

BitLab dokumentacja

Przetwarzanie Równoległe i Rozproszone

Skład zespołu: Marek Pałdyna, Jacek Giedronowicz, Konrad Cendrowski, Bartłomiej Kozłowski, Michał Różycki

Spis treści

[Diagram interakcji (sequence diagram) 2](#_Toc62755902)

[Diagram klas 3](#_Toc62755903)

[Docelowe funkcje oprogramowania 8](#_Toc62755904)

[Kroki pośrednie 8](#_Toc62755905)

[Połączenie z peerami i poszukiwanie peerów 8](#_Toc62755906)

[Podtrzymywanie połączenia z peerem 9](#_Toc62755907)

[Obsługa transakcji w systemie Bitcoin 9](#_Toc62755908)

[Polecenie getdata 9](#_Toc62755909)

[Używane biblioteki 10](#_Toc62755910)

[Harmonogram (wykres Gantta) 10](#_Toc62755911)

Diagram interakcji (sequence diagram)

Diagram przepływu dla getAddr:

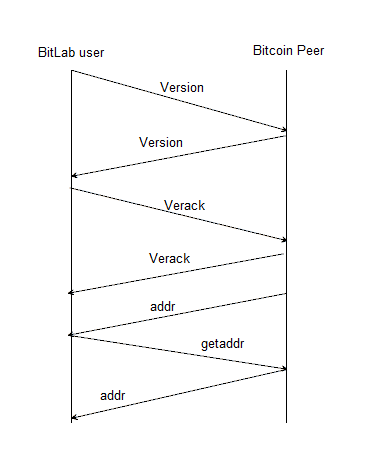
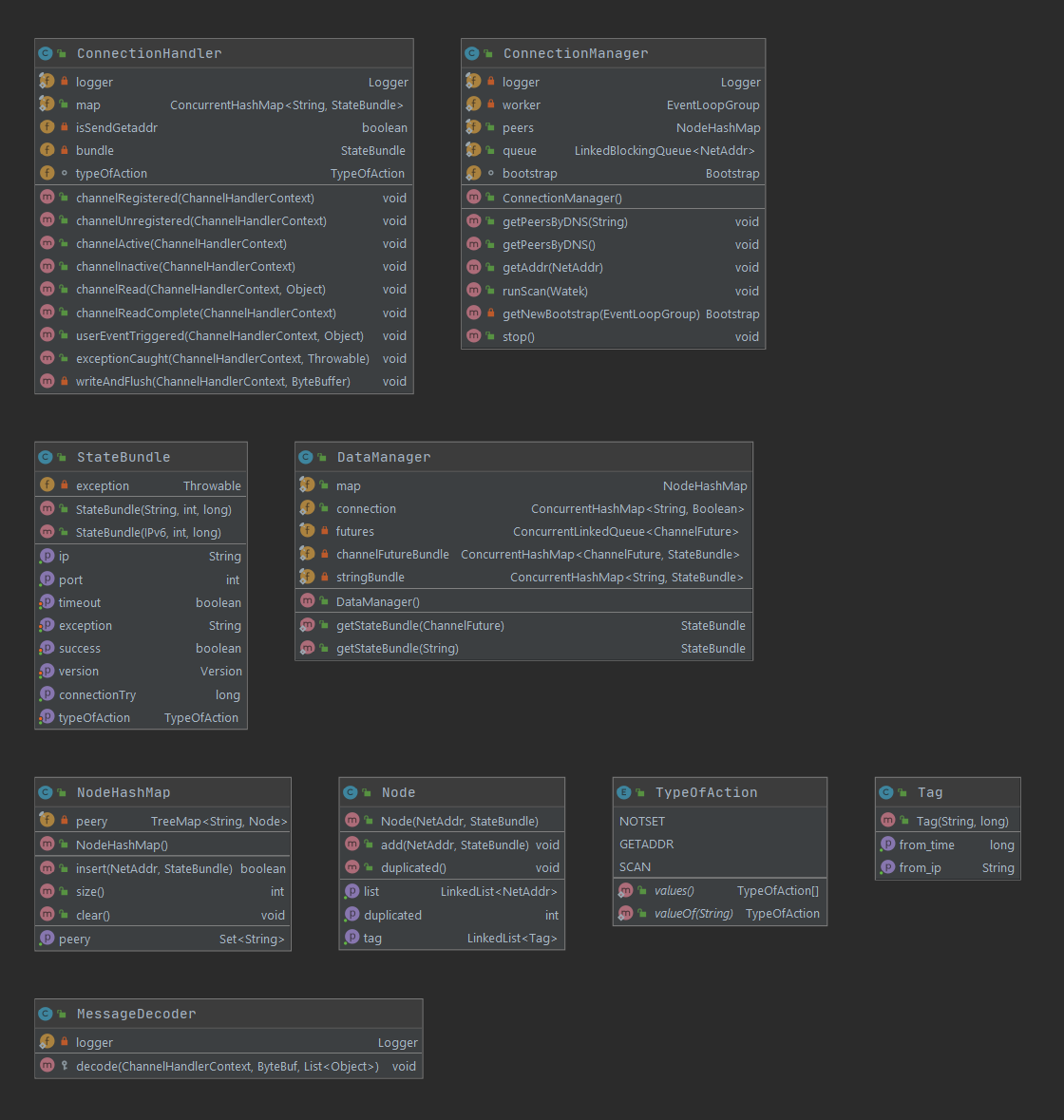


