Politechnika Warszawska

WYDZIAŁ ELEKTRYCZNY

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

Praca dyplomowa magisterska

na kierunku Informatyka
w specjalności Inżynieria oprogramowania

Problemy współbieżności w algorytmach węzła sieci czujnikowej na przykładzie modelu w języku Go.

Jakub Młynarczyk
nr albumu 288226

promotor
dr inż. Łukasz Makowski

WARSZAWA 2019

Problemy współbieżności w algorytmach węzła sieci czujnikowej na przykładzie modelu w języku Go.

Streszczenie

Praca składa się z krótkiego wstępu jasno i wyczerpująco opisującego oraz uzasadniającego cel pracy. Rozdział drugi 'Wykorzystane technologie' opisuje technologie, narzędzia oraz systemu wykorzystane w celu wykonania pracy. Rozdział trzeci 'Bezprzewodowe sieci czujnikowe' przedstawia podstawowe zagadnienia związane z bezprzewodowymi sieci czujników, algorytmami i protokołami stosowanymi w celach uzyskania lepszej efektywności procesu wymiany informacji. Rozdział czwarty 'Architektura systemu' opisuje wymagania oraz implementację autorskiego systemu środowiska symulacyjnego. Rozdział piąty 'Opracowanie wyników eksperymentów' definiuje zakres testów, prezentuje uzyskane wyniki oraz krótko opisuje uzyskane wartości. Ostatni rozdział pracy 'Podsumowanie' to holistyczny opis uzyskanych wyników, zaobserwowanych zależności w bezprzewodowych sieciach czujnikowych oraz możliwościach dalszego rozwoju projektu.

Słowa kluczowe: wsn, sieci czujnikowe, golang, LEACH, PEGASIS

Challenges of concurrency in wireless sensor network, based on a model developed in Go.

Abstract

This thesis presents a novel way of using a novel algorithm to present complex problems of concurrency in wireless sensor networks. In the first chapter briefly presents presents the objectives and goals of the document. The second chapter describes all available tools, technologies and utilities used in the process of writing. The third chapter presents the fundamentals of wireless sensor networks and technologies. The fourth chapter presents requirements and implementation details of custom-made simulation environment written in Go programming language. The fifth chapter defines a set of tests and sub-tests, as well as presents results of those tests, which enable to compare existing wireless sensor network protocols and proves the

correctness of end-to-end simulator. Final chapter summarizes all the tests results, discovered dependencies and points some new possibilities of further development of the simulator.

Keywords: wsn, wireless sensor networks, golang, LEACH, PEGASIS

WARSZAWA, 31 marca 2019

POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY

OŚWIADCZENIE

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa magisterska pt. Problemy współbieżności w algorytmach węzła sieci czujnikowej na przykładzie modelu w języku Go.:

- 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.

Jakub Młynarczyk.....

Spis treści

Podziękowania

Dziękuję bardzo serdecznie wszystkim pracownikom uczelni, rodzinie oraz pracodawcy za poświęcony czas i energię.

Jakub Młynarczyk

Rozdział 1

Wstęp

W ciąg ostatnich lat obserwujemy szybki rozwój technologii informatycznych i teleinformatycznych w zakresie bezprzewodowych sieci czujnikowych. Pierwsze implementacje bezprzewodowych czujników wykorzystano w celach zbrojeniowych. Aktualnie, możliwości oferowane przez bezprzewodowe czujniki znajdują zastosowanie w nieskończonej liczbie aplikacji, zarówno w przemyśle (np. medycynie), jak również w sektorze prywatnym (np. inteligentny dom).

1.1 Cel pracy

Celem pracy są badania problemów współbieżności w algorytmach sieci czujnikowej. Praca wykorzystuje ogólnie znane i udokumentowane algorytmy umożliwiające transmitowanie danych w sieci, w których najważniejszym ograniczeniem jest skończona ilość energii czujnika.

Istotnym elementem pracy jest autorski symulator sieci sensorowej, która stanowi podstawowe narzędzie umożliwiająca implementację oraz testowanie nowych algorytmów. Aplikacja umożliwia tworzenie symulacji porównujących efekty zastosowania różnych algorytmów transmisji danych, bez potrzeby wykorzystania fizycznych urządzeń.

1.2 Tworzenie projektu

Koncepcja projektu informatycznego została opracowana przez autora pracy. Początkowy etap tworzenia oprogramowania polegał na zebraniu wymagań systemu. Niezbędnym było zdefiniowanie danych wejściowych, oraz oczekiwanych danych wyjściowych będących rezultatem działania oprogramowania symulacyjnego. W momencie zatwierdzenia założeń projektu przez

promotora pracy, autor przystąpił do stworzenia prototypu, który miał na celu zweryfikowanie założeń, przetestowanie fundamentów architektury systemu, oraz oszacowanie czasu niezbędnego do ukończenia całości projektu. Pierwszy prototyp uwidocznył istotne braki w funkcjonalności (np. brak wielokrotnego odpalania scenariuszy), które zostały zaimplementowane w finalnej wersji. Projekt zrealizowano w języku programowania Go, wraz z systemem kontroli wersji git, którego zdalna kopia dostępna jest na serwisie GitHub na profilu autora pracy (github.com/keadwen).

Rozdział 2

Wykorzystane technologie

Rozdział ten zawiera opis technologii oraz narzędzi wykorzystanych w pracy dyplomowej. Przedstawiony zostanie język oprogramowania Go, w którym napisanym zostały najważniejsze komponenty pracy dyplomowej. Rozdział zawiera również metody i technologie niezbędne do wygenerowania danych wejściowych, technik serializacji danych oraz ich reprezentacji przy wykorzystaniu Protocol Buffers, GNUPlot, itd.

2.1 Język programowania Go

Go (Golang) to język programistyczny stworzony jako wolne oprogramowanie (open-source) na potrzeby firmy Google, Inc. Głównymi architektami języka są Robert Griesemer, Rob Pike i Ken Thompson. Golang umożliwia tworzenie komercyjnego oprogramowania i jest wspierany na wielu systemach operacyjnych (Linux, Windows, Mac OS X).

Język Go należy do kategorii języków kompilowanych, z statycznym definiowaniem typu zmiennych. Poniżej przedstawiony został fragment kodu źródłowego napisane w języku Go.

```
package main

import (
  "fmt"

  "github.com/golang/example/stringutil"
)

type Label struct {
```

```
  Text string
  X, Y int
}

func (l *Label) ReverseText() {
  l.Text = stringutil.Reverse(l.Text)
}

func main() {
  var x = 100
  fmt.Printf("<%T>: %v", x, x)
  // Expected output: <int>: 100
  hello := Label{
    Text: "Hello",
    X:    100,
    Y:    200,
  }

  fmt.Println(hello.Text)
  // Expected output: hello
  hello.ReverseText()
  fmt.Println(hello.Text)
  // Expected output: olleh
}
```

W powyższym przykładzie przedstawione zostało kilka innowacyjnych funkcjonalności języka Go. Składnia języka oraz funkcjonalności zbliżone są do C/C++ czy Python, jednakże występuje kilka cech unikatowych dla Go.

Porównanie Go z Python

- Go jest językiem wspomagającym tworzenie aplikacji wielowątkowych.
- Go jest językiem statycznym, co pozwala wyeliminować błędy typu runtime wynikające z typu zmiennej.
- Go jest językiem samodokumentującym. Określony odgórnie format komentarzy umożliwia automatyczne tworzenie przejrzystej dokumentacji.
- Go jest językiem kompilowanym, co pozwala na szybsze uruchamianie i egzekucję oprogramowania.
- Go wykorzystuje mniej pamięci. Przykład: W Go zmienna typu int32 wymaga 4 bajty pamięci, w Python 24 bajty.

- Python umożliwia runtime reflection.
- Python posiada większy bazę publicznych bibliotek.

Porównanie Go z C++

- Go posiada system zarządzania pamięcią (garbage collector).
- Go jest językiem wspomagającym tworzenie aplikacji wielowątkowych.
- Go jest językiem samodokumentującym. Nie wymaga tworzenia plików typu header.
- Go nie jest językiem obiektowym. Zdolność dziedziczenia (inheritance) została zastąpiona osadzaniem (embedding).
- Go posiada możliwość użycia (zaimportowania) dowolnej biblioteki C, C++.
- C++ posiada osiągnąć szybszą egzekucję oprogramowania.
- C++ posiada tworzenie kodu niezależnie od typu zmiennej (generics).
- C++ nie posiada systemu zarządzania pamięcią (garbage collector), co umożliwia większą kontrolę nad zasobami pamięci (np. w mikro kontrolerach).

Unikatowe cechy Go

- Produktem kompilacji jest plik egzekucyjny posiadający wszystkie niezbędne zależności.
- Zarządzanie pakietami pozwala na importowanie rozwiązań bezpośrednio z GitHub (lub innych serwisów zewnętrznych).
- Dynamiczna alokacja typu zmiennej statycznej.
- Natywne wspieranie tworzenie oprogramowania wykorzystującego wielowątkowość i współbieżność procesów.
- Natywna metoda testowania funkcjonalności i bibliotek.
- Pełny zestaw wbudowanych narzędzi pozwalających na testowanie wydajności kodu (benchmarking), formatowania składni kodu zgodnie ze standardem (gofmt), oraz wiele innych funkcjonalności.

2.2 Serializacja danych Protocol Buffers

Protocol Buffers (proto, protobuf) to mechanizm serializacji danych stworzony na potrzeby firmy Google, Inc. Protocol Buffers to mechanizm współpracującym niezależnie od języka oprogramowania aplikacji czy platformy na którym uruchamiana jest aplikacja. Technologia ta definiuje strukturę danych (proto schema) za pomocą dedykowanego języka, składającego się z prostych zmiennych (np.: int64, string) oraz złożonych komunikatów (message). Poniżej przedstawiona została przykładowa struktura.

```
// example.proto
syntax = "proto3";

// Citizen represents a single citizen of Poland.
message Citizen {
  // The name of a citizen.
  string name = 1;
  // The surname of a citizen.
  string surname = 2;
  // (required) Unique Polish national identification number.
  PESEL pesel = 3;
}

// PESEL represents Polish Universal Electronic System for
// Registration of the Population.
message PESEL {
  // (required) Unique Polish national identification number.
  uint64 number = 1;
  bool active = 2;
}
```

Struktura ta przechowywana jest w plikach o rozszerzeniu ‘.proto‘, które są następnie kompilowane do dowolnego z wspieranych języków oprogramowania (w przypadku Protocol Buffers w wersji 3, wspierana jest generacja kodu w Java, C++, Python, Java Lite, Ruby, JavaScript, Objective-C, C# oraz PHP). Następnie, dane zserializowane zostają zapisane w formacie binarnym (wire format), który umożliwia na uzyskanie wyższego poziomu kompresji danych oraz transmisję danych bez potrzeby wykonania dalszego kodowania. Wynikiem kompilacji plików ‘.proto‘ jest zestaw bibliotek zawierający wygenerowany kod źródłowy, wraz z gotowymi strukturami, funkcjami i metodami niezbędnymi do operowania danymi w sposób natywny dla wybranego

języka programowania.

2.3 Narzędzia dodatkowe

Sekcja ta przedstawia zestaw narzędzi których funkcjonalności umożliwiły stworzenie oraz ułatwiły zarządzanie oprogramowaniem stworzonym w celach pracy dyplomowej.

2.3.1 System kontroli wersji git

Git to rozproszony system kontroli wersji stworzony jako wolne oprogramowanie (open source). Głównymi architektem narzędzia jest Linus Torvalds. Git to oprogramowanie powszechnie stosowanym w przypadku zarządzania oprogramowaniem. Narzędzie to umożliwia tworzenie pobocznych gałęzi (branch) niezależnych od głównej gałęzi. Funkcjonalność ta pozwala na niezależne wprowadzanie zmian w kodzie na określonej wersji kontrolnej, które mogą następnie zostać wprowadzone ponownie do gałęzi głównej (merge). Architektura rozproszona git (w przeciwieństwie do scentralizowanych systemów kontroli wersji) umożliwia programistom na posiadanie lokalnej kopii repozytorium, której zmiany mogą zostać następnie wprowadzone do gałęzi głównej.

2.3.2 GNUPlot

GNUPlot to narzędzie do generowania wykresów funkcji w oparciu o dane wejściowe. Program dostępny jest niemal na każdym systemie operacyjnym. Przy pomocy GNUPlot generować można dwu- oraz trójwymiarowe wykresy, które zapisane mogą zostać w różnych formatach t.j. PNG, SVG czy JPEG.

2.3.3 Docker

Docker to narzędzie stworzone jako wolne oprogramowanie (open source) napisane w języku Go przez firmę Docker, Inc. Narzędzie to pozwala na tworzenie kontenerów, które izolują aplikację na poziomie systemu operacyjnego. W przeciwieństwie do maszyn wirtualnych, kontener nie wymaga wirtualizowania systemu operacyjnego dla każdego z kontenerów. Wszystkie równolegle działające kontenery aplikacji działające na pojedynczym urządzeniu współdzielą parametry fizyczne maszyny oraz jądro systemu operacyjnego (np. Linux). Izolacja kontenerów widoczna jest na poziomie zależności (depen-

dency) do określonych wersji bibliotek (libraries), narzędzi (binaries), plików konfiguracyjnych czy parametrów.

Rozdział 3

Bezprzewodowe sieci czujnikowe

W tym rozdziale przedstawiona zostanie najważniejsza część teoretyczna bezprzewodowych sieci sensorowych, która jest niezbędna w celu zrozumienia problemu rozwiązywanego w pracy dyplomowej.

3.1 Wstęp do bezprzewodowych sieci sensorowych

Bezprzewodowe sieci sensorowe (wireless sensor network) nazywa się również bezprzewodowymi sieciami czujników. Sieć czujnikowa składa się z urządzeń, których funkcją jest realizowanie określonego zadania. Pierwsze aplikacje i zastosowania bezprzewodowych sieci czujników zostały wdrożone na potrzeby wojskowe, jednakże przeciągu ostatnich lat, technologie te znalazły wiele nowych zastosowań w przemyśle (np. pomiary meteorologiczne) i aplikacjach codziennych (np. systemy domów inteligentnych).

Rozwój technologii, malejący koszt elektroniki oraz dostępność produktów bezprzewodowej sieci czujnikowej, umożliwia tworzenie dedykowanych aplikacji. Na rynku dostępnych jest wiele urządzeń oraz rozwiązań, a proces wyboru uzależniony jest przede wszystkim od wymogów projektu oraz budżetu. Dla uproszczenia przyjąć można, że pojedynczy czujnik powinien składać się z procesora zdolnego wykonywać określone zadanie, pamięci zdolnej do przechowywania informacji oraz anteny, która umożliwia nadawanie i odbieranie informacji.

Proces wymiany informacji między urządzeniami zależy od protokołów i implementacji. Niezależnie jednak od protokołu i implementacji, cel pozostaje niezmienny. Informacje posiadane przez węzeł w sieci (np. pomiar temperatury otoczenia) muszą zostać przekazane z węzła pomiarowego do węzła głównego. Węzeł główny, zwany również ‘sink‘ jest odpowiedzialnym

za agregację wszystkich informacji oraz ich dalsze przetwarzanie. W dalszej części pracy, przedstawione zostaną dwa protokoły trasowania (routingu), których zadaniem jest zoptymalizowanie energetyczne procesu komunikacji, podwyższenie niezawodności systemu oraz umożliwienie zautomatyzowanej organizacji topologii sieci. By móc zrozumieć istotę i korzyści wykorzystania protokołów trasowania, niezbędnym jest przedstawienie najprostszego schematu komunikacji, komunikacji bezpośredniej.

3.2 Model transmisji

W pracy wykorzystany został model transmisji przedstawiony w pracy [2]. Model jest powszechnie wykorzystywany w planowaniu oraz symulowaniu aplikacji bezprzewodowych sieci czujników. Model przedstawia zależności transmisji danych w wolnej przestrzeni pomiędzy dwoma węzłami (nadawcą i odbiorcą) uwzględniając wielotorowość sygnału. Poniższy wzór pozwala na wyznaczenie całkowitego kosztu transmisji k-bitów do węzła oddalonego o d-metrów:

$$E_{TX}(k,d) = E_{TX-elec}(k) + E_{TX-amp}(k,d) = \begin{cases} k \times E_{elec} + k \times \varepsilon_{fs} \times d^2 & d < d_{d_0} \\ k \times E_{elec} + k \times \varepsilon_{fs} \times d^4 & d \geqslant d_{d_0} \end{cases}$$

gdzie E_{elec} jest energią wykorzystywaną przez nadajnik (np. praca mikroprocesora), d_0 wyznaczamy przez $\sqrt{\varepsilon_{fs}/\varepsilon_{mp}}$. W zależności od odległości na której odbywa się komunikacja, ε_{fs} reprezentuje transmisję w wolnej przestrzeni, natomiast ε_{mp} wielotorowość transmisji.

Poniższy wzór pozwala na wyznaczenie całkowitego kosztu odbioru k-bitów przez węzeł:

$$E_{RX}(k,d) = E_{RX}(k) = k \times E_{elec}$$

3.3 Protokoły

Warstwa sieciowa w bezprzewodowych sieciach czujników jest elementem krytycznym działania całości systemu. Warstwa ta w przeciwieństwie do warstwy fizycznej, odpowiedzialna jest za logiczne skonfigurowanie węzłów i zaimplementowanie algorytmu trasowania danych pomiędzy węzłami.

Istnieje wiele protokołów WSN, których dobór zależy od przede wszystkim od naszej aplikacji oraz parametrów sieci które wymagają optymalizacji.

Poniższej przedstawione zostały cztery aspekty, których dokładne określenie pozwoli na dopasowanie najbardziej optymalnego algorytmu:

- koszty energetyczne - w sieci w której ilość energii węzła jest ograniczona, zalecanym jest stosowanie algorytmów pozwalających na efektywną wymianę danych.
- adresowania i rozpoznawania węzłów - w sieci w której występuje duża ilość węzłów, należy określić metodę adresacji. Metoda statyczna (np. wykorzystanie pliku konfiguracyjnego) lub dynamiczna (np. wygenerowanie unikatowego numeru identyfikującego).
- skalowalności sieci - w sieci w której występuje duża ilość węzłów wymiana informacji może być utrudniona zarówno pod kątem fizycznym (np. brak dostępnej przepustowości w paśmie bezprzewodowym), jak i logicznym (np. długi czas detekcji węzłów i budowania topologii sieci).
- niezawodności wymiany danych - w sieci w której dane generowane przez poszczególne węzły nie mogą zostać utracone, zalecane jest wykorzystanie algorytmu umożliwiającego detekcję brakujących fragmentów danych i retransmisję.

W następnej części pracy przedstawione zostaną 3 protokoły trasowania danych, które zostały zaimplementowane w symulatorze. Wybór ukierunkowany był na ograniczenie kosztów energetycznych w sieci, które umożliwia na wydłużenie czasu życia każdego z węzłów.

3.3.1 Komunikacja bezpośrednia

Komunikacja bezpośrednia jest rozwiązaniem najprostszym z perspektywy implementacji, jednakże nieefektywnym z poziomu energetycznego. Węzły znajdujące się sieci nie są zaangażowane w podejmowanie decyzji dotyczących optymalizacji kosztów transmisji. Adres węzła głównego ('sink') może być zaprogramowany na poziomie pliku konfiguracyjnego. Dane wysyłane przez węzeł nadawane są bezpośrednio do węzła głównego. Model środowiska symulacyjnego definiuje koszt transmisji komunikacji. Wartość ta zależna jest od odległości między dwoma węzłami. Czas życia węzłów (posiadających identyczną ilość energii początkowej oraz wielkość przesyłanych informacji) wydłuża się, wraz z malejącym dystansem do węzła głównego.

Poniższa ilustracja przedstawia trzy węzły w sieci, które komunikują się bezpośrednio z węzłem głównym:

3.3.2 LEACH

LEACH (Low-Energy Adaptive Clustering Hierarchy) jest algorytmem wykorzystywanych w protokołach routingu bezprzewodowej sieci czujnikowej. W przeciwieństwie do komunikacji bezpośredniej, LEACH jest protokołem w którym występuje hierarchia.

Zestaw operacji w protokole LEACH nazywa się rundami (round). Każda z rund podzielona jest na dwie fazy. W pierwszej fazie, fazie konfiguracyjnej (setup), węzły dokonują podziału sieci na niezależne klastry (cluster). W drugiej fazie, fazie komunikacyjnej (steady state), węzły uczestniczące w sieci dokonują wymiany informacji za pomocą agregacji danych w klastrze, które następnie przesłane są do stacji bazowej.

Faza konfiguracji (setup) zbudowana jest z trzech etapów. W pierwszym etapie węzły dokonują wyboru roli w rundzie. W LEACH występują dwa rodzaje węzłów:

- węzły typu cluster head (CH)
- węzły standardowe (non-CH)

Każdy z węzłów generuje losową wartość ‘n‘ z zakresu [0, 1]. Wartość ta podstawiona w poniższe równanie:

$$T(n) = \begin{cases} \frac{P}{1-P[r*mod(1/P)]} & \text{jeżeli } n \in G \\ 0 & \text{w innym przypadku} \end{cases}$$

gdzie P oznacza wartość z zakresu (0, 1] definiująca prawdopodobieństwa wyznaczenia węzła typu cluster head. G jest zbiorem węzłów które nie pełniły roli cluster head w ostatnich 1/P rundach.

