Pawel Gora

Remote Queues Protocol

Opis protokolu

Pawel Gora
4/17/2008

Zawartosc dokumentu

Streszczenie

Niniejszy dokument opisuje specyfikacje protokolu zdalnych kolejek Remote Queues Protocol (zwanego dalej protokolem RQP), jego cele, zalozenia, strukture i sposob dzialania.

Opis celow protokolu

Protokol RQP ma za zadanie rozszerzyc mozliwosc korzystania ze standardowych uniksowych kolejek komunikatow. Rozszerzenie to polega na umozliwieniu uzytkownikom zapisywania komunikatow do kolejek znajdujacych sie na zdalnych maszynach.

Protokol ma stworzyc warunki, aby zdalne przesylanie wiadomosci bylo mozliwe:

- zdalny nadawcy powinien miec mozliwosc zlokalizowania w sieci interesujacej go kolejki
- nadawca oraz odbiorca komunikatu musza miec mozliwosc nawiazania polaczenia
- protokol powinien zagwarantowac obsluge sytuacji wyjatkowych oraz niezawodna transmisje danych

Opis zalozen protokolu

Protokol RQP dziala w warstwie aplikacji i jest w pelni zdecentralizowany, obowiazuje architektura P2P (ang. *peer-to-peer*). Bedzie on korzystal z *kanalu transmisyjnego* realizowanego za pomoca protokolu *TCP/IP* oraz protokolu *UDP*.

Protokol TCP/IP zapewni niezawodnosc w dostarczania pakietow.

Protokol UDP daje mozliwosc rozglaszania komunikatu pomiedzy wszystkie maszyny i bedzie wykorzystywany jedynie do zgloszenia zapytania o dostep do kolejki komunikatow.

Kanal transmisyjny sluzy do przesylania wiadomosci przez *agenta nadajacego* do zdalnej kolejki komunikatow, do ktorej dostep ma *agent odbierajacy*.

Opis formatu komunikatow

Zalozenia wstepne

Przyjeto nastepujace zalozenia odnosnie przesylanych danych:

- sieciowy porzadek oktetow w liczbach
- brak interpretacji znakow (przesyla sie liczby bezznakowe)
- nie uzywa sie liczb zmiennoprzecinkowych
- brak uzupelnienia formatu

Ogolny format komunikatow

Ogolny format komunikatow ma nastepujaca postac:

Nazwa pola	Opis pola	Jednostka	Ilosc jednostek	Rozmiar pola (w bajtach)
type	Zawiera identyfikator przesylanego komunikatu	uint32	1	4
ip	Zawiera adres IP nadawcy	uint32	1	4
port	Zawiera port nadawcy komunikatu, na ktory trzeba odpowiedziec	uint32	1	4
key	Identyfikator uniksowej kolejki komunikatow	uint32	1	4
inne dane	Pole opcjonalne, jego zawartosc i rozmiar zaleza od typu komunikatu	Octet	x	x

Razem bajtow: 16 + x.

W dalszej czesci dla kazdego konkretnego komunikatu pojawi sie specyfikacja jakie pola sa zawarte w miejscu oznaczonym jako *inne dane*.

Powyzszy format specyfikuje jakie dane powinny zostac dostarczone drugiej stronie, aby cala komunikacja przebiegala poprawnie, ale dopuszcza sie, ze mozna zrezygnowac z niektorych pol komunikatu (np pola *ip*), jezeli ich zawartosc da sie uzyskac z innych zrodel w trakcie implementacji.

Komunikat REQ_MSG

Jest to komunikat, ktory jest wysylany do zdalnych maszyn odczytanych przez nadawce z pliku konfiguracyjnego. Celem tego komunikaty jest zgloszenie zapytania o konkretna uniksowa kolejke komunikatow. Komunikat bedzie przesylany przy pomocy protokolu UDP.

Pola komunikatu REQ_MSG

Nazwa pola	Opis pola	Jednostka	Ilosc jednostek	Rozmiar pola (w bajtach)
type	Zawiera identyfikator przesylanego komunikatu	uint32	1	4
ip	Zawiera adres IP nadawcy	uint32	1	4
port	Zawiera port nadawcy komunikatu, na ktory trzeba odpowiedziec	uint32	1	4
key	Klucz kolejki, do ktorej adresowany jest komunikat	uint32	1	4

Razem bajtow: 16.

Komunikat OK_REQ_MSG

Jest to komunikat, ktorym zglaszane jest posiadanie potrzebnej kolejki.

Pola komunikatu OK_REQ_MSG

Nazwa pola	Opis pola	Jednostka	Ilosc jednostek	Rozmiar pola (w

				bajtach)
type	Zawiera identyfikator przesylanego komunikatu	uint32	1	4
ip	Zawiera adres IP nadawcy	uint32	1	4
port	Zawiera port nadawcy komunikatu, na ktory trzeba odpowiedziec	uint32	1	4
key	Klucz kolejki, ktorej posiadanie zglasza nadawca	uint32	1	4

Razem bajtow: 16.