Diagram klas

Ogólny diagram powiązań pakietówObraz zawierający tekst, sprzęt elektroniczny

Opis wygenerowany automatycznie

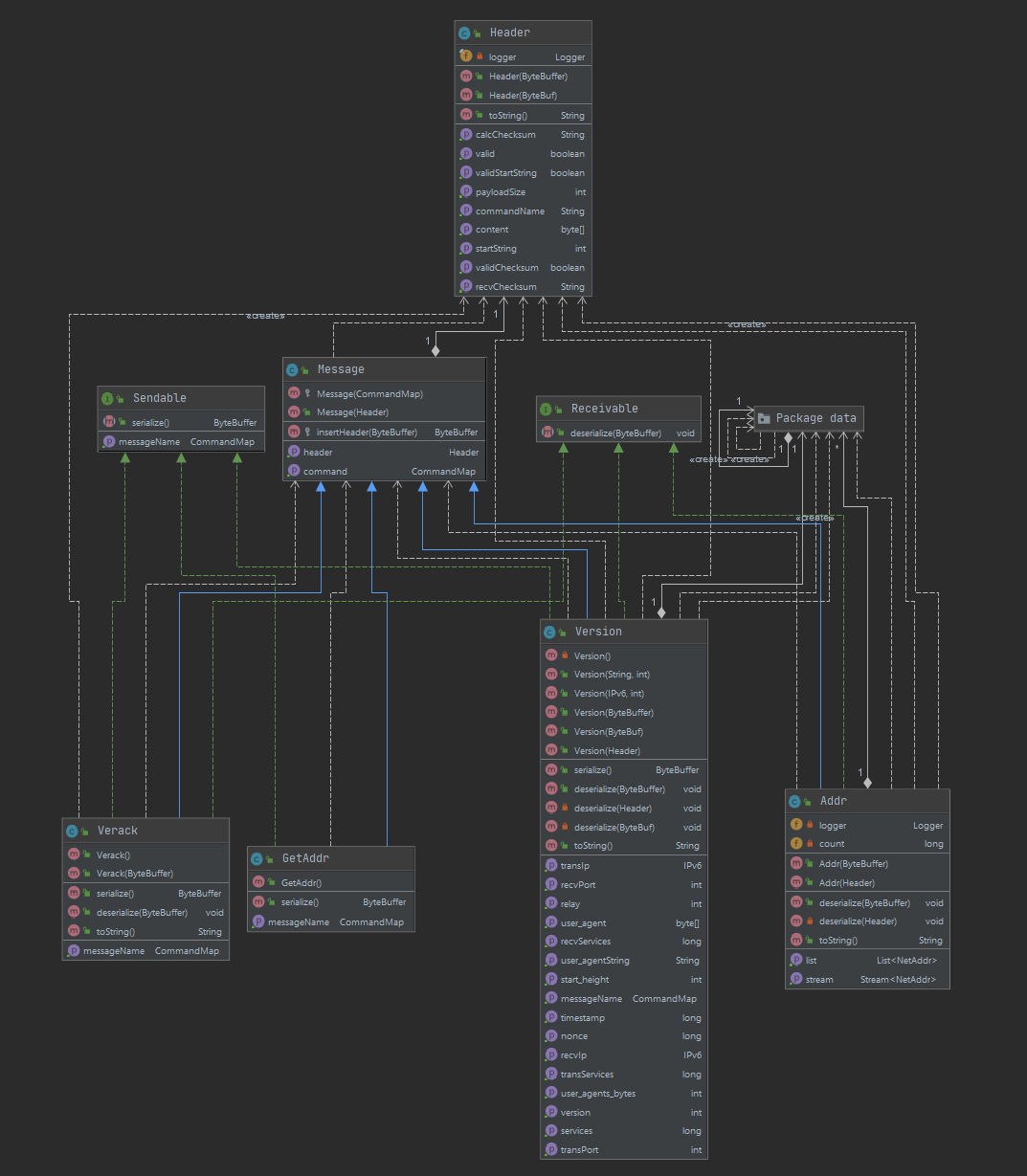
Pakiet connect  


Pakiet constant  
Obraz zawierający tekst

Opis wygenerowany automatycznie

pakiet exception  
Obraz zawierający tekst

Opis wygenerowany automatycznie

pakiet Message  


pakiet UI  
Obraz zawierający tekst, czarny, zrzut ekranu, tabliczka

Opis wygenerowany automatycznie

pakiet util  
Obraz zawierający tekst

Opis wygenerowany automatycznie

Docelowe funkcje oprogramowania

* Rekurencyjne poszukiwanie wszystkich peerów w sieci Bitcoin (obsługa wiadomości i poleceń getaddr, addr, print IP addresses)
* Nawiązywanie i podtrzymywanie połączenia z peerami (obsługa poleceń/wiadomości version i verack oraz ping i pong)
* Obsługa przychodzących transakcji i dalsze jej wysyłanie (obsługa poleceń: inv, getdata, getblocks, getheaders, tx, block, headers)

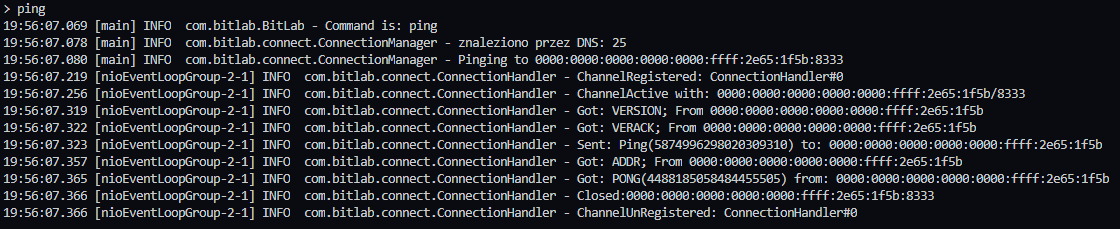
Kroki pośrednie

## Połączenie z peerami i poszukiwanie peerów

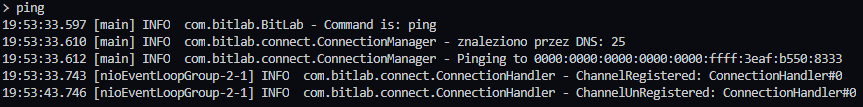
Przy nawiązywaniu połączenia najpierw użytkownik programu BitLab wysyła do znanego sobie peera wiadomość (klasa dziedzicząca po klasie Message) Version, w odpowiedzi peer powinien wysłać swoją wiadomość Version, następnie użytkownik wysyła Verack, a w odpowiedzi powinien otrzymać Verack oraz addr zawierający jeden adres.