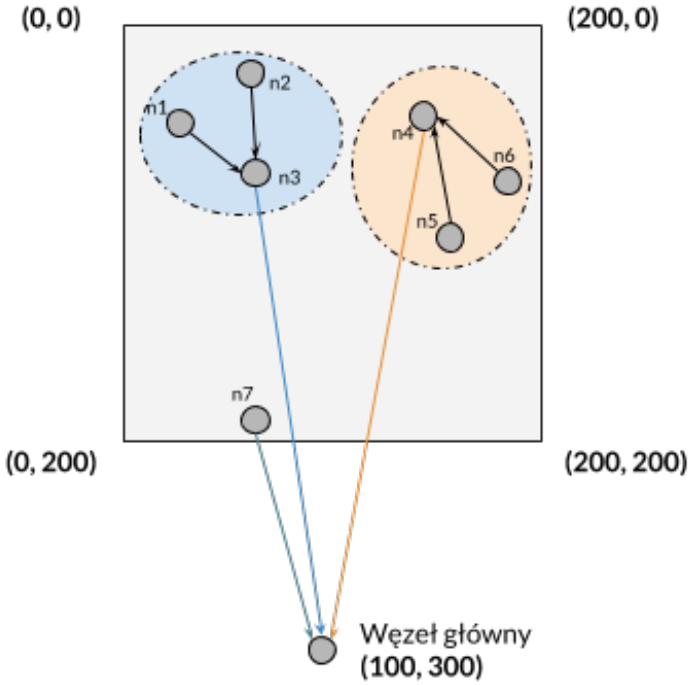
W momencie ustalenia roli przez węzły, każdy z węzłów informuje pozostałe węzły uczestniczące w sieci o swojej roli w rundzie.

Kolejnym etapem fazy konfiguracji jest wyznaczenie klastra do którego zostaną przyłączone węzły standardowe. Każdy z węzłów standardowych wybiera jeden węzeł typu cluster head. W przypadku otrzymania informacji od kilku węzłów typu cluster head, węzeł standardowy wybiera węzeł CH znajdujący się najbliższej (węzeł którego sygnał jest najmocniejszy).

Ostatnim etapem jest ukończenie formowania klasterów w którym znajduje się dokładnie jeden węzeł typu cluster head. Wystąpić może sytuacja w której węzeł typu cluster head nie posiada przynależących węzłów standardowych. Pozytywnie zakończona faza formowania klastrów tworzy stabilną sieci wymiany informacji.

W tym momencie rozpoczyna się faza komunikacji (‘steady state‘) w której węzły standardowe dokonują transmisji danych do węzła typu cluster head. Węzły CH dokonują agregacji zebranych informacji oraz bezpośrednie przekazanie ich do węzła głównego (‘sink‘). Stworzenie środowiska w którym węzły dokonują pośredniczenia informacji, pozwala na ograniczenie kosztów energetycznych transmisji danych przez standardowe węzły (dystans do węzła typu cluster head jest mniejszy niż do ‘sink‘). Należy jednak pamiętać, że koszt energetyczny transmisji węzłów typu cluster head rośnie, gdyż stają się one odpowiedzialne za odbieranie informacji, przetworzenie ich, a następnie wysłanie do całości do ‘sink‘. Wielkość informacji (mierzona w bajtach), przetransmitowana z węzłów typu cluster head jest zależna od ilości standardowych węzłów przynależących do CH oraz metody agregacji informacji.

Poniższa ilustracja przedstawia zestaw węzłów w sieci, które wykorzystują protokół LEACH:



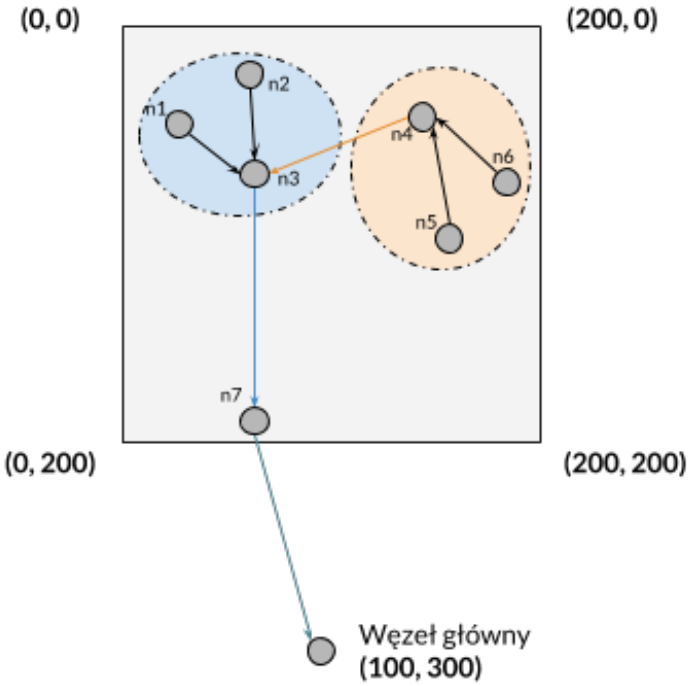
3.3.3 PEGASIS

PEGASIS (Power-Efficient GAathering in Sensor Information Systems) jest algorytmem wykorzystywanym w protokołach routingu bezprzewodowej sieci czujnikowej. W przeciwieństwie do komunikacji bezpośredniej, PEGASIS podobnie jak LEACH jest protokołem w którym występuje hierarchia.

PEGASIS jest protokołem dokonującym usprawnień w protokole LEACH. W przypadku sieci czujnikowej, urządzenia pełniące rolę węzłów posiadają ograniczone ilości energii, wynikające np. z zasilania bateryjnego. Przebudowana metoda przekazywania danych zebranych przez węzły typu cluster head, pozwala na ukończenie rundy przy wykorzystaniu mniejszej ilości energii niż LEACH.

Pierwsza faza, faza komunikacji, w której ustalane są role węzłów jest identyczna. Przebudowana została metoda transmisji agregacji informacji od węzłów typu cluster head to węzła głównego (‘sink‘). Węzły typu cluster head formują łańcuch, w którym najdalej oddalony węzeł typu cluster head, przekazuje zebrane i dane agregowane do innego węzła typu cluster head znajdującego się w najbliższym sąsiedztwie. Algorytm trasowania najmniejszym kosztem nazywana się typu ‘greedy‘.

Efektem niekorzystnym w protokole PEGASIS jest wydłużenie czasu każdej z rund, wynikające z sekwencyjnej wymiany danych w łańcuchu. Generowane w ten sposób są dodatkowe koszty energetyczne związane z utrzymaniem węzłów w stanie pracy, kosztem odbioru i przetworzenia informacji. Należy jednak podkreślić, że zysk energetyczny wynikający z transmisji danych na krótsze dystanse kompensuje poniesione straty, tym samym pozwalając na wydłużenie czasu pracy każdego z węzłów.



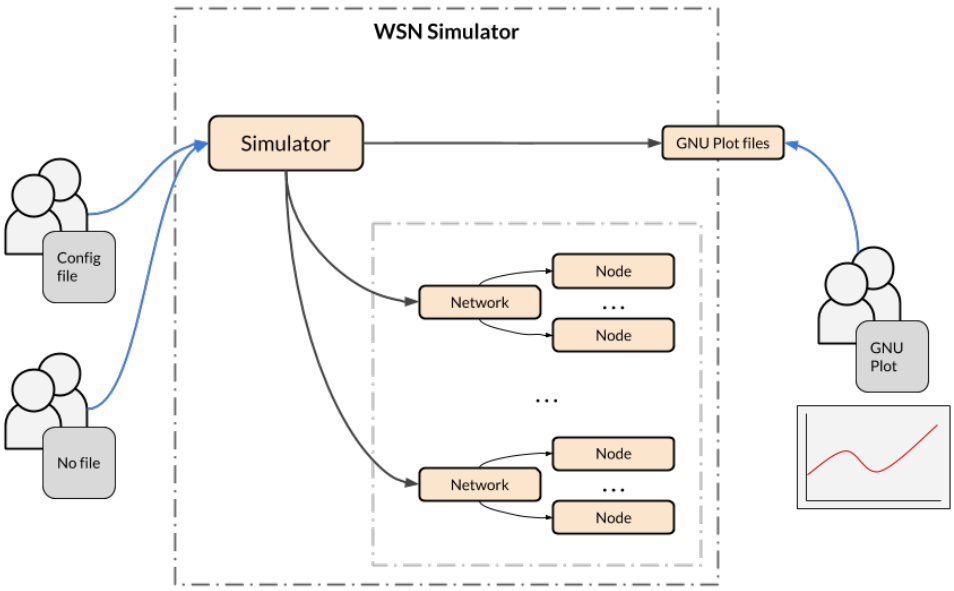
Rozdział 4

Architektura systemu

System informatyczny stworzony na potrzeby pracy dyplomowej ma za zadanie dostarczenie wyników oraz wykresów niezbędnych do zbadania poniższych zależności sprawności energetycznej węzłów dla wybranych algorytmów w sieci WSN (komunikacji bezpośredniej, LEACH, PEGASIS).

4.1 Komponenty systemu

Poniższy diagram przedstawia dekompozycję modułów, całkowitą architekturę systemu oraz kierunki interakcji i przepływu informacji.



Użytkownik (znajdujący się po lewej stronie) posiada możliwość uruchomienia oprogramowania na dwa sposoby:

- symulacja wstępnie konfigurowana za pomocą pliku konfiguracyjnego
- symulacja bez pliku konfiguracyjnego

Opcja symulacji bez pliku konfiguracyjnego powoduje wygenerowanie pliku konfiguracyjnego w którym znajduje się 200 węzłów, rozproszonych w sposób pseudolosowy na przestrzeni 200 x 200.

Symulator tworzy wewnętrzną strukturę sieci i węzłów, które poddawane są symulacji. Wynik końcowy poszczególnych sieci zwracany jest do symulatora, który generuje pliki w formacie kompatybilnym z narzędziem GNU Plot.

4.1.1 Plik konfiguracyjny

Plik konfiguracyjny (‘config file‘) posiada dane wejściowe pozwalające na zbudowanie środowiska testowego. Informacje zawarte w pliku konfiguracyjnym można podzielić na dwie sekcje:

- konfiguracja symulatora i sieci
- konfiguracja węzła

Konfiguracja symulatora i sieci

Konfiguracja symulatora i sieci składa się z pięciu zmiennych w komunikacie ‘Config‘. Każda z tych wartości może być modyfikowana bez potrzeby ponownej kompilacji oprogramowania symulacyjnego. Poniższa lista przedstawia oraz opisuje znaczenie poszczególnych parametrów konfiguracyjnych:

- protokoł (‘protocol‘) - zmienna ta zdefiniowana za pomocą komunikatu typu ‘enum E_Protocol‘ pozwala symulatorowi wybrać odpowiedni protokół sterujący symulacją. Dokładny opis działania protokołów zostanie przedstawiony w dalszej części pracy.
- wartości maksymalnej rund pomiarowych (‘max_rounds‘) - zmienna ta pozwala określić wartość rund pomiarowych aktywnych węzłów w pojedynczej symulacji, po której całkowity przebieg symulacji zostanie zatrzymany. Wyznaczenie tej wartości umożliwia użytkownikowi określenie dowolnej granicy, bez potrzeby oczekiwania na zakończenie symulacji (wykorzystanie całkowitej energii dostępnej przez węzły pomiarowe).

- proporcji węzłów typu klaster do wszystkich węzłów (‘p_cluster_heads‘) - zmienna ta pozwala określić stosunek ilości węzłów pomiarowych odpowiedzialnych za pośredniczenie w przesyłaniu danych pomiarowych (klastrów) do ilości wszystkich węzłów w sieci. Parametr ten wykorzystywany jest zależnie od protokołu.
- długość wiadomości (‘msg_length‘) - zmienna ta określa całkowity rozmiar wiadomości generowanych przez pojedynczy węzeł pomiary. Długość wyrażona jest w bajtach i jest sumą dwóch elementów, danych pomiarowych oraz dodatkowych danych generowanych w procesie enkapsulacji (np. adresowanie, preambuły, itd.)
- konfiguracja węzłów (‘nodes‘) - zmienna ta zdefiniowana za pomocą listy komunikatów typu Node. Symulacja musi składać się przynajmniej z dwóch węzłów. Kolejność listy ma znaczenie, gdyż pierwszy węzeł pełni rolę głównego odbiornika danych (‘sink‘). Dokładniejszy opis parametrów poszczególnych węzłów przedstawiony został w kolejnym podrozdziale ‘Konfiguracja węzła‘.

Poniżej przedstawiony został fragment pliku konfiguracyjnego dla konfiguracji symulatora i sieci:

```
enum E_Protocol {
  UNSET = 0;
  DIRECT = 1;
  LEACH = 2;
  APTEEN = 3;
  PEGASIS = 4;
}

message Config {
  // Simulation protocol.
  E_Protocol protocol = 1;
  // Number of maximum rounds in simulation.
  int64 max_rounds = 2;
  // Percentage of cluster heads among all nodes [0, 1].
  double p_cluster_heads = 3;
  // Size of data sent by individual node (in Bytes).
  int64 msg_length = 4;
  // Nodes points to configuration for each node.
  repeated Node nodes = 5;
}
```

Konfiguracja węzła

Konfiguracja węzła składa się z pięciu zmiennych w komunikacje ‘Node‘. Każda z tych wartości może być modyfikowana bez potrzeby ponownej kompilacji oprogramowania symulacyjnego. Poniższa lista przedstawia oraz opisuje znaczenie poszczególnych parametrów konfiguracyjnych:

- indeks (‘id‘) - zmienna ta określa unikatowy identyfikator węzła. Parametr ten umożliwia identyfikację węzła podczas symulacji.
- energia początkowa (‘initial_energy‘) - zmienna ta określa ilość energii (mierzonej w [J]), którą posiada węzeł w momencie rozpoczęcia symulacji. W przypadku zdefiniowania zerowej energii początkowej, węzeł nie będzie brał udziału w komunikacji ze względu na brak zasobów energetycznych na przeprowadzenia jakiejkolwiek operacji.
- pozycja (‘location‘) - zmienna ta zdefiniowana za pomocą komunikatu typu ‘Location‘ pozwala symulatorowi na umieszczenie węzła w dwuwymiarowej przestrzeni. Pozycja węzła wykorzystywana jest do określania odległości między węzłami i aplikowania kosztów energetycznych operacji (np. transmisji danych).
- koszt energetyczny (‘energy_cost‘) - zmienna ta zdefiniowana za pomocą komunikatu typu ‘EnergyCost‘ pozwala na wprowadzenie dodatkowych kosztów energetycznych operacji dla poszczególnych węzłów. Podstawowy model transmisji posiada zdefiniowane koszty energetyczne operacji. W przypadku zdefiniowana zmiennej ‘EnergyCost‘ dla węzła, koszt operacji (np. transmisji, odbioru, pomiaru i przetwarzania danych) ulega zmianie.
- opóźnienia czasowe (‘time_delay‘) - zmienna ta zdefiniowana za pomocą komunikatu typu ‘TimeDelay‘ pozwala symulatorowi na wprowadzenie dodatkowych parametrów czasowych dla operacji wykonywanych przez węzeł. Iloczyn zmiennej (np. czasu przetwarzania danych węzła [ns]) i kosztu energetycznego działania węzła [J/s] reprezentuje dodatkowe parametry symulacji, które umożliwiają tworzenie zaawansowanych i precyzyjnych scenariuszy.

Poniżej przedstawiony został fragment pliku konfiguracyjnego dla konfiguracji węzła:

```
// Node defines a configuration for a single node.
message Node {
  // Unique ID for a node.
```

```
  int64 id = 1;
  // Initial value of energy (in Joules).
  double initial_energy = 2;

  // Location of a node in 2D space.
  Location location = 3;
  // Energy consumption of node operations.
  EnergyCost energy_cost = 4;
  // Time delays introduced by node operations
  TimeDelay time_delay = 5;
}
```

Poniżej przedstawiony został fragment pliku konfiguracyjnego dla konfiguracji lokalizacji węzła:

```
// Location defines a X, Y coordinates of a node.
message Location {
  double X = 1;
  double Y = 2;
}
```

Poniżej przedstawiony został fragment pliku konfiguracyjnego dla konfiguracji kosztów energetycznych pracy węzła:

```
// EnergyCost defines energy consumption for common node operations.
message EnergyCost {
  // Energy required to transmit one byte (in Joules).
  double transmit = 1;
  // Energy required to receive one byte (in Joules).
  double receive = 2;
  // Energy required to listen the channel for a second (in Joules).
  double listen = 3;
  // Energy required to process sensor data (in Joules).
  double sensor_data_process = 4;
  // Energy required to wake up MCU (in Joules).
  double wake_up_mcu = 5;
}
```

Poniżej przedstawiony został fragment pliku konfiguracyjnego dla konfiguracji opóźnień czasowych pracy węzła:

```
// TimeDelay defines time delays for common node operations.
message TimeDelay {
  // Time required to process sensor data (in nanoseconds).
  int64 sensor_data_process = 1;
  // Time required to wake up MCU (in nanoseconds).
  int64 wake_up_mcu = 2;
}
```

4.1.2 Moduł główny (Core)

Moduł główny posiada najważniejsze cechy i funkcjonalności umożliwiające modelowanie systemu symulatora WSN. Wszystkie zaimplementowane struktury (‘structs‘) znajdują się w jednej bibliotece (‘package‘) o nazwie ‘core‘.

Węzeł (Node)

Węzeł (‘Node‘) reprezentuje model węzła, będącego elementem podstawowym w sieci. Struktura ta przechowuje dane konfiguracyjne węzła, poziom energii czy dane historyczne, pozwalające na monitorowanie pracy oraz tworzenie grafów.

Poniżej przedstawiona została struktura węzła oraz definicje funkcji przynależące do struktury.

```
type Node struct {
  Conf   config.Node
  Ready  bool
  nextHop *Node  // As a default set to Base Station.
  Energy float64 // Energy level of a node.

  transmitQueue int64
  receiveQueue int64
  // Statistics and aggregation variables.
  dataSent    int64
  dataReceived int64
}

func (n *Node) Transmit(msg int64, dst *Node) error
func (n *Node) Receive(msg int64, src *Node) error
func (n *Node) Info() string
```

```
func (n *Node) distance(dst *Node) float64
func (n *Node) consume(e float64) error
```

Poniższa lista przedstawia oraz opisuje znaczenie poszczególnych zmiennych struktury węzła (Node):

- konfiguracja (‘Conf‘) - zmienna publiczna przechowuje konfigurację węzła w formacie protocol buffer (szczegółowe informacje dostępne w ‘Konfiguracja węzła‘).
- gotowość (‘Ready‘) - zmienna publiczna przechowuje stan gotowości węzła. Wartość ‘true‘ oznacza, że węzeł jest gotowy do nadawania i odbierania informacji.
- następny skok (‘nextHop‘) - zmienna prywatna przechowuje adres do zmiennej węzła, będącego odbiorcą informacji nadawanych przez węzeł. Wartością domyślna w momencie rozpoczęcia symulacji jest adres głównego odbiornika danych (‘sink‘).
- energia (‘Energy‘) - zmienna publiczna przechowuje aktualny stan energetyczny węzła. W przypadku wyczerpania energii, zmienna Ready zostaje ustawiona na ‘false‘.
- kolejka nadawania (‘transmitQueue‘) - zmienna prywatna przechowuje informację o ilości bajtów gotowych do przekazania do następnego węzła na końcu rundy.
- kolejka odbioru (‘receiveQueue‘) - zmienna prywatna przechowuje informację o ilości bajtów odebranych przez węzeł na początku rundy.
- dane nadane (‘dataSent‘) - zmienna prywatna przechowuje sumę bajtów nadanych przez węzeł.
- dane odebrane (‘dataReceived‘) - zmienna prywatna przechowuje sumę bajtów odebranych przez węzeł.

Poniższa lista przedstawia oraz opisuje znaczenie poszczególnych funkcji węzła (Node):

- Transmit - funkcja pozwala na transmisję danych do węzła.
- Receive - funkcja pozwala na odbieranie danych przez węzeł.
- Info - funkcja generuje ciąg znaków w podstawowymi informacjami na temat węzła.
- distance - funkcja wyznacza wartość odległości pomiędzy dwoma węzłami.
- consume - funkcja obciąża energetycznie węzeł.

Sieć (Network)

Sieć (‘Network‘) reprezentuje model środowiska w którym odbywa się symulacja. Struktura ta przechowuje kontroluje przepływ informacji pomiędzy węzłami, zbiera i eksportuje informacje z poszczególnych rund.

