HAPPY BOSS
CZYLI BĄDŹ ADMINEM WE WŁASNEJ FIRMIE

Katarzyna Janocha, Michał Piekarz

26 lutego 2014

Spis treści

1 Wstęp **3**
- 1.1 Po co taki projekt? . . . 3
- 1.2 Naukowe określenie problemu . . . 3

2 Dokumentacja użytkowa **4**
- 2.1 Czego użytkownik powinien oczekiwać? . . . 4
- 2.2 Tworzenie wykresu . . . 5
- 2.3 Przykładowy wygląd wykresu . . . 5
- 2.4 Jak uruchomić program? . . . 6
- 2.5 Podsumowanie . . . 6

3 Dokumentacja implementacyjna **7**
- 3.1 Algorymiczna idea rozwiązania . . . 7
- 3.2 Utrudnienie i rozwiązanie tego problemu . . . 7
- 3.3 Podstawowe dane na temat specyfikacji projektu . . . 7
- 3.4 Dlaczego golang? . . . 8
- 3.5 Dlaczego Google charts? . . . 8
- 3.6 Szczegóły rozwiązania: dodawanie nowego komputera . . . 8
- 3.7 Szczegóły rozwiązania: pierwotna idea na tworzenie wykresu 8
- 3.8 Szczegóły rozwiązania: problem z tworzeniem wykresu i jego rozwiązanie . . . 8
- 3.9 Wybrane protokoły . . . 9
- 3.10 Protokół SIP . . . 9
- 3.11 Protokół STP . . . 9
- 3.12 Krótkie uzasadnienie . . . 10
- 3.13 Bezpieczeństwo . . . 10

4 Dokumentacja techniczna **11**
- 4.1 Klient - serwer . . . 11
- 4.2 Drzewo . . . 11
- 4.3 Protokoły . . . 11
- 4.4 Pakiet resources . . . 12

4.5 Szczegółowy opis działania wszystkich funkcji 12

5 Podział pracy **13**
5.1 Słowo wstępu 13
5.2 Część, nad którą pracowaliśmy wspólnie 13
5.3 Michał 13
5.4 Kasia 13

1 Wstęp

Czyli o co chodzi w naszym projekcie?

1.1 Po co taki projekt?

W dzisiejszych czasach młoda firma, która chce prężnie się rozwijać, często potrzebuje sieć komputerową na własne, wewnętrzne potrzeby. Postanowiliśmy stworzyć aplikację, która pomoże rozwiązać ten problem. Dzięki naszej aplikacji Użytkownik, który dokładnie i ze zrozumieniem przeczyta instrukcję, może samodzielnie zbudować prostą, ale niezwykle wydajną sieć komputerową na potrzeby swojej firmy.

Ma to dla Użytkownika duże znaczenie - początkujące, młode firmy niekoniecznie mogą pozwolić sobie na zatrudnienie specjalisty.

Dla ułatwienia Użytkownikowi pracy i zrozumienia działania aplikacji, wyposażyliśmy ją w dodatkową funkcjonalność - tworzenie wykresu sieci komputerowej. Dzięki niej Użytkownik pozna strukturę sieci, stanie się ona również niezwykłą pomocą w razie awarii sieci.

1.2 Naukowe określenie problemu

Tworzymy sieć komputerową o strukturze drzewowej, dla każdej maszyny istnieje z góry określona maksymalna pojemność - ilość maszyn, z którymi może być połączona (zakładamy, że jest to liczba większa od 2). Chcemy zachować najmniejszą możliwą średnicę sieci, tj. chcemy, aby odległość między najbardziej oddalonymi wierzchołkami była jak najmniejsza. Dochodzi nowy komputer, który musimy dodać on-line. Co robimy?

2 Dokumentacja użytkowa

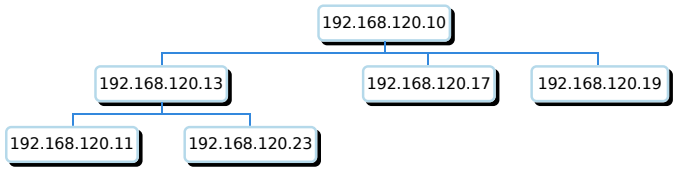
Czyli co użytkownik powinien wiedzieć

2.1 Czego użytkownik powinien oczekiwać?

Prostej aplikacji, która pozwoli mu utworzyć własną sieć komputerową. Użytkownik powinien przygotować się na używanie linii komend. Aplikacja, zainstalowana na danym komputerze, przyłączy komputer do sieci.

