UniLANChat

Podręcznik programisty

Tomasz Wasilczyk, Piotr Gajowiak

17 czerwca 2010

Spis treści

1. Wstęp

UniLANChat to (docelowo) wieloprotokołowy, wieloplatformowy i wielojęzykowy klient czatów p2p[1]. 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;

[1] patrz: Słownik pojęć

- 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[2], 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]patrz: Słownik pojęć

2.4. Zarys przepływu danych

W programie wykorzystano wzorzec MVC, w związku z czym[3]:

- **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 wzorzec obserwatora.

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[4]. 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[5], 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`;

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

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

[5] patrz: Słownik pojęć

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. W oryginalnym kliencie jest inicjowany wartością unix timestamp[6], w UniLANChat losową;

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`, tam też znajdują się ich kody);
- `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.

3.1.1. `NOOP` – pusty pakiet

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

3.1.2. `ENTRY` – powiadomienie o obecności

Rozpoczęcie sesji, lub odświeżenie listy. Może być wysyłany zarówno na adres broadcast (aby powiadomić wszystkich o swojej obecności), jak i pojedyńcze adresy (aby sprawdzić ich dostępność). Odebranie takiego pakietu powoduje dodanie nadawcy do listy kontaktów oraz wysłanie mu odpowiedzi `ANSENTRY`.

Flagi:

- `ABSENCE` – nadawca pakietu ma ustawiony status *zaraz wracam*;
- `SERVER` – zarezerwowany;
- `DIALUP` – oznacza to, że nadawca nie może odbierać wiadomości broadcast. Należy wysyłać do niego takie wiadomości indywidualnie (UniLANChat i tak korzysta z unicastu[7], kiedy tylko się da);
- `FILEATTACH`;
- `ENCRYPT`.

[6] patrz: Słownik pojęć
[7] patrz: Słownik pojęć

Sekcja danych:

`<nick>[opis]\0<grupa>`

gdzie:

nick – konfigurowalny alias użytkownika, do wyświetlenia na liście kontaktów;

opis – status opisowy użytkownika;

grupa – grupa, do której należy użytkownik (także wybierana przez niego).

3.1.3. `EXIT` – powiadomienie o opuszczeniu sieci

Powiadomienie innych użytkowników sieci o jej opuszczeniu przez nadawcę. Po otrzymaniu takiego pakietu należy usunąć nadawcę z listy kontaktów. Flagi i sekcja danych jest taka sama, jak w `ENTRY`.

3.1.4. `ANSENTRY` – potwierdzenie obecności

Potwierdzenie odebrania pakietu `ENTRY`. Nasłuchiwanie tych pakietów jest wykorzystywane do uzupełnienia listy obecności nowych użytkowników. Flagi i sekcja danych jest taka sama, jak w `ENTRY`.

3.1.5. `ABSENCE` – powiadomienie o zmianie statusu

Powiadomienie o zmianie statusu. Flagi i sekcja danych jest taka sama, jak w `ENTRY`. Zasadniczą różnicą między `ABSENCE` a `ENTRY` jest nie wysyłanie pakietu `ANSENTRY` po otrzymaniu tego pierwszego.

3.1.6. `SENDMSG` – wysłanie wiadomości

Pakiet z wysyłaną wiadomością, może zawierać także informacje o załącznikach (które są później przesyłane osobnym kanałem). Wiadomość może być szyfrowana (format w tej chwili nie jest tutaj opisany).

Flagi:

- `SENDCHECK` – nadawca oczekuje na potwierdzenie dostarczenia wiadomości (`RECVMSG`), jeżeli ono nie dojdzie, wiadomość jest wysyłana ponownie;
- `SECRET` – wiadomość jest zapieczętowana, w implementacji UniLANChat ignorowana;
- `READCHECK` – nadawca zostanie poinformowany o próbie otworzenia wiadomości, używana razem z SECRET. W implementacji UniLANChat ignorowana;
- `BROADCAST` – wiadomość do wszystkich obecnych w sieci. Na takie pakiety nie są odsyłane potwierdzenia odbioru. Oryginalna implementacja IPMsg korzysta z tej flagi także dla pakietów, które są rozsyłane tylko do wybranej grupy odbiorców;
- `MULTICAST` – wiadomość do wybranej grupy odbiorców. Na te pakiety są odsyłane potwierdzenia odbioru. Ta flaga jest używana w implementacji UniLANChat dla wszystkich grupowych rozmów (ze względu na wspomniane potwierdzenia odbioru);

- `MULTICAST_NEW` – odpowiednik `MULTICAST`, ale zarezerwowany dla przyszłych wersji protokołu;
- `NOPOPUP` – flaga nieprawidłowa w wersji protokołu Draft-9, ignorowana;
- `AUTORET` – flaga oznaczająca, że odpowiedź jest automatyczna. Jest to zabezpieczenie przed efektem "ping-pong", między dwoma osobami z ustawioną automatyczną informacją o nieobecności;
- `RETRY`;
- `PASSWORD` – flaga zapieczętowania i zabezpieczenia hasłem (ignorowana). W oryginalnej implementacji ipmsg użytkownik przed odpieczętowaniem (i przeczytaniem wiadomości) musi podać swoje hasło, ustawione wcześniej w programie (domyślnie puste);
- `NOLOG` – flaga sugerująca, aby nie zapisywać danej wiadomości w logu (ignorowana);
- `NOADDLIST`.

