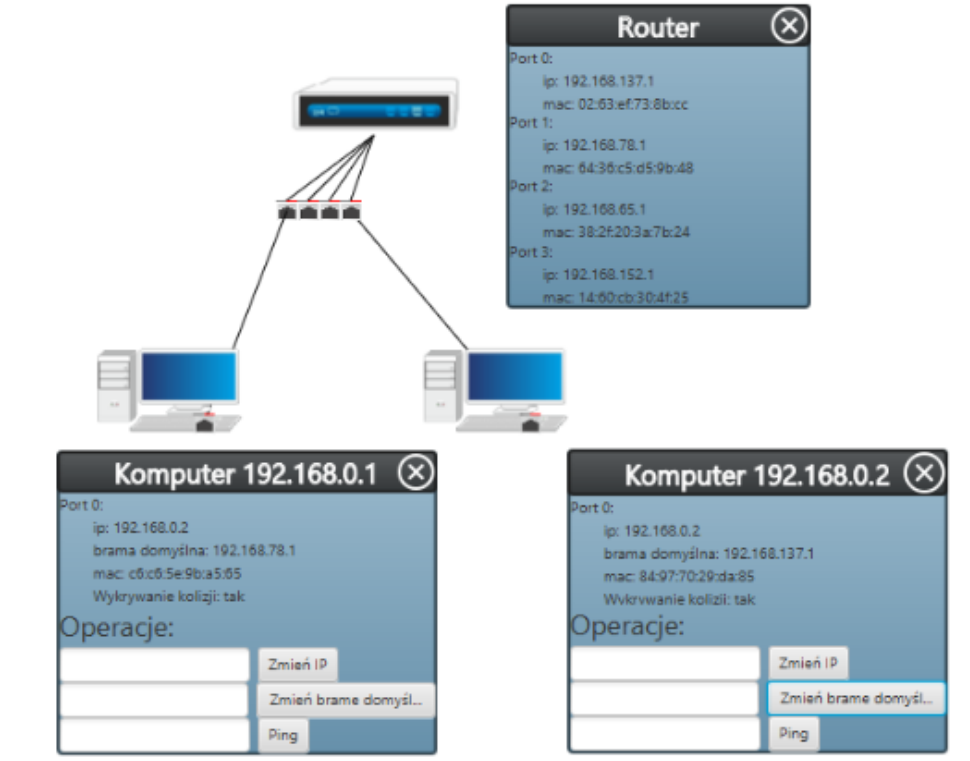
Rys. 16 Menu dodawania urządzenia sieciowego

Kolejnym krokiem po utworzeniu urządzeń jest ich połączenie. Wykonujemy to przez kliknięcie prawym przyciskiem myszy na portach urządzeń, które chcemy połączyć i wybranie z wyświetlonego menu opcji Podłącz kabel.

Rys. 17 Menu łączenia urządzeń sieciowych

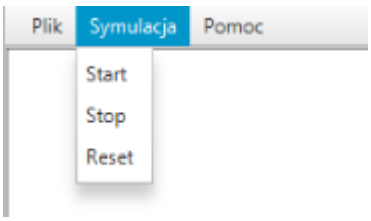
Rys. 18 Połączony segment sieciowy

Po utworzeniu segmentu sieci należy przystąpić do jego konfiguracji. Adresy IP przypisane poszczególnym interfejsom możemy edytować zarówno poprzez wywołanie opcji Zmień adres IP jak i w oknie pojawiającym się po wybraniu opcji Pokaż Stan Urządzenia, które to znajdują się w menu wyświetlanym po kliknięciu prawym przyciskiem myszy na urządzenie, które mamy zamiar edytować. Należy zwrócić uwagę na poprawność konfiguracji, w przeciwnym wypadku nie będzie możliwe poprawne wykonanie symulacji.