Poniżej przedstawiona została struktura sieci oraz definicje funkcji przynależące do struktury.

```
type Network struct {
  Protocol   Protocol
  BaseStation *Node
  Nodes      sync.Map

  Round    int64
  MaxRounds int64
  MsgLength int64

  GNUPlotNodes     []string
  GNUPlotTotalEnergy []string

  PlotTotalEnergy *plot.Plot // An amount of total energy in the
     network per Round.
  PlotNodes       *plot.Plot // A number of alive nodes in the
     network per Round.
  NodesAlivePoints plotter.XYs
  NodesEnergyPoints map[int64]plotter.XYs
}

func (net *Network) AddNode(n *Node) error
func (net *Network) Simulate() error
func (net *Network) CheckNodes() int
func (net *Network) PopulateEnergyPoints()
func (net *Network) PopulateNodesAlivePoints()
```

Poniższa lista przedstawia oraz opisuje znaczenie poszczególnych zmiennych struktury sieci (Network):

- protokół (‘Protocol‘) - zmienna publiczna przechowuje obiekt definiujący protokół komunikacji pomiędzy węzłami.
- stacja bazowa (‘BaseStation‘) - zmienna publiczna przechowuje obiekt węzła głównego (‘sink‘).

- węzły (‘Nodes‘) - zmienna publiczna przechowuje obiekty węzłów w sieci. Implementacja przy wykorzystaniu dziennika (hashmap), który umożliwia operacje zapisu i odczytu w procesach równoległych. Kluczem dziennika jest unikatowy identyfikator węzła.
- runda (‘Round‘) - zmienna publiczna przechowuje numer aktualnej rundy symulacji.
- maksymalna ilość rund (‘MaxRounds‘) - zmienna publiczna przechowuje maksymalną ilość rund symulacji. W przypadku osiągnięcia wartości ‘Round‘ równej ‘MaxRounds‘, symulacja zostanie przerwana.
- długość wiadomości (‘MsgLength‘) - zmienna publiczna przechowuje informację o całkowitej wielkości wiadomości (mierzonej w bajtach) jaka generowana jest podczas rundy przez każdy z węzłów. W skład tej wartości wchodzą dane pomiarowe i dodatkowy nakład informacji powstały w wyniku enkapsulacji.
- GNUPlotNodes, GNUPlotTotalEnergy - zmienne publiczne przechowujące parametry rund wykorzystywane do generowania grafów przy użyciu narzędzia GNUPlot.
- PlotTotalEnergy, PlotNodes, NodesAlivePoints, NodesEnergyPoints - zmienne publiczne przechowujące parametry rund wykorzystywane do generowania grafów przy użyciu biblioteki plotter.

Poniższa lista przedstawia oraz opisuje znaczenie poszczególnych funkcji węzła (Node):

- AddNode - funkcja pozwala na dodanie węzła do sieci.
- Simulate - funkcja umożliwia rozpoczęcie symulacji.
- CheckNodes - funkcja sprawdza ilość sprawnych węzłów w sieci.
- PopulateEnergyPoints - funkcja sprawdza oraz przechowuje stany energetyczne węzłów.
- PopulateNodesAlivePoints - funkcja sprawdza oraz przechowuje ilość sprawnych węzłów w sieci.

Koszty transmisji

Transmisja i odbiór danych w sieci obarczony jest kosztem energetycznym. Model transmisji przedstawiony w rozdziale ‘Model transmisji‘ wymaga zdefiniowania wartości liczbowych dla czterech parametrów:

```
const (
  // Energy values measured in [J/byte].
  E_ELEC = 40e-9
  E_RX   = 4e-9
  E_MP   = 0.0104e-12
  E_FS   = 80e-12
)
```

Protokół (Protocol)

Protokół (‘Protocol‘) reprezentuje model protokołu komunikacji między węzłami.

Poniżej przedstawiona został interfejs protokołu oraz definicje funkcji przynależące do interfejsu.

```
type Protocol interface {
  Setup(net *Network) ([]int64, error)
  SetNodes(int)
  SetClusters(int)
}
```

Poniższa lista przedstawia oraz opisuje znaczenie poszczególnych funkcji węzła (Node):

- Setup - funkcja konfiguruje węzły w sieci zgodnie z zaimplementowanym protokołem.
- SetNodes - funkcja definiuje ilość węzłów w sieci. Wartość ta jest niezbędna do wyznaczania parametrów w protokołach (np. LEACH, PEGASIS).
- SetClusters - funkcja definiuje ilość klastrów w sieci. Wartość ta jest niezbędna do wyznaczania parametrów w protokołach (np. LEACH, PEGASIS).

Poniższa lista przedstawia zaimplementowane protokoły:

- Komunikacja bezpośrednia (Direct Communication) - wszystkie węzły w sieci komunikują się bezpośrednio z węzłem głównym (‘sink‘).
- LEACH (LEACH) - wszystkie węzły w sieci komunikują się zgodnie z topologią ustaloną w procesie konfiguracji LEACH.

- PEGASIS (PEGASIS) - wszystkie węzły w sieci komunikują się zgodnie z topologią ustaloną w procesie konfiguracji PEGASIS.

4.1.3 Moduł symulatora (Simulator)

Symulator (‘Simulator‘) odpowiada za tworzenie środowisk symulacyjnych. Pojedynczy obiekt symulatora pozwala na zbudowanie wielu scenariuszy symulacji, a następnie ich uruchomienie oraz wygenerowanie metryk. Dane wyjściowe mogą zostać podane dalszej analizie przy wykorzystaniu zewnętrznych narzędzi (np. GNUPlot).

```
type Simulator struct {
  namespace map[string]bool
  config    map[string]*config.Config
  network   map[string]*core.Network

  plotTotalEnergy *plot.Plot // An amount of total energy in the
      network per round.
  plotNodes      *plot.Plot // A number of alive nodes in the
      network per round.
}

func Create() (*Simulator, error)

func (s *Simulator) AddScenario(name string, conf *config.Config)
    error
func (s *Simulator) Run() error
func (s *Simulator) ExportPlots(filepath string) error
func (s *Simulator) ExportGNUPlots(filepath string) error

func createAndPopulateFile(filepath string, data []string) error
func (s *Simulator) create(name string, conf *config.Config) error
func createPlot(title, x, y string) (*plot.Plot, error)
func (s *Simulator) plotter() error
```

Poniższa lista przedstawia oraz opisuje znaczenie poszczególnych zmiennych struktury symulatora (‘Simulator‘):

- przestrzeń nazw (‘namespace‘) - zmienna prywatna przechowuje nazwy symulacji, które w jednoznaczny sposób identyfikują sieć i konfigurację.
- przestrzeń konfiguracji (‘config‘) - zmienna prywatna przechowuje obiekt

konfiguracji (Config) dla każdej symulacji. Kluczem dziennika jest nazwa symulacji.

- przestrzeń sieci (‘network‘) - zmienna prywatna przechowuje obiekt sieci (Network) dla każdej symulacji. Kluczem dziennika jest nazwa symulacji.

Poniższa lista przedstawia oraz opisuje znaczenie poszczególnych funkcji symulatora (‘Simulator‘):

- Setup - funkcja konfiguruje węzły w sieci zgodnie z zaimplementowanym protokołem.
- SetNodes - funkcja definiuje ilość węzłów w sieci. Wartość ta jest niezbędna do wyznaczania parametrów w protokołach (np. LEACH, PEGASIS).
- SetClusters - funkcja definiuje ilość klastrów w sieci. Wartość ta jest niezbędna do wyznaczania parametrów w protokołach (np. LEACH, PEGASIS).

4.2 Obsługa systemu

4.2.1 Konfiguracja środowiska i kompilacja

Poprawna konfiguracja środowiska wymaga instalacji niezbędnych bibliotek i pakietów.

1. Instalacja kompilatora i bibliotek Golang w wersji 1.10.1, lub wyższej.
2. Instalacja kompilatora Protocol Buffer w wersji 3.6, lub wyżej.

Weryfikacja konfiguracji Golang:

```
! Poprawna konfiguracja Golang.
$ which go
/usr/local/go/bin/go
$ go version
go version go1.10.1 linux/amd64
```

Weryfikacja konfiguracji protoc:

```
! Poprawna konfiguracja protoc.
$ which protoc
/usr/local/bin/protoc
$ protoc -version
libprotoc 3.6.0
```

Proces kompilacji (z poziomu folderu z kodem projektu):

```
$ pwd
/home/<username>/go/src/github.com/keadwen/msc_project
$ go build .
```

4.2.2 Generowanie konfiguracji

Wcześniej opisany plik konfiguracyjny jest elementem niezbędnym do uruchomienia symulacji.

Przykładowy plik konfiguracyjny (example.pbtxt) znajdują się w folderze proto:

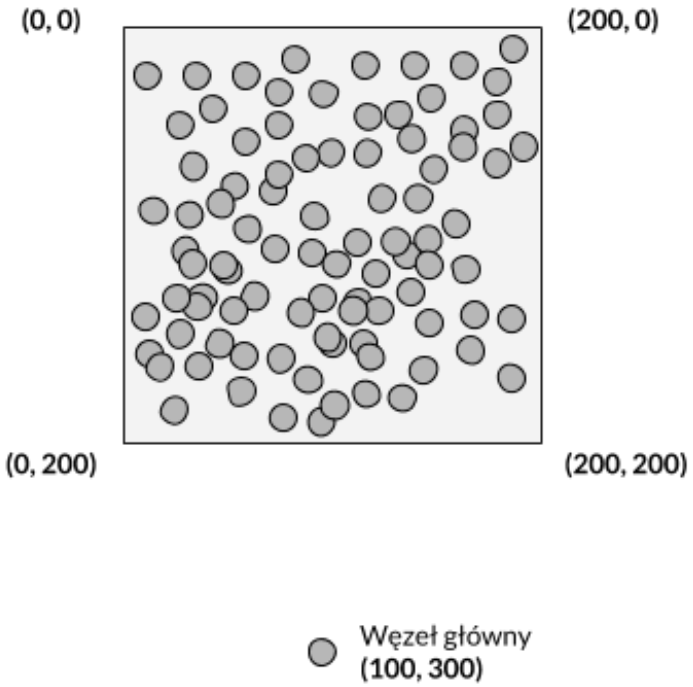
```
protocol: 1
nodes: <
  id: 0
  initial_energy: 1.0e4
  location: <
    X: 0
    Y: 0
  >
  energy_cost <
  >
  time_delay: <
  >
>
nodes: <
  id: 1
  initial_energy: 10.0e-6
  location: <
    X: 300.0
    Y: 0
  >
  energy_cost <
```

```
  >
  time_delay: <
  >
>
nodes <
  id: 2
  initial_energy: 20.0e-6
  location: <
    X: 500.0
    Y: 0
  >
  energy_cost: <
  >
  time_delay: <
  >
>
```

Oprogramowanie umożliwia również generowanie scenariuszy w dynamiczny sposób. W tym przypadku podczas uruchamiania oprogramowania (proces opisany w sekcji 'Uruchamianie scenariuszy') użytkownik nie podaje pliku konfiguracyjnego. Oprogramowanie stworzy identyczny zestaw konfiguracyjny dla każdego zaimplementowanego protokołu. Parametry wygenerowanej konfiguracji dostępne poniżej:

- Protokół: Bezpośrednia komunikacja, LEACH, PEGASIS
- Maksymalna ilość rund: 25000
- Proporcja węzłów typu klaster do wszystkich węzłów: 0.15
- Liczba węzłów głównych ('sink'): 1
- Energia węzła głównego: 100 [J]
- Lokalizacja węzła głównego na osi X: 100
- Lokalizacja węzła głównego na osi Y: 300
- Liczba węzłów: 200
- Energia węzłów: 1 [J]
- Lokalizacja węzłów na osi X: [0, 200]
- Lokalizacja węzłów na osi Y: [0, 200]

Poniższa ilustracja przedstawia rozstawienie węzłów w sieci wygenerowanych dynamicznie:



4.2.3 Uruchamianie scenariuszy

Przykład uruchamiania projektu z dwoma plikami konfiguracyjnymi:

```
$ /go/src/github.com/keadwen/msc_project/msc_project \
    --config_file=example1.proto,example2.proto
```

Przykład uruchamiania projektu bez pliku konfiguracyjnego:

```
$ /go/src/github.com/keadwen/msc_project/msc_project
```

4.2.4 Generowanie wykresów

Generowanie wykresów odbywa się przy wykorzystaniu narzędzia GNUPlot. Proces tworzenia wykresów nie jest zautomatyzowany. Jest to dodatkowa czynność, która musi zostać wykonana manualnie po pomyślnie zakończonym procesie symulacji. W początkowym etapie tworzenia oprogramowania, zastosowane zostało rozwiązanie przy wykorzystaniu biblioteki plotter, która generowała dwa wykresy:

- Wykres ilości aktywnych węzłów w każdej rundzie.

- Wykres energii całkowitej posiadanej przez wszystkie węzły w każdej rundzie.

Rozwiązanie to pomimo zalet związanych z automatycznym tworzeniem wykresów, nie pozwalało na generowanie ich w jakości spełniającej wymagania pracy dyplomowej.

W momencie poprawnego zakończenia symulacji, folder plotdata powinien posiadać zestaw plików w formacie GNU plot. Pliki te należy następnie przekierować do GNUPlot. Poniżej przedstawiona została lista operacji umożliwiająca wygenerowanie wykresów:

```
~/go/src/github.com/keadwen/msc_project$ gnuplot
gnuplot> set grid
gnuplot> plot \
  "< cat ./latex/gnuplot_data/test_1/leach200_m32/*" using 1:2
     smooth sbezier ls 1 title "LEACH (m=32B)", \
  "< cat ./latex/gnuplot_data/test_1/pegasis200_m32/*" using 1:2
     smooth sbezier ls 2 title "PEGASIS (m=32B)", \
  "< cat ./latex/gnuplot_data/test_1/direct200_m32/*" using 1:2
     smooth sbezier ls 4 title "DIRECT (m=32B)"
```

4.2.5 Dodawanie nowych protokołów

Architektura systemu umożliwia tworzenie nowych protokołów wymiany informacji pomiędzy węzłami. Poprawna implementacja wymaga modyfikacji oprogramowania w kilku miejscach:

1. Rozszerzenie definicji enum E_Protocol w proto/config.proto.
2. Rozszerzenie mapProtocol w simulator/simulator.go
3. Stworzenie nowego pliku .go w folderze core. Struktura reprezentująca nowy protokołu musi posiadać zestaw funkcji zgodny z interfejsem Protocol.

Po wykonaniu wszystkich z powyższych kroków, protokół może zostać wykorzystany w symulacji. Należy pamiętać, że oprogramowanie i protocol buffers będą wymagały ponownej kompilacji. W przeciwnym wypadku zmiany nie będą widoczne.

Rozdział 5

Opracowanie wyników eksperymentów

W tym rodziale przedstawione zostaną wyniki uzyskane w procesie symulacji przy wykorzystaniu autorskiego oprogramowania.

5.1 Dane wejściowe

Oprogramowanie posiada implementacje trzech algorytmów: komunikację bezpośrednią, LEACH oraz PEGASIS (dokładniejszy opis działania protokołów został przedstawiony w poprzednich rozdziałach). W celu porównania efektywności energetycznej pomiędzy trzema rozwiązaniami, niezbędne jest precyzyjne zdefiniowanie testów. W systemie posiadamy cztery parametry, których wartości będą modyfikowane i efekty zmian będą porównywane:

- Wielkość wiadomości - parametr ten definiuje wielkość wiadomości wysyłanej pomiędzy węzłami (mierzone w Bajtach).
- Ilość węzłów w sieci - parametr ten definiuje liczbę węzłów w pojedynczej sieci i symulacji.
- Wielkość obszaru sieci - parametr ten definiuje wielkość obszaru wymiany danych w sieci.
- Parametr ‘p‘ - wartość prawdopodobieństwa wypromowania węzła na węzeł typu cluster head.

W celu zapewnienia poprawnej jakości porównywania testów, każdy z scenariuszy posiada dedykowany plik konfiguracyjny. W ten sposób zapewnione jest stworzenie identycznego środowiska testowego, niezależnego od testowanego protokołu.

5.2 Scenariusze

Każdy z testów oraz wartości przedstawione na grafie są wynikami uzyskanymi w procesie wielokrotnego uruchomienia identycznego scenariusza. Ilość uruchomień zdefiniowana dla każdego z testów (domyślenie wynosi 100). W celach testów wygenerowana została ‘konfiguracja domyślna‘, w której zdefiniowana jest lokalizacja 200 unikatowych węzłów o energii początkowej wynoszącej 2 [J], rozdystrybuowanych w sposób pseudolosowy na przestrzeni X: [0, 200], Y: [0, 200]. Lokalizacja węzła typu ‘sink‘ to X:100, Y: 450. Scenariusze wykorzystujące np. 50 węzłów, korzystają z podzbioru węzłów domyślnych, czyli z pierwszych 50 węzłów konfiguracji domyślnej.

Poniższa tabela przedstawia lokalizację 200 węzłów znajdujących się w 200 węzłów w konfiguracji domyślnej.

Tabela lokalizacji węzłów		
Numer węzła	Pozycja X	Pozycja Y
1	199.36168179984253	67.36600933711391
2	88.91310863948563	36.60553297569458
3	19.295149762559976	72.49368923637391
4	181.33926864734022	2.0102636220499934
5	195.51239470437187	111.14328544319282
6	75.89088472075748	146.59505205457364
7	153.23642097371282	57.8575190160736
8	91.11610376949123	126.81760013585746
9	22.168573090360173	17.731684004119654
10	171.53390619509764	138.11474775008918
11	178.78074650585523	80.7496337663683
12	34.99559928880805	59.17180490772036
13	198.34907062745805	141.7869157169524
14	110.5802228177258	160.62083671385693
15	6.589663573066959	73.00097663707615
16	33.38369542490563	87.94381364572581
17	192.71636682180403	29.850811471937412
18	186.64160279465443	4.960927074057211
19	19.103311503198448	37.63818457057886
20	153.59829982210294	104.56247228410325
21	137.0831945195012	45.91688906631662
22	11.917939854883572	152.02973161502598
23	142.44031003121216	22.648902126490743
24	167.57507873516485	63.058120837358466
25	163.86836016322584	80.18689721687169

26	129.33679162442746	36.0562585246913
27	197.36395310621828	143.53930309169507
28	144.7119462938826	156.4741142403833
29	144.6435676420312	71.10641124434568
30	105.01074049743391	158.3798337695642
31	25.59723410509157	135.01628097402926
32	159.35684382610015	31.07145409951828
33	195.4998943585755	78.74753711725046
34	66.17656299978923	197.86588903103794
35	145.05855800949772	30.505805193407117
36	10.405251931328802	178.62885596208508
37	186.2949365520907	12.879258649207234
38	27.331688606572452	64.13705161884809
39	65.98931262074079	167.81945921360898
40	11.740319035407376	115.08640487974763
41	143.27864880590639	175.1440505536343
42	1.021168110783018	85.1441576545403
43	138.17164894151819	112.12873499632224
44	179.1612530841057	184.59869529107561
45	79.69621017689192	107.48873839754025
46	184.27435018999392	67.76831575856659
47	164.0666858114393	158.88321509362515
48	122.17595616791097	131.27859256586234
49	10.604440333326474	61.54801233368917
50	74.72911207259237	163.8145393737179
51	163.824642852362	19.340884972113727
52	60.38344661544656	128.72369102082766
53	154.91415741206794	13.363467080664002
54	173.23978252241287	29.96077828911183
55	35.49837932709627	107.3900801668683
56	5.679935610996528	115.94203226988064
57	107.64533248438022	59.60024137504178
58	123.88900849502306	6.636102135199909
59	127.15338908711131	71.35956883976591
60	66.83544395200622	126.97706574942518
61	47.98012999327529	83.94561033367512
62	176.54577387669218	95.1033047863116
63	122.74547206915622	150.42099582791752
64	188.60947212270017	125.56388147877313
65	167.31399429921188	129.8661815583073
66	192.74321424257602	113.00098456208536

67	59.66450032025723	95.30165349825657
68	102.14207496740319	68.63869717540874
69	120.16610716143093	70.24492600120034
70	137.78811563373782	155.19617715119978
71	24.470724817234647	5.184575998240362
72	43.8705621654678	104.66703173985512
73	13.253805021893925	82.79368905658592
74	14.123609457464465	65.46157949748054
75	197.96011639718708	164.03154695042716
76	199.4757688521392	59.09752667435134
77	176.49465208085067	96.05417914229349
78	176.50488885715166	99.04400119700234
79	132.64305462163384	20.900413261323397
80	176.5890688486683	55.407048824799176
81	18.350890499650514	83.46951690823204
82	105.58091748461553	102.02132605898467
83	169.59807974941262	45.025932649133
84	91.62996074036289	71.07506728413101
85	41.53882313046381	27.95626060211661
86	126.53583705160749	16.57548495738593
87	130.62855665163445	64.32618645648812
88	196.50891780437055	49.196425529157956
89	137.87255271222824	102.95824277715488
90	123.12621895552498	174.96796265673586
91	24.807861660493703	144.5934962380802
92	194.1384235524454	171.3560749350443
93	92.72040033399135	33.20862484294133
94	53.518017397133455	165.27777018747446
95	110.437934352195	186.28501182761832
96	101.09876278713969	32.189509769062894
97	98.07156701924082	124.6875904840318
98	1.9040331497067826	188.00024939659156
99	46.150885396423476	137.81500883535043
100	177.98634098294173	57.79724576017953
101	28.346175929760832	90.54446697252153
102	53.78336249998632	51.92871571977163
103	151.3835071542002	41.317477362666786
104	66.80205068557619	9.18111181695748
105	39.50534861219433	44.122427313386595
106	94.02768478329232	33.35707447001591
107	80.33262457093635	36.045851962231524

