07.06.2015

**Autor**

Paweł Maciejewski

Podstawy teleinformatyki – dokumentacja

Scentralizowane zarządzanie routerami OpenWrt

2017

[1. Wstęp 4](#_Toc484654497)

[1.1. Temat 4](#_Toc484654498)

[1.2. Cel projektu 4](#_Toc484654499)

[1.3. Uzasadnienie wyboru 4](#_Toc484654500)

[2. Planowanie 5](#_Toc484654501)

[2.1. Planowana funkcjonalność 5](#_Toc484654502)

[2.2. Zrealizowana funkcjonalność 5](#_Toc484654503)

[2.3. Podział prac 6](#_Toc484654504)

[3. Praca wstępna 7](#_Toc484654505)

[3.1. Dostępne urządzenia 7](#_Toc484654506)

[3.2. Wybór technologii 7](#_Toc484654507)

[3.3. Uzasadnienie wybory danych technologii 7](#_Toc484654508)

[3.4. Usługi zewnętrze 8](#_Toc484654509)

[3.5. Składnia UCI 8](#_Toc484654510)

[3.6. Klasa SshConnection i RoutersConnection 8](#_Toc484654511)

[4. Napotkane problemy 10](#_Toc484654512)

[4.1. Problem z udostępnieniem routera oddalonego o 70 km. 10](#_Toc484654513)

[4.2. Problem z bazą danych w usłudze Amazon Web Services 10](#_Toc484654514)

[4.3. Problem z zespołem 11](#_Toc484654515)

[4.4. Pozostałe problemy 11](#_Toc484654516)

[5. Wyświetlanie pełnej konfiguracji 12](#_Toc484654517)

[5.1. Wykorzystano polecenie „uci show” wyświetlającą pełną konfigurację 12](#_Toc484654518)

1. Wstęp

Temat

Projekt miał na celu stworzenie aplikacji pozwalającej na zarządzanie routerem opartym na systemie OpenWRT. OpenWRT jest oparty na podstawie systemu operacyjnego GNU/Linux stworzonego dla jednej z serii routerów Linksys przez ich producenta. Firma udostępniła go na licencji GNU GPL co oznacza, że musiała w całości opublikować kod. System ten został bardzo pozytywnie przyjęty przez społeczność i dzięki sporemu gronu entuzjastów doczekał się kolejnych wersj, m.in.: [DD- RT](https://pl.wikipedia.org/wiki/DD-WRT), [Tomato](https://pl.wikipedia.org/wiki/Tomato), [HyperWRT](https://pl.wikipedia.org/wiki/HyperWRT), [OpenWrt](https://pl.wikipedia.org/wiki/OpenWrt), Gargoyle czy FreeWRT.

Cel projektu

Głównym założeniem było stworzenie przyjaznego interfejsu użytkownika, dzięki któremu będzie on mógł wprowadzać zmiany w routerze nie mając specjalistycznej wiedzy z zakresu architektury Linuxa oraz poleceń i komunikatów obsługiwanych przez system. Projekt zakładał podstawową automatyzację, jednak z powodu odejścia członka zespołu, tempo tworzenia kodu zdecydowanie spadło. W ostatecznej wersji projektu zaimplementowana została większość funkcjonalności w całości konfigurowana manualnie przez użytkownika.

Uzasadnienie wyboru

Za wyborem tego tematu przemawiało już skromne doświadczenie w konfiguracji domowego routera opartego na tym systemie. Warto więc było poszerzyć tę wiedzę. Dodatkowo, obecnie na rynku nie istnieje podobne narzędzie, więc interesująca okazała się możliwość komercjalizacji projektu. Na chwilę obecną jednak jest to niemożliwe, gdyż z powodu problemów z zespołem projekt nie jest na tyle rozbudowany, aby mógł być sprzedawany. Jednakże z pewnością zostanie udostępniony na licencji Open Source.