## Podtrzymywanie połączenia z peerem

Program obsługuje komendy ping i pong. Po wpisaniu komendy ping, wysyłamy wiadomość ***ping*** do jednego ze znanych hostów. Wiadomość ***ping*** zawiera losową liczbę typu *long*. W odpowiedzi dostajemy wiadomość ***pong*** która również zawiera liczbę losową, ale wygenerowaną z liczby wysłanej w ping. Komenda ta standardowo używana jest do sprawdzania aktywności połączenia z danym peerem i wysyłana jest co 0.5-3 godzin. Błąd TCP oznacza, że połączenie nie jest już nawiązane. W programie zakodowano również, automatyczne odpowiadanie na ***ping*** – ***pong****iem.*



Wykonanie oznaczające nieaktywne połączenie (*brak PONG*):



## Obsługa transakcji w systemie Bitcoin

Obsługa poleceń: inv, getdata, getblocks, getheaders, tx, block, headers, które pozwolą na obsługę bloków transakcji- odbiór i przesyłanie ich dalej.

### Polecenie getdata

Aktualnie program obsługuje polecenie getdata, które po wpisaniu prosi o podanie hasha bloku. Następnie program wykonuje zapytanie do api bitcoina, wysyłając podany hash jako parametr i otrzymuje w odpowiedzi obiekt reprezentujący blok. Obiekt ten zawiera między innymi:

• hash reprezentujący blok

• rozmiar bloku

• wysokość bloku

• ilość transakcji

• hash poprzedzającego bloku w łańcuchu

• tablicę hashów transakcji

obiekt bloku jest wypisywany w konsoli.

Używane biblioteki

W projekcie używana jest biblioteka Javy Netty. Jest to framework umożliwiający pisanie aplikacji sieciowych klient-serwer. Ułatwia obsługę połączeń TCP nie blokując jednocześnie wykonania innych wątków programu. Program BitLab używa jej do połączeń z peerami.

Harmonogram (wykres Gantta)

Obraz zawierający stół

Opis wygenerowany automatycznie