108	73.31183316777766	137.7715601112402
109	123.18502187759933	75.25057034738076
110	131.5728567058488	92.57808894243522
111	156.8688785278494	123.4357274952122
112	102.89479352232294	137.20546890272942
113	71.09296009677784	180.16879931073152
114	49.944712048786776	140.08737037990522
115	43.746093590341175	192.77875927001097
116	156.41911737223057	156.8820863460496
117	95.0790408921825	186.9725637190851
118	73.67868942887657	78.59410356376412
119	94.30629261358024	30.148219204604707
120	3.2850631350552413	155.0318308459969
121	184.28082004775192	97.80131098450379
122	38.482242153448595	89.52534507513428
123	24.148929644094125	69.26363721845851
124	96.21308909480939	56.92838777087266
125	149.59459577311887	67.80032045784897
126	192.27050010053873	59.04263769530825
127	121.85846816087307	193.63296526619345
128	86.26177182968621	60.630471185467684
129	15.387170600013992	111.80259677603111
130	82.14863453232417	136.61981109316076
131	194.39160178925673	162.71156399100673
132	147.98543420188847	28.470915776268228
133	27.762796030980713	89.65978227373299
134	185.30337564811637	145.5815784174491
135	33.73228615428452	69.90790135380783
136	140.4354411868145	17.241086286242332
137	63.81350760314003	125.83350552497879
138	85.59301169653692	43.243829604777574
139	103.4650837926725	56.151901362463455
140	27.81743063475214	117.50411165780702
141	40.87014586990372	16.909314045119142
142	55.302768403109695	69.09931885235329
143	85.67648787609042	134.8471182913608
144	35.96476647909297	82.07092729684129
145	163.05452105325094	93.82638959981294
146	56.85714314313321	187.7953338341786
147	67.46209164307635	62.88995698209582
148	37.502844231978635	93.17577891611545

149	181.8706643376419	160.0226423034456
150	95.10828057306911	68.52969346816535
151	161.19258565658868	9.565083989283302
152	48.537515571144155	87.38233511342835
153	9.629064734291505	116.53698640353387
154	170.52196054473708	163.1971902793061
155	83.23778651861842	148.91024991919292
156	140.00313072214695	154.75550886729854
157	37.147993352437595	188.1620606045412
158	8.927735348662594	131.77270933985886
159	181.64147905279125	125.32315535019829
160	12.31235124063389	191.3324999546166
161	98.54813259291232	152.82530490346866
162	101.10265288350571	10.066208381414626
163	129.35968741623685	185.43679021550352
164	10.538280379630459	156.13539005105923
165	17.329080439748225	156.41672906357815
166	78.4045885855135	180.8373074677437
167	2.742143115224063	137.3742617758146
168	139.5575339004708	139.28314823261775
169	164.3606756996568	6.275350032138586
170	144.29341379298802	128.885824619697
171	4.508531144480791	6.256340981494829
172	127.48477137799001	162.96149098444434
173	19.60585389438154	64.39562115571947
174	167.1953365853049	11.415006127185945
175	125.84457010959306	9.1849650086987
176	18.146692688474836	108.47665527426997
177	27.661756044703665	138.0571696600632
178	132.82684329463973	142.8921981101254
179	176.5410953367216	187.96538711106678
180	158.4595296735704	37.448735872827996
181	106.05638261813495	25.086073342347078
182	157.53272233091792	171.76822155338965
183	22.777398787876628	151.98453581828787
184	106.27756263081163	150.12442786389838
185	197.14349605575129	170.6186168464708
186	6.781040491692945	177.8787258459002
187	58.74610836822448	40.306197532384225
188	172.85217790739807	142.0893278172008
189	105.54334798614644	105.3466341523146

190	198.5663924947574	73.58824568518179
191	124.0349644276351	109.31900752872563
192	46.936398211275545	72.30324882020078
193	146.22393181273557	75.29273815758663
194	195.495593662256	53.36494610271914
195	197.47820095322837	190.37079023341855
196	35.96318684562386	166.52359672298164
197	120.22163681844546	26.10858953526762
198	56.72077212977173	97.22674324729
199	132.02125288048953	41.84481484452638
200	154.9545524215235	65.2514440252824

5.2.1 Test 1 - Wielkość wiadomości

Test 'Wielkość wiadomości' ma na celu przedstawienie zależności efektywności energetycznej dla trzech algorytmów dla różnych wielkości wiadomości wysyłanej pomiędzy węzłami.

Lista plików konfiguracyjnych:

```
~/go/src/github.com/keadwen/msc_projec$ tree testdata/test_1/
testdata/test_1/
  direct200_m128.pbtxt
  direct200_m256.pbtxt
  direct200_m32.pbtxt
  direct200_m64.pbtxt
  leach200_m128.pbtxt
  leach200_m256.pbtxt
  leach200_m32.pbtxt
  leach200_m64.pbtxt
  pegasis200_m128.pbtxt
  pegasis200_m256.pbtxt
  pegasis200_m32.pbtxt
  pegasis200_m64.pbtxt

0 directories, 12 files
```

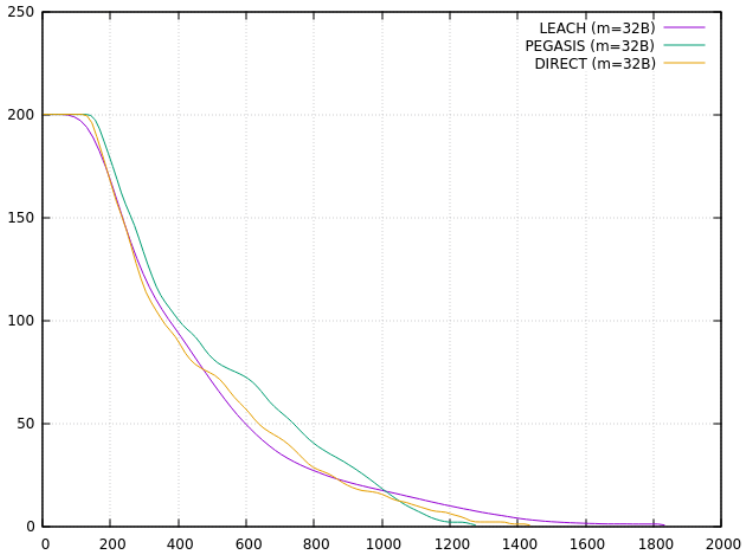
Uruchomienie oprogramowania:

```
~/go/src/github.com/keadwen/msc_project$ ./msc_project
    --repeat_config=100 --config_files=\
```

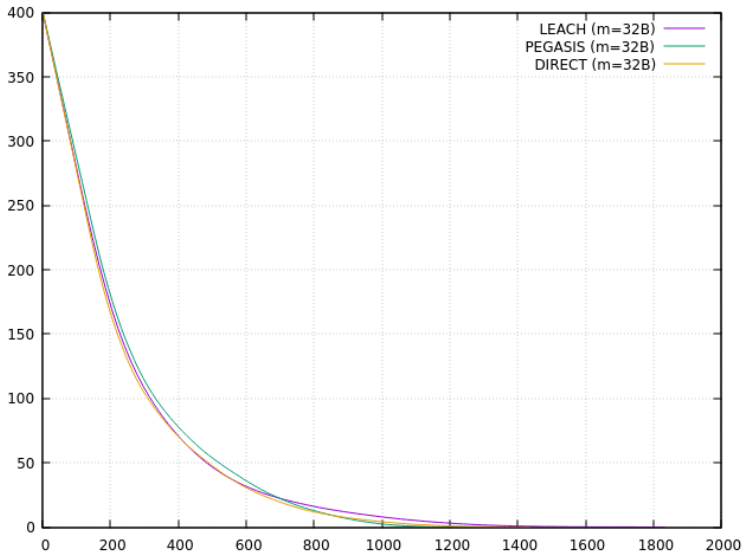
```
testdata/test_1/direct200_m32.pbtxt,\
testdata/test_1/direct200_m64.pbtxt,\
testdata/test_1/direct200_m128.pbtxt,\
testdata/test_1/direct200_m256.pbtxt,\
testdata/test_1/leach200_m32.pbtxt,\
testdata/test_1/leach200_m64.pbtxt,\
testdata/test_1/leach200_m128.pbtxt,\
testdata/test_1/leach200_m256.pbtxt,\
testdata/test_1/pegasis200_m32.pbtxt,\
testdata/test_1/pegasis200_m64.pbtxt,\
testdata/test_1/pegasis200_m128.pbtxt,\
testdata/test_1/pegasis200_m256.pbtxt
```

Tabela uzyskanych wyników:

Uzyskane rezultaty			
Protokół	Wielkość wiadomości (B)	Kwantyl	Wartość
DIRECT	32	-	1434.00
LEACH	32	1	1430.85
		5	1499.10
		10	1518.00
		25	1570.75
		50	1635.00
		75	1696.75
		90	1764.50
		95	1783.05
		99	1805.26
		100	1831.00
PEGASIS	32	1	1135.76
		5	1146.90
		10	1152.00
		25	1165.75
		50	1184.50
		75	1203.00
		90	1224.00
		95	1234.25
		99	1262.12
		100	1274.00



Rysunek 5.1: Funkcja ilości węzłów aktywnych w rundzie (m=32B).



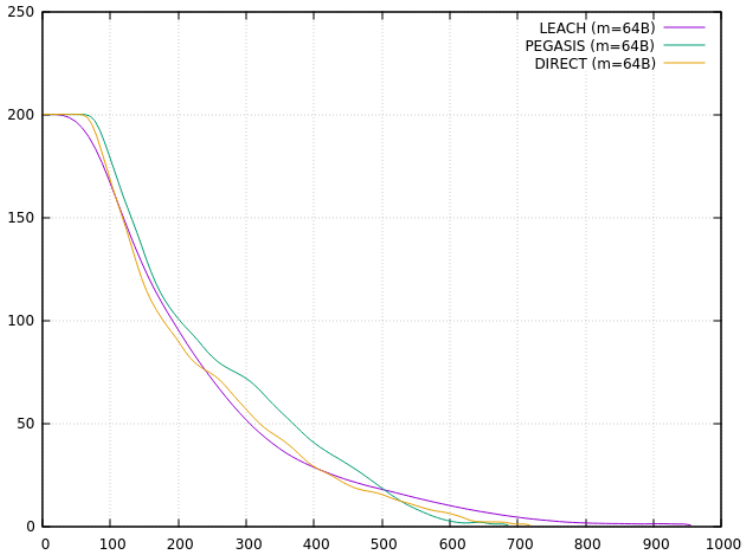
Rysunek 5.2: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (m=32B).

Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 1434. Dla LEACH uzyskano skrócenie czasu pracy na poziomie -1.3% dla wartości minimalnej (1416.00), oraz wydłużenie czasu pracy na poziomie +14.0% dla mediany (1635.00), +27.7% dla wartości maksymalnej (1636.00). Dla PEGASIS uzyskano skrócenie czasu pracy

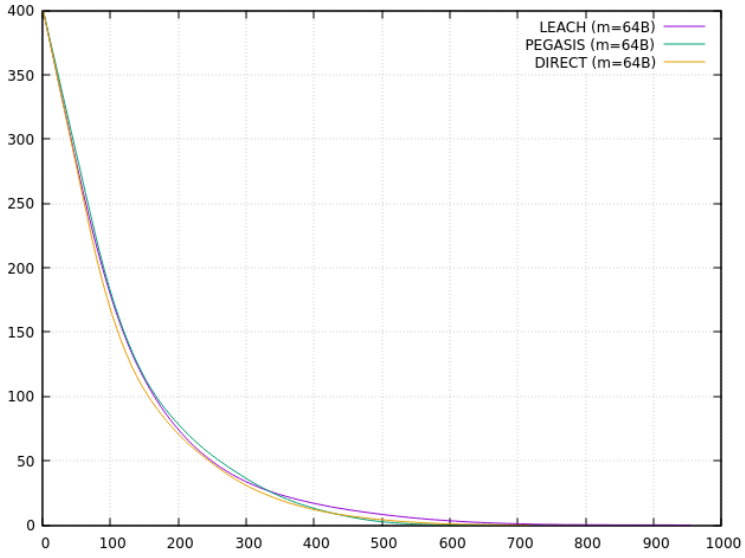
na poziomie -22.5% dla wartości minimalnej (1112.00), -17.4% dla mediany (1184.50), -11.2% dla wartości maksymalnej (1274.00). Pomimo niższej wartości maksymalnej uzyskanych rund pracy węzłów w PEGASIS, ilość węzłów aktywnych w przeciągu symulacji jest wyższa niż w przypadku DIRECT czy LEACH. Zjawisko to pozwala na uzyskanie danych od większej puli węzłów aktywnych.

Rozkład ilości energii całkowitej wszystkich węzłów w rundzie jest zbliżony. Zauważyć należy jednak, że PEGASIS cechuje się lepszą sprawnością energetyczną przez pierwsze 1000 rund.

Uzyskane rezultaty			
Protokół	Wielkość wiadomości (B)	Kwantyl	Wartość
DIRECT	64	-	717.00
LEACH	64	1	774.87
		5	759.65
		10	769.80
		25	800.75
		50	830.50
		75	872.25
		90	893.20
		95	908.40
		99	935.19
		100	954.00
PEGASIS	64	1	575.95
		5	583.95
		10	587.00
		25	594.00
		50	607.50
		75	622.25
		90	629.10
		95	635.10
		99	669.16
		100	685.00



Rysunek 5.3: Funkcja ilości węzłów aktywnych w rundzie (m=64B).



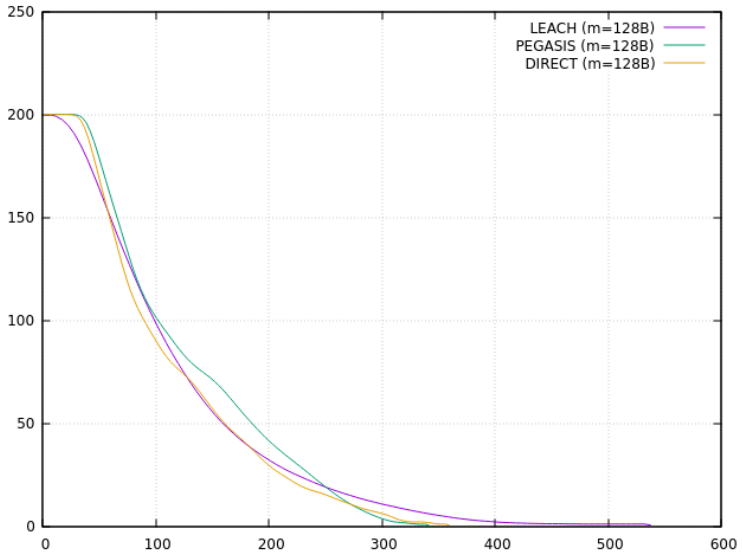
Rysunek 5.4: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (m=64B).

Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 717. Dla LEACH uzyskano wydłużenie czasu pracy na poziomie +2.1% dla wartości minimalnej (732.00), +15.8% dla mediany (830.50), +31.1% dla wartości maksymalnej (954.00). Dla PEGASIS uzyskano skrócenie czasu pracy na poziomie -20.4% dla wartości minimal-

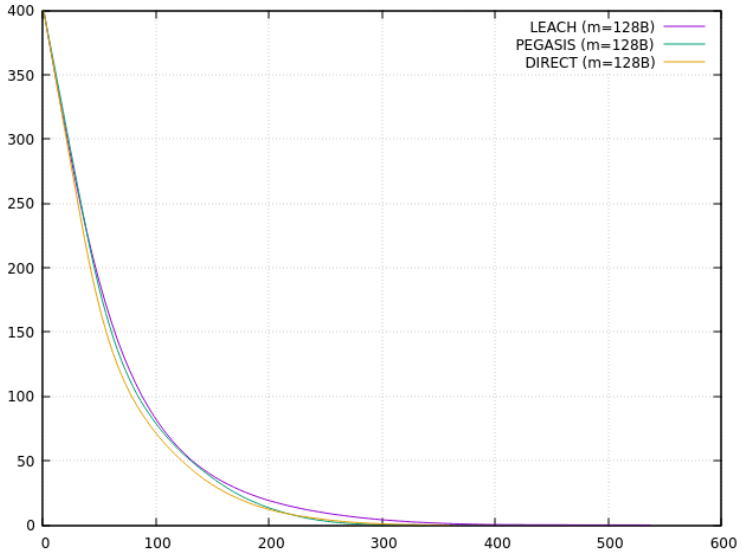
nej (571.00), -15.3% dla mediany (607.50), -4.5% dla wartości maksymalnej (685.00). Pomimo niższej wartości maksymalnej uzyskanych rund pracy węzłów w PEGASIS, ilość węzłów aktywnych w przeciągu symulacji jest wyższa niż w przypadku DIRECT czy LEACH. Zjawisko to pozwala na uzyskanie danych od większej puli węzłów aktywnych.

Rozkład ilości energii całkowitej wszystkich węzłów w rundzie jest zbliżony. Zauważyć należy jednak, że PEGASIS cechuje się lepszą sprawnością energetyczną przez pierwsze 500 rund.

Uzyskane rezultaty			
Protokół	Wielkość wiadomości (B)	Kwantyl	Wartość
DIRECT	128	-	359.00
LEACH	128	1	387.85
		5	391.95
		10	398.70
		25	408.75
		50	432.00
		75	453.25
		90	478.60
		95	492.05
		99	508.29
		100	537.00
PEGASIS	128	1	290.99
		5	295.00
		10	299.90
		25	305.00
		50	312.00
		75	318.00
		90	327.00
		95	330.10
		99	336.05
		100	341.00



Rysunek 5.5: Funkcja ilości węzłów aktywnych w rundzie (m=128B).



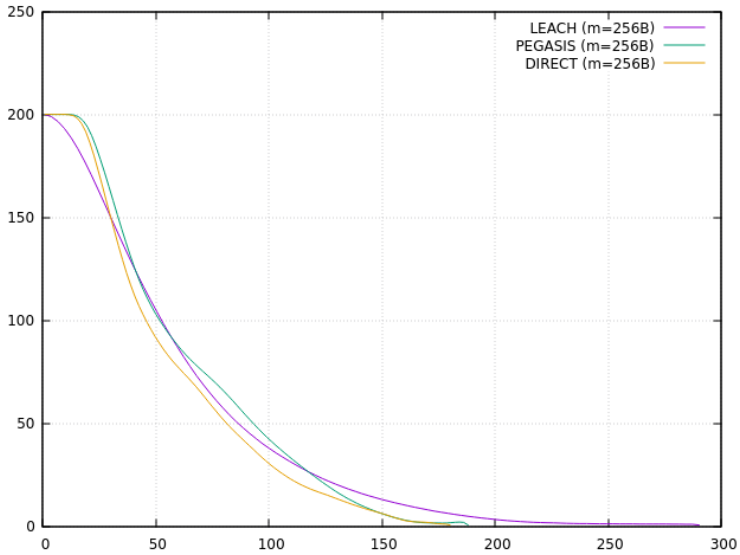
Rysunek 5.6: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (m=128B).

Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 359. Dla LEACH uzyskano wydłużenie czasu pracy na poziomie +3.9% dla wartości minimalnej (373.00), +20.3% dla mediany (432.00), +49.6% dla wartości maksymalnej (537.00). Dla PEGASIS uzyskano skrócenie czasu pracy na poziomie -19.2% dla wartości minimal-

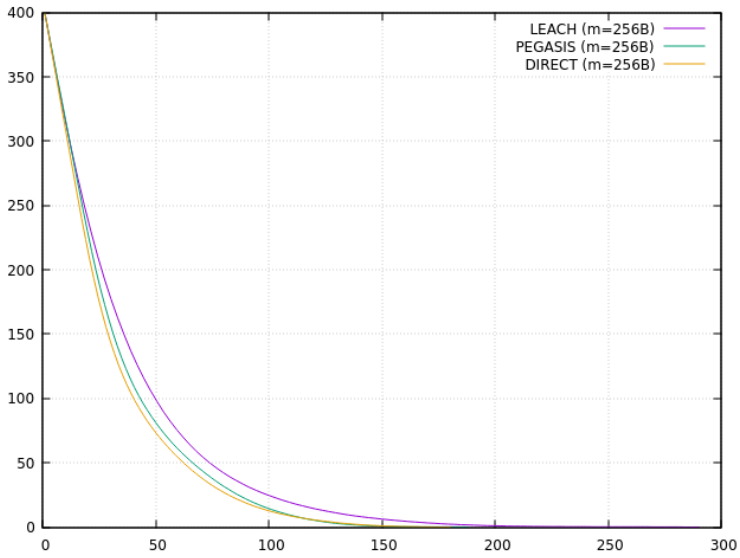
nej (290.00), -13.1% dla mediany (312.00), -5.0% dla wartości maksymalnej (341.00). Pomimo niższej wartości maksymalnej uzyskanych rund pracy węzłów w PEGASIS, ilość węzłów aktywnych w przeciągu symulacji jest wyższa niż w przypadku DIRECT czy LEACH. Zjawisko to pozwala na uzyskanie danych od większej puli węzłów aktywnych.