Komunikat SEND_MSG

Komunikat sluzy do przesylania wiadomosci, ktora ma zostac umieszczona w zdalnej kolejce.

Pola komunikatu SEND_MSG

Nazwa pola	Opis pola	Jednostka	Ilosc jednostek	Rozmiar pola (w bajtach)
type	Zawiera identyfikator przesylanego komunikatu	uint32	1	4
ip	Zawiera adres IP nadawcy	uint32	1	4
port	Zawiera port nadawcy komunikatu, na ktory trzeba odpowiedziec	uint32	1	4
key	Klucz kolejki, do ktorej adresowana jest wiadomosc	uint32	1	4
msg_id	Identyfikator wiadomosci	uint32	1	4
msg_len	Dlugosc wiadomosci	uint32	1	4
msg	Tresc wiadomosci	Octet	msg_len	msg_len

Razem bajtow: 24 + msg_len

Pole *msg_id* zawiera identyfikator komunikatu. Wlasciwym identyfikatorem komunikatu jest liczba *msg_id* / 2. Najmniej znaczacy bit pola msg_id oznacza typ komunikatu (1 = pewny, 0 = niepewny).

Komunikaty CONF_MSG

Sa to komunikaty do potwierdzania wstawienia/odrzucenia wiadomosci pewnej.

Ogolny format komunikatow z tej grupy jest nastepujacy:

Nazwa pola	Opis pola	Jednostka	Ilosc jednostek	Rozmiar pola (w bajtach)
type	Zawiera identyfikator przesylanego komunikatu	uint32	1	4
ip	Zawiera adres IP nadawcy	uint32	1	4
key	Klucz kolejki, do ktorej adresowana byla wiadomosc	uint32	1	4
msg_id	Identyfikator wiadomosci, ktorej dotyczy odpowiedz	uint32	1	4

Razem bajtow: 16.

Komunikaty z tej grupy beda roznily sie zawartoscia pola *type*.

Komunikat OK_CONF_MSG
Komunikat informuje, ze wiadomosc zostala pomyslnie wstawiona do kolejki docelowej.

Komunikat NOT_CONF_MSG
Komunikat informuje, ze wiadomosc nie zostala wstawiona do kolejki docelowej oraz brak jest kolejki DEAD.LETTER.Q po stronie odbiorcy. Oznacza to, ze agent nadajacy wiadomosc powinien zakonczyc dzialanie.

Komunikat QUEUE_CONF_MSG
Komunikat informuje, ze nie istnieje kolejka, do ktorej adresowana byla wiadomosc. Wiadomosc zostala natomiast wstawiona do kolejki DEAD.LETTER.Q.

Komunikat EXIST_CONF_MSG
Komunikat informuje, ze wiadomosc zostala wstawiona przez agenta odbierajacego do kolejki DEAD.LETTER.Q, poniewaz wiadomosc o tym identyfikatorze juz istnieje w kolejce komunikatow.

Komunikat ERROR_CONF_MSG
Komunikat informuje, ze wiadomosc zostala wstawiona przez agenta odbierajacego do kolejki DEAD.LETTER.Q, poniewaz wiadomosc ta byla bledna.

Komunikat OTHER_CONF_MSG
Komunikat informuje, ze wiadomosc zostala wstawiona przez agenta odbierajacego do kolejki DEAD.LETTER.Q, a powod nie jest wyspecyfikowany.

Opis mozliwych stanow

Mozliwe stany agenta wysylajacego
Agent wysylajacy moze znajdowac sie nastepujacych stanach:

1. STOPPED – proces nie jest uruchomiony
2. NO_MESSAGE – w tym stanie agent nie ma zadnej wiadomosci do wyslania i musi odczytac wiadomosc z kolejki transmisyjnej (lub czekac az taka wiadomosc sie w niej pojawi). Po odczytaniu wiadomosci z kolejki transmisyjnej agent sprawdza, czy wiadomosc jest poprawna (np. ma poprawny naglowek). W przypadku stwierdzenia bledu wiadomosc zostaje wstawiona do kolejki DEAD.LETTER.Q, a agent pozostaje w stanie NO_MESSAGE. Jezeli wiadomosc jest poprawna, to agent przechodzi do stanu SEARCH_QUEUE.
3. SEARCH_QUEUE – w tym stanie agent wie jaka wiadomosc musi wyslac i zna jej docelowa kolejke. Rozglasza komunikat REQ_MSG do maszyn odczytanych z pliku konfiguracyjnego i przechodzi do stanu WAIT_FOR_REPLY.

