Dokumentacja

Mateusz Banaszek Szymon Kozakiewicz

Joanna Świętosławska

30 stycznia 2020

1 Zespól i jego zadania

Zespół projektowy składał się z 3 osób. Podział obowiązków był następujący:
Mateusz Banaszek - nadzorowanie postepów projektu, testy, dokumentacja,
Szymon Kozakiewicz - programowanie I,
Jaonna świetosławka - programowanie II.

Przegląd posępu prac prezentuje diagram Gantta(patrz rys. 1).

2 Opis programu

Program ma za zadanie udostępniać interfejs do komunikacji z full nodem sieci bitcoin. Pozawala na nawiązanie z nim połączenie oraz podstawową wymianę danych. Można więc uzyskać informacje o adresach innych węzłów czy otrzymać informacje o blokach i ich zawartości. Udostępnia też wysyłanie

week	M.Banaszek	S.Kozakiewicz	J.Swiętosławska
1	zaznajomienie z tematem		
2	Podział obowiązków zespołu	wybór środowiska i metod programowania	
3	Stworzenie Harmonogramu testów i wybór elemtntów do testowania	stworzenie architektury sytemu	
4	testy jednostkowe	programowanie sieciowe	
5	Święta, spotkania kontrolne		
6			
7	testy jednsotkowe i integracyjne	naprawa błedów	
8	przygotowanie dokumentacji		
9	testy akceptacyjne	programowanie sieciowe i komunikacji z uzytkownikiem	
10	testy wydajnościowe,dokończenie dokumentacji	naprawa błedów i refaktoryzacja	Programowanie pozostałych elemntów
11	zamykanie projektu		

Rysunek 1: Wykres Gantta.

wiadomości czysto serwisowych takich jak ping, version czy verack. Możliwe są też różne sposoby na nawiązanie połączenia czyli UDP i TCP.

2.1 Realizowane funkcjonalności

- Ustanowienie połączenia
  - TCP
  - UDP
- Komunikacja z użytkownikiem za pomocą konsoli
- Znajdywanie adresów ip węzłów za pomocą DNS seed
- Wysyłanie wiadomości:
  - getaddr
  - addr
  - version
  - verack
  - ping
  - inv
  - getdata
  - getblocks
- Odbiór wiadomości
  - addr
  - version
  - verack
  - ping
  - inv
  - tx

2.2 Uruchomienie programu

By uruchomić program należy wywołać polecenie

$$python3Console.py$$

Aplikacja była testowana dla pythona w wersji 3.6 na innych wersjach może on nie działać poprawnie.

2.3 Opis działania

Po uruchomieniu programu do konsoli można wpisać następujące polecenia:

- *ping*:* wysyła wiadomość ping do wybranego hosta.
- *polacz*:* ustanawia połączenie z wybranym węzłem. Jeśli wcześniej żaden adres ip nie był ustawiony użytkownik zostanie poproszony o jego podanie. Wykonanie polecenia skutkuje wysłaniem do docelowego węzła wiadomości version. Następnie odebrana zostaje zwrotna wiadomość version od węzła. Na koniec następuje wymiana wiadomościami verack. Jeżeli nie uda się nawiązać połączenia TCP w przeciągu 5 sekund to próba połączenia kończy się niepowodzeniem.
- *help*:* Wyświetla możliwe do wpisania polecenia
- *ustaw adres*:* ustawia adres docelowego węzła
- *dns*:* wyszukiwanie adresów węzłów za pomocą dns. Wyszukane węzły zostają wypisane.
- *getaddr*: uzyskanie adresów innych węzłów z wybranego węzła
- *addr*: wysyła wiadomość addr do wybranego węzła
- *getblocks*: wyświetla bloki wybranego węzła(i hashe)
- *getdata*: wyświetla dane z wybranego bloku. Użytkownik musi wcześniej podać hash identyfikujący wybrany blok.
- *inv*: Wysyła wiadomość inv

Uwaga! Tylko polecenia oznaczone gwiazdką da się wykonać bez wcześniejszego nawiązania połączenia z węzłem(za pomocą polecenia *polacz*)

3 Implementacja

Aplikacja została stworzona przy użyciu języka *python* w wersji 3.6. Nie były używane żadne zewnętrzne biblioteki.