Rozkład ilości energii całkowitej wszystkich węzłów w rundzie jest zbliżony. Zauważyć należy jednak, że PEGASIS cechuje się lepszą sprawnością energetyczną przez pierwsze 250 rund.

Uzyskane rezultaty			
Protokół	Wielkość wiadomości (B)	Kwantyl	Wartość
DIRECT	256	-	180.00
LEACH	256	1	206.88
		5	211.90
		10	214.90
		25	224.00
		50	239.00
		75	252.00
		90	267.40
		95	281.10
		99	289.01
		100	290.00
PEGASIS	256	1	154.97
		5	157.00
		10	158.00
		25	162.00
		50	166.00
		75	170.00
		90	175.10
		95	179.00
		99	181.08
		100	188.00



Rysunek 5.7: Funkcja ilości węzłów aktywnych w rundzie (m=256B).



Rysunek 5.8: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (m=256B).

Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 180. Dla LEACH uzyskano wydłużenie czasu pracy na poziomie +8.3% dla wartości minimalnej (195.00), +32.8% dla mediany (239.00), +61.1% dla wartości maksymalnej (290.00). Dla PEGASIS uzyskano skrócenie czasu pracy na poziomie -15.6% dla wartości minimalnej

(152.00), -7.8% dla mediany (166.00), oraz wydłużenie czasu pracy na poziomie +4.4% dla wartości maksymalnej (188.00). Pomimo niższej wartości maksymalnej uzyskanych rund pracy węzłów w PEGASIS, ilość węzłów aktywnych w przeciągu symulacji jest wyższa niż w przypadku DIRECT czy LEACH. Zjawisko to pozwala na uzyskanie danych od większej puli węzłów aktywnych.

Rozkład ilości energii całkowitej wszystkich węzłów w rundzie jest zbliżony. Zauważyć należy jednak, że PEGASIS cechuje się lepszą sprawnością energetyczną przez pierwsze 120 rund.

5.2.2 Test 2 - Ilość węzłów w sieci

Test 'Ilość węzłów w sieci' ma na celu przedstawienie zależności efektywności energetycznej dla trzech algorytmów względem ilości węzłów znajdujących się w sieci.

List plików konfiguracyjnych:

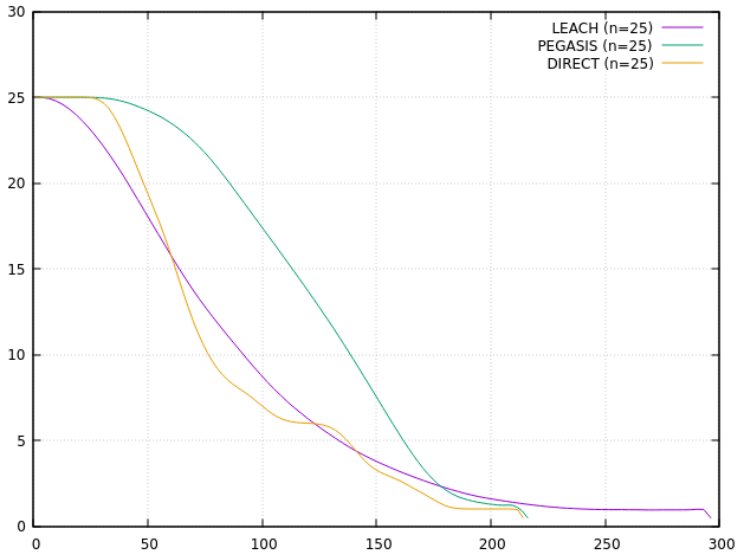
```
~/go/src/github.com/keadwen/msc_project$ tree testdata/test_2/
testdata/test_2/
  direct100.pbtxt
  direct150.pbtxt
  direct200.pbtxt
  direct25.pbtxt
  direct50.pbtxt
  leach100.pbtxt
  leach150.pbtxt
  leach200.pbtxt
  leach25.pbtxt
  leach50.pbtxt
  pegasis100.pbtxt
  pegasis150.pbtxt
  pegasis200.pbtxt
  pegasis25.pbtxt
  pegasis50.pbtxt

0 directories, 15 files
```

Uruchomienie oprogramowania:

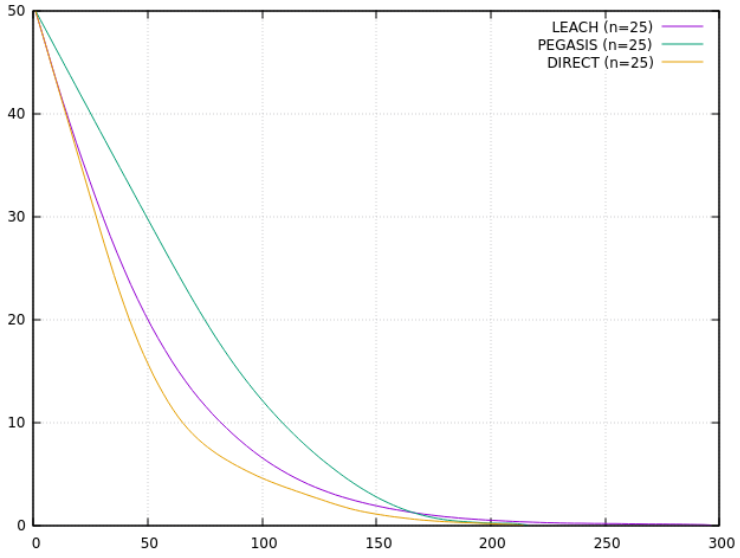
```
~/go/src/github.com/keadwen/msc_project$ ./msc_project
    --repeat_config=100 --config_files=\
```

```
testdata/test_2/direct25.pbtxt,\
testdata/test_2/direct50.pbtxt,\
testdata/test_2/direct100.pbtxt,\
testdata/test_2/direct150.pbtxt,\
testdata/test_2/direct200.pbtxt,\
testdata/test_2/leach25.pbtxt,\
testdata/test_2/leach50.pbtxt,\
testdata/test_2/leach100.pbtxt,\
testdata/test_2/leach150.pbtxt,\
testdata/test_2/leach200.pbtxt,\
testdata/test_2/pegasis25.pbtxt,\
testdata/test_2/pegasis50.pbtxt,\
testdata/test_2/pegasis100.pbtxt,\
testdata/test_2/pegasis150.pbtxt,\
testdata/test_2/pegasis200.pbtxt
```

Uzyskane rezultaty			
Protokół	Ilość węzłów	Kwantyl	Wartość
DIRECT	25	-	214.00
LEACH	25	1	178.99
		5	185.90
		10	196.90
		25	211.75
		50	229.50
		75	246.00
		90	263.20
		95	272.00
		99	287.09
		100	296.00
PEGASIS	25	1	169.95
		5	171.95
		10	175.90
		25	181.75
		50	186.50
		75	195.00
		90	201.10
		95	206.00
		99	215.01
		100	216.00



Rysunek 5.9: Funkcja ilości węzłów aktywnych w rundzie (n=25).



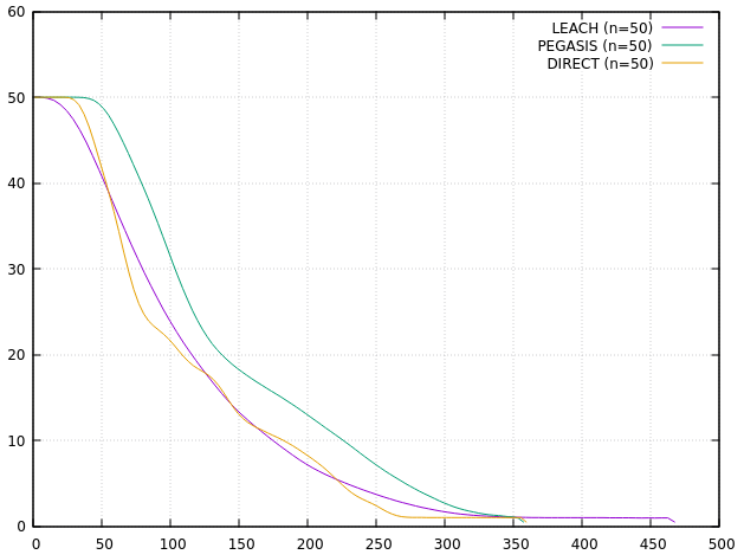
Rysunek 5.10: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (n=25).

Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 214 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano skrócenie czasu pracy na poziomie -16.8% dla wartości minimalnej (178.00), oraz wydłużenie czasu pracy na poziomie +7.2% dla mediany (229.50), +38.3% dla wartości mak-

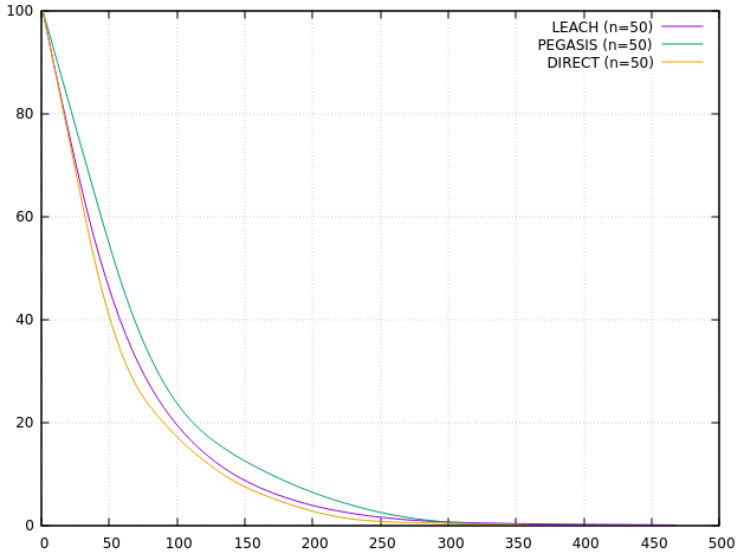
symalnej (296.00). Dla PEGASIS uzyskano skrócenie czasu pracy na poziomie -32.9% dla wartości minimalnej (165.00), -12.9% dla mediany (186.50), oraz wydłużenie czasu pracy na poziomie +0.9% dla wartości maksymalnej (216.00). Pomimo niższej wartości maksymalnej uzyskanych rund pracy węzłów w PEGASIS, ilość węzłów aktywnych w przeciągu symulacji jest wyższa niż w przypadku DIRECT czy LEACH. Zjawisko to pozwala na uzyskanie danych od większej puli węzłów aktywnych.

Rozkład ilości energii całkowitej wszystkich węzłów uwidacznia różnice pomiędzy efektywnością energetyczną protokołów. Zauważyć można, że PEGASIS cechuje się lepszą sprawnością energetyczną przez pierwsze 150 rund.

Uzyskane rezultaty			
Protokół	Ilość węzłów	Kwantyl	Wartość
DIRECT	50	-	359.00
LEACH	50	1	293.85
		5	307.00
		10	321.40
		25	356.75
		50	382.00
		75	410.00
		90	435.40
		95	443.20
		99	461.06
		100	467.00
PEGASIS	50	1	284.87
		5	291.90
		10	301.90
		25	310.00
		50	320.00
		75	331.50
		90	342.30
		95	348.00
		99	354.03
		100	357.00



Rysunek 5.11: Funkcja ilości węzłów aktywnych w rundzie (n=50).



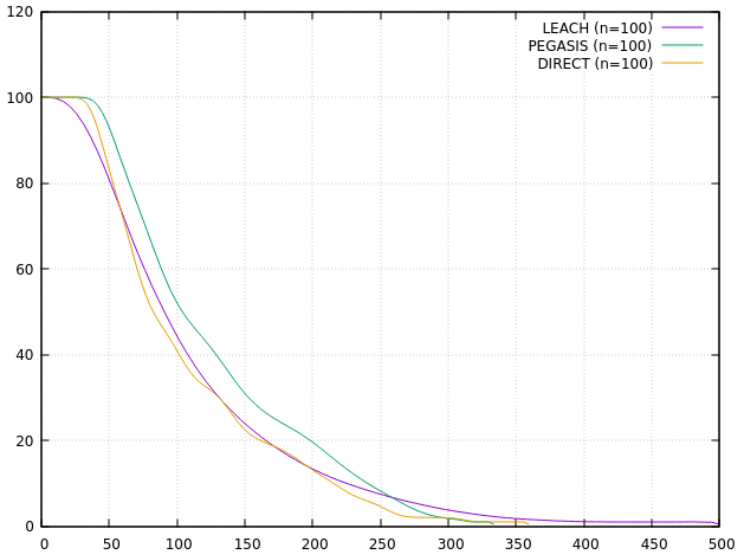
Rysunek 5.12: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (n=50).

Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 359 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano skrócenie czasu pracy na poziomie -22.3% dla wartości minimalnej (279.00), oraz wydłużenie czasu pracy na poziomie +6.4% dla mediany (382.00), +30.1% dla wartości mak-

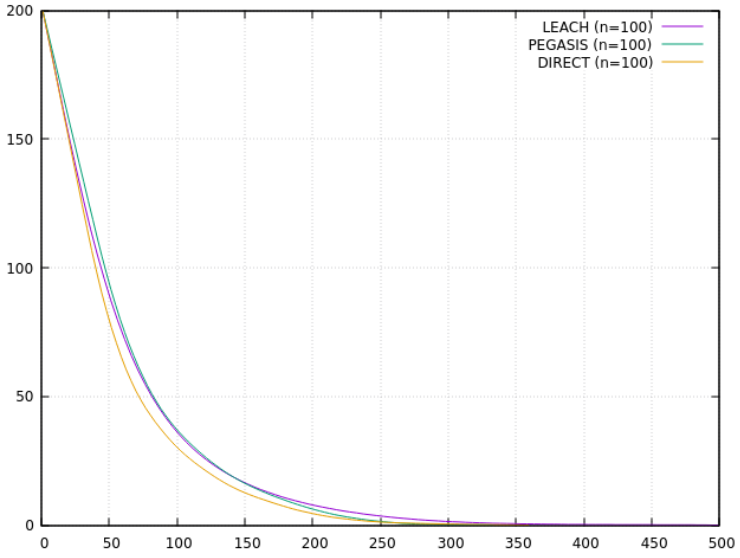
symalnej (467.00). Dla PEGASIS uzyskano skrócenie czasu pracy na poziomie -34.2% dla wartości minimalnej (272.00), -10.9% dla mediany (320.00), -0.6% dla wartości maksymalnej (357.00). Pomimo niższej wartości maksymalnej uzyskanych rund pracy węzłów w PEGASIS, ilość węzłów aktywnych w przeciągu symulacji jest wyższa niż w przypadku DIRECT czy LEACH. Zjawisko to pozwala na uzyskanie danych od większej puli węzłów aktywnych.

Rozkład ilości energii całkowitej wszystkich węzłów w rundzie jest zbliżony. Zauważyć można, że PEGASIS cechuje się lepszą sprawnością energetyczną przez pierwsze 250 rund. Komunikacja bezpośrednia jest widocznie mniej efektywna.

Uzyskane rezultaty			
Protokół	Ilość węzłów	Kwantyl	Wartość
DIRECT	100	-	359.00
LEACH	100	1	325.98
		5	346.75
		10	355.90
		25	384.75
		50	412.50
		75	439.25
		90	464.30
		95	476.60
		99	493.06
		100	499.00
PEGASIS	100	1	268.98
		5	274.00
		10	278.00
		25	284.50
		50	291.00
		75	301.00
		90	311.10
		95	314.15
		99	319.14
		100	333.00



Rysunek 5.13: Funkcja ilości węzłów aktywnych w rundzie (n=100).



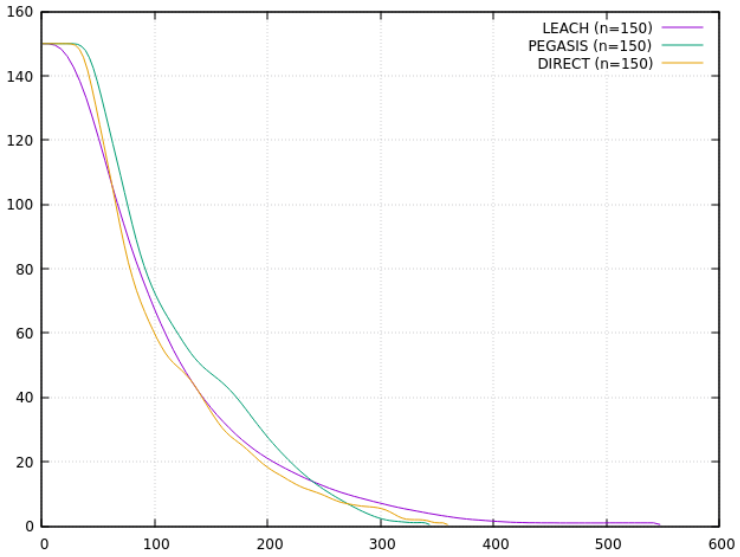
Rysunek 5.14: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (n=100).

Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 359 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano skrócenie czasu pracy na poziomie -9.7% dla wartości minimalnej (324.00), oraz wydłużenie czasu pracy na poziomie +14.9% dla mediany (412.50), +39.0% dla wartości mak-

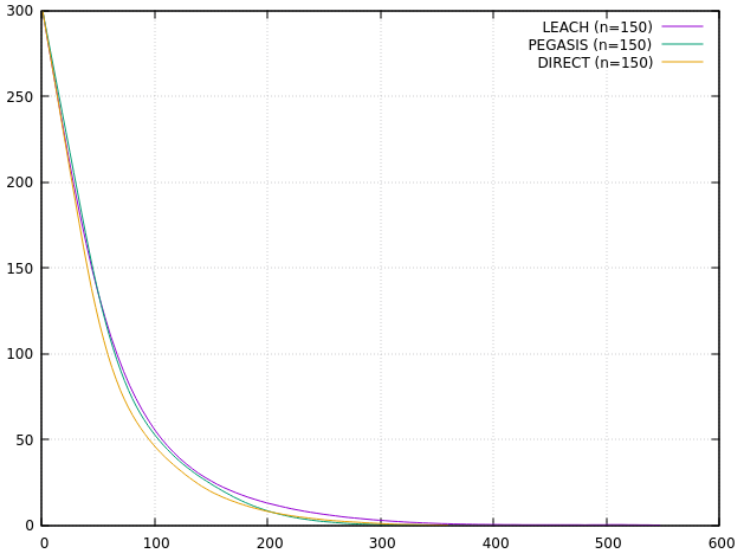
symalnej (499.00). Dla PEGASIS uzyskano skrócenie czasu pracy na poziomie -25.6% dla wartości minimalnej (267.00), -18.9% dla mediany (291.00), -7.2% dla wartości maksymalnej (333.00). Pomimo niższej wartości maksymalnej uzyskanych rund pracy węzłów w PEGASIS, ilość węzłów aktywnych w przeciągu symulacji jest wyższa niż w przypadku DIRECT czy LEACH. Zjawisko to pozwala na uzyskanie danych od większej puli węzłów aktywnych.

Rozkład ilości energii całkowitej wszystkich węzłów w rundzie dla LEACH i PEGASIS jest niemal identyczny. Komunikacja bezpośrednia jest widocznie mniej efektywna.

Uzyskane rezultaty			
Protokół	Ilość węzłów	Kwantyl	Wartość
DIRECT	150	-	359.00
LEACH	150	1	367.92
		5	377.85
		10	383.80
		25	398.75
		50	424.50
		75	446.50
		90	468.20
		95	484.15
		99	499.48
		100	547.00
PEGASIS	150	1	286.99
		5	292.90
		10	296.80
		25	303.50
		50	312.00
		75	322.25
		90	330.00
		95	333.00
		99	339.04
		100	343.00



Rysunek 5.15: Funkcja ilości węzłów aktywnych w rundzie (n=150).



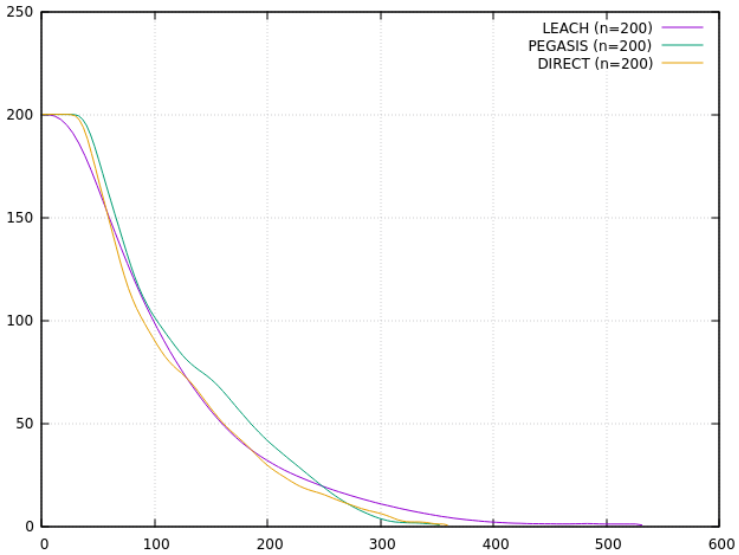
Rysunek 5.16: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (n=150).

Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 359 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano wydłużenie czasu pracy na poziomie +0.3% dla wartości minimalnej (360.00), +18.2% dla mediany (424.50), +52.4% dla wartości maksymalnej (547.00). Dla PEGASIS uzy-

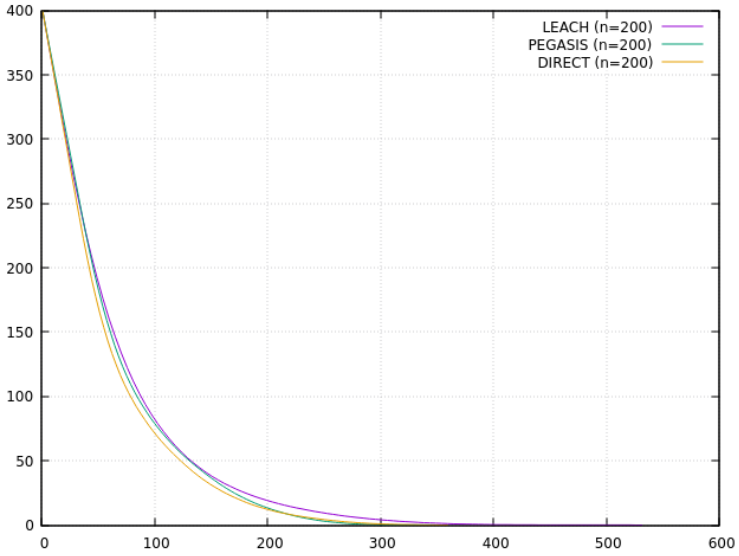
skano skrócenie czasu pracy na poziomie -20.3% dla wartości minimalnej (286.00), -13.1% dla mediany (312.00), -4.5% dla wartości maksymalnej (343.00). Pomimo niższej wartości maksymalnej uzyskanych rund pracy węzłów w PEGASIS, ilość węzłów aktywnych w przeciągu symulacji jest wyższa niż w przypadku DIRECT czy LEACH. Zjawisko to pozwala na uzyskanie danych od większej puli węzłów aktywnych.

Rozkład ilości energii całkowitej wszystkich węzłów w rundzie jest niemal identyczny.

Uzyskane rezultaty			
Protokół	Ilość węzłów	Kwantyl	Wartość
DIRECT	200	-	359.00
LEACH	200	1	381.91
		5	398.00
		10	405.00
		25	415.00
		50	429.50
		75	449.25
		90	467.10
		95	475.05
		99	499.32
		100	531.00
PEGASIS	200	1	296.96
		5	298.95
		10	302.00
		25	307.00
		50	313.00
		75	321.25
		90	329.00
		95	335.20
		99	349.03
		100	352.00



Rysunek 5.17: Funkcja ilości węzłów aktywnych w rundzie (n=200).



Rysunek 5.18: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (n=200).

Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 359 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano wydłużenie czasu pracy na poziomie +3.9% dla wartości minimalnej (373.00), +19.6% dla mediany (429.50), +47.9% dla wartości maksymalnej (531.00). Dla PEGASIS uzy-

skano skrócenie czasu pracy na poziomie -18.4% dla wartości minimalnej (293.00), -12.8% dla mediany (313.00), -1.9% dla wartości maksymalnej (352.00). Pomimo niższej wartości maksymalnej uzyskanych rund pracy węzłów w PEGASIS, ilość węzłów aktywnych w przeciągu symulacji jest wyższa niż w przypadku DIRECT czy LEACH. Zjawisko to pozwala na uzyskanie danych od większej puli węzłów aktywnych.

Rozkład ilości energii całkowitej wszystkich węzłów w rundzie jest niemal identyczny.

5.2.3 Test 3 - Wielkość obszaru sieci

Test 'Wielkość obszaru sieci' ma na celu przedstawienie zależności efektywności energetycznej dla trzech algorytmów względem wielkości obszaru sieci na której znajdują się węzły w sieci.

Istotna informacja: lokalizacja węzłów w scenariuszu 200 jest niezmienna względem pozostałych testów. Wielkość obszaru sieci była manipulowana przy użyciu skali (0.25, 0.50, 2.00, 4.00). Oznacza to, że koordynaty (X, Y) węzłów zostały przemnożone przez wartość skali. Proporcja odległości pomiędzy węzłami zostaje taka sama, jednakże zmianie ulega koszt komunikacji, gdyż funkcja kosztów energetycznych jest zależna od dystansu.

List plików konfiguracyjnych:

```
~/go/src/github.com/keadwen/msc_project$ tree testdata/test_3/
testdata/test_3/
  direct_xy050.pbtxt
  direct_xy100.pbtxt
  direct_xy200.pbtxt
  direct_xy400.pbtxt
  direct_xy800.pbtxt
  leach_xy050.pbtxt
  leach_xy100.pbtxt
  leach_xy200.pbtxt
  leach_xy400.pbtxt
  leach_xy800.pbtxt
  pegasis_xy050.pbtxt
  pegasis_xy100.pbtxt
  pegasis_xy200.pbtxt
  pegasis_xy400.pbtxt
  pegasis_xy800.pbtxt
```

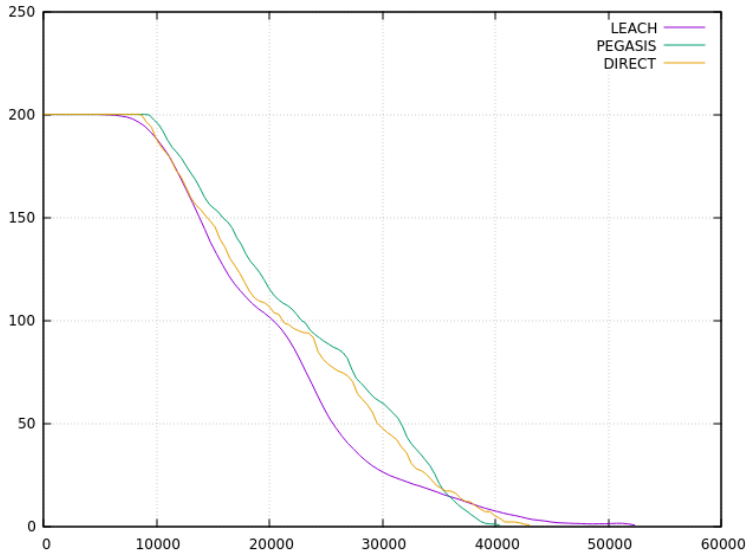
Uruchomienie oprogramowania:

```
~/go/src/github.com/keadwen/msc_project$ ./msc_project
   --repeat_config=100 --config_files=\
testdata/test_3/direct_xy050.pbtxt,\
testdata/test_3/direct_xy100.pbtxt,\
testdata/test_3/direct_xy200.pbtxt,\
testdata/test_3/direct_xy400.pbtxt,\
testdata/test_3/direct_xy800.pbtxt,\
testdata/test_3/leach_xy050.pbtxt,\
testdata/test_3/leach_xy100.pbtxt,\
testdata/test_3/leach_xy200.pbtxt,\
testdata/test_3/leach_xy400.pbtxt,\
testdata/test_3/leach_xy800.pbtxt,\
testdata/test_3/pegasis_xy050.pbtxt,\
testdata/test_3/pegasis_xy100.pbtxt,\
testdata/test_3/pegasis_xy200.pbtxt,\
testdata/test_3/pegasis_xy400.pbtxt,\
testdata/test_3/pegasis_xy800.pbtxt
```

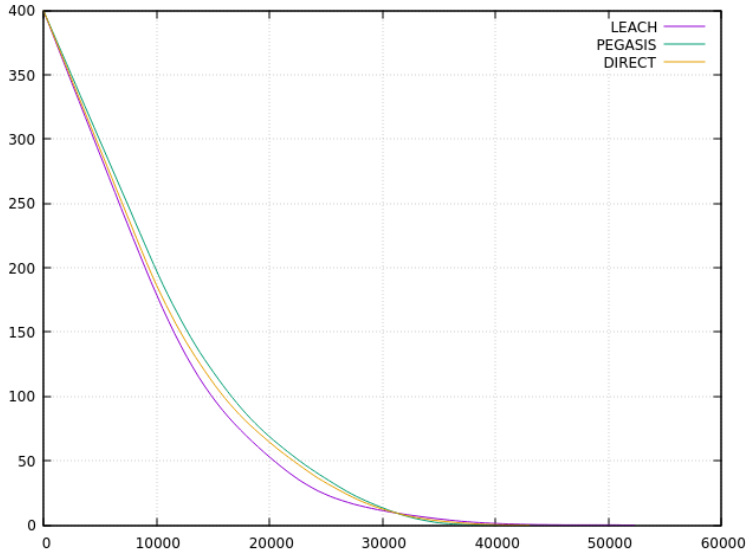
Poniższa tabela przedstawia wyniki uzyskane w ramach symulacji, powtórzonej 100 razy na każdym z plików konfiguracyjnych.

Uzyskane rezultaty			
Protokół	Skala sieci	Kwantyl	Wartość
DIRECT	0.25	-	42976.00
LEACH	0.25	1	43417.42
		5	44197.10
		10	45130.10
		25	45789.50
		50	46909.50
		75	48151.00
		90	49309.70
		95	50096.95
		99	51908.73
		100	52278.00

Uzyskane rezultaty			
Protokół	Skala sieci	Kwantyl	Wartość
PEGASIS	0.25	1	38521.39
		5	38723.70
		10	38847.20
		25	39050.25
		50	39278.50
		75	39543.75
		90	39756.10
		95	39827.65
		99	40037.81
		100	40316.00



Rysunek 5.19: Funkcja ilości węzłów aktywnych w rundzie.

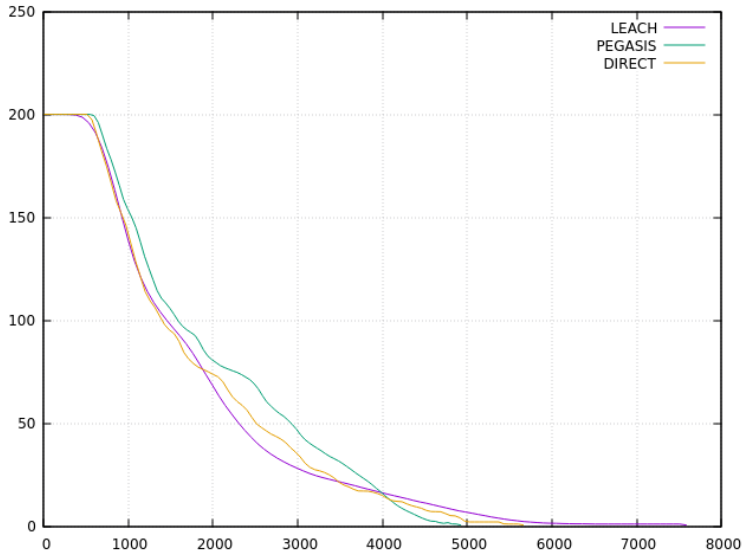


Rysunek 5.20: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie.

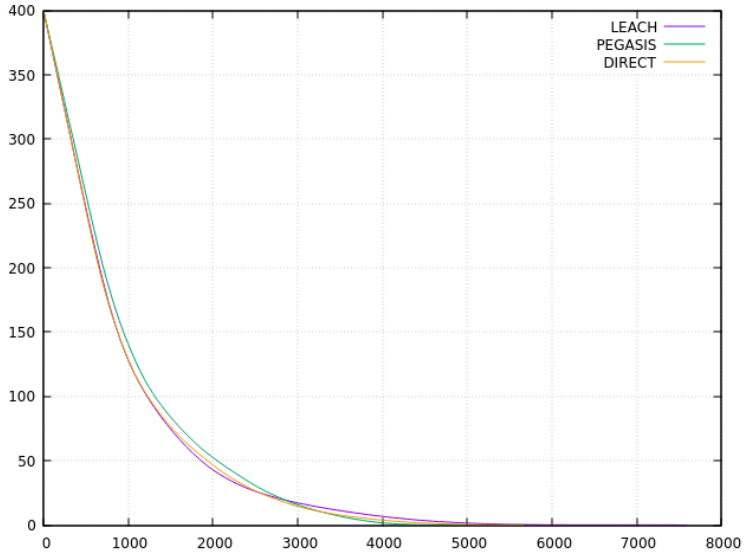
Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 42976 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano wydłużenie czasu pracy na poziomie +0.9% dla wartości minimalnej (43360.00), +9.2% dla mediany (46909.50), +21.6% dla wartości maksymalnej (52278.00). Dla PEGASIS uzyskano skrócenie czasu pracy na poziomie -10.7% dla wartości mi-

nimalnej (38362.00), -8.6% dla mediany (39278.50), -6.2% dla wartości maksymalnej (40316.00). Odległości węzłów do 'sink' są czterokrotnie mniejsze niż w konfiguracji podstawowej. Czas pracy węzłów uległ znaczącemu wydłużeniu, gdyż koszt energetyczny transmisji jest niższy.

Uzyskane rezultaty			
Protokół	Skala sieci	Kwantyl	Wartość
DIRECT	0.50	-	5657.00
LEACH	0.50	1	5598.30
		5	5744.20
		10	5872.20
		25	6035.00
		50	6292.50
		75	6563.50
		90	6779.70
		95	6919.75
		99	7213.70
		100	7580.00
PEGASIS	0.50	1	4406.99
		5	4454.90
		10	4472.70
		25	4516.00
		50	4576.50
		75	4627.00
		90	4691.20
		95	4708.45
		99	4776.45
		100	4920.00



Rysunek 5.21: Funkcja ilości węzłów aktywnych w rundzie.

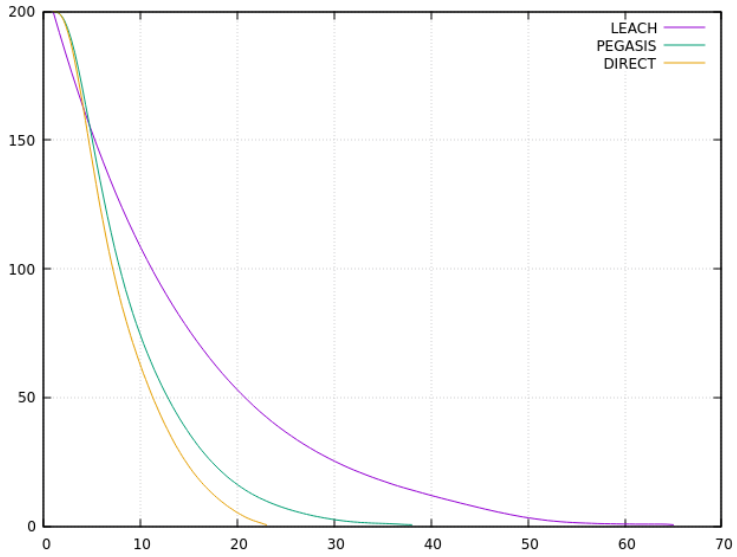


Rysunek 5.22: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie.

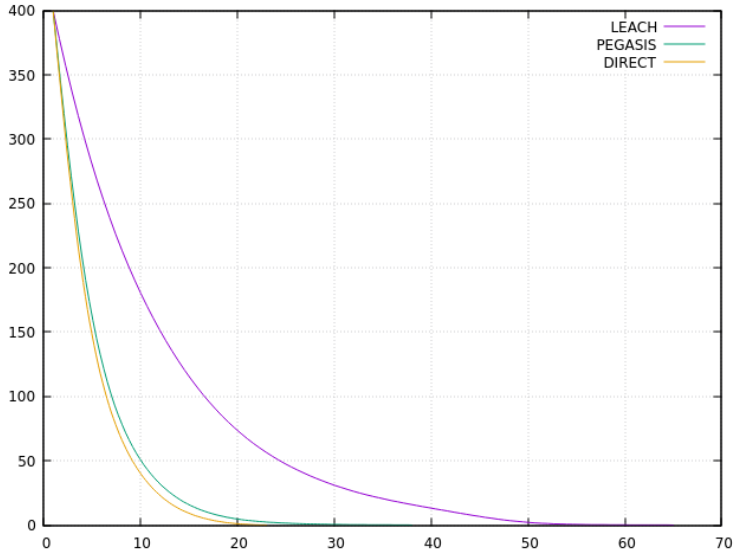
Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 5657 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano skrócenie czasu pracy na poziomie -2.3% dla wartości minimalnej (5529.00), oraz wydłużenie czasu pracy na poziomie +11.2% dla mediany (6292.50), +34.0% dla wartości maksymalnej (7580.00). Dla PEGASIS uzyskano skrócenie czasu pracy

na poziomie -22.1% dla wartości minimalnej (4406.00), -19.1% dla mediany (4576.50), -13.0% dla wartości maksymalnej (4920.00). Odległości węzłów do 'sink' są dwukrotnie mniejsze niż w konfiguracji podstawowej. Czas pracy węzłów uległ znaczącemu wydłużeniu, gdyż koszt energetyczny transmisji jest niższy.

Uzyskane rezultaty			
Protokół	Skala sieci	Kwantyl	Wartość
DIRECT	2.00	-	23.00
LEACH	2.00	1	49.00
		5	50.95
		10	51.00
		25	53.00
		50	55.00
		75	58.00
		90	60.00
		95	61.05
		99	63.02
		100	65.00
PEGASIS	2.00	1	27.00
		5	28.95
		10	29.00
		25	30.00
		50	32.00
		75	33.00
		90	34.10
		95	36.05
		99	37.01
		100	38.00



Rysunek 5.23: Funkcja ilości węzłów aktywnych w rundzie.

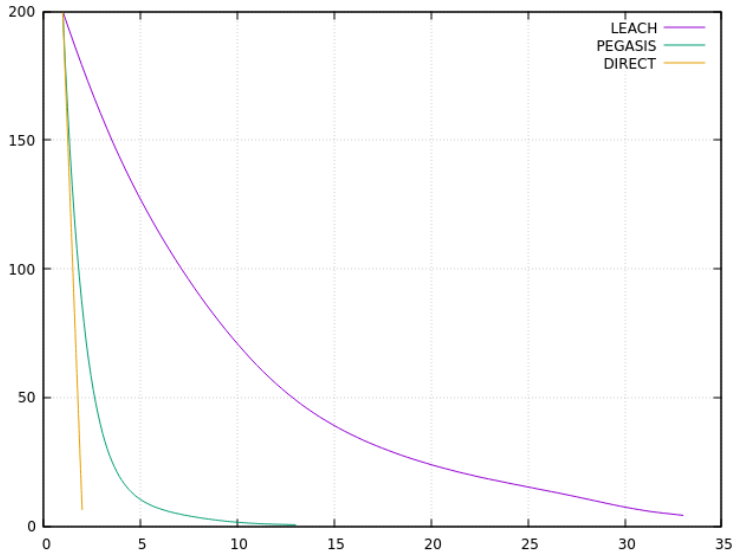


Rysunek 5.24: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie.

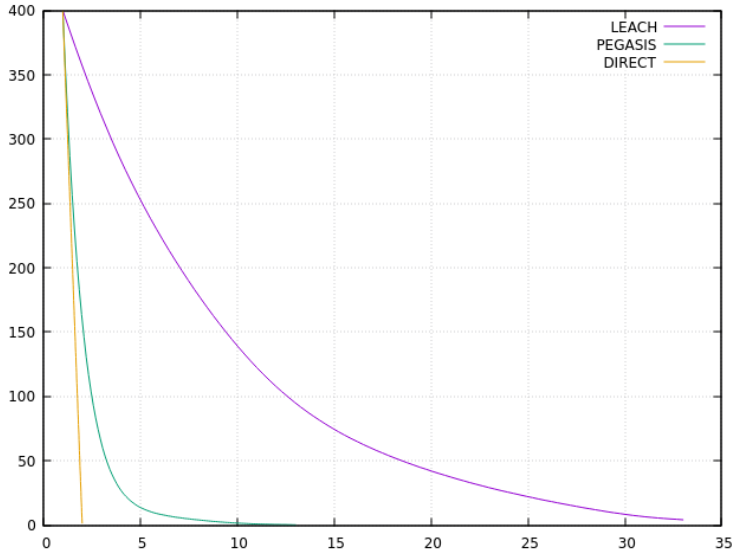
Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 23 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano wydłużenie czasu pracy na poziomie +213.0% dla wartości minimalnej (49.00), +239.1% dla mediany (55.00), +282.6% dla wartości maksymalnej (65.00). Dla PEGASIS uzyskano wydłużenie czasu pracy na poziomie +117.4% dla wartości mini-

malnej (27.00), +139.1% dla mediany (32.00), +165.2% dla wartości maksymalnej (38.00). Odległości węzłów do 'sink' są dwukrotnie większa niż w konfiguracji podstawowej. Czas pracy węzłów uległ znaczącemu skróceniu, gdyż koszt energetyczny transmisji jest wyższy.

Uzyskane rezultaty			
Protokół	Skala sieci	Kwantyl	Wartość
DIRECT	4.00	-	2.00
LEACH	4.00	1	27.00
		5	28.00
		10	29.00
		25	29.00
		50	30.00
		75	31.00
		90	32.00
		95	32.00
		99	33.00
		100	33.00
PEGASIS	4.00	1	7.00
		5	8.00
		10	8.90
		25	9.00
		50	10.00
		75	11.00
		90	11.00
		95	12.00
		99	13.00
		100	13.00



Rysunek 5.25: Funkcja ilości węzłów aktywnych w rundzie.