4. WAIT_FOR_REPLY – po wyslaniu komunikatu REQ_MSG agent czeka na odpowiedz przez okreslony czas (*timeout).* Jezeli otrzyma odpowiedz (komunikat OK_REQ_MSG), to przechodzi do stanu SEND_MESSAGE. Jezeli odebrany zostanie komunikat typu CONF_MSG, to agent wykonuje stosowne czynnosci:
    - usuwa odpowiednia wiadomosc z kolejki transmisyjnej, jezeli zostala ona umieszczona w kolejce komunikatow po stronie odbiorcy lub w kolejce DEAD.LETTER.Q i przechodzi do stanu WAIT_FOR_REPLY
    - przechodzi do stanu STOPPED, jezeli wiadomosc nie zostala umieszczona w kolejce komunikatow ani w kolejce DEAD.LETTER.Q po stronie odbiorcy

   Jezeli uplynie okreslony czas czekania (timeout), to agent przechodzi do stanu NO_MESSAGE.

5. SEND_MESSAGE – w tym stanie agent posiada juz zlokalizowanego agenta odbierajacego i rozpoczyna wysylanie mu wiadomosc. Jezeli wiadomosc byla oznaczona jako *pewna*, to po jej wyslaniu agent przechodzi do stanu WAIT_FOR_CONF. W przeciwnym razie przechodzi do stanu NO_MESSAGE.
6. WAIT_FOR_CONF – w tym stanie agent czeka przez okreslony czas (timeout) na potwierdzenie od odbiorcy czynnosci wykonanych na wiadomosci pewnej (komunikat typu CONF_MSG). Jezeli wiadomosc zostala wstawiona do kolejki komunikatow (komunikat OK_CONF_MSG), to agent usuwa ja z kolejki transmisyjnej i przechodzi do stanu NO_MESSAGE. Jezeli zostala wstawiona do kolejki DEAD.LETTER.Q po stronie odbiorcy, to:
    - Agent usuwa wiadomosc z kolejki transmisyjnej, jezeli byla ona juz wczesniej wstawiona do kolejki komunikatow (komunikat EXIST_CONF_MSG) i przechodzi do stanu NO_MESSAGE
    - Agent przechodzi do stanu SEARCH_QUEUE, jezeli dostal informacje, ze kolejka nie istnieje (komunikat QUEUE_CONF_MSG).
    - Agent ponownie sprawdza poprawnosc wiadomosci, jezeli otrzyma komunikat ERROR_CONF_MSG lub OTHER_CONF_MSG. Jezeli wiadomosc jest niepoprawna, to zostaje wstawiona przez agenta do kolejki DEAD.LETTER.Q i agent przechodzi do stanu NO_MESSAGE. W przeciwnym razie agent od razu przechodzi do stanu NO_MESSAGE i wiadomosc pozostaje w kolejce (opcjonalnie moze rowniez przejsc do stanu SEND_MESSAGE lub SEARCH_QUEUE).

   Jezeli odebrany zostanie komunikat NOT_CONF_MSG, to proces agenta konczy sie (stan STOPPED).

   Jezeli odebrany zostanie komunikat OK_REQ_MSG, to jest on ignorowany i agent pozostaje w stanie WAIT_FOR_RESPOND.

   Jezeli przekroczony zostanie timeout, to agent przechodzi do stanu NO_MESSAGE.

Mozliwe stany agenta odbierajacego

Agent odbierajacy moze znajdowac sie w nastepujacych stanach:

1. STOPPED – w tym stanie proces agenta nie jest uruchomiony
2. WAIT_FOR_MSG – w tym stanie agent czeka na zapytania od innych agentow. Jezeli odbierze komunikat REQ_MSG, to sprawdza czy jest w posiadaniu odpowiedniej kolejki komunikatow. Jezeli posiada kolejke, to przesyla do agenta nadajacego komunikat zwrotny OK_REQ_MSG. Po odebraniu komunikatu SEND_MSG agent spradza, czy ma dostep do odpowiedniej kolejki komunikatow, czy odebrana wiadomosc jest poprawna i czy istnieje juz w kolejce komunikatow. Nastepnie wstawia wiadomosc do odpowiedniej kolejki (komunikatow lub DEAD.LETTER.Q), o ile jest to mozliwe. Po wykonaniu tych czynnosci przesyla do nadawcy odpowiedni komunikat typu CONF_MSG.

   Jezeli proces agenta zostanie zakonczony, to przyjmuje sie, ze agent przechodzi do stanu STOPPED.