3.1 Opis konstrukcji poszczególnych wiadomości

3.1.1 Nagłówek wiadomości

Każda wiadomość protokołu bitcoin musi rozpoczynać się od nagłówka. Składa się on z pól przedstawionych w tabeli 2

nazwa pola	liczba bajtów	komentarz
time	4	Czas. Pole nie jest obecne gdy adres jest częścią wiadomości version
services	8	-
ipv6/4	16	Pierwsze 12 bajtów przechowuje adres ipv6 pozostałe 4 bajty trzymają ipv4. W naszej implementacji wykorzystujemy tylko ipv4.
port	2	Numer portu

Tablica 1: Zawartość adresu internetowego

nazwa pola	liczba bajtów	Opis
magic letters	4	Każda wiadomość rozpoczyna się od tzw liczb magicznych.Zwykle mają one postać f9beb4d9 (hex)
command	12	Zawiera nazwę wiadomości
length	4	Długość wiadomości w bajtach z pominięciem nagłówka
checksum	4	Suma kontrolna(przez nas nie odczytywana)

Tablica 2: Zawartość nagłówka wiadomości

3.1.2 Adres

w wielu wiadomościach trzeba wysłać adres ip. Zawartość części wiadomości z adresem zaprezentowano w tabeli 1.

3.1.3 Version

Wiadomość version składa się z pól przedstawionych w tabeli 3. Wiadomość wysyłana jest w celu ustalenia zasad komunikacji.

3.1.4 Verack

Wiadomość składa się z samego nagłówka. Wysyłana jest w celu potwierdzenia nawiązania komunikacji.

3.1.5 Getaddr

Wiadomość składa się z samego nagłówka. Wysyłana jest w celu uzyskania nowych adresów ip węzłów.

nazwa pola	liczba bajtów	Opis
version	4	Wskazuje wersje protokołu używaną przez wysyłającego. W naszym wypadku było to 60002
services	8	-
timestamp	8	czas wysłania wiadomości
addr_recv	26	Adres adresata wiadomości
addr_from	26	Adres wysyłającego
nonce	8	-
user_agent	?	-
start_height	4	-
relay	1	-

Tablica 3: Zawartość wiadomości version

nazwa pola	liczba bajtów	komentarz
count	1+	Określa liczbę adresów wysłanych wraz z wiadomością. Wykorzystywany jest tu typ danych zwany varint dlatego nie można jednoznacznie określić wielkości pola
addr_list	?	Lista adresów wysyłanych w wiadomości

Tablica 4: Zawartość wiadomości addr

3.1.6 Addr

Zawartość wiadomości addr została przedstwaiona w tabeli 4

3.1.7 Getblocks

Wiadomość getblocks składa się z pól przedstawionych w tabeli 5. Wysłanie tej wiadomości jest równoznaczne z zarządaniem odpowiedzi w postaci wiadomości inv.

3.1.8 Getdata

Wiadomość getblocks składa się z pól przedstawionych w tabeli 6. Wysłanie tej wiadomości jest równoznaczne z zarządaniem odpowiedzi w postaci wiadomości tx. Do wysłania wiadomości getdata potrzebny jest hash transakcji, której szczegóły chcemy otrzymać w odpowiedzi.

nazwa pola	liczba bajtów	Opis
version	4	Wskazuje wersje protokołu używaną przez wysyłającego. W naszym wypadku było to 60002
liczba hashy	1+	-
hashe nagłówków bloków	32*liczba hashy	
hash stopu	32	Hash składający się z samych zer

Tablica 5: Zawartość wiadomości getblocks

nazwa pola	liczba bajtów	Opis
liczba par typ+hash	1+	-
typ obiektu	4	Wartość to liczba naturalna od 1 do 7. Zastosowaliśmy typ 1, który oznacza transakcję.
hash obiektu	32	Hash obiektu, którego detali żądamy

Tablica 6: Zawartość wiadomości getdata

3.1.9 Inv

Wiadomość inv składa się z pól przedstawionych w tabeli 7. Wysłanie tej wiadomości jest równoznaczne z wysłaniem informacji "posiadam te bloki/transakcje".

