ChaTIN

Dokumentacja końcowa

Prowadzący projekt:
Jacek Wytrębowicz
Autorzy:
Maciej Grzybek
Andrzej Fiedukowicz
Jan Ignatowicz

Wstęp

Niniejsza dokumentacja została podzielona na trzy części.

1. **Dla użytkownika** – opisuje wszystkie wiadomości potrzebne użytkownikowi do zapoznania się z aplikacją i pracy z nią.
2. **Implementacja** – opisuje metody implemetacji, główne założenia i podstawowy projekt architektoniczny aplikacji. **Szczegóły** implemetacji dostępne w załączniku nr. 1.
3. **Szczegóły techniczne** – opisuje wymagania dla aplikacji, wykorzystane narzędzia, zależności (zewnętrzne elementy niezbędne do wykorzystania/kompilowania aplikacji).

Dla użytkownika

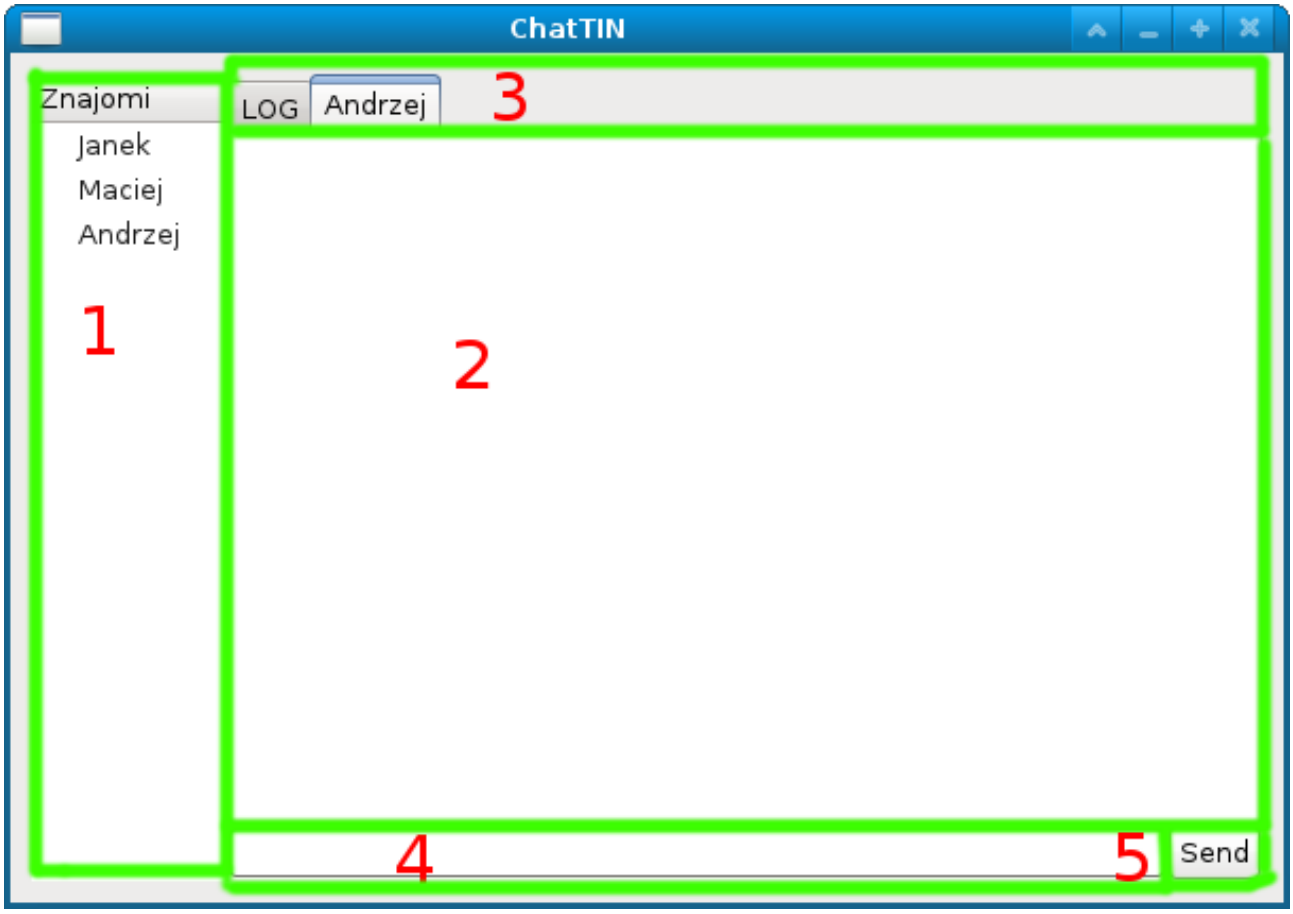
Opis aplikacji:

