WYDZIAŁ PODSTAWOWYCH PROBLEMÓW TECHNIKI

POLITECHNIKA WROCŁAWSKA

WIZUALIZACJA POŁĄCZEŃ

SIECI INTERNETOWEJ

MARCIN ADAMCZYK

NR INDEKSU: 221 429

Praca inżynierska napisana

pod kierunkiem

Dr. Przemysława Kobylańskiego



WROCŁAW 2017

Spis treści

1. Wstęp 1
2. Analiza problemu 2
3. Projekt systemu 5
   1. Grupy użytkowników i założenia 5
   2. Diagramy klas 6
   3. Diagramy sekwencji 7
   4. Opis struktur danych 8
   5. Opis algorytmów 8
4. Implementacja systemu 9

4.1 Opis użytych technologii 9

4.2 Omówienie kodów źródłowych 10

1. Instalacja i wdrożenie 12
2. Podsumowanie 13

Bibliografia 15

A Zawartość płyty 16

Wstęp

Internet. W łatwy i prosty sposób daje dowolnemu użytkownikowi dostęp do usług, czy serwisów z całego świata. W dzisiejszych czasach z Internetu korzysta prawie każdy. Jednak stosunkowo niewielu użytkowników zdaje sobie sprawę, jak właściwie działa Internet. W jaki sposób są w stanie w dowolnym momencie nawiązać kontakt z osobą znajdującą się po drugiej stronie globu i jak wiele urządzeń zaangażowanych jest w przekazanie pojedynczego pakietu do adresata. Do takich właśnie osób skierowane jest oprogramowanie tworzone w ramach tej pracy.

Swoim zakresem praca obejmuje szereg zagadnień powiązanych z Internetem oraz sieciami komputerowymi. Spośród nich szczególnie ważnym jest *routing*, czyli wyznaczanie trasy, za pomocą której pakiet ma dotrzeć do celu. Do analizy sieci wykorzystywane są odpowiednie narzędzia takie jak *Wireshark* czy *traceroute.* Dodatkowo system wykorzystuje usługę geolokalizacji na podstawie adresu IP.

Celem pracy jest stworzenie oprogramowania, które poprzez nasłuchiwanie ruchu sieciowego oraz jego analizę, naniesie na mapę świata punkty, w których znajdują się urządzenia biorące udział w transmisji pakietów użytkownika. W efekcie powstanie wizualizacja, mająca pomóc wspomnianym w pierwszym akapicie osobom w zrozumieniu pewnych podstawowych faktów dotyczących działania sieci, jaką jest Internet. Co najważniejsze, system będzie działał w czasie rzeczywistym. Dzięki temu użytkownik będzie mógł na bieżąco obserwować, jak jego działania wpływają na urządzenia rozlokowane na świecie.

Obecnie nie ma oprogramowania udostępniającego podobne możliwości. Istnieje pewien serwis internetowy pokazujący na mapie trasę pojedynczego pakietu – jednak nie ma to zastosowania w przypadku chęci analizowania całego ruchu. Samo odpowiednie połączenie programów *Wireshark* i *traceroute* może dać pewne efekty, jednak w postaci wydruku na konsoli, zupełnie nieczytelnego dla przeciętnego użytkownika Internetu.

Praca zawiera:

* omówienie zagadnień związanych z Internetem oraz sieciami komputerowymi wykorzystanych podczas tworzenia oprogramowania,
* informacje na temat dodatkowych narzędzi wykorzystywanych do działania systemu,
* informacje na temat stworzonego oprogramowania: dokumentację, zasadę działania, schematy UML,
* wyniki działania, instrukcję instalacji i uruchamiania, listę dodatkowego oprogramowania niezbędnego do prawidłowego działania programu, jak również instrukcje instalacji tegoż oprogramowania.

Analiza problemu

W niniejszym rozdziale przedstawiono środowisko pracy systemu oraz omówiono podstawowe pojęcia z tym środowiskiem powiązane. Następnie wyjaśniono procesy zachodzące w programie oraz powiązania pomiędzy tymi procesami. Dalsza część zawiera opis założeń funkcjonalnych jak i niefunkcjonalnych przedstawianego systemu. Na koniec przeprowadzono analizę oprogramowania trzeciego, używanego przez system, jak również porównanie do innych programów dostępnych dla użytkownika, realizujących podobne zadania.