3.2 Opis działania pozostałych elementów aplikacji

3.2.1 Ping

Ping jest wysyłany przy użyciu systemowego polecenia ping. Wysyłany jest tylko jednokrotnie, jeśli nie osiągnie hosta pokazywany jest komunikat o porażce.

nazwa pola	liczba bajtów	Opis
liczba par typ+hash	1+	-
typ obiektu	4	Wartość to liczba naturalna od 1 do 7. Zastosowaliśmy typ 1, który oznacza transakcję.
hash obiektu	32	Hash obiektu, który posiadamy

Tablica 7: Zawartość wiadomości inv

TCP

Offset	Oktet	0								1								2								3							
Oktet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Port nadawcy																Port odbiorcy														
4	32	Numer sekwencyjny																															
8	64	Numer potwierdzenia (jeżeli flaga ACK jest ustawiona)																															
12	96	Długość nagłówka				Zarezerwowane				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Szerokość okna														
16	128	Suma kontrolna																Wskaźnik priorytetu (jeżeli flaga URG jest ustawiona)														
20	160	Opcje (jeżeli długość nagłówka > 5, to pole jest uzupełniane "0")																															
...	...	...																															

Rysunek 2: Segment TCP, źródło: wilkiepedia.

3.2.2 DNS seed

Wykorzystywane w celu wyszukiwania adresów węzłów. W tym celu do wyszukiwania dns wrzucana jest nazwa *seed.bitcoin.sipa.be*. Wynikiem wyszukiwania jest kilkadziesiąt adresów węzła bitcoin.

3.2.3 TCP

Protokół transmisji danych, zapewniający dostarczenie pakietów. Obsługuje połączenie klient-serwer (1:1). Klient inicjuje połaczenie. Segment protokołu ukazuje rys. 3.2.3)

3.2.4 UDP

Protokół transmisji danych, dopuszczający pewne straty danych. Przez to ramka jest znacznie mniejsza, ukazuje to rys. 3.2.4). Protokól koncetruje się na szybkości dostarczenia pakietu.

3.3 Diagramy sekwencji

Przedstawiono następujące dogramy sekwencji:

- nawiązywanie połączenia (version, verack) grafika 5
- Uzyskiwania adresów ip z węzła(getaddr, addr) na grafice 4

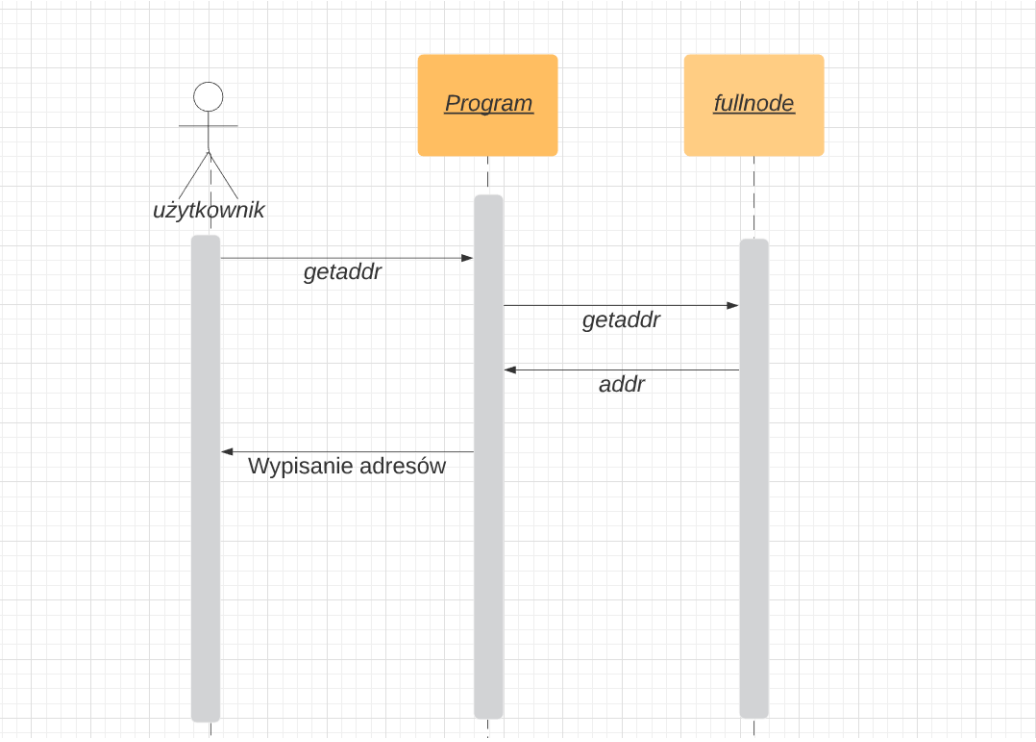
4 FAQ

- Dlaczego program nie działa?
  Należy sprawdzić czy na komputerze zainstalowana jest odpowiednia