1. Planowanie

Planowana funkcjonalność

* Dodawanie wielu routerów, przechowywanie ich danych dostępowych w bazie danych oraz określanie ich dostępności w danym momencie
* Wyświetlenie pełnej konfiguracji danego routera
* Edycja danych sieci WiFi (nazwa, klucz, kanał itp.)
* Obsługa Firewall’a, czyli blokowanie klientom dostępu do sieci po ich adresie MAC lub IP oraz ruchu sieciowego z danego portu, na dany port, na dany adres IP
* Dodawanie i konfiguracja skryptów uruchamianych przy uruchomieniu systemu
* Podgląd aktualnych parametrów systemu (RAM, procesor)
* Proste monitorowanie ruchu sieciowego
* Automatyczna aktualizacja każdego z tych parametrów w routerze (o ile zostały one zmodyfikowane np. zewnętrznie)

Zrealizowana funkcjonalność

Ze względu na odłączenie się od zespołu jednej osoby niemożliwe było zrealizowanie pełnej funkcjonalności. Jednakże zrealizowano jej ważniejszą część udostępniając ładnie prezentującą się aplikację z podstawową funkcjonalnością. Stopień realizacji założonych celów na 2 osoby można uczciwie określić na 70% (praca tylko jednej osoby). Oznacza to, że jeden autor wykonał 140% pracy, która mu powinna przypadać.

Zrealizowano:

* Całą pracę wstępną, czyli wybór technologii, konfigurację projektu, analizę sposobu komunikacji, przygotowanie bazy danych, zabezpieczenie dostępu do aplikacji loginem i hasłem, zaimplementowanie wzorców projektowych
* 2 z 3 zadań trudnych – zrealizowano dodawanie routerów i przełączanie pomiędzy nimi i dość rozbudowaną opcję Firewall’a, nie zrealizowano automatycznej aktualizacji parametrów w routerze wraz z watchdog’iem monitorującym ,czy ustawienia nie zostały zmodyfikowane zewnętrznie
* 3 z 5 zadań łatwych – zrealizowano wyświetlanie pełnej konfiguracji routera, edycję sieci danych sieci WiFi oraz wyświetlanie parametrów systemu, nie zrealizowano dodawania i konfiguracji skryptów uruchamianych przy starcie systemu i prostego monitorowania ruchu sieciowego
* Frontend z intuicyjnym interfejsem użytkownika opartym na Bootstrapie

Podział prac

Wstępny podział prac zakładał współpracę dwóch członków, ale z powodu rezygnacji jednego z członków z uczelni całość została wykonanana przez jedną osobę.

1. Praca wstępna

Dostępne urządzenia

* Tp-Link WR1043ND
* TP-Link TL-WR740N
* Serwer zdalny z systemem routingu do zdalnego podłączenia jednego z routerów

Wybór technologii

* ASP .NET MVC
* IDE: Visual Studio 2015
* SSH do łączenia z routerami, wykorzystano bibliotekę SSH.NET
* Wzorce projektowe: MVC, Repository Pattern
* Baza danych: Microsoft SQL Serwer
* Bootstrap, Razor i JS jako frontend

Uzasadnienie wybory danych technologii

* Dobra znajomość technologii Microsoftu i praca zarobkowa z ich użyciem, dlatego wykorzystano technologię webową ASP .NET MVC, środowisko deweloperskie Visual Studio 2015 oraz bazę danych w standardzie Microsoft SQL
* Zastosowano wzorce projektowe, aby ułatwić pisanie kodu i poprawić jego czytelność. Wzorzec MVC był konieczny, bowiem jest zintegrowany z technologią ASP .NET MVC. Wzorzec repozytoriów jest chyba najprostszym wzorcem do łatwej obsługi baz danych obsługiwanych za pomocą narzędzia ORM Entity Framework.
* SSH ponieważ nie da się prościej skomunikować z systemem OpenWRT. Inną możliwością byłoby wykorzystanie gotowego interfejsu graficznego i jego analiza lub połączenia szeregowego. Nawiązanie połączenia przez SSH jest więc najprostrze i najbardziej funkcjonalne, gdyż daje dostęp do np. składni UCI.
* Bootstrap ponieważ wiele rozwiązń jest już zaimplementowanych
* JavaScript ponieważ część logiki typu sortowanie wyników musiało być wykonane po stronie klienta
* Razor ponieważ jest zintegrowany z ASP .NET MVC
  1. Usługi zewnętrze
* Baza danych w Amazon Web Services
* Serwer Ubuntu w Digital Ocean
* Repozytorium w serwisie GitHub

