

Politechnika Warszawska

W Y D Z I A Ł E L E K T R Y C Z N Y



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

Praca dyplomowa magisterska

na kierunku Informatyka
w specjalności Inżynieria oprogramowania

Tytuł pracy dyplomowej

Jakub Młynarczyk
nr albumu 288226

promotor
dr inż. Łukasz Makowski

WARSZAWA 2019

Problem 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, dwóch rozdziałów (2-3) zawierających opis istniejących technologii, dostępnych rozwiązań oraz opisuje autorski system środowiska symulacyjnego. Rozdział trzeci przedstawia szczegóły implementacji środowiska symulacyjnego. Ostatni rozdział pracy to opis możliwości dalszego rozwoju projektu.

Słowa kluczowe: wsn, sieci czujnikowe, golang

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 presents the fundamentals of wireless sensor networks and technologies, currently available solutions, as well as authored new design proposal. The second chapter presents the authored simulation tool. In the third chapter presents the implementation of the simulation environment in Go programming language. The fourth chapter contains results of tests which compare existing wireless sensor network protocols and proves simulator correctness. Finally some possibilities of further development of the invented algorithms are proposed.

Keywords: wsn, wireless sensor networks, golang

WARSZAWA, 1 lutego 2019

POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY

OŚWIADCZENIE

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa magisterska pt. Tytuł pracy dyplomowej:

- została napisana przeze mnie samodzielnie,
- nie narusza niczych 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

1	Wstęp	1
1.1	Wprowadzenie	1
1.2	Cel pracy	1
1.3	Tworzenie projektu	1
2	Wykorzystane technologie	3
2.1	Język programowania Go	3
2.1.1	Porównanie Go z Python	3
2.1.2	Porównanie Go z C++	4
2.2	Serializacja danych Protocol Buffers	4
2.3	System kontroli wersji git	5
2.4	GNUPlot	5
2.5	Docker	6
2.6	Latex	6
3	Architektura systemu	7
3.1	Opis	7
3.2	Komponenty systemu	7
3.2.1	Plik konfiguracyjny	7
3.2.2	Moduł główny (Core)	10
3.2.3	Moduł symulatora (Simulator)	10
3.3	Obsługa systemu	11
3.3.1	Generowanie scenariuszy	11
3.3.2	Uruchamianie scenariuszy	11
3.3.3	Generowanie wykresów	11
3.3.4	Dodawanie nowych protokołów	11
	Bibliografia	12

Podziękowania

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

Jakub Młynarczyk

Rozdział 1

Wstęp

1.1 Wprowadzenie

W ciągu 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 ilości aplikacji, zarówno w przemyśle (np. medycynie), jak również w sektorze prywatnym (np. inteligentny dom).

1.2 Cel pracy

Celem pracy jest przedstawienie 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 umożliwiające implementację oraz testowanie nowych algorytmów. Aplikacja umożliwia tworzenie symulacji porównujących efekty zastosowania różnych algorytmów transmisji danych, bez potrzeby wykorzystania fizycznych urządzeń.

1.3 Tworzenie projektu

Koncepcja i projekt systemu symulującego model bezprzewodowej sieci czujnikowej został opracy 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 zająć 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.

Rozdział 2

Wykorzystane technologie

Rozdział ten zawiera opis technologii oraz narzędzi wykorzystanych w pracy dyplomowej.

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. Składnia języka jest zbliżona do C/C++ czy Python.

2.1.1 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 ogólnie format komentarzy umożliwia automatyczne tworzenie przejrzystej dokumentacji.
- Go jest językiem kompilowalnym, 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 większą bazę publicznych bibliotek.

2.1.2 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 mikrokontrolerach).

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). 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). Natomiast zserializowane dane 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.

```
// 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;
}
```

2.3 System kontroli wersji git

Git to rozproszony system kontroli wersji stworzony jako wolne oprogramowanie (open source). Głównym 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 kontroli, które mogą następnie zostać wprowadzon ponownie do gałęzi głównej (merge). Architektura rozproszona git (w przeciwieństwie do zcentralizowanych systemów kontroli wersji) umożliwia programistom na posiadanie lokalnej kopii repozytorium, której zmiany mogą zostać następnie wprowadzone do gałęzi głównej.

2.4 GNUPlot

TODO

2.5 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 równoległe działające kontenery aplikacji działające na pojedynczym urządzeniu współdzielą parametry fizyczne maszyny oraz jądro systemu operacyjnego (np. Linux). Izolacja kontenerów widoczna jest na poziomie zależności (dependency) do określonych wersji bibliotek (libraries), narzędzi (binaries), plików konfiguracyjnych czy parametrów.

2.6 Latex

Latex to narzędzie do formatowania tekstu przy wykorzystaniu znaczników. Praca nad dokumentem przy wykorzystaniu Latex ułatwia redagowanie artykułu, ze względu na odeseparowanie treści od struktury tekstu.

Rozdział 3

Architektura systemu

3.1 Opis

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).

3.2 Komponenty systemu

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

3.2.1 Plik konfiguracyjny

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

- konfiguracja symulatora i sieci
- konfiguracja węzła

```
syntax = "proto3";
```

```
package config;
```

```
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;
}

// 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 = 4;
    // Energy consumption of node operations.
    EnergyCost energy_cost = 5;
    // Time delays introduced by node operations
    TimeDelay time_delay = 6;
}

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

// EnergyCost defines energy consumption for common node operations.
message EnergyCost {
    // Energy required to transmit one byte (in nanojoules).

```

```

int64 transmit = 1;
// Energy required to receive one byte (in nanojoules).
int64 receive = 2;
// Energy required to listen the channel for a second (in
    nanojoules).
int64 listen = 3;
// Energy required to process sensor data (in nanojoules).
int64 sensor_data_process = 4;
// Energy required to wake up MCU (in nanojoules).
int64 wake_up_mcu = 5;
}

// 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;
}

```

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:

- protokół (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 kluster do wszystkich węzłów (p_cluster_heads) - zmienna ta pozwala określić stosunek ilości węzłów pomiarowych odpowiedzialnych za pośredniczenie w przesyłaniu danych pomiarowych

(kłastrów) 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 pomiarowy. 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 `Nodes`. 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 porozdziale ‘Konfiguracja węzła’.

Konfiguracja węzła

TODO

- identyfikatora (`id`) zdefiniowanego jako 64 bajtowy integer.

3.2.2 Moduł główny (Core)

TODO

Węzeł (Node)

TODO

Sieć (Network)

TODO

Protokół (Protocol)

TODO

3.2.3 Moduł symulatora (Simulator)

TODO

3.3 Obsługa systemu

3.3.1 Generowanie scenariuszy

TODO

3.3.2 Uruchamianie scenariuszy

TODO

3.3.3 Generowanie wykresów

TODO

3.3.4 Dodawanie nowych protokołów

TODO

Bibliografia

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

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 rozpatrywanych jako 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 rozpatrywanych jako 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 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)