Diagamy stanow

Diagram stanow agenta nadajacego

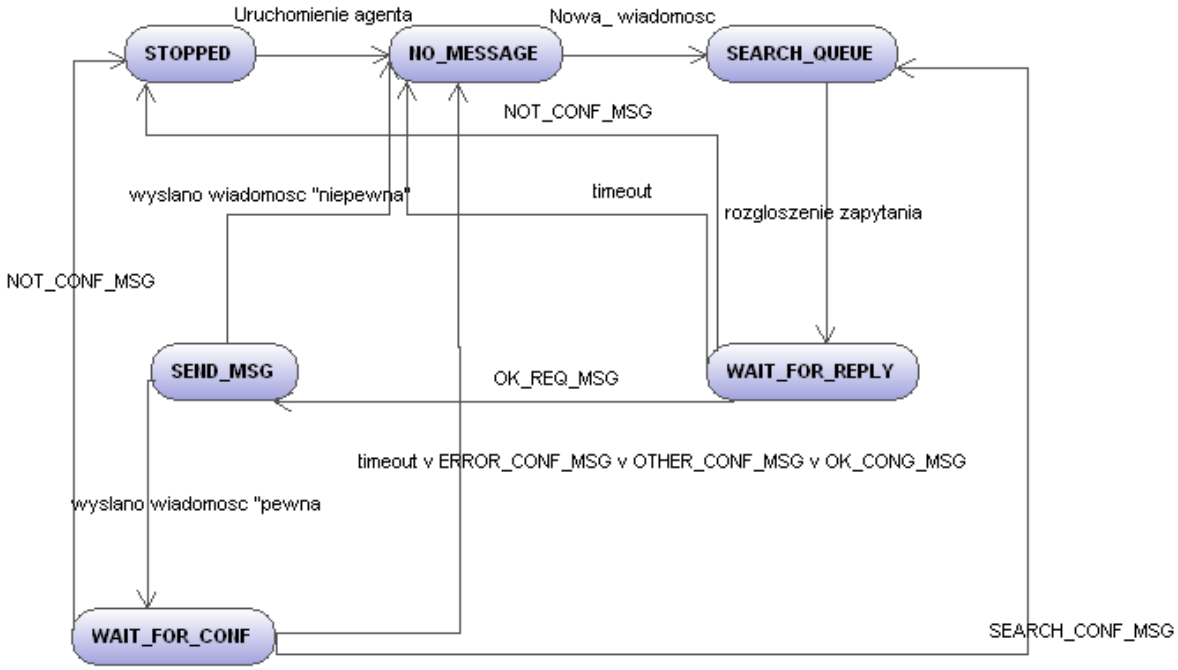
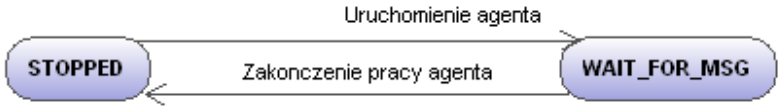


Diagram stanow agenta odbierajacego



Inicjowanie protokolu

Dla ustalenia uwagi przyjmuje sie, ze wszyscy agenci odbierajacy beda czekali na zapytania na domyslnym porcie o numerze 7777. Wlasciwe wiadomosci moga byc juz przesylane na inny port (podany przez agenta). Kazdy z agentow nadajacych bedzie posiadal plik konfiguracyjny z wykazem maszyn, do ktorych moze kierowac zapytanie o dostep do kolejki komunikatow.

Mozliwe rozszerzenia i wskazowki implementacyjne

Dopuszcza sie, aby agenci nadajacy posiadali w cache'u wykaz kolejek komunikatow, ktore moga znajdowac sie na poszczegolnych maszynach. Poniewaz kolejki moga byc dynamicznie dodawane i usuwane, wiec dane te nie musza byc precyzyjny. Agent nadajacy moze jednak tuz po rozpoczeciu pracy odpytac agentow odbierajacych o posiadane przez nich kolejki, a te informacje moga zostac wykorzystane do proby odpytania niektorych agentow o dostep do kolejki przed rozgloszeniem zapytania o kolejke.

Agent odbierajacy moze zostac zmodyfikowany tak, aby dla kazdej przesylanej wiadomosci tworzyl osobny watek, ktory bedzie obslugiwal transmisje, dzieki czemu bedzie mozliwe jednoczesne odbieranie wielu wiadomosci przez agenta.

Uzywane numery

Komunikaty nadawcy

Komunikat	Numer
REQ_MSG	1000
SEND_MSG	1001

Komunikaty odbiorcy

Komunikat	Numer
OK_REQ_MSG	2000
OK_CONF_MSG	2001
NOT_CONF_MSG	2002
WAIT_CONF_MSG	2003
QUEUE_CONF_MSG	2004
OTHER_CONF_MSG	2005
EXIST_CONF_MSG	2006

Stany agenta nadajacego

Stan	Numer
STOPPED	100
NO_MESSAGE	101
SEARCH_QUEUE	102
WAIT_FOR_REPLY	103

SEND_MESSAGE	104
WAIT_FOR_CONF	105

Stany agenta odbierajacego

Stan	Numer
STOPPED	200
WAIT_FOR_MSG	201