Rysunek 5.26: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie.

Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 2 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano wydłużenie czasu pracy na poziomie +1350.0% dla wartości minimalnej (27.00), +1500.0% dla mediany (30.00), +1650.0% dla wartości maksymalnej (33.00). Dla PEGASIS uzyskano wydłużenie czasu pracy na poziomie +350.0% dla wartości minimalnej

(7.00), +500.0% dla mediany (10.00), +650.0% dla wartości maksymalnej (13.00). Odległości węzłów do 'sink' są czterokrotnie większa niż w konfiguracji podstawowej. Czas pracy węzłów uległ znaczącemu skróceniu, gdyż koszt energetyczny transmisji jest wyższy.

5.2.4 Test 4 - Parametr 'p'

Test 'Parametr p' ma na celu przedstawienie zależności efektywności energetycznej dla trzech algorytmów względem wartości prawdopodobieństwa wypromowania węzła na węzeł typu cluster head.

Istotna informacja: w przypadku algorytmu komunikacji bezpośredniej parametr 'p' nie istnieje. Wynika to z charakterystyki komunikacji, gdyż w protokole tym nie występuje koncept węzła typu cluster head i każdy z węzłów przesyła informacje bezpośrednio do węzła głównego (sink). Dokonać można jednak założenia, że transmisja w protokole komunikacji bezpośredniej jest specjalnym przypadkiem sieci typu LEACH, w której parametr

$$p = 1.0$$

Oznacza to, że każdy z węzłów pełni rolę węzła cluster head, który nie posiada przynależących węzłów standardowych i dokonuje transmisji tylko swoich dane.

Lista plików konfiguracyjnych:

```
~/go/src/github.com/keadwen/msc_project$ tree testdata/test_4/
testdata/test_4/
  direct200.pbtxt
  leach200_p005.pbtxt
  leach200_p010.pbtxt
  leach200_p015.pbtxt
  leach200_p020.pbtxt
  leach200_p025.pbtxt
  leach200_p030.pbtxt
  pegasis200_p005.pbtxt
  pegasis200_p010.pbtxt
  pegasis200_p015.pbtxt
  pegasis200_p020.pbtxt
  pegasis200_p030.pbtxt

0 directories, 13 files
```

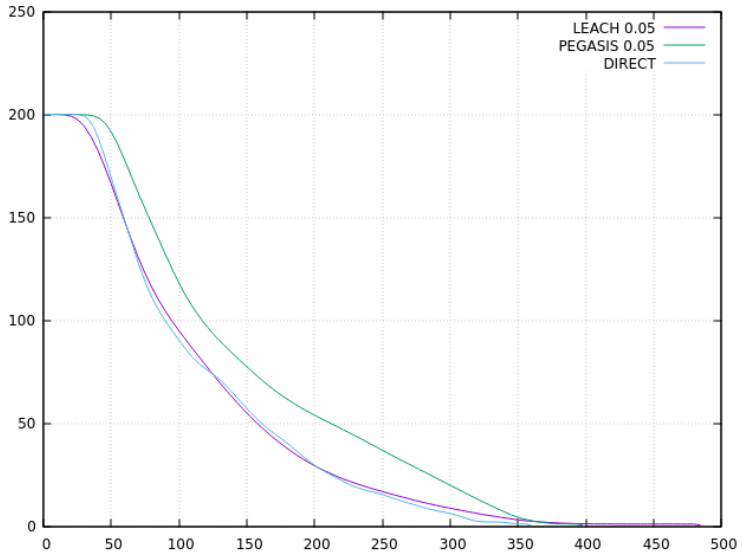
Uruchomienie oprogramowania:

```
~/go/src/github.com/keadwen/msc_project$ ./msc_project
   --repeat_config=100 --config_files=\
testdata/test_4/direct200.pbtxt,\
testdata/test_4/leach200_p005.pbtxt,\
testdata/test_4/leach200_p010.pbtxt,\
testdata/test_4/leach200_p015.pbtxt,\
testdata/test_4/leach200_p020.pbtxt,\
testdata/test_4/leach200_p025.pbtxt,\
testdata/test_4/leach200_p030.pbtxt,\
testdata/test_4/pegasis200_p005.pbtxt,\
testdata/test_4/pegasis200_p010.pbtxt,\
testdata/test_4/pegasis200_p015.pbtxt,\
testdata/test_4/pegasis200_p020.pbtxt,\
testdata/test_4/pegasis200_p025.pbtxt,\
testdata/test_4/pegasis200_p030.pbtxt
```

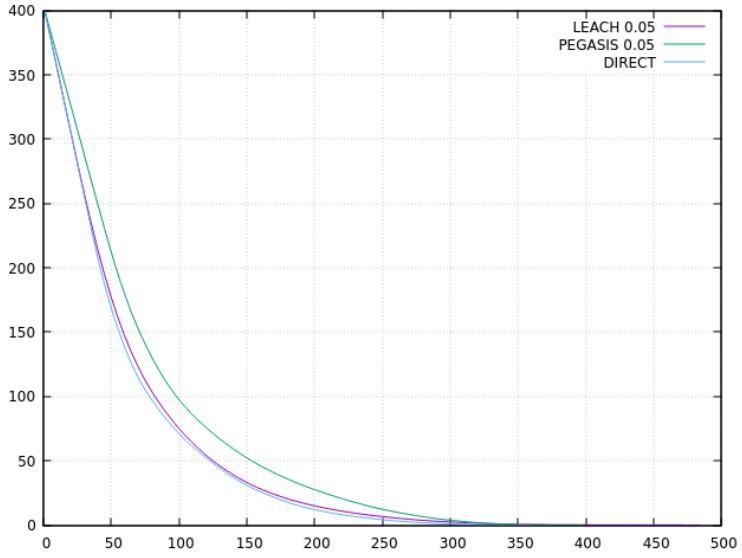
Poniższa tabela przedstawia wyniki uzyskane w ramach symulacji, powtórzonej 100 razy na każdym z plików konfiguracyjnych.

Uzyskane rezultaty			
Protokół	Parametr ‘p‘	Kwantyl	Wartość
DIRECT	-	-	359.00
LEACH	0.05	1	355.98
		5	362.90
		10	364.90
		25	383.00
		50	396.00
		75	417.00
		90	435.20
		95	455.05
		99	476.08
		100	484.00

Uzyskane rezultaty			
Protokół	Parametr ‘p‘	Kwantyl	Wartość
PEGASIS	0.05	1	345.00
		5	351.00
		10	351.90
		25	356.00
		50	365.00
		75	372.00
		90	377.00
		95	380.05
		99	387.02
		100	389.00



Rysunek 5.27: Funkcja ilości węzłów aktywnych w rundzie (p=0.05).

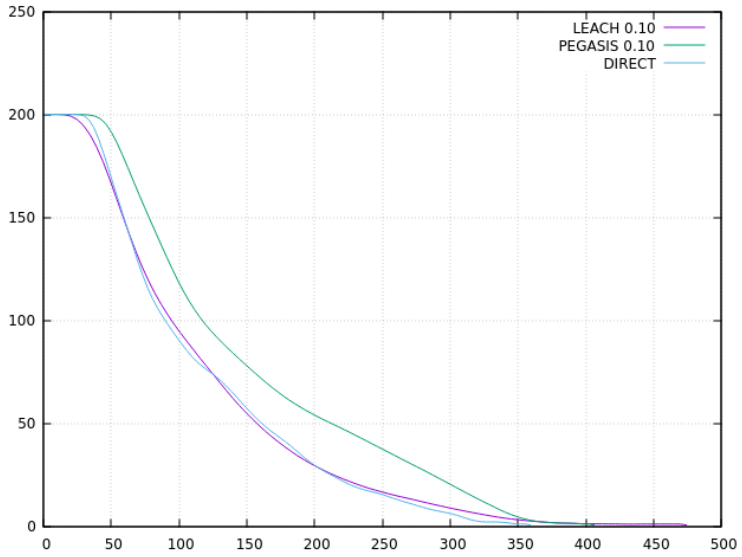


Rysunek 5.28: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (p=0.05).

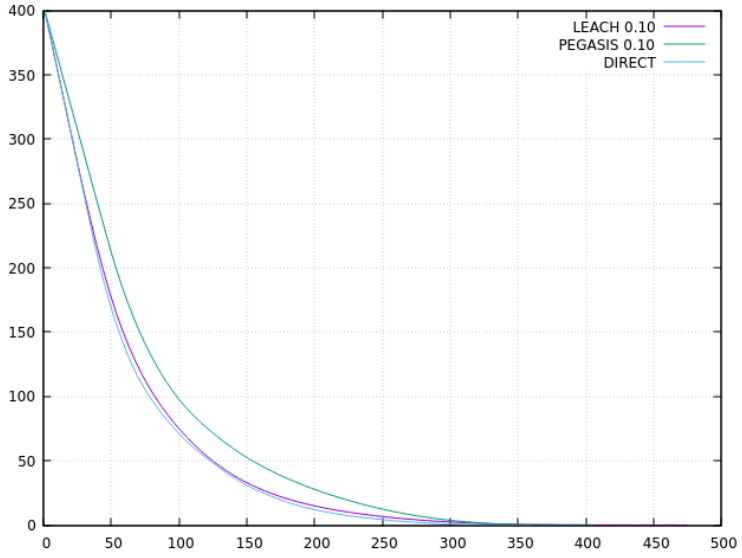
Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 359 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano skrócenie czasu pracy na poziomie -1.4% dla wartości minimalnej (354.00), oraz wydłużenie czasu pracy na poziomie +10.3% dla mediany (396.00), +34.8% dla wartości mak-

symalnej (484.00). Dla PEGASIS uzyskano skrócenie czasu pracy na poziomie -3.9% dla wartości minimalnej (345.00), oraz wydłużenie czasu pracy na poziomie +1.7% dla mediany (365.00), +8.4% dla wartości maksymalnej (389.00). Ilość węzłów typu cluster head jest dwukrotnie mniejsza niż w konfiguracji podstawowej.

Uzyskane rezultaty			
Protokół	Parametr ‘p‘	Kwantyl	Wartość
DIRECT	-	-	359.00
LEACH	0.10	1	364.00
		5	366.00
		10	372.90
		25	385.75
		50	400.00
		75	413.00
		90	430.00
		95	436.00
		99	455.19
		100	474.00
PEGASIS	0.10	1	340.99
		5	346.95
		10	351.00
		25	358.75
		50	363.50
		75	371.50
		90	377.00
		95	380.05
		99	388.07
		100	395.00



Rysunek 5.29: Funkcja ilości węzłów aktywnych w rundzie (p=0.10).

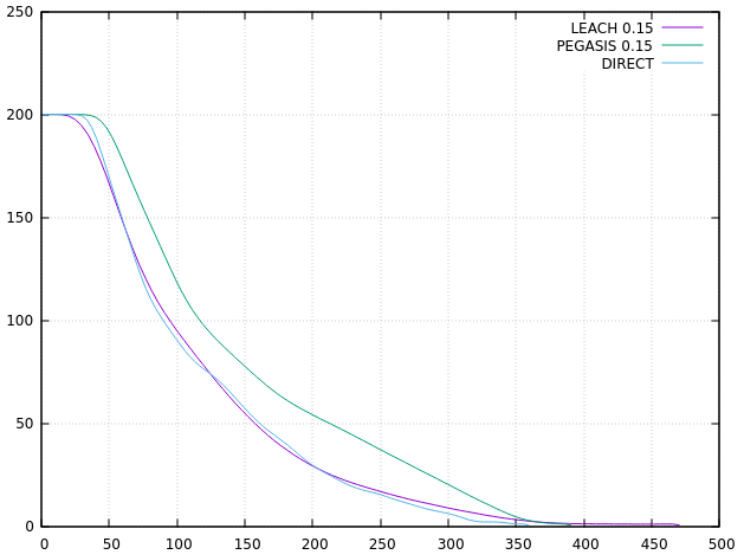


Rysunek 5.30: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (p=0.10).

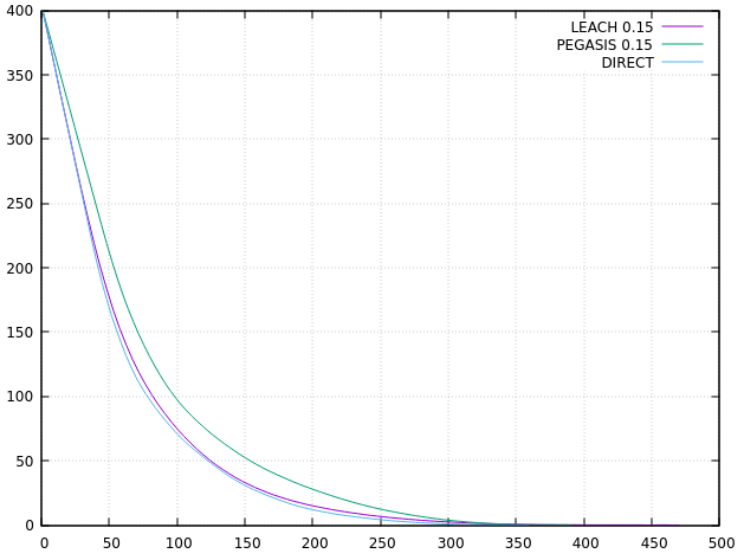
Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 359 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano wydłużenie czasu pracy na poziomie +1.4% dla wartości minimalnej (365.00), +11.4% dla mediany (400.00), +32.0% dla wartości maksymalnej (474.00). Dla PEGASIS uzy-

skano skrócenie czasu pracy na poziomie -5.3% dla wartości minimalnej (345.00), oraz wydłużenie czasu pracy na poziomie +1.3% dla mediany (363.50), +10.0% dla wartości maksymalnej (395.00).

Uzyskane rezultaty			
Protokół	Parametr ‘p‘	Kwantyl	Wartość
DIRECT	-	-	359.00
LEACH	0.15	1	356.93
		5	366.95
		10	371.00
		25	386.00
		50	398.50
		75	418.75
		90	434.30
		95	442.25
		99	455.19
		100	470.00
PEGASIS	0.15	1	339.95
		5	346.00
		10	350.00
		25	355.00
		50	364.50
		75	371.25
		90	382.10
		95	386.25
		99	402.09
		100	411.00



Rysunek 5.31: Funkcja ilości węzłów aktywnych w rundzie (p=0.15).

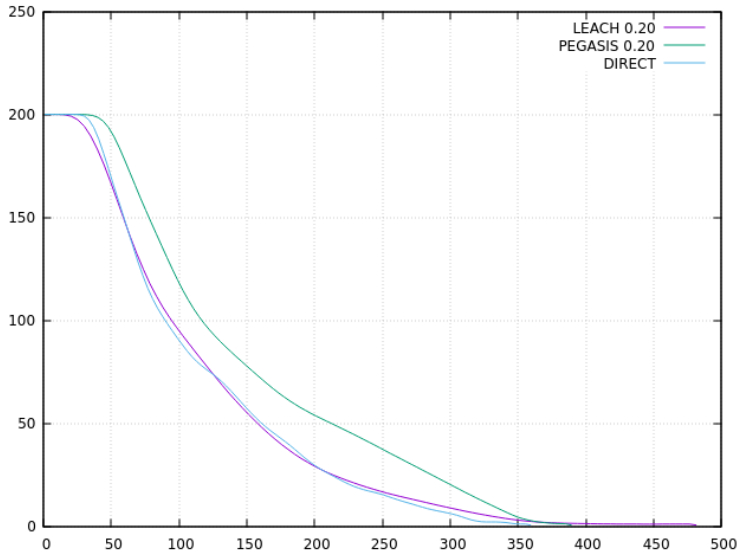


Rysunek 5.32: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (p=0.15).

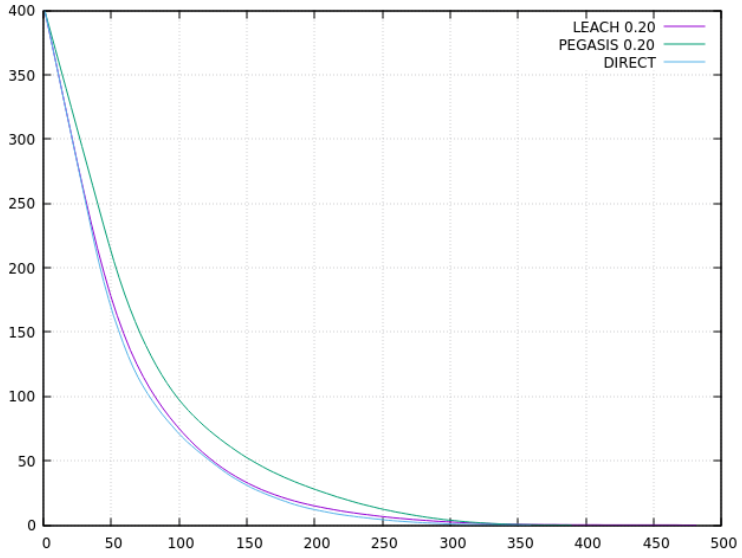
Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 359 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano skrócenie czasu pracy na poziomie -2.5% dla wartości minimalnej (350.00), oraz wydłużenie czasu pracy na poziomie +10.0% dla mediany (398.50), +30.9% dla wartości mak-

symalnej (470.00). Dla PEGASIS uzyskano skrócenie czasu pracy na poziomie -6.7% dla wartości minimalnej (335.00), oraz wydłużenie czasu pracy na poziomie +1.5% dla mediany (364.50), +14.5% dla wartości maksymalnej (411.00). Ilość węzłów typu cluster head jest o 50% większa niż w konfiguracji podstawowej.

Uzyskane rezultaty			
Protokół	Parametr ‘p‘	Kwantyl	Wartość
DIRECT	-	-	359.00
LEACH	0.20	1	362.00
		5	365.95
		10	371.80
		25	385.75
		50	402.50
		75	423.00
		90	442.20
		95	451.00
		99	471.10
		100	481.00
PEGASIS	0.20	1	346.97
		5	349.00
		10	352.00
		25	358.00
		50	367.00
		75	373.25
		90	378.00
		95	382.05
		99	388.09
		100	397.00



Rysunek 5.33: Funkcja ilości węzłów aktywnych w rundzie (p=0.20).

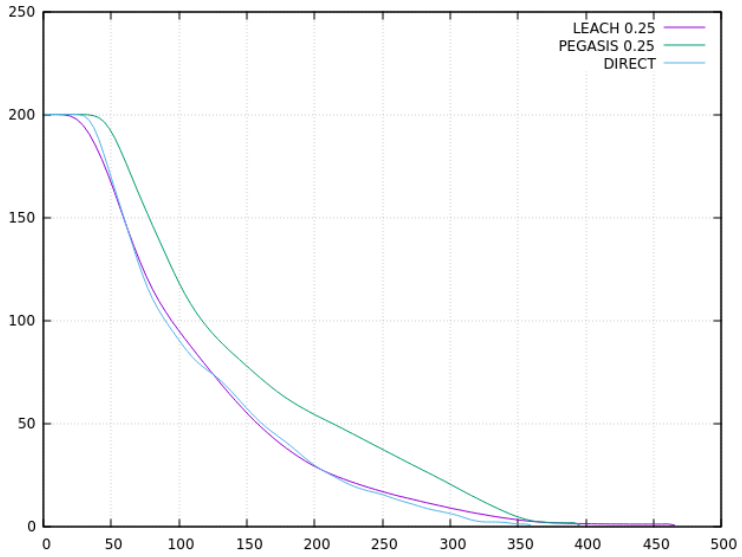


Rysunek 5.34: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (p=0.20).

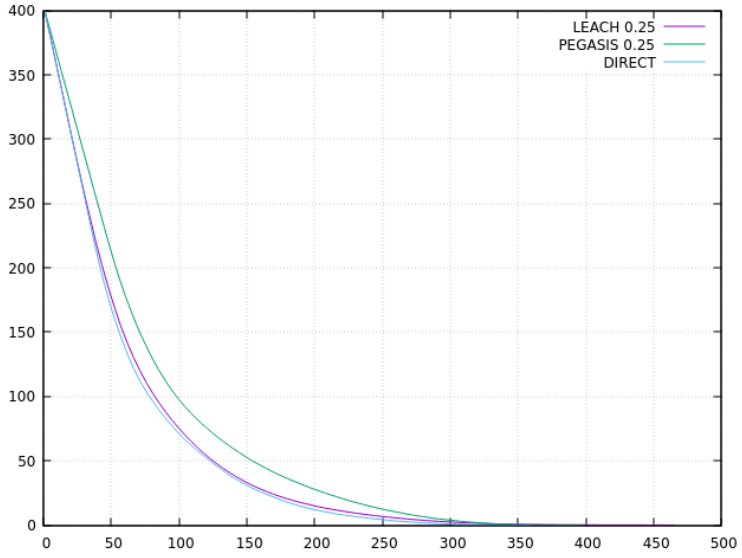
Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 359 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano wydłużenie czasu pracy na poziomie +0.8% dla wartości minimalnej (362.00), +12.1% dla mediany (402.50), +34.0% dla wartości maksymalnej (481.00). Dla PEGASIS uzy-

skano skrócenie czasu pracy na poziomie -4.2% dla wartości minimalnej (344.00), oraz wydłużenie czasu pracy na poziomie +2.2% dla mediany (367.00), +10.6% dla wartości maksymalnej (397.00). Ilość węzłów typu cluster head jest dwukrotnie większa niż w konfiguracji podstawowej.