Sekcja danych:

```
<treść wiadomości>[\0<<numer pliku>:<nazwa pliku>:<rozmiar pliku>:
<czas ostatniej modyfikacji>:<atrybut>
<:dodatkowy atrybut=wartość<,wartość>*>*:\b:>+]
```

gdzie:

treść wiadomości – treść przesyłanej wiadomości, może być pusta;

numer pliku – numer pliku w obrębie pakietu, w którym jest inicjowane jego wysyłanie;

nazwa pliku – nazwa pliku, jeśli w nazwie występuje znak ':' należy go zastąpić przez '::';

rozmiar pliku – rozmiar pliku, dla folderu oryginalny klient ustawia to pole równe 0, w implementacji UniLANChat jest to suma długości wszystkich plików znajdujących się w folderze;

atrybut – jedna z wartości:

`FILE_REGULAR` – plik właściwy;

`FILE_DIR` – folder;

dodatkowy atrybut – dodatkowe informacje o pliku;

wartość – jedna z wartości dodatkowego atrybutu.

\b – znak Bell, kod ASCII o numerze 7.

3.1.7. `RECVMSG` – potwierdzenie odebrania wiadomości

Potwierdzenie odebrania wiadomości informuje odbiorcę, że wiadomość przez niego wysłana została odebrana.

Sekcja danych: identyfikator pakietu z potwierdzaną wiadomością (w formacie dziesiętnym).

3.1.8. `READMSG` – potwierdzenie przeczytania wiadomości

Potwierdzenie otworzenia zapieczętowanej wiadomości (pakiet ignorowany) – jeżeli w wiadomości (`SENDMSG`) była ustawiona flaga `READCHECK`.

3.1.9. `DELMSG` – powiadomienie o odrzuceniu wiadomości

Odpowiednik `READMSG`, ale informujący o odrzuceniu wiadomości.

3.1.10. `ANSREADMSG` – potwierdzenie odebrania `READMSG` lub `DELMSG`

Potwierdzenie odebrania pakietów `READMSG` lub `DELMSG`, działa podobnie do `RECVMSG`. Pakiet ignorowany.

3.2. Nagłówki strumieni TCP

Nagłówki strumieni TCP mają tą samą strukturę jak te opisane w sekcji Pakiety UDP z wyjątkiem nagłówka kontrolującego wysyłanie folderów.

Sekcja danych:

```
<numer pakietu>:<numer pliku>[:domiar]:
```

gdzie:

numer pakietu – numer pakietu, który inicjował wysyłanie danego pliku;

numer pliku – numer pliku, z pakietu inicjującego identyfikowanego przez numer pliku;

domiar – liczba początkowych bajtów, która ma zostać pominięta podczas wysyłania, występuje jedynie w przypadku polecenia `GETFILEDATA` i jest wtedy polem wymaganym.

Wszystkie powyższe wartości są zapisane w systemie szesnastkowym.

3.2.1. `GETFILEDATA` – odbieranie pliku

Wysyłany, gdy użytkownik żąda od serwera wysłania pliku.

3.2.2. `GETDIRFILES` – odbieranie folderu

Wysyłany, gdy użytkownik żąda od serwera wysłania folderu.

3.2.3. Nagłówek kontrolujący wysyłanie folderów

```
<rozmiar nagłówka>:<nazwa pliku>:<rozmiar pliku>:
<atrybut><:dodatkowy atrybut=wartość<,wartość>*>*:<dane>
```

gdzie:

rozmiar nagłówka – rozmiar nagłówka w bajtach, liczony aż do ostatniego bajtu przed danymi;

nazwa pliku – nazwa pliku, jeśli w nazwie występuje znak ':' należy go zastąpić przez '::';

rozmiar pliku – oznacza rozmiar pliku w bajtach, określa jednocześnie długość pola dane;

atrybut – jedna z wartości:

`FILE_REGULAR` – plik właściwy;

`FILE_DIR` – folder;

`FILE_RETPARENT` – nakaz powrotu do katalogu nadrzędnego, używany odczas przechodzenia drzewa katalogów;

dodatkowy atrybut – dodatkowe informacje o pliku;

wartość – jedna z wartości dodatkowego atrybutu;

dane – dane z pliku, o długości równej polu rozmiar pliku.

Wszystkie powyższe wartości są zapisane w systemie szesnastkowym z wyjątkiem nazwy pliku oraz danych.