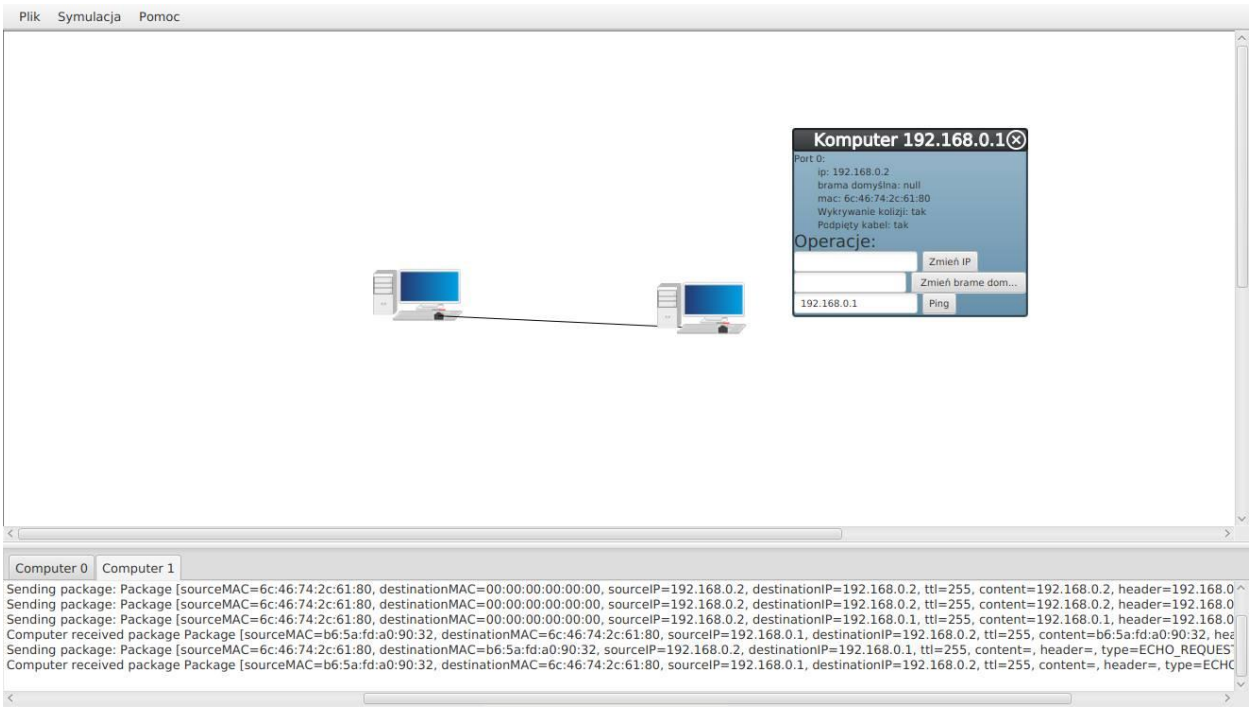


Rys. 19 Przykładowa konfiguracja prostej sieci komputerowej

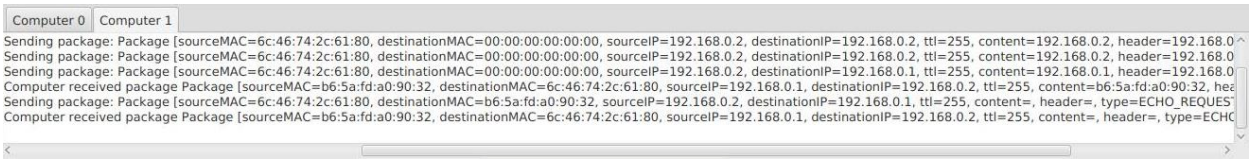
Mając gotową do testu sieć komputerową możemy przystąpić do uruchomienia symulacji. W celu rozpoczęcia wybieramy opcję Start z Menu Symulacji.



Rys. 20 Menu kontroli symulacji



Rys. 21 Okno działa aplikacji w trakcie symulacji



Rys. 22 Zakładka logów prowadzonych dla każdego urządzenia

5. Wnioski

Celem projektu było zaprojektowanie oraz zaimplementowanie aplikacji umożliwiającej przeprowadzenie symulacji działania oraz zachowania sieci komputerowej. Projekt został wykonany przez wykorzystanie symulacji w dziedzinie zdarzeń dyskretnych, co pozwoliło na zapoznanie się z jej charakterystyką oraz mechanizmami. Program realizuje wszystkie wyznaczone mu wymagania i można go uznać za pełnowartościową symulację działania sieci komputerowej opartej o model ISO/OSI.

Projekt pozwala na dalsze jego rozwijanie oraz zwiększanie skomplikowania przez co możliwe jest upodabnianie go w coraz większym stopniu do rzeczywistych sieci komputerowych. Osiągnąć można to poprzez dodanie obsługi podsieci o wszystkich maskach czy też wprowadzenie enkapsulacji pakietów danych. Ciekawym doświadczeniem mogłaby być również próba przeniesienia symulacji w dziedzinę czasu oraz dokonanie analizy porównawczej z aktualnym projektem w dziedzinie zdarzeń dyskretnych.