2.2 Tworzenie wykresu

2.3 Przykładowy wygląd wykresu



2.4 Jak uruchomić program?

2.5 Podsumowanie

Jesteśmy pewni, iż dzięki powyższej instrukcji nawet Użytkownicy niezwiązani z branżą komputerową poradzą sobie z utworzeniem sieci komputerowej w swojej firmie!

Tych bardziej zorientowanych w temacie oraz ciekawych szczegółów technicznych zapraszamy do zapoznania się z dokumentacją implementacyjną oraz techniczną.

3 Dokumentacja implementacyjna

Czyli szczegóły implementacyjne

3.1 Algorymiczna idea rozwiązania

Tworzymy sieć w formie ukorzenionego drzewa (korzeń stanowi pierwsza maszyna). Nowy komputer dołączamy do takiego wierzchołka, który nie osiągnął maksymalnej pojemności i ma najniższą głębokość. Jeśli takich wierzchołków jest więcej, wybieramy ten najbardziej na lewo.

1. Nowo dodany komputer połączony będzie z co najwyżej jednym komputerem dodanym wcześniej. Dlatego nigdy nie powstanie nam cykl i sieć zachowa strukturę drzewa.

2. Każde drzewo ma liście. Pojemność wierzchołków jest większa od 2, zatem zawsze istnieje wierzchołek, do którego możemy coś dodać.

3. Zauważmy, że średnica powiększa się jedynie wtedy, gdy tworzymy nowy poziom w drzewie. Nasze postępowanie prowadzi jednak do tego, by nowy poziom był tworzony jak najrzadziej.

Oczywiście, globalnie rozwiązanie to nie jest optymalne, ale my musimy radzić sobie z problemem on-line!

3.2 Utrudnienie i rozwiązanie tego problemu

JEDNAKŻE, POJAWIA SIĘ PEWIEN PROBLEM: Co będzie, jeśli dostaniemy dwa requesty o dołączenie do sieci na raz? Możemy rozwiązać go w następujący sposób: root będzie znał strukturę drzewa. To on będzie znajdował maszynę, do której podpięty zostanie nowo dodany komputer.

3.3 Podstawowe dane na temat specyfikacji projektu

Zatem... jak się do tego zabrać technicznie? Na początku dokonaliśmy następujących wyborów:

WYBRANY JĘZYK: go

POŁĄCZENIE MIĘDZY KOMUPUTERAMI: socket

POZOSTAŁE TECHNOLOGIE: Google Charts (https://developers.google.com/chart/)

3.4 Dlaczego golang?

Go oferuje łatwość tworzenia aplikacji sieciowych oraz wydajność. Aplikacja napisana w pythonie byłaby podobnie skomplikowana, natomiast mało wydajna. Tworzenie aplikacji sieciowych w C++ jest na tyle utrudnione, że nie chcieliśmy się w nie zagłębiać.

3.5 Dlaczego Google charts?

Jest to prosta, przydatna technologia, którą warto znać. Pozwala na tworzenie nawet bardzo skomplikowanych wykresów stosunkowo niewielkim kosztem - należy znać jedynie podstawy JavaScript (a właściwie - schemat HTML/CSS/JAVASCRIPT). Duża ilość dostępnych na stronie przykładów pozwala każdemu zagłębić się w tę technologię. Dlatego uznaliśmy, że jest ona idealna dla tworzenia wykresu naszej sieci.

3.6 Szczegóły rozwiązania: dodawanie nowego komputera

Informację o tym, że przychodzi nowy komputer, otrzymuje dowolna maszyna w naszej sieci. Następnie każdy wysyła zapytanie o wynik do swojego rodzica, w końcu dojdzie ono do korzenia.
Korzeń znajduje wierzchołek, do którego należy podpiąć nową maszynę na wspomniany wcześniej sposób. Można do tego użyć zwyczajnego algorytmu DFS.
Rozsyła później informację do wszystkich swoich dzieci, które przekazują ją dalej. Wiadomość rozprzestrzenia się w całej sieci.