Uzyskane rezultaty			
Protokół	Parametr ‘p‘	Kwantyl	Wartość
DIRECT	-	-	359.00
LEACH	0.25	1	358.99
		5	363.00
		10	370.70
		25	384.00
		50	403.50
		75	429.00
		90	439.20
		95	450.00
		99	454.11
		100	465.00
PEGASIS	0.25	1	338.99
		5	346.90
		10	349.90
		25	355.75
		50	362.00
		75	370.00
		90	381.00
		95	385.05
		99	397.09
		100	406.00



Rysunek 5.35: Funkcja ilości węzłów aktywnych w rundzie (p=0.25).

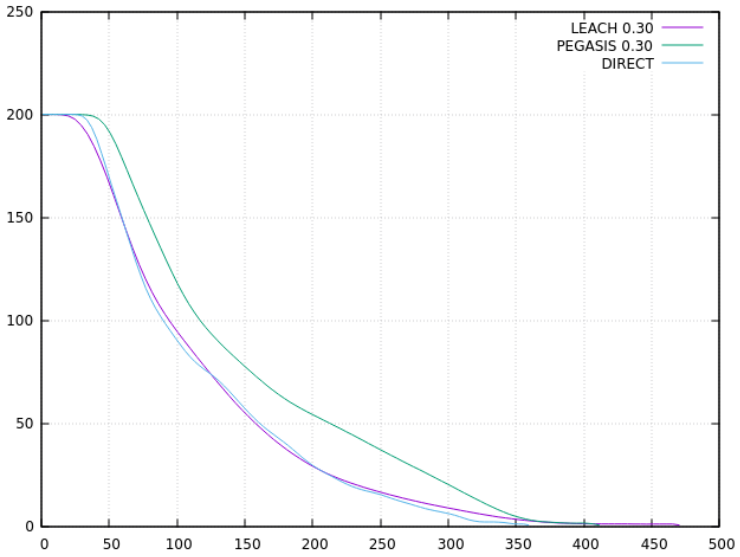


Rysunek 5.36: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (p=0.25).

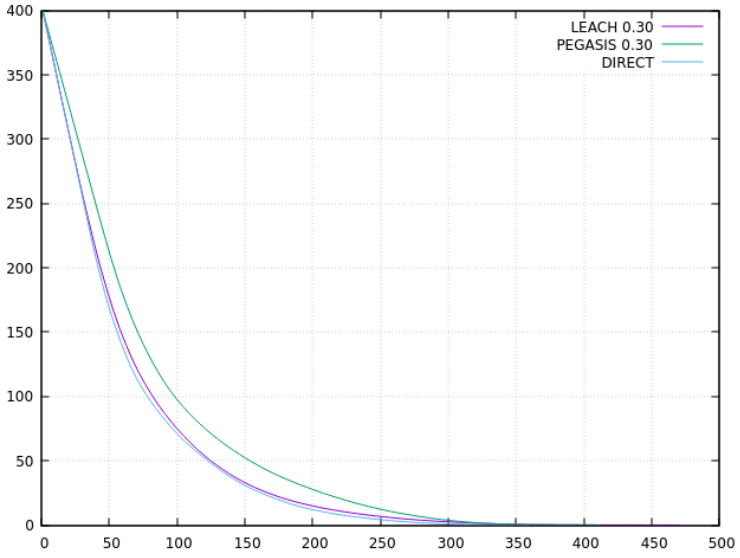
Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 359 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano skrócenie czasu pracy na poziomie -0.3% dla wartości minimalnej (358.00), oraz wydłużenie czasu pracy na poziomie +12.4% dla mediany (403.50), +29.5% dla wartości mak-

symalnej (465.00). Dla PEGASIS uzyskano skrócenie czasu pracy na poziomie -4.2% dla wartości minimalnej (344.00), oraz wydłużenie czasu pracy na poziomie +2.2% dla mediany (367.00), +10.6% dla wartości maksymalnej (397.00). Ilość węzłów typu cluster head jest 150% większa niż w konfiguracji podstawowej.

Uzyskane rezultaty			
Protokół	Parametr ‘p‘	Kwantyl	Wartość
DIRECT	-	-	359.00
LEACH	0.30	1	351.99
		5	368.90
		10	372.90
		25	385.00
		50	399.00
		75	418.00
		90	431.20
		95	444.10
		99	456.14
		100	470.00
PEGASIS	0.30	1	347.94
		5	349.95
		10	354.00
		25	359.00
		50	365.50
		75	372.25
		90	377.10
		95	385.05
		99	389.01
		100	390.00



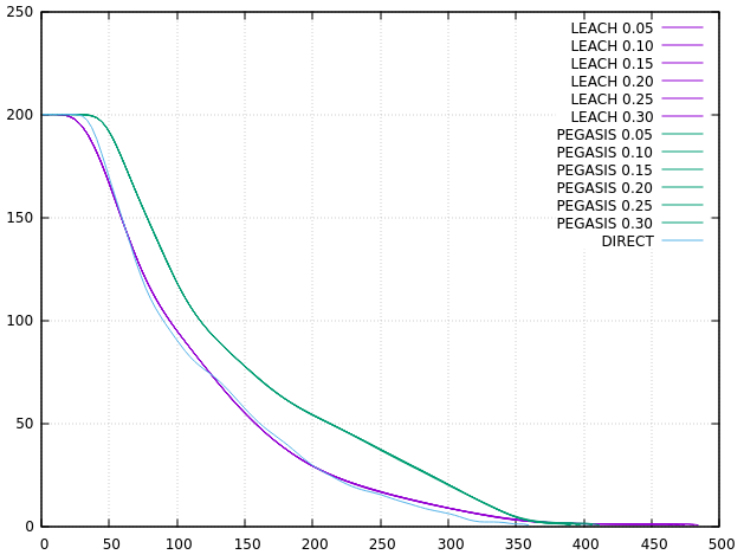
Rysunek 5.37: Funkcja ilości węzłów aktywnych w rundzie (p=0.30).



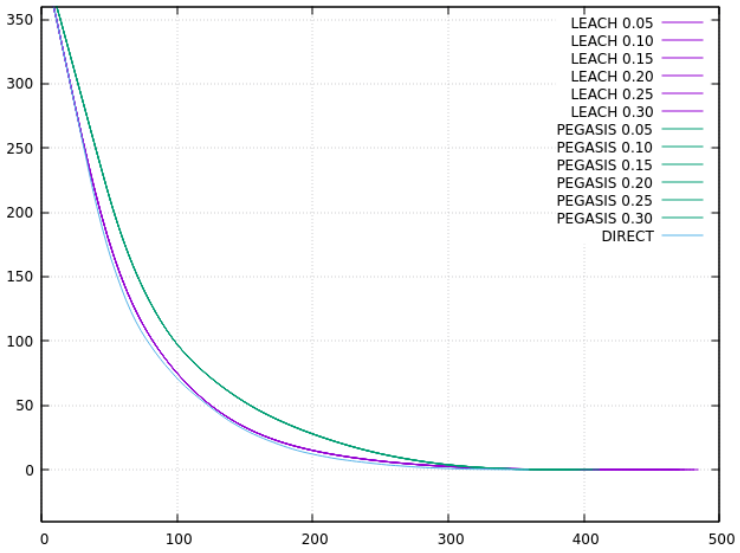
Rysunek 5.38: Funkcja całkowitej ilości energii wszystkich węzłów w rundzie (p=0.30).

Maksymalna ilość rund dla komunikacji bezpośredniej (DIRECT) jest parametrem stałym i wynosi 359 (uzyskana przez węzeł znajdujący się najbliższej węzła głównego ‘sink‘). Dla LEACH uzyskano skrócenie czasu pracy na poziomie -2.2% dla wartości minimalnej (351.00), oraz wydłużenie czasu pracy na poziomie +11.1% dla mediany (399.00), +30.9% dla wartości mak-

symalnej (470.00). Dla PEGASIS uzyskano skrócenie czasu pracy na poziomie -4.7% dla wartości minimalnej (342.00), oraz wydłużenie czasu pracy na poziomie +1.7% dla mediany (364.50), +8.6% dla wartości maksymalnej (390.00). Ilość węzłów typu cluster head jest trzykrotnie większa niż w konfiguracji podstawowej.



Rysunek 5.39: Funkcje ilości węzłów aktywnych w rundzie (wszystkie konfiguracje parametru p).



Rysunek 5.40: Funkcje całkowitej ilości energii wszystkich węzłów w rundzie (wszystkie konfiguracje parametru p).

Powyższe dwa wykresy nakładają przebiegi funkcji uzyskane dla wszystkich sześciu scenariuszy testu. Zauważyć można zależność, że efektywność energetyczna LEACH oraz PEGASIS przynosi pozytywne rezultaty, pozwalające na wydłużenie czasu pracy węzłów.

Rozdział 6

Podsumowanie

Koncepcja i projekt systemu symulującego model bezprzewodowej sieci czujnikowej został opracowany przez autora pracy. Pierwszym etapem pracy było określenie wymagań oraz funkcjonalności systemu końcowego. Pierwsze szkice dokumentacji były regularnie konsultowane z promotorem pracy. Proces ten pozwolił na zebranie szczegółowych wymagań projektowych. W końcowym etapie zbierania wymagań, autor zdecydował się tworzeniem prototypu systemu symulującego. Prace te pozwoliły na wczesne przetestowanie założeń i pomysłów dotyczących dekompozycji problemu, których efekt zaowocował uproszczeniem całkowitej architektury systemu. Projekt zrealizowano w języku Go, przy wykorzystaniu rozszerzonego systemu kontroli wersji git.

Proces zbieranie wyników generowanych przez system symulacyjny został podzielony na cztery główne testy. W każdym z testów wybrany został jeden parametr konfiguracji, który podlegał wielokrotnym zmianom mającym na celu zaobserwowanie zależności czasu i efektywności pracy sieci.

Najistotniejszym wnioskiem zaobserwowanym zarówno w momencie tworzenia pracy, jak i procesie tworzenia wyników jest zależność między uzyskanymi wynikami maksymalnego czasu pracy, a konfiguracją sieci. Oznacza to, że niemalże niemożliwym jest określenie stałej (przewidywalnej) efektywności energetycznej sieci w momencie wdrożenia LEACH, PEGASIS czy innego algorytmu usprawniającego wymianę danych w bezprzewodowej sieci czujników. W przypadku scenariusza wygenerowanego w ramach pracy dyplomowej zauważyć można, że węzeł 171 (X: 4.508531144480791, Y: 6.256340981494829) znajduje się bardzo blisko ‘sink‘. Czas życia tego węzła będzie kilkukrotnie dłuższy niż w przypadku węzła najbardziej oddalonego, węzła 195 (X: 197.47820095322837, Y: 190.37079023341855).

Każdy z testów definiuje kilka wartości parametru modyfikowanego, np. domyślna wielkość wiadomości wynosi 128 [B], dlatego też test składa się

z mniejszych testów w których praca węzłów podlega symulacji przy wykorzystaniu identycznej lokalizacji węzłów zdefiniowanych w konfiguracji domyślnej. Jedyną wartością ulegającą modyfikacji była wielkości wiadomości wysyłanej pomiędzy węzłami. Wielkości te wynosiły 32, 64, 128 oraz 256 [B].

Najlepsze korzyści płynące z implementacji protokołów LEACH oraz PEGASIS uzyskano dla testu numer 3, w którym dystanse pomiędzy węzłami uległy powiększeniu. Dla sieci powiększone czterokrotnie, czyli o wielkości X: 800, Y: 800 (konfiguracja domyślna X: 200, Y: 200), uzyskano jedynie 2 rundy dla komunikacji bezpośredniej. Gdzie w przypadku identycznego scenariusza w wykorzystaniem PEGASIS uzyskano w najgorszym przypadku 7 rund, w najlepszym 13. Natomiast LEACH umożliwiło utrzymanie pracy węzłów w 27 rundach dla najgorszego przypadku, oraz 33 rundach dla przypadku najlepszego.

W początkowym etapie pracy oczekiwanym było, że algorytm PEGASIS będący usprawnieniem protokołu LEACH, umożliwi uzyskanie dłuższego czasu maksymalnego pracy sieci. Na pierwszy rzut oka, wyniki tego nie odzwierciedlają. Dystrybucja maksymalnego czas pracy w LEACH jest dłuższa, jednakże wykresy pozwalają zauważyć pewną, że ilość węzłów aktywnych w protokole PEGASIS jest wyższa w początkowej fazie symulacji scenariusza (zwłaszcza w pierwszej połowie uzyskanych rund). Oznacza to, że węzeł główny ‘sink‘ uzyskuje większą ilość danych z większej puli węzłów. W scenariuszach w których odległości pomiędzy węzłami zwiększają się, koszt przekazywania informacji PEGASIS przy pomocy algorytmu ‘greedy‘ staje się bardzo kosztowny. Każdy kolejny węzeł odpowiedzialny za przekazanie (‘relay‘) danych, musi ponieść spory koszt energetyczny transmisji, który w końcowych rundach bardzo szybko eliminuje węzły niezdolne do przekazywania danych (ze względu na niewystarczające zasoby energetyczne).

Bibliografia

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, „Wireless sensor networks: a survey", Georgia Institute of Technology, 2001.prez

[2] W. R. Heinzelman, A. Chandrakasan, H. Balakrishnan, „EnergyEfficient Communication Protocol for Wireless Microsensor Networks", MIT, 2000.

[3] J. Yick, B. Mukherejee, D. Ghosal, „Wireless sensor network survey", Elsevier Computer Networks, Vol. 52, 2008.

[4] I. F. Akyildiz, M. Can Vuran, „Wireless Sensor Networks", Georgia Institute of Technology, Wiley, ISBN 9780470036013, 2010.

[5] G. J. Pottie, W. J. Kaiser, Wireless integrated network sensors. Communications of the ACM, 43:51–58, May 2000.

[6] C.Y. Chong and S.P. Kumar, „Sensor networks: evolution, opportunities, and challenges", In Proceedings of IEEE, Vol. 91, s. 1247-1256, 2003.

[7] K. Sohrabi, J. Gao, V. Ailawadhi, G.J. Pottie, „Protocols for self-organization of a wireless sensor network" IEEE Personal Communications, Vol. 7, s. 16-27, 2000.

[8] M. Bajelan, H. Bakhshi, „An adaptive LEACH-based clustering algorithm for wireless sensor networks.", J. Commun. Eng. Vol. 2(4), s. 351–365 2013.

[9] J. Ma, S. Wang, C. Meng, i in., Journal on Wireless Communications and Networking, s. 102, 2018

[10] G. Kavitha, R.S.D. Wahidabanu, „Improved cluster head selection for efficient data aggregation in sensor networks“ Research Journal of Applied Sciences, Engineering and Technology, Vol. 7(24), s. 5135–5142, 2014.

[11] D. Jia, H. Zhu, S. Zou, P. Hu, „Dynamic cluster head selection method for wireless sensor network.“, IEEE Sens. J. Vol. 16(8), s. 2746–2754, 2016.

[12] A. Donowan, B. Kerninghan, „The Go Programming Language“, Addison-Wesley, ISBN 9780134190440, 2015.

Opinia

o pracy dyplomowej magisterskiej wykonanej przez dyplomanta
Zdolnego Studenta i Pracowitego Kolegę
Wydział Elektryczny, kierunek Informatyka, Politechnika Warszawska
Temat pracy
TYTUŁ PRACY DYPLOMOWEJ

Promotor: **dr inż. Miły Opiekun**
Ocena pracy dyplomowej: **bardzo dobry**

Treść opinii

Celem pracy dyplomowej panów dolnego Studenta i Pracowitego Kolegi było opracowanie systemu pozwalającego symulować i opartego o oprogramowanie o otwartych źródłach (ang. Open Source). Jak piszą Dyplomanci, starali się opracować system, który łatwo będzie dostosować do zmieniających się dynamicznie wymagań, będzie miał niewielkie wymagania sprzętowe i umożliwiał dalszą łatwą rozbudowę oraz dostosowanie go do potrzeb. Przedstawiona do recenzji 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 przygotowanego przez Dyplomantów środowiska obejmujący opis konfiguracji środowiska oraz przykładowe ćwiczenia laboratoryjne. Ostatni rozdział pracy to opis możliwości dalszego rozwoju projektu. W ramach przygotowania pracy Dyplomanci zebrali i przedstawili w bardzo przejrzysty sposób duży zasób informacji, co świadczy o dobrej orientacji w nowoczesnej i ciągle intensywnie rozwijanej tematyce stanowiącej zakres pracy i o umiejętności przejrzystego przedstawienia tych wyników. Praca zawiera dwa dodatki, z których pierwszy obejmuje wyniki eksperymentów i badań nad wydajnością, a drugi to źródła skryptów budujących środowisko.

Dyplomanci dość dobrze zrealizowali postawione przed nimi zadanie, wykazali się więc umiejętnością zastosowania w praktyce wiedzy przedstawionej w rozdziałach 2-4. Uważam, że cele postawione w założeniach pracy zostały pomyślnie zrealizowane. Proponuję ocenę bardzo dobrą (5).

(data, podpis)

Recenzja

pracy dyplomowej magisterskiej wykonanej przez dyplomanta
Zdolnego Studenta i Pracowitego Kolegę
Wydział Elektryczny, kierunek Informatyka, Politechnika Warszawska
Temat pracy
TYTUŁ PRACY DYPLOMOWEJ

Recenzent: **prof. nzw. dr hab. inż. Jan Surowy**
Ocena pracy dyplomowej: **bardzo dobry**

Treść recenzji

Celem pracy dyplomowej panów dolnego Studenta i Pracowitego Kolegi było opracowanie systemu pozwalającego symulować i opartego o oprogramowanie o otwartych źródłach (ang. Open Source). Jak piszą Dyplomanci, starali się opracować system, który łatwo będzie dostosować do zmieniających się dynamicznie wymagań, będzie miał niewielkie wymagania sprzętowe i umożliwiał dalszą łatwą rozbudowę oraz dostosowanie go do potrzeb. Przedstawiona do recenzji praca składa się z krótkiego wstępu jasno i wyczerpująco opisującego oraz uzasadniającego cel pracy, trzech rozdziałów (2-4) zawierających bardzo solidny i przejrzysty opis: istniejących podobnych rozwiązań (rozdz. 2), komponentów rozpatrywanychjako kandydaci do tworzonego systemu (rozdz. 3) i wreszcie zagadnień wydajności wirtualnych rozwiązań, zwłaszcza w kontekście współpracy kilku elementów sieci (rozdział 4). Piąty rozdział to opis przygotowanego przez Dyplomantów środowiska obejmujący opis konfiguracji środowiska oraz przykładowe ćwiczenia laboratoryjne (5 ćwiczeń). Ostatni, szósty rozdział pracy to krótkie zakończenie, które wylicza także możliwości dalszego rozwoju projektu. W ramach przygotowania pracy Dyplomanci zebrali i przedstawili w bardzo przejrzysty sposób duży zasób informacji o narzędziach, Rozdziały 2, 3 i 4 świadczą o dobrej orientacji w nowoczesnej i ciągle intensywnie rozwijanej tematyce stanowiącej zakres pracy i o umiejętności syntetycznego, przejrzystego przedstawienia tych wyników. Drobne mankamenty tej części pracy to zbyt skrótowe omawianie niektórych zagadnień technicznych, zakładające dużą początkową wiedzę czytelnika i dość niestaranne podejście do powołań na źródła. Utrudnia to w pewnym stopniu czytanie pracy i zmniejsza jej wartość dydaktyczną (a ta zdaje się być jednym z celów Autorów), ale jest zrekompensowane zawartością merytoryczną. Praca zawiera dwa dodatki, z których pierwszy obejmuje wyniki eksperymentów i badań nad wydajnością, a drugi to źródła skryptów budujących środowisko. Praca zawiera niestety dość dużą liczbę drobnych błędów redakcyjnych, ale nie wpływają one w sposób istotny na na jej czytelność i wartość. W całej pracy przewijają się samodzielne, zdecydowane wnioski

Autorów, które są wynikiem własnych i oryginalnych badań. Rozdział 5 i dodatki pracy przekonują mnie, że Dyplomanci dość dobrze zrealizowali postawione przed nimi zadanie. Pozwala to stwierdzić, że wykazali się więc także umiejętnością zastosowania w praktyce wiedzy przedstawionej w rozdziałach 2-4. Kończący pracę rozdział szósty świadczy o dużym (ale moim zdaniem uzasadnionym) poczuciu własnej wartości i jest świadectwem własnego, oryginalnego spojrzenia na tematykę przedstawioną w pracy dyplomowej. Uważam, że cele postawione w założeniach pracy zostały pomyślnie zrealizowane. Proponuję ocenę bardzo dobrą (5).

(data, podpis)