Aplikacja umożliwia komunikację z użytkownikami sieci poprzez wysyłanie do nich wiadomości tekstowych. Wiadomość tekstowa może być wysłana do jednej osoby lub do grupy osób (konferencji). Użytkownicy identyfikowani są w sieci wg. adresu IP jednak aplikacja umożliwia tworzenie lokalnych aliasów do adresów zapisywanych w bazie danych. Użytkownik może się posługiwać aliasem tak samo jak oryginalnym adresem IP.

Funkcjonalności aplikacji:

- Dialog dwóch osób rozpoznawanych po adresie IPv6 lub aliasie
- Tworzenie i utrzymywanie rozmowy konferencyjnej, w trakcie której wszystkie wpisywane komunikaty rozsyłane są do jej uczestników
- Tworznie powiązań ciąg_znaków => IPv6 zwanych aliasami, które mogą być stosowane zamiennie z adresami IPv6.
- Możliwość subskrybowania obecności osób wg. protokołu określonego w dokumentacji wstępnej.

Obsługa interfejsu (instrukcja użytkownika):
Interfejs użytkownika wzraz z oznaczeniami najważniejszych elementów pokazano na rysunku poniżej:



Legenda:

1. Lista zarejestrowanych aliasów (lista aliasów)
2. Okno rozmowy/logów
3. Lista kart
4. Linia wprowadzania (linia komend)
5. Przycisk akceptacji (analogicznie do naciśnięcia klawisza ENTER)

Głównym sposobem komunikacji użytkownika z aplikacją jest linia komend która pełni dwie role.

1. Niezależnie od aktualnie wybranej karty pozwala na wpisywanie poleceń (lista i opis niżej), których wykonanie ma odzwierciedlenie w działaniu aplikacji. Wszystkie komendy należy poprzedzać znakiem '/' w przeciwnym razie wprowadzony tekst zostanie zinterpretowany jak w pkt. 2.
2. W zależności od wybranej karty:
    1. Dla karty dialogu - wysyła do danego rozmówcy wpisaną wiadomość.
    2. Dla karty konferencji – wysyła do wszystkich uczestników konfrencji wpisaną wiadomość.
    3. Dla karty logów – nie ma zdefiniowanego żadnego działania.

Oprócz tego użytkownik może wybierać z listy aliasów znajomych (przy użyciu podwójnego kliknięcia), z którymi chce rozpocząć rozmowę. W przypadku takiego wyboru zostaje otwarta (i/lub ustawiona jako aktywna, w przypadku gdy wcześniej była otwarta) karta rozmowy z wybranym użytkownikiem.
Ponadto użytkownik może przełączać się między kartami pojedyńczym kliknięciem w kartę, na którą chce się przełączyć. W przypadku otworzenia większej ilości kart niż może się zmieścić na pasku wyboru, karty można przewijać przy użyciu pojawiających się przycisków < oraz >.
Oprócz tego karta aktualnie wybrana jest specjalnie wyróżniona.

Lista poleceń:

W linii komend można wykonać następujące polecenia:

- **/open *Alias/IP*** – otwiera kartę rozmowy z danym aliasem lub adresem IP, program automatycznie sprawdza czy karta już istnieje – w takim wypadku karta ta jest ustawiana jako aktywna. Ponadto wpisanie adresu IPv6 w przypadku gdy adres ten ma znany aplikacji alias, spowoduje otwarcie karty dla tego aliasu (by zapobiec problemowi dwóch kart dla jednego użytkownika).
- **/confopen *nazwa_konferencji Alias/IP{2+}*** - otwiera kartę rozmowy konferencyjnej. Wymaga podania minimum dwóch aliasów/adresów IP. Karta konferencyjna identyfikowana jest po podanej nazwie konferencji. W sieci mogą istnieć dwie konferencje o tej samiej nazwie ale innym właścicielu (użytkownik który wywoła /confopen staje się właścicielem konferencji).
- **/addalias *alias IP*** – powoduje dodanie aliasu o nazwie *alias* wskazującego na adres *IP*
- **/close** – zamyka bieżącą kartę, lub gdy zostanie wywołane na karcie logów – zamyka aplikację
- **/sub *alias/IP*** – wysyła prośbę o subskrypcję do użytkownika wskazanego przez alias/IP
- **/subakc *alias/IP* –** akceptuje prośbę wskazanego użytkownika
- **/subdec *alias/IP*** – odrzuca prośbę wskazanego użytkownika