3.7 Szczegóły rozwiązania: pierwotna idea na tworzenie wykresu

wierzchołek otrzymuje polecenie "zbuduj wykres". Wysyła wówczas do swojego rodzica tę wiadomość. Gdy wiadomość dochodzi do korzenia, tworzy on skrypt budujący wykres oraz rozsyła go do swoich dzieci z komunikatem mówiącym ZBUDOWAŁEM SKRYPT. Zostaje ona przekazana dalej.

3.8 Szczegóły rozwiązania: problem z tworzeniem wykresu i jego rozwiązanie

Jednakże, dziecko nie wie, czy zmiany w drzewie zaszły, zatem przy częstych prośbach o wykres nasza sieć zostanie zasypana niepotrzebnymi danymi.

Zatem nowy wykres będzie tworzony razem z dodaniem nowego wierzchołka i wówczas rozsyłany, co oznacza, że każda maszyna trzyma sobie update'owany wykres.

3.9 Wybrane protokoły

PROTOKÓŁ WARSTWY TRANSPORTOWEJ: tcp. Nie zależy nam na szybkości, ponieważ mamy do czynienia z małą, wewnętrzną siecią. Natomiast przy tworzeniu sieci bardzo ważne są dla nas bezpieczeństwo danych oraz dokładność.
Poniżej opisane są protokoły warstwy aplikacji.

3.10 Protokół SIP

PROTOKÓŁ 1 - PRZESYŁANIE INFORMACJI MIĘDZY KOMPUTERAMI (podczas tworzenia sieci, przekazywania informacji o klientach, "dogadywania się" w sprawie przesłania skryptu wykresu): wystarczy bardzo prosty protokół tekstowy. Jedyne informacje, jakie potrzebujemy, to: typ wiadomości, jej treść oraz ewentualna informacja o błędzie. Zauważmy, że jest to wiadomość, która jest bardzo niewielka, nie potrzebujemy zatem w żaden sposób "porcjować" danych. Będzie ona przekazywana w taki sposób, że wysyłamy tę samą wiadomość póki nie otrzymamy informacji zwrotnej - potwierdzenia jej otrzymania.
Opis uzytych przez nas wiadomości oraz format danych znajduje się w dokumentacji.

Protokołowi temu zdecydowaliśmy nadać nazwę Simple Information Protocol - w skrócie SIP.

3.11 Protokół STP

PROTOKÓŁ 2 - PRZESYŁANIE WYKRESU: plik z wykresem może być duży, musimy zatem przesyłać go w mniejszych pakietach. Możemy użyć protokołu tekstowego i przekazać po prostu kod wykresu w JavaScripcie. Zauważmy, że komputery mogą ustalić między sobą fakt, że zaraz nastąpi tranfser danych za pomocą naszego protokołu. Następnie możemy podzielić przesyłany skrypt na odpowiednio małe fragmenty i przesyłać je do momentu otrzymania informacji zwrotnej na ich temat.

Protokół ten nazwaliśmy Script Transfet Protocol - STP.

3.12 Krótkie uzasadnienie

Dlaczego zamiast używania SIP oraz STP nie użyjemy jednego protokołu, o formacie jak SIP, skoro równie dobrze można wysyłać podzielony plik za pomocą SIP? Zwróćmy uwagę na to, ile niepotrzebnych informacji zostałoby wówczas przekazane i jak bardzo spowolniłoby to transfer danych!

3.13 Bezpieczeństwo

Tworzona sieć jest na użytek wewnętrzny, zatem uznaliśmy, że nie musimy czynić dodatkowych wysiłków w celu gwarancji jej większego bezpieczeństwa. Jest ono częściowo zapewnione poprzez używanie sieci lokalnej. Aplikacja ma na celu utworzenie sieci bardzo prostej, służącej do wewnętrznej komunikacji pracowników, nie do przesyłania danych poufnych, które w wypadku firm niewielkich mogą znajdować się poza siecią. W wypadku, gdy Użytkownik będzie chciał przesyłać dane o znaczących rozmiarach czy dane o szczególnej poufności należałoby dorobić przynajmniej o jeden protokół więcej, nie posiadamy bowiem protokołu typu FTP.
Jednakże, projektując aplikację zadbaliśmy o wprowadzenie metodyki Agile - "programowanie zwinne". Zakłada ona częste zmiany preferencji użytkowników, a co za tym idzie, wymaga łatwości w przeprowadzaniu modyfikacji kodu. Daje to możliwość dodania zabezpieczeń w wypadku dalszego rozwoju naszej aplikacji.
Możliwości, jeśli chodzi o wprowadzenie bezpieczeństwa, są niezliczone - pierwszą, dość oczywistą, jest zabezpieczenie sieci hasłem. Każdy komputer w sieci posiadałby hash hasła, przed dołączeniem do sieci rodzic wymagałby od nowego dziecka podania hasła.

