UniLANChat

Podręcznik programisty

Tomasz Wasilczyk, Piotr Gajowiak

10 czerwca 2010

Spis treści

1. Wstęp **1**
1.1. Autorzy 2

2. Ogólna dokumentacja programistyczna **2**
2.1. Numeracja wersji 2
2.2. Zasady formatowania 3
2.3. Struktura aplikacji 3
2.4. Przepływ danych 3

3. Protokół: IP Messenger **4**
3.1. Pakiety UDP 4
3.1.1. `NOOP` – pusty pakiet 5
3.1.2. reszta pakietów 5
3.2. Nagłówki strumieni TCP 5
3.3. Scenariusze komunikacji 5

1. Wstęp

UniLANChat to (docelowo) wieloprotokołowy, wieloplatformowy i wielojęzykowy klient czatów p2p. Program powstał w celu zastąpienia komunikatora IP Messenger, używanego między innymi w akademikach Uniwersytetu Wrocławskiego. Podstawowe założenia:

- zgodność z protokołem IP Messenger, komunikatory powinny ze sobą współdziałać w ramach tej samej sieci;
- program powinien dobrze działać zarówno pod systemem Windows, jak i Linux – napisany jest więc w JavaSE;
- ergonomiczny interfejs użytkownika, wzorowany jest na komunikatorze Pidgin;
- udostępniony na otwartej licencji LGPL;
- obsługa wielu języków (w przyszłych wersjach).

Pierwowzór odbiegał od części z powyższych założeń – interfejs jest w nim jeszcze z czasów Windows 3.11, a jego działanie pod systemem Linux pozostawia wiele do życzenia.

W miarę możliwości czasowych, być może będą realizowane następujące cele dodatkowe:

- zgodność z innymi protokołami, np. LanChat, winpopup;
- różne interfejsy użytkownika: graficzny, tekstowy dla konsoli (czyli okienka ASCII) oraz tekstowy dla terminala (np. do obsługi standardowymi strumieniami);
- protokół automatycznego routingu: sieć p2p sama mogła by dbać, aby różni klienci, którzy nie posiadają bezpośredniego połączenia, mogli ze sobą rozmawiać (per proxy, przez innych użytkowników sieci);

- komunikacja głosowa;
- całkowicie zdecentralizowana sieć wymiany plików p2p.

1.1. Autorzy

Autorzy programu:

- Tomasz Wasilczyk (www.wasilczyk.pl),
- Piotr Gajowiak.

Strona projektu: unilanchat.googlecode.com
Pierwsze wersje programu powstały w ramach projektu licencjackiego, prowadzonego na Instytucie Informatyki Uniwersytetu Wrocławskiego (www.ii.uni.wroc.pl).

2. Ogólna dokumentacja programistyczna

Dokumentacja kodu źródłowego znajduje się w oddzielnym dokumencie, wygenerowanym narzędziem javadoc.

2.1. Numeracja wersji

Wersje programu są numerowane według schematu:

$$< major > . < minor > [. < release >][(nightly)]$$

gdzie:

- **major** – główny numer wersji programu, związany z bardzo znaczącymi zmianami. Zaczyna się od 0 (wersja testowa), następne są wersjami przeznaczonymi dla użytkowników końcowych;
- **minor** – kolejne wydania programu, wprowadzają nowe funkcje, lub znaczne poprawki błędów. Numeracja może być wielocyfrowa, zaczyna się od liczby 1 w przypadku gałęzi 0.x oraz od 0 w przypadku kolejnych;
- **release** – drobne poprawki, które nie zostały uwzględnione w wydaniu minor, ale wymagające szybkiej naprawy, np. poprawki bezpieczeństwa, lub błędów uniemożliwiających korzystanie z programu. Wersja 0 nie jest dodawana do ciągu oznaczającego pełny numer wersji;
- **nightly** – kompilacja zrzutu z svn. Jest to wersja kandydująca do tej bez oznaczenia nightly, zazwyczaj do wydania wersji finalnej się znacznie zmienia. Nie jest przeznaczona dla użytkowników końcowych, ani betatesterów.

2.2. Zasady formatowania

- wcięcia tabulacjami;
- pusta linia na końcu pliku, bez białych znaków na końcach linii (również pustych);
- klamry otwierające i zamykające w nowej linii, w tej samej kolumnie co blok nadrzędny, blok podrzędny wcięty o jedną tabulację;
- komentarze javadoc:
  - opis metody z dużej litery, zakończony kropką;
  - opisy parametrów i zwracanych wartości z małej litery, bez kropki na końcu;
  - deklaracje zwracanych wartości boolean: `@return <code>true</code>, jeżeli (...);`
- importowane paczki w dwóch oddzielonych pustą linią sekcjach: java[x] oraz reszta.