2.1 Opis środowiska

Na początek wspomniany we wstępie Internet. To właśnie on stanowi swego rodzaju środowisko pracy systemu. Na potrzeby opisu funkcjonalności oprogramowania, wystarczy przytoczyć jedynie kilka podstawowych faktów na temat struktury i działania Internetu. Przede wszystkim jest to sieć złożona z wielu mniejszych sieci. Te z kolei składają się z dalszych, jeszcze mniejszych sieci. Idąc dalej w ten sposób można dojść do pojedynczego urządzenia znajdującego się w pewnej lokalnej sieci, należącej do potencjalnego użytkownika Internetu. Aby zapewnić komunikację pomiędzy wszystkimi urządzeniami, potrzeba zarówno tysięcy urządzeń sterujących przepływem danych, kilometrów kabli i przewodów, jak również niezwykle zaawansowanych algorytmów. Algorytmów zajmujących się między innymi wyliczaniem *routingu*, czyli opracowywaniem trasy jaką pakiet wysłany przez nadawcę ma dotrzeć do adresata. Wizualizacją właśnie tego zjawiska zajmować ma się system opracowany w ramach tej pracy. Ponieważ wyświetlenie listy adresów IP urządzeń „odwiedzonych” przez dany pakiet niewiele powie przeciętnemu użytkownikowi Internetu (a właśnie do takich osób kierowany jest stworzony system), w projekcie wykorzystywana jest usługa geolokalizacji. Pozwala ona określić współrzędne geograficzne obiektu na podstawie jego adresu IP. Sama usługa geolokalizacji do działania używa baz danych zawierających adresy IP oraz lokalizacje do których te adresy są przypisane.

2.2 Opis procesów

System składa się z trzech głównych procesów: **wczytywanie**, **analiza**, **wyświetlanie**. Ogólny sposób działania procesów został opisany poniżej.

1. Wczytywanie danych

Bazą tego procesu jest program *Wireshark* - popularny sniffer, czyli program nasłuchujący ruch sieciowy. Dla wygody, wykorzystywana jest wbudowana w ów sniffer opcja filtrowania danych, dzięki której na standardowym wyjściu pojawiają jedynie interesujące system rekordy. W tym przypadku są to adresy IP urządzeń do których użytkownik wysyła pakiety. Dane ze sniffera przekazywane są przy pomocy unixowego pipe’a na wejście głównego programu. Ten po sprawdzeniu poprawności danych przekazuje je do następnego procesu.

1. Analiza danych

Proces ten odpowiada za przeprowadzenie analizy oraz za przetworzenie na jej podstawie otrzymanych danych. Jest to najważniejszy element całego systemu, bowiem to właśnie na nim spoczywa cała logika. Podstawą jego działania jest drzewiasta struktura, przechowująca informacje na temat odebranych z wejścia danych. Konkretnie jest to drzewo czerwono – czarne, którego kluczem jest łańcuch zawierający adres IP, a wartością - wskaźnik na obiekt przechowujący statystyki dotyczące odpowiadającego połączenia. Statystyki zawierają dane na temat łącznej ilości pakietów wysłanych w ramach konkretnego połączenia oraz ilości pakietów wysłanych w określonym odcinku czasu. Dzięki przechowywaniu statystyk jako obiektu osobnej klasy, uzyskujemy izolację logiki związanej z przetwarzaniem danych statystycznych, a co za tym idzie, łatwą rozszerzalność typów informacji jakie chcemy gromadzić i przetwarzać.

Kolejnym zadaniem analizatora jest określanie trasy jaką przebywa pakiet   
w drodze do adresata. W tym celu system ponownie korzysta z oprogramowania trzeciego. Tym razem jest to program *traceroute*. Oczywiście nie ma pewności, że trasa wskazana przez wspomniany program, będzie w pełni zgodna z rzeczywistą trasą jaką przebył pakiet przechwycony przez sniffer. Jednak warto pamiętać, że system ma pełnić rolę jedynie edukacyjną, więc tego typu niedokładności w niczym nie przeszkadzają.