4 Dokumentacja techniczna

Czyli szczegóły techniczne

4.1 Klient - serwer

Podstawową jednostką naszej sieci jest klient, czyli użytkownik. Klientem takim jest nowo dołączany komputer. Najważniejszą funkcjonalnością sieci jest komunikacja klienta z serwerem.
Za obsługę tejże komunikacji odpowiedzialny jest package joinservice.go, a konkretnie - znajdujące się w nim implementacje client.go i server.go.
Techniczne szczegóły dotyczące zamieszczonych w nich konstruktorów i metod znnajdują się w komentarzach, a zatem również w wygenerowanym docu. Komentarze są na tyle dokładne, ze nie wymagają dodatkowego tłumaczenia - zapraszamy do ich lektury!

4.2 Drzewo

Funkcjonalności naszej sieci opierają się na strukturze drzewa, które tworzy. Odpowiada za nie joinservice/tree.
Klasa node.go trzyma reprezentację drzewa jako drzewa wskaźnikowego - każdy wierzchołek ma listę wskaźników do swoich dzieci.
Funkcje DFS oraz FindSolution przeszukują drzewo metodą DFS na potrzeby obu funkcjonalności, które oferujemy.
Dlatego obsługę drzewa zdecydowaliśmy się umieścić w pakiecie z klientem i serwerem. Jest to klasa kluczowa dla działania całej algorytmiki w aplikacji.

4.3 Protokoły

...znajdują się w pakiecie protocols. Opis ich działania zamieściliśmy w dokumentacji implementacyjnej.
Z kodu w oczywisty sposób wynikają uzasadnienia poszczególnych naszych decyzji.
Funkcje znajdujące się w protokołach to głównie funkcje pomocnicze służące do interpretacji, parse'owania i obsługi danych.
Za budowę wiadomości odpowiada struct Message (w każdym protokole z osobna). W przypadku STP jest niezwykle prosty. Sposób, w jaki używamy go w SIP jasno wynika z komentarzy. Są w nich wyszczególnione możliwe typy wiadomości - są one kluczowe w naszej aplikacji, warto się z nimi zapoznać!

4.4 Pakiet resources

odpowiada za skrypt tworzenia grafu. Jest w niego wpisany kod skryptu, komputery budują odpowiedni plik łącząc pliki odpowiadające za początek i koniec skryptu.
Zdecydowaliśmy się na takie rozwiązanie, ponieważ musimy wpisać "środek skryptu", który jest zmienny i odpowiada za kształt wykresu (patrz klasa node.go - jego funkcja DFS wpisuje do odpowiedniego pliku tenże "środek").

4.5 Szczegółowy opis działania wszystkich funkcji

...znajduje się w załączonym anglojęzycznym docu.
Opisy są bardzo dokładne - zapraszamy do lektury!

5 Podział pracy

Co kto robił?

5.1 Słowo wstępu

Większa część projektu była pracą wspólną, dlatego w niektórych momentach ciężko jest wyodrębnić role poszczególnych członków zespołu. Większość czasu spędzonego nad projektem to dyskusje i projektowanie - w tym oboje mieliśmy równy udział. Dlatego też postanowiliśmy podzielić poszczególne zadania na robione wspólnie i te, którymi zajmowała się głównie jedna osoba.

5.2 Część, nad którą pracowaliśmy wspólnie

1. Ogólny projekt aplikacji;
2. Ogólny projekt protokołów;
3. Szczegółowa implementacja serwera.

5.3 Michał

1. Ogólny projekt i implementacja schematu protokołów;
2. Implementacja schematu klient-server;
3. Testowanie;
4. Większość debugu.

5.4 Kasia

1. Większość funkcji pomocniczych protokołu SIP;
2. Budowanie drzewa, implementacja node;
3. Część związana z obsługą Google Charts i ogólną budową wykresu;
4. Algorytmika dodawania nowego węzła sieci.