Raport C: Wstępna Implementacja

Zespół C: Mikołaj Kowalczewski, Stanisław Borkowski

## Przypomnienie opisu projektu

Celem projektu jest stworzenie systemu niezależnych agentów komunikujących się między sobą w celu optymalizacji trasy ich posiadacza.

Agenty na podstawie komunikatów broadcast wysyłanych przez pozostałe urządzenia aktualizują swoje lisy urządzeń sąsiednich i ich planowane trasy. Jeżeli trasa któregoś z urządzeń będzie kolidowała z aktualnym agentem, to ten spróbuje nawiązać komunikację w celu zmiany swojej lub jego trasy w taki sposób, by sumacyjnie ostateczna trasa obu urządzeń była lepsza niż przedtem.

## Zastosowane technologie

* Język programowania: Python 3.6
* Biblioteka do obsługi systemów agentowych: SPADE
* Biblioteka do przetwarzania komunikatów: Json
* Serwer XMPP: Prosody
* System operacyjny: Linux Ubuntu

## Opis działania

Agenty: “**MobileAgent**”, to domyśle urządzenia które będą się komunikowały i optymalizowały swoje trasy. Okresowo wysyłają informacje rozgłoszeniowe i aktualizujące do urządzeń znajdujących się w pobliżu. Posiadają wbudowane listy sąsiadów, ich trasy (harmonogramy) oraz własną trasę (harmonogram).

Docelowa wersja programu zakłada działanie na urządzeniach mobilnych, gdzie urządzenia sąsiednie będą wykrywane przy pomocy czujników wbudowanych (bluetooth, wifi itp.). Aktualnie implementowana wersja pełni rolę symulacji działania takiego systemu i rola środowiska (eteru) została powierzona specjalnemu agentowi: "**EnvironmentManager**", który zna położenie każdego urządzenia i sztucznie zapewnia połączenie tylko z sąsiadującymi agentami.\

Dla skutecznej komunikacji zostało utworzone kilka klas odpowiedzialnych za przetwarzanie komunikatów, a przydzielenie komunikatu do odpowiedniej klasy działa na zasadzie rozróżniania zdefiniowanych szablonów (templates).

Treść komunikatów jest przesyłana w formacie json, co pozwala na łatwe zapisywanie/odczytywanie konkretnych wartości/parametrów.

## Opis poszczególnych plików

**EnvironmentManagerAgent.py** - Plik odpowiedzialny za agenta specjalnego: EnvironmentManager

Co zostało zrobione:

* Odbiera komunikaty od agentów o ich pozycjach a następnie je aktualizuje w swojej bazie danych.
* Odbiera komunikaty Broadcast od agentów, porównuje lokalizację wysyłającego urządzenia z pozycjami pozostałych agentów i odsyła komunikat tylko do tych, które są w określonej odległości od nadawcy.

**MobileAgent.py** - Plik odpowiedzlany za domyślnego agetna: MobileAgent

Co zostało zrobione:

* Wysyła okresowo komunikaty do EnvironmentManager’a ze swoją pozycją.
* Wysyła okresowo komunikaty broadcast do EnvironmentManager’a, któe ten dostarcza do odpowiednich agentów.
* Odbiera ewentualne komunikaty broadcast od sąsiednich urządzeń i aktualizuje swoją listę sąsiadów.

Co trzeba jeszcze zrobić:

* Zaimplementować trasę/harmonogram, którym kierowałby się agent
* Komunikację pomiędzy sąsiadami w celu optymalizacji trasy
* Pobieranie listy sąsiadów swoich sąsiadów (lista z drugiej ręki), co pozwoli na zwiększenie zasięgu zbieranych informacji

**Templates.py** - Plik przechowujący szablony komunikacji rozróżniana za pomocą atrybutu: performative. Pozwalają one zaklasyfikować odpowiedni komunikat do odpowiedniego sposobu obsługi.

**adduser.sh** / **deluser.sh** - skrypty powłoki bash odpowiedzialne to za tworzenie/usuwanie kont na serwerze prosody. Przed uruchomieniem docelowego programu należy wywołać adduser.sh, by agenty mogły zostać utworzone przez bibliotekę SPADE.

**main.py** - plik uruchomieniowy. Tworzy jednego agenta: EnvironmentManager i zadaną ilość agentów: MobileAgent. Następnie je uruchamia i wchodzi w pętlę sprawdzającą poprawność systemu i w przypadku awarii zatrzymuje agentów.