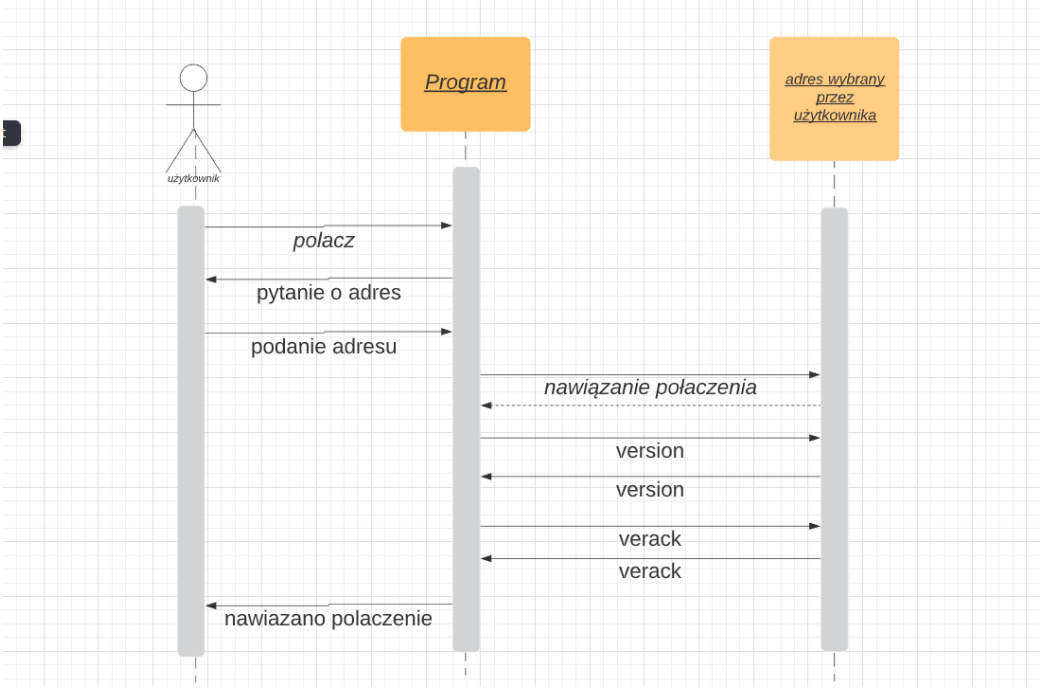
+	Bity 0 – 15	16 – 31
0	Port nadawcy	Port odbiorcy
32	Długość	Suma kontrolna
64	Dane	

Rysunek 3: nagłówek UDP, źródło: wilkiepedia.

Rysunek 4: Diagram sekwencji dla getaddr. Pominięto nawiązywanie połączenia(tcp, version,verack)



Rysunek 5: Diagram sekwencji dla nawiązywania połączenia(version,verack).



wersja języka Python jak i systemu operacyjnego(patrz rozdział subsection 2.2).

- Czemu nie działa komunikacja z serwerem?
  Sprawdź za pomocą polecenia ping czy istnieje połączenie, jeśli tak wpisz polecenie polacz. w innym przypadku skontatuj się z dostawcą internetu.

- Czemu nie dzaiała polecenie?
  Zweryfikuj czy jest to polecenie z "gwiazdką", jesli nie najpierw wpisz polecenie polacz.

- Nie otrzymuje odpowiedzi na wiadomość version gdy używam UDP!
  UDP często *gubi* dane a wiadomość *version* musi dojść w całości żeby otrzymać na nią odpowiedź. Jest to standardowe zachowanie i nie należy się nim przejmować.

- getaddr zwraca zawsze tylko jeden adres ip
  Za pierwszym razem zawsze zwracany jest tylko jeden adres ip. Gdy ponownie wyśle się getaddr zwracanych jest zwykle 1000 adresów