Opis protokołu RQP

Piotr Butryn 233978

21 kwietnia 2008

1 Streszczenie

Niniejszy dokument opisuje protokół RQP (Remote Queues Protocol) służący do przesyłania informacji między kolejkami na zdalnych maszynach.

2 Cele

Celem protokołu RQP jest rozszerzenie możliwości korzystania ze zwykłych uniksowych kolejek komunikatów, w ten sposób iż dane zapisywane są przez sieć do kolejek na systemach zdalnych.

3 Założenia

Protokół będzie działał w warstwie aplikacji. Model komunikacji będzie oparty o model Klient-Serwer. Rolę klientem będzie pełnił Message Channel Agent (oznaczany później jako MCA) wysyłający, a serwera MCA odbierający. Kanał transmisyjny będzie zaimplementowany za pomocą protokołu TCP ze względu na niezawodną transmisję danych, którą zapewnia.

MCA wysyłający jest tworzony, gdy po raz pierwszy zostanie wywołana funkcja openConnection dla danego klucza kolejki. Na początku czyta on z pliku konfiguracyjnego (o nazwie .rqprc umieszczonego w katalogu bieżącym) listę maszyn, na których mogą działać MCA odbierający. Jednocześnie tworzy też kolejkę DEAD.LETTER.Q. Następnie pobiera on w nieskończonej pętli komunikaty (wraz z nagłówkami), które