Implementacja

Główne moduły aplikacji:
Socket – biblioteka stanowiąca obiektową nakładkę (w zakresie wymaganym do realizacji aplikacji ChaTIN) na bibliotekę BSD Socktes, wykorzystuje ona mechanizmy języka C++ by zapewnić możliwie prosty interfejs do komunikacji sieciowej pozwalający na wysyłanie i odbieranie wiadomości w formie obieków klasy std::string.
DialogManager – element modelu aplikacji stanowiący swoiste przejście (proxy) między elementami kontrolera a bilioteką Socket. DialogManager zarządza otwartymi połączeniami sieciowymi oraz kontroluje ich stan. Jego podstawową zaletą jest możliwość przyjmowania i odbierania wiadomości tekstowych w postaci zmiennej std::string do wskazanego adresu bez konieczności przejmowania się wywołującego o otwieranie/zamykanie/utrzymywanie połączeń. DialogManager realizuje funkcje serwera chatu dla połczeń przychodzących oraz klienta dla połączeń wychodzących.
FromViewParser – odbiera wiadmości od użytkownika (z widoku), parsuje ich znaczenie po czym wykonuje powiązane z nimi zadania odwołując się, w razie potrzeby zarówno do elementów modelu jak i uaktualniając widok.
ToViewParser – odbiera wiadomości od DialogManagera w postaci obiektów typu std::string. Parsuje je zgodnie z przyjętym standardem (wiadomości XML) opisanym niżej. Po czym wykonuje powiązane z danymi typami wiadomości XML akcje odwołując się zarówno do modelu jak i do widoku.
AliasManager – zarząda zarówno listą aliasów pozwalając na ich rejestrację oraz translacją alias=>IP oraz IP=>alias, jak i listą subskrypcji oraz powiązanym z nim automatem stanowym opisującym stan każdej z nich. Posiada metody pozwalające obsługiwać subskrypcję zarówno na podstawie sygnałów pochodzących od użytkownika jak i bezpośrednio z sieci.
ConferenceManager – zarządza znanymi dla danego hosta konferencjami istniejącymi w sieci. Zapisuje w bazie powiązania nazwy i właściciela z uczestnikami konferencji. Wykorzystywany jest również w przypadku wysyłania wiadomości do konfrencji w celu ustalenia listy adresów, które powinny otrzymać daną wiadmość.
ChatWindow – główna klasa widoku, odpowiada za renderowania okna. Otrzymuje od kontrolera komunikaty dotyczące aktualizacji stanu widoku, wyświetla przychodzące wiadomości, obsługuje interakcję z użytkownikiem itp.
XMLPackageCreator – tworzy pakiety XML na podstawie zadanych informacji zgodnie z ustalonym standardem.

Komunikacja między modułami:
Moduły komunikują się na dwa sposoby:

1. **Synchronicznie –** przez wywołanie metod innych modułów.
   Wykorzystane np:
   - *FromViewParser => DialogManager* – wołanie metod send();
   - *DialogManager => ToViewParser* – jednak jedynie na poziomie dodawania do kolejki komunikatów, gdyż dalsza obsługa komunikatów jest już realizowana asynchronicznie. Wołania zasilenia kolejki SafeQueue są realizowane bezzwłocznie i umożliwiają natychmiastowy powrót do wołającego (DialogManagera).
2. **Asynchronicznie –** z wykorzystaniem kolekcji SafeQueue stanowiącej blokującą (na operacjach push i pop w przypadku niemożliwości ich wykonania) kolejkę dowolnego typu, do której wiele wątków pisze, a zwykle (w przypadku aplikacji ChaTIN w każdym wypadku) jeden wątek jest wydelegowany do cyklicznego sprawdzania zawartości kolejki i obsługiwania otrzymanych przez nią danych zgodnie z ich semantyką.
   Wykorzystane np:
   - ChatWindow => FormViewParser – przesyłanie zdarzeń przez kolejkę elementów klasy Event