3.3. Scenariusze komunikacji

3.3.1. Przyłączenie do sieci, odświeżenie listy kontaktów

1. Klient wysyła na adres broadcast pakiet `NOOP` (tylko w oryginalnej implementacji);
2. klient wysyła na adres broadcast pakiet `ENTRY`;
3. inni klienci, którzy otrzymają powyższy pakiet, odsyłają na adres nadawcy pakiet `ANSENTRY`.

UniLANChat dodatkowo, każdemu kontaktowi ze swojej listy, który nie odpowie na ten pakiet, wysyła pakiet `ENTRY` na adres unicast.

3.3.2. Opuszczenie sieci

Wykonywane dwukrotnie, aby zwiększyć pewność, że wszyscy otrzymają tą informację:

1. Klient wysyła na adres broadcast pakiet `NOOP` (tylko w oryginalnej implementacji);
2. klient wysyła na adres broadcast pakiet `EXIT`.

3.3.3. Ustawienie statusu

UniLANChat wykorzystuje to także do ustawienia statusu opisowego (technicznie – zmiany nicka).

1. Klient wysyła na adres broadcast pakiet `NOOP` (tylko w oryginalnej implementacji);
2. klient wysyła na adres broadcast pakiet `ABSENCE`.

3.3.4. Wysłanie wiadomości nieszyfrowanej

1. Nadawca pliku wysyła do odbiorcy pakiet `SENDMSG`, z wiadomością;
2. odbiorca odsyła nadawcy pakiet `RECVMSG`;
3. jeżeli nadawca w określonym czasie nie otrzyma pakietu z potwierdzeniem, ponawia próbę wysłania.

3.3.5. Przesyłanie plików

1. Nadawca pliku wysyła do odbiorcy wiadomość (patrz Wysłanie wiadomości nieszyfrowanej) wraz z ustawioną flagą `FILEATTACH` oraz z sekcją danych opisaną w `SENDMSG` – wysłanie wiadomości
2. dla każdego pliku zawartego w sekcji danych odbiorca wykonuje jedno połączenie do nadawcy używając protokołu TCP na domyślnym porcie. Wysyła pakiet `GETFILEDATA` w przypadku odbierania pliku właściwego, albo `GETDIRFILES` w przypadku folderu z sekcją danych opisaną w Nagłówki strumieni TCP na początek strumienia TCP i oczekuje na napływające dane.
3. nadawca odczytuje żądanie wysłania pliku ze strumienia, odnajduje go na swojej liście plików oczekujących na wysłanie i zależnie od rodzaju przesyłanego pliku wysyła dane na dwa sposoby:
   - jeśli żądanie dotyczyło pliku właściwego to kopiuje dane pliku bajt po bajcie do strumienia TCP. Komunikacja kończy się wraz z ostatnim wysłanym bajtem
   - jeśli żądanie dotyczyło folderu to przechodzi drzewo katalogu algorytmem DFS[8] wypisując nagłówki opisane w Nagłówek kontrolujący wysyłanie folderów do strumienia TCP pamiętając o następujących zasadach:
     - jako pierwszy wypisywany jest nagłówek `FILE_DIR` dla katalogu, który jest wysyłany
     - wchodząc katalog niżej wypisywany jest nagłówek `FILE_DIR` dla katalogu, który jest właśnie odwiedzany
     - napotykając na plik wypisywany jest nagłówek `FILE_REGULAR` wraz z danymi tego pliku w polu dane
     - wychodząc z rekursji wypisywany jest nagłówek `FILE_RETPARENT`

     odbiorca rekonstruuje drzewo katalogu odczytując nagłówki wraz z danymi, wykonując symetryczne akcje do każdego z nagłówków. Komunikacja kończy się, gdy zostanie wypisany nagłówek `FILE_RETPARENT` wychodzący z katalogu będącego korzeniem drzewa DFS

4. Słownik pojęć

broadcast – rozsyłanie jednego pakietu do wszystkich znajdujących się w sieci komputerów.

[8] patrz: Słownik pojęć

czat p2p – protokół rozmów tekstowych w sieci bez pośrednictwa serwera, w której każdy z użytkowników trzyma lokalnie listę wszystkich uczestników rozmowy.

JNI, Java Native Interface – metoda pozwalająca w programach napisanych w Javie używać kodu natywnego, czyli kompilowanego dla konkretnego typu maszyny. Pozwala to korzystać z innych języków programowania, takich jak C++.

unicast – przeciwieństwo *broadcast*u, czyli wysłanie jednego pakietu do jednego odbiorcy.

unix timestamp – czas reprezentowany jako liczba sekund od 1 stycznia 1970, godz. 0:00.

DFS – algorytm przechodzenia grafu w głąb, czyli przeszukiwanie zaczyna się od korzenia i przechodzi w dół do gałęzi, a następnie cofa się o jeden poziom w górę i przeszukuje kolejnego potomka bieżącego węzła.