Po określeniu trasy, potrzebne jest już tylko ustalenie lokalizacji (współrzędnych geograficznych) poszczególnych punktów odwiedzonych przez pakiet użytkownika. W tym celu system wykorzystuje API serwisu geolokalizacyjnego. W zapytaniu wysłanym przy pomocy programu *curl* zawarty jest adres IP urządzenia, którego lokalizacja jest potrzebna. Jako odpowiedź system otrzymuje potrzebne mu współrzędne w postaci długości i szerokości geograficznej.

Na koniec wszelkie potrzebne dane zostają opakowane w odpowiednią strukturę i przekazane do następnego procesu.

1. Wyświetlanie obrazu wynikowego

Zadaniem tej części systemu jest przetworzenie danych odebranych z wejścia celem wyświetlenia ich w odpowiedniej formie na ekranie. Co ważne, ten proces nie przeprowadza żadnej analizy – zakłada, że to co odebrał, jest informacją prawidłową i gotową do wyświetlenia. Dokonywana jest tu przede wszystkim konwersja współrzędnych geograficznych na położenie na ekranie (piksele). Dodatkowo na podstawie opcjonalnych danych przekazanych przez analizator, mogą zostać nałożone dodatkowe efekty wizualne (przykładowo: pogrubienie lub zmiana koloru wyświetlanej linii w przypadku połączenia o wysokim natężeniu ruchu).

Po ustaleniu wszystkich potrzebnych danych, system przystępuje do wizualizacji, czyli nanoszenia na wcześniej przygotowaną mapę świata punktów oraz krzywych je łączących.

2.3 Opis działania wykorzystanego oprogramowania trzeciego

1. Wireshark:

Podstawowa funkcjonalność, oraz sposób w jaki jest ona wykorzystywana w systemie został opisany przy okazji objaśniania działania procesu wczytywania danych. Warto jedynie wspomnieć, że samego przechwytywania sniffer dokonuje przy pomocy biblioteki *libpcap* (w przypadku systemów UNIX), działającej na znacznie niższym poziomie od samego *Wireshark’a*, co zapewnia odseparowanie działania całego programu od warstwy sprzętowej. Istotny jest również fakt wymagania uprawnień administratora/roota w celu wykonania przechwytywania pakietów.

1. Traceroute:

W celu uzyskania listy pośrednich urządzeń, *traceroute* wysyła do adresata serię pakietów o określonym TTL (*time-to-live*), zaczynając od wartości 1, inkrementując ją aż do momentu osiągnięcia celu, bądź przekroczenia maksymalnej wartości. W ten sposób każde z pośrednich urządzeń odeśle do nadawcy komunikat błędu związany z przekroczonym „czasem życia pakietu” (ICMP Time exceeded), co pozwoli na określenie trasy jaką przebywa pakiet w drodze do adresata.

1. Ip-api

Dupa

Każde z wymienionego oprogramowania zostało wykorzystane w celu stworzenia czegoś w rodzaju *”proof of concept”*. W przypadku dalszych prac nad rozwojem projektu zarówno *Wireshark* jak i *traceroute* zostaną zastąpione autorskimi programami napisanymi w oparciu o biblioteki *libpcap* oraz *libnet.* Zapewni to ograniczenie funkcjonalności do wymaganego minimum, co zwiększy wydajność. Pozwoli to również na łatwiejsze implementowanie funkcjonalności nie wspieranych przez zastąpione programy.