2.3. Struktura aplikacji

UniLANChat składa się z następujących komponentów:

- program działający w maszynie wirtualnej Javy, którego kod źródłowy znajduje się w katalogu `java`;
- biblioteki JNI, wykorzystywane przez powyższy program, których kod źródłowy jest w katalogu `jni`. Aby uprościć proces kompilacji projektu (oraz zmniejszyć wymagane zależności), te biblioteki są dostarczane w postaci binarnej (dla wszystkich obsługiwanych platform) z możliwością ich rekompilacji;
- natywnych programów startowych (znajdujących się w katalogu `launcher`) dla systemów Windows oraz Linux. Pierwszy z nich jest generowany programem launch4j (launch4j.sourceforge.net), drugi jest skryptem w bashu.

2.4. Przepływ danych

W programie wykorzystano wzorzec MVC, w związku z czym[1]:

- **model** przechowuje pełne dane na temat implementowanych rozwiązań, np. treść wiadomości, jej autora, datę odebrania. Nie ma jednak możliwości bezpośredniego oddziaływania na widok, ani kontroler (chyba, że tamte go obserwują);
- **widok** służy do wyświetlania danych z modelu oraz wywoływania odpowiednich metod kontrolera, w celu operowania na danych. Widok nie powinien modyfikować bezpośrednio modelu;
- **kontroler** odpowiada za wykonywanie akcji zleconych przez widok, a także przechowywanie wygenerowanych obiektów z modelu (czyli np. listy użytkowników, listy pokoi rozmów). Kontroler nie może bezpośrednio modyfikować widoku, ale może na niego oddziaływać przez wzorzez obserwatora.

[1] w Internecie krąży wiele sprzecznych interpretacji tego wzorca – aby uniknąć nieporozumień, zostanie to tutaj uściślone

Wzorzec obserwatora pozwala na przekazywanie informacji do obiektów, nad którymi obserwowany obiekt nie ma kontroli. Np. model kontaktu (osoby na liście kontaktów) sam w sobie nie może modyfikować widoku listy kontaktów, ale jest przez nią obserwowany[2]. Wtedy w przypadku jego zmiany (np. zmiana statusu dostępności) wysyła o tym powiadomienie wszystkim obserwatorom, a więc także wspomnianemu widokowi.

3. Protokół: IP Messenger

Protokół jest bezpołączeniowy, opiera się o rozgłaszanie swojej obecności poprzez broadcast, za pomocą pakietów UDP. Wykorzystuje opcjonalne szyfrowanie, transfer plików (oparty o TCP), pozwala na użycie opcjonalnego serwera pośredniczącego.

Strona projektu: ipmsg.org.

Domyślnie wykorzystywanym portem (zarówno TCP, jak i UDP) jest 2425 – aby zachować możliwość pełnej komunikacji, należy mieć w firewallu otworzone je także na połączenia przychodzące.

3.1. Pakiety UDP

Pakiety UDP są przesyłane w następującym formacie (jako czysty tekst):

```
<wersja protokołu>:<numer pakietu>:<login nadawcy>:<host nadawcy>:
<polecenie i flagi>:<dane>\0
```

gdzie:

- **wersja protokołu** – ipmsg korzysta tylko z wartości `1`, a implementacja iptux – `1_iptux_0`;
- **numer pakietu** (w formacie dziesiętnym) – unikalny (w skali każdego rozmówcy, z którym wymieniamy pakiety) identyfikator pakietu, dzięki któremu można realizować m.in. kontrolę dostarczania wiadomości;
- **login nadawcy** – w oryginalnej implementacji jest to login zalogowanego użytkownika w systemie nadawcy. W implementacji UniLANChat jest to po prostu jego nick (nie chcemy zdradzać wspomnianej nazwy użytkownika);
- **host nadawcy** – nazwa hostu nadawcy;
- **polecenie i flagi** (w formacie dziesiętnym) – przechowuje zarówno identyfikator polecenia, jak i jego flagi. Aby się do nich odwołać, należy wykonać operacje bitową AND z wartościami:
  - `0x000000FF` – identyfikator polecenia (opisane w klasie `IpmsgPacket`, w pakiecie `protocols.ipmsg`);
  - `0xFFFFFF00` – flagi bitowe, wykorzystywane zależnie od polecenia (opisane j/w);
- **dane** zależne od konkretnego identyfikatora polecenia (np. treść wiadomości, dane n/t klucza itp.), mogą zawierać kolejne separatory (dwukropki);
- `\0` – znak null.

[2] w rzeczywistości model kontaktu jest obserwowany przez model listy kontaktów, a ta dopiero przez widok listy kontaktów

3.1.1. `NOOP` – pusty pakiet

Pusty pakiet, nie udało nam się odkryć jego zastosowania. Nie obsługuje żadnych flag, ani niczego w sekcji danych.

3.1.2. reszta pakietów

TODO

3.2. Nagłówki strumieni TCP

TODO

3.3. Scenariusze komunikacji

TODO