Składnia UCI

Interfejs "uci" (Unified Configuration Interface - ujednolicony interfejs konfiguracyjny) jest sposobem konfiguracji systemu stosowanym w OpenWrt. W przeciwieństwie do stosowanego w starych wydaniach nvram też inne zalety - mniejsze zużycie flash (nie jest ciągle zapisywany ten sam obszar, tylko przydzielane jest losowo miejsce do zapisania pliku - wynika z właściwości systemu plików) czy możliwość zastosowania go w normalnych systemach plików (np. dla x86).

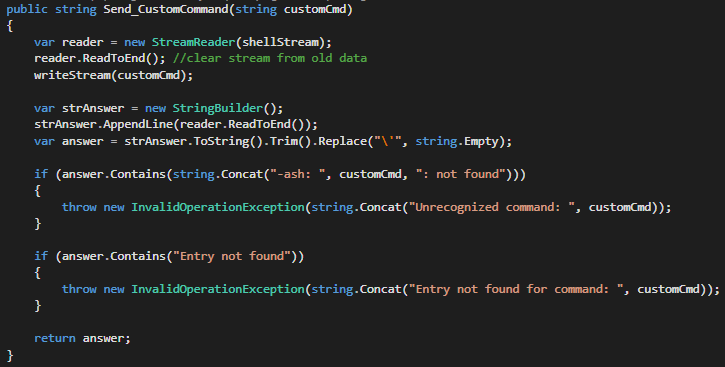
Domyślnie pliki konfiguracyjne zawarte są w katalogu "/etc/config”. Ważna jest nazwa pliku (przez nią odwołujemy się do poszczególnych sekcji) oraz jego budowa w postaci wydzielonych sekcji.

Odwołanie zawsze następuje wg schematu:

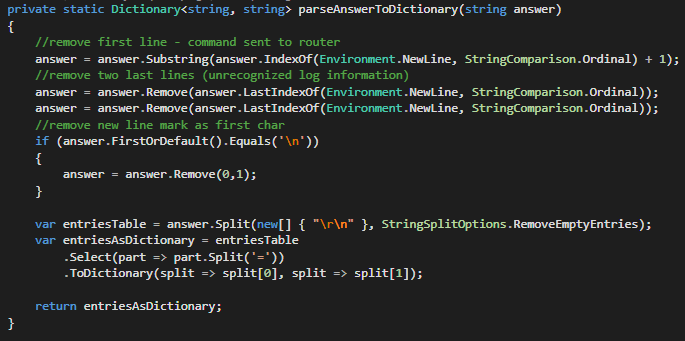
konfig.sekcja.nazwa\_opcji=wartość

Wysyłanie poleceń z uwzględnieniem składni UCI

Polecenie jest wysyłane do streamu writera, a odczyt następuje ze streamu readera. Następnie następuje sprawdzenie, czy odpoweidź nie zawiera komunikatu błędu.