2.4 Założenia systemu

Podstawowym założeniem dotyczącym funkcjonalności systemu, jest jego działanie   
w czasie rzeczywistym. Ma na celu to przede wszystkim zapewnienie użytkownikowi wrażenia wyższej responsywności używanego oprogramowania. Gdyby system najpierw zbierał wszystkie dane, a dopiero potem je przetwarzał, użytkownik nie wiedziałby, która z jego akcji wywołała taki, a nie inny efekt. Dzięki zbieraniu danych na bieżąco, efekt interakcji użytkownika z siecią widoczny jest w ciągu ułamków sekund. Oczywiście taki sposób pracy systemu wiąże się z określonymi problemami. Przede wszystkim program nie może wstrzymywać swojej pracy w przypadku nieplanowanych opóźnień w działaniu którejkolwiek z funkcjonalność. Dlatego też, z założenia, napisany system ma działać wielowątkowo. Konkretnie, każdy z opisanych w poprzednim podrozdziale procesów ma działać jako osobny wątek. Komunikacja pomiędzy nimi, odbywa się przez umieszczanie wyjścia jednego procesu w buforze postaci nieblokującej kolejki FIFO, z której następny proces pobiera dane, gdy tylko jest na to gotowy. Uproszczony schemat przedstawiający komunikację między procesami przedstawiony jest na *rysunku 2.1*.

F

I

F

O

F

I

F

O

ANALIZA

Wątek 2

WYŚWIETLANIE

Wątek 3

WCZYTYWANIE

Wątek 1

*Rysunek 2.1 – komunikacja między procesami*

2.5 Porównanie do istniejącego oprogramowania

Oczywiście każdy z wymienionych wcześniej programów wykorzystanych do stworzenia projektu sam z siebie realizuje jakąś część jego funkcjonalności. Jednak w przypadku analizatorów, ich wyjście samo z siebie jest zupełnie nieczytelne dla przeciętnego użytkownika.

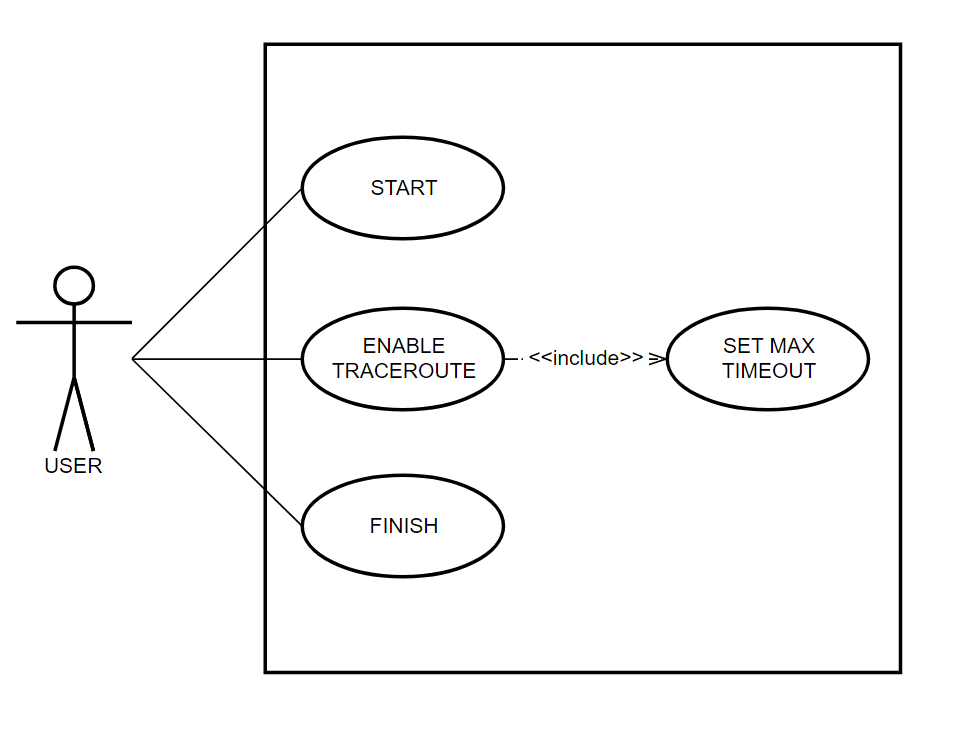
Istnieją serwisy internetowe oferujące możliwość przeprowadzenia wizualizacji trasy pojedynczego pakietu. Jednak nie ma to żadnego zastosowania chcąc uzyskać podgląd „na żywo”.