- FromViewParser, ToViewParser, AliasManager => ChatWindow – ponieważ gtkmm nie jest bezpieczny wątkowo, zdarzenia mające aktualizować stan widoku są wpychane do kolejki a następnie obsługiwane przez wątek gtk zgodnie z zadaną implementacją jako akcja typu idle (wykonywana "w czasie wolnym"). Stanowi to swoisty odpowiednik rozwiązania tego problemu znanego z języka Java – metody *invokeLater*.

Wątki aplikacji:
Aplikacja posiada następującą pulę wątków:

1. Wątek serwera nasłuchiwania, oczekujący na inicjację połączenia z zewnątrz
2. Po jednym wątku dla każdego użytkownika, z którym jest otwarte połączenie
3. Wątek gtkmm obsługi widoku
4. Wątek ToViewParsera obsługujący asynchronicznie przychodzące Eventy z kolejki SafeQueue
5. Wątek FromViewParsera obsługujący asynchroniczne przychodzące pakiety z DialogManagera.

Standard wiadomości:
Standardowa wiadomość wysyłana za pośrednictwem sieci przez aplikację ChaTIN ma następujące właściwości:

- Obowiązująca wersja standardu XML to 1.0
- Węzłem korzeniem (root) jest węzeł **<message></message>**
- W ramach węzła korzenia wymagany jest węzeł **<type></type>** zawierający jako zawartość jedną ze zdefiniowanych wartości określających rodzaj wiadomości.
  Dostępne typy wiadmości
  - *msg* – wiadmość zwykła
  - *cmsg* – wiadomość konrefencyjna
  - *iky* - (I know you) zapytanie o subskrypcję
  - *ikya* – (I know you accept) zakceptowanie prośby
  - *idky* – (I don't know you) odrzucenie prośby
- W ramach węzła korzenia wymagany jest węzeł **<text></text>** który zawiera niepusty tekst opisujący dany pakiet. Semantyka tego tekstu może różnić się w zależności od typu pakietu jednak zwyczajowo jest to wiadomość przeznaczona do zaprezentowania użytkownikowi.
- W ramach węzła korzenia dozwolona jest dowolna ilość innych informacji tzw. o*therAttributes,* które nabierają znaczenia w kontekście pewnych typów wiadomości np. dla typu *cmsg* dozwolone jest wysłanie dowolnej ilości węzłów o nazwie *memberip* opisujących listę uczestników konferencji.

Szczegóły implementacji a w szczególności opis interfejsu klas, oraz kodu znajduje się w załączonej do tej dokumentacji dokumentacji kodu.

Szczegóły techniczne

Wymagania:

Do wykorzystania:

- Połączenie sieciowe zgodne z IPv6
- Dostępność bibliotek linkowanych dynamicznie (lista niżej)

Do kompilacji:

- gcc >= 4.7.0 lub inny kompilator wspierający taki sam lub większy zakres standardu C++11
- make – wersja zgodna ze standardem POSIX
- headery bibliotek wymaganych w ścieżce include kompilatora lub dostępne jako flaga kompilacji przez narzędzie pkg-config

Technologia:

Program napisano z wykorzystaniem języka C++ wg. najnowszej wersji standardu C++11.

Wykorzystane zewnętrzne biblioteki:

- **BSD Socket**
- boost::optional
- boost::bimap
- boost::bind
- boost::function
- boost::thread
- boost::variant
- boost::lexical_cast
- boost::test – testy nie regresji
- gtkmm 2.4
- wt::dbo - orm dla SQLite3

Wykorzystane narzędzia:

- gcc 4.7.0 – kompilator
- make – narzędzie wspomagające budowanie
- Git 1.7.3.4 – system kontroli wersji, wspierający jego utrzymanie

Załączniki

- Dokumentacja kodu
- Kod źródłowy (do pobrania z repozytorium, znajdującego się pod adresem: https://github.com/maciejgrzybek/ChaTIN.git)
- Skrypt budujący aplikację (Makefile)