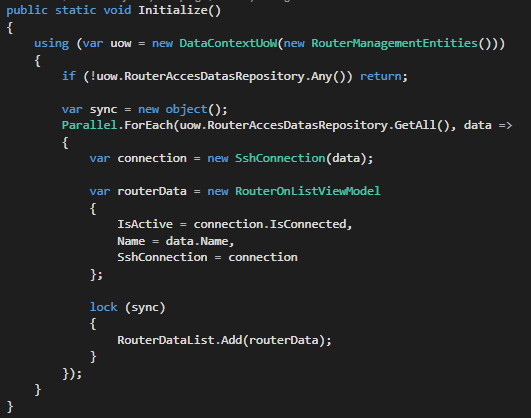


Następnie odpowiedź jest parsowana do słownika, zawierającego nazwę klucza wraz z sekcją i jego wartość.

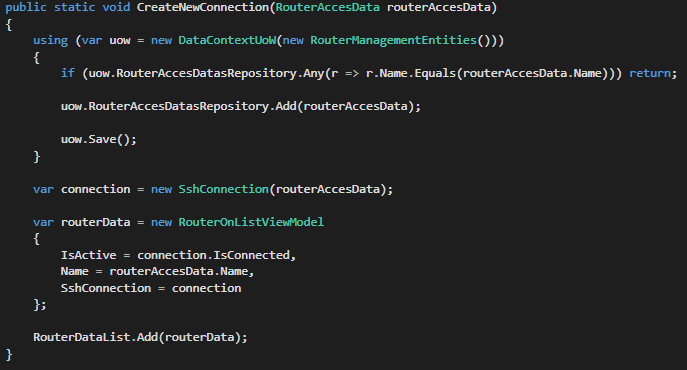


Klasa SshConnection i RoutersConnection

* SshConnection zawiera niskopoziomowe metody do zarządzania routerem – dodawanie, odczyt i edycja ustawień oraz metody parsujące wynik na postać łatwą do obsługi w logice aplikacji (postać obiektowa). Ważniejsze funkcjonalności będą omówione w dalszej części dokumentacji.
* RoutersConnection zawiera listę routerów i metody pobierające routery i dane o nich wg. Kryteriów, np. pobieranie aktywnych routerów czy pobieranie samych nazw routerów. Każdy z obiektów routerów zawiera własną klase SShConnection, dzięki czemu połączenie z każdym z nich odbywa się niezależnie.
* Inicjalizacja routerów odbywa się poprzez odczyt z bazy danych za pomocą repozytorium danych o nich. Następnie wielowątkowo odbywa się ich inicjalizacja w tle.



* Dodawanie routera do aplikacji składa się z dodania danych dostępowych do bazy danych oraz dodania go do listy routerów z nawiązanych połączeniem (lub też nie – połączenie może nawiązać użytkownik później ręcznie



1. Napotkane problemy

Problem z udostępnieniem routera oddalonego o 70 km.

System miał za zadanie zarządzać kilkoma routerami, w związku z czym podjęto próbę udostępnienia zdalnie drugiego routera. Otworzenie portu 22 na zewnątrz nie było problemem, problemem okazał się operator Play. Nie udostępnia on stałego IP, ale bez problemu można ten problem rozwiązać usługą dynamicznego DNS, do którego router loguje się co określoną ilość czasu i aktualizuje swój obecny adres IP na serwerze, a w zamian otrzymuje adres mnemoniczny. Poważniejszym problemem okazała się natomiast blokada połączeń przychodzących. Taka blokada jest nakładana dla bezpieczeństwa użytkowników sieci mobilnych, aby złośliwy serwer z zewnątrz nie naraził użytkownika na opłaty z tytułu opłaty za transfer. Operator odmówił również zdjęcia tej blokady po prośbie w zgłoszeniu do konsultanta.

 (…) dziękuję za przesłane przez Pana zgłoszenie. Chciałbym wyjaśnić, że ze względów bezpieczeństwa, zablokowaliśmy porty dla ruchu przychodzącego. Konsultowałem Pana prośbę z naszym działem technicznym, bardzo mi przykro ale nawet na Pana prośbę, nie jesteśmy w stanie ich odblokować. (…)

Jako rozwiązanie zdecydowano się wykorzystać opcję Reverse-SSH, dzięki której połączenie nie jest nawiązywane bezpośrednio do routera, tylko do zdalnego serwera, który przekazuje dane do routera. Serwer nie łączy się z routerem, tylko router z serwerem, a połączenie jest cały czas podtrzymywane, dzięki czemu omija się blokadę połączeń przychodzących. Serwer utworzono w usłudze Digital Ocean wykorzystując darmowe 50$ otrzymane za fakt bycia studentem. Usługa działała, ale był problem z podtrzymaniem połączenia. W związku z brakiem czasu na znalezienie rozwiązania koncepcję porzucono.

Problem z bazą danych w usłudze Amazon Web Services

Do realizacji projektu miała zostać użyta baza danych zdalna. Znalezienie bezpłatnego serwera baz danych SQL Server było trudne. Darmowy serwer Microsoft Azure dla studentów był już użyty w innym projekcie. Rozwiązaniem okazał się bezpłatna oferta w serwisie AWS o dość dobrych parametrach zlokalizowana w USA. Pojawiały się jednak problemy z nawiązaniem połączenia i prędkością transmisji. Ostatecznie zrezygnowano ze zdalnego serwera na rzecz bazy lokalnej, ponieważ zdalny serwer miał sens wówczas, gdy w projekcie przewidywane były dwie osoby.

Problem z zespołem

Nieoczekiwanie ze studiów zrezygnował członek zespołu. W związku z czym niemożliwe okazało się zrealizowanie pełnej planowanej funkcjonalności.

Pozostałe problemy

Pozostałe problemy związane z poszczególnymi funkcjonalnościami zostały poruszone w odrębnych sekcjach przeznaczonych dla nich.

1. Zarządzenie routerami

Dodawanie nowego

Modyfikacja starego

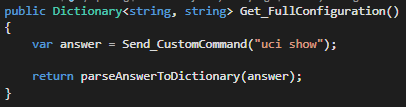
Usuwanie

Reconnect

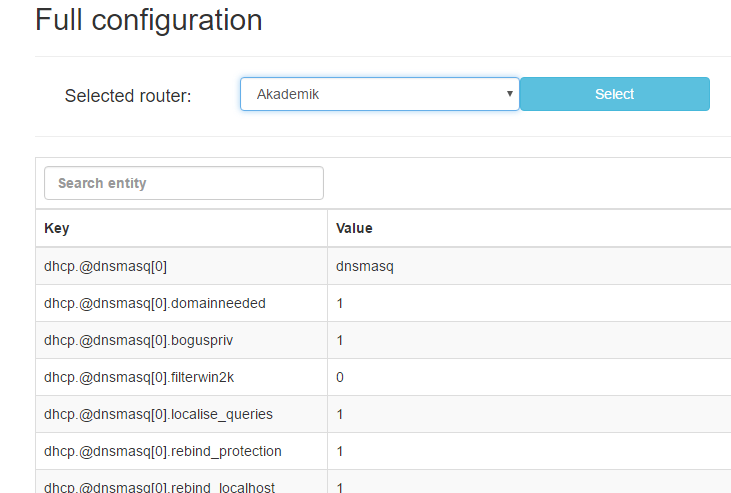
1. Wyświetlanie pełnej konfiguracji

Wykorzystano polecenie „uci show” wyświetlającą pełną konfigurację

Funkcja wysyłająca polecenie UCI Show wykorzystuje zaimplemntowany już wrapper SSH.



Dane w postaci słownika sa przesyłane do widoku, gdzie połączenie silnika Razor z plugin JavaScript’owym ListJS umożliwia wyświetlenie danych w przystępnej formie z jednoczesną możliwością sortowania wyników.



1. Wybór zapisanego routera
2. Konfiguracja sieci WiFi
3. Konfiguracja Firewalla
4. Wyświetlanie podstawowych danych o połączeniu