Projekt systemu

W tym rozdziale przedstawiono opis systemu za pomocą schematów UML (ang. Unified Modelling Language).

3.1 Grupy użytkowników i założenia

1. Odbiorcy to idioci
2. Z założenia wielowątkowy,
3. odseparowane i niezależne komponenty
4. Komunikacja przez nieblokujące kolejki
   1. Przypadki użycia i scenariusze

 Mimo, iż program sam w sobie zaprojektowany jest do działania „w tle”, bez interakcji użytkownika, ten ma możliwość wybrania trybu w jakim pracować będzie oprogramowanie. Dokładnie, użytkownik może (w zależności od tego jaki efekt chce uzyskać) włączyć obsługę programu *traceroute* (domyślnie jest on wyłączony). Jeśli użytkownik zdecyduje się włączyć obsługę *traceroute’a*, może on także ustawić parametr mówiący o maksymalnym czasie oczekiwania na odpowiedź przez program *traceroute*. W przypadku gdy użytkownikiem jest osoba nie mająca wiedzy wystarczającej by określić jaką wartość powinien mieć dany parametr, oprogramowanie wykorzysta własną, domyślną wartość. Ostatecznie do użytkownika należą również dwie dodatkowe akcje: uruchomienie oraz zakończenie pracy systemu.

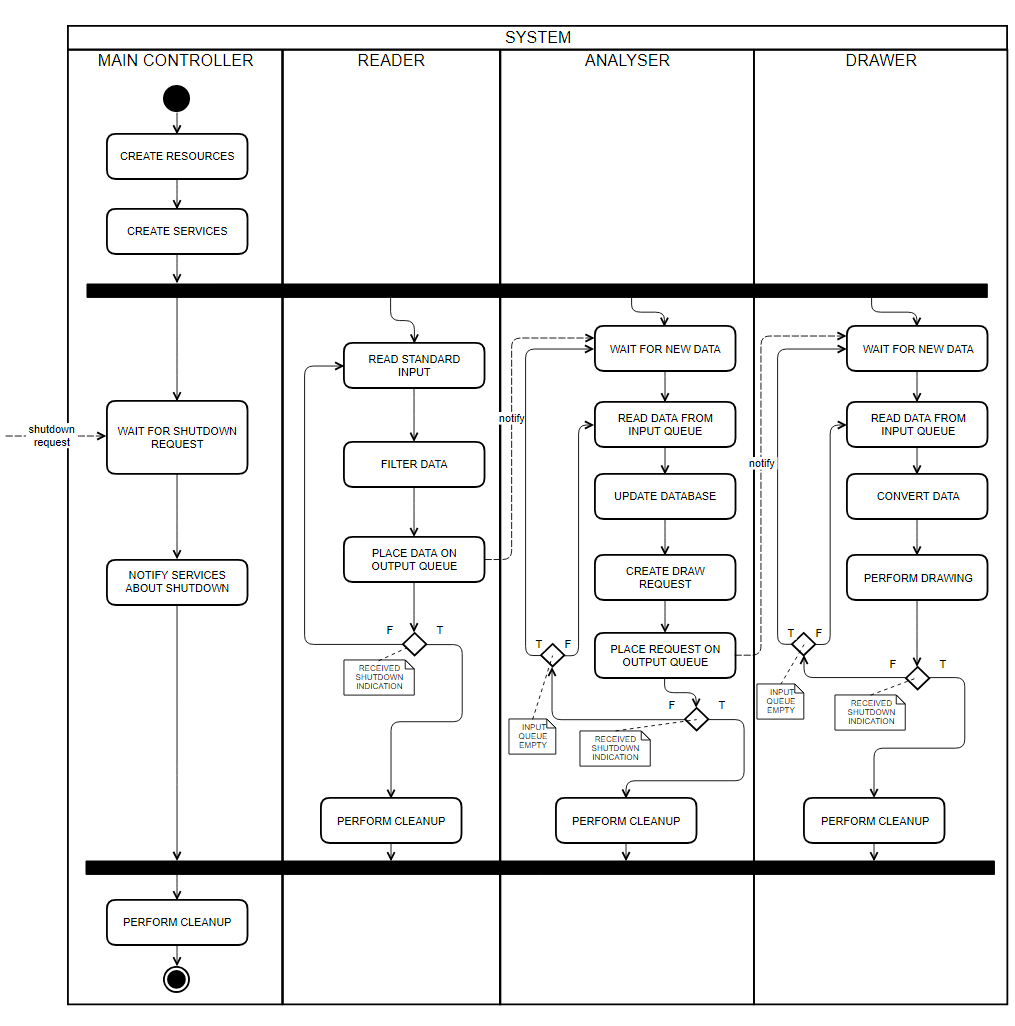
3.3 Diagramy klas

1. Po chuj to…

W tej sekcji należy przedstawić diagramy klas dla odpowiednich elementów systemu zidentyﬁkowane na podstawie wcześniejszych rozważań

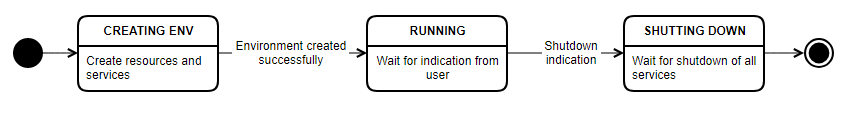
* 1. Diagram aktywności

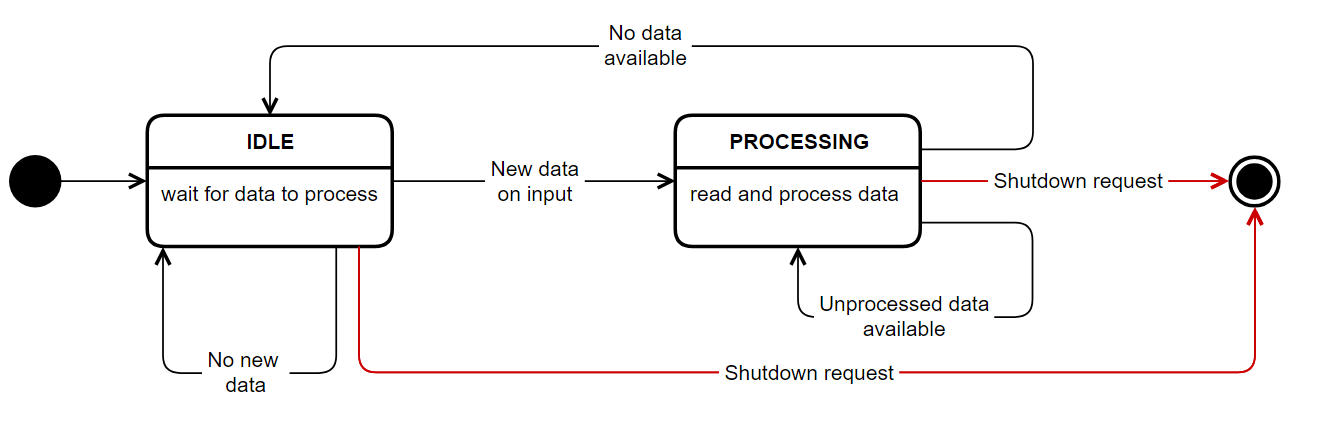
Diagram na rysunku 3xx przedstawia przepływ sterowania oraz wysokopoziomowy podgląd na sposób realizacji zadań przez poszczególne komponenty. Całością pracy systemu zarządza kontroler. Do jego zadań należą utworzenie środowiska pracy systemu (stworzenie zasobów w postaci buforów komunikacyjnych oraz serwisów działających jako osobne wątki) oraz zapewnienie, że w otrzymaniu od użytkownika polecenia zakończenia pracy systemu, wszystkie zasoby zostaną poprawnie zwolnione.

W tej sekcji należy przedstawić diagramy aktywności dla elementów systemu i odpowiednich procesów wynikające z wcześniejszej analizy. W niniejszym rozdziale przedstawiono diagramy aktywności .... Diagram na rysunku 3.1 przedstawia ....

3.6 Diagramy stanów

Poniżej przedstawiono diagramy stanów dla poszczególnych komponentów systemu. Komponenty zostały podzielone na dwie kategorie: serwisy i kontroler.

Diagram na rysunku 3xx przedstawia stany w jakich znajdować się może komponent kontrolera systemu.

Diagram na rysunku 3xx przedstawia stany w jakich może znajdować się komponent będący serwisem.

3.8 Opis protokołów

W systemie komunikacja pomiędzy serwisami odbywa się za pomocą przekazywania odpowiednio skonstruowanej wiadomości do bufora przez jeden serwis i odebrania tej wiadomości przez inny serwis. Możemy rozróżnić dwa protokoły komunikacji między serwisami:

1. Reader → Analyser,
2. Analyser → Drawer,

Poniżej zostały przedstawione dokładne struktury wiadomości wyżej wymienionych protokołów.

Ad 1:

Po wczytaniu i przefiltrowaniu danych, utworzona zostaje wiadomość w postaci łańcucha znaków (std::string) zawierającego adres IP urządzenia docelowego.

Ad 2:

W tym wypadku struktura wiadomości jest nieco bardziej skomplikowana. Do bufora przekazywana jest krotka (std::tuple<float, float, unsigned int>) zawierająca dwie wartości typu float (współrzędne geograficzne otrzymane w odpowiedzi od serwisu geolokacyjnego) oraz dodatnią wartość całkowitoliczbowa typu unsigned int (wartość określająca natężenie ruchu na danym odcinku).

3.9 Opis algorytmów

W tej sekcji należy wymienić i przedyskutować algorytmy wykorzystywane w systemie. Algorytmy należy przedstawić w pseudokodzie (wykorzystać pakiet algorithm2e). Omówienia poszczególnych kroków algorytmów powinny zawierać odwołania do odpowiednich linii pseudokodu. Dla zaproponowanych autorskich algorytmów należy przeprowadzić analizę ich złożoności czasowej i pamięciowej. Algorytm bąblowania jest przedstawiony w Pseudokodzie 3.1.

Implementacja systemu

4.1 Opis technologii

Do implementacji systemu użyto języka C++ w wersji określanej jako C++17. Język ten charakteryzuje się wysoką wydajnością, przy zachowaniu stosunkowo dobrej czytelności kodu. Dodatkowo daje on do dyspozycji szereg zewnętrznych bibliotek. Kilka z niż zostało wykorzystanych:

1. Boost: potężna, rozwijana przez społeczność biblioteka udostępniająca szereg gotowych implementacji zarówno algorytmów jak i struktur danych. W systemie została wykorzystana nieblokująca kolejka FIFO, stanowiąca bufor dla komunikatów przesyłanych pomiędzy komunikatów.
2. Qt: biblioteka graficzna (przez kogo utzymywana i tworzona). W opisywanym systemie wykorzystana do ostatecznej wizualizacji.
3. Libpcap i libnet (optional)

4.2 Omówienie kodów źródłowych

1. Kod przekształcający coordy na pixele
2. Przedstawienie klasy connection z timestampami i ich update-y

Instalacja i wdrożenie

4.1 Opis technologii

Do implementacji systemu użyto języka C++ w wersji określanej jako C++17. Język ten charakteryzuje się wysoką wydajnością, przy zachowaniu stosunkowo dobrej czytelności kodu. Dodatkowo daje on do dyspozycji szereg zewnętrznych bibliotek. Kilka z niż zostało wykorzystanych:

1. Boost: potężna, rozwijana przez społeczność biblioteka udostępniająca szereg gotowych implementacji zarówno algorytmów jak i struktur danych. W systemie została wykorzystana nieblokująca kolejka FIFO, stanowiąca bufor dla komunikatów przesyłanych pomiędzy komunikatów.
2. Qt: biblioteka graficzna (przez kogo utzymywana i tworzona). W opisywanym systemie wykorzystana do ostatecznej wizualizacji.
3. Libpcap i libnet (optional)

4.2 Omówienie kodów źródłowych

1. Kod przekształcający coordy na pixele
2. Przedstawienie klasy connection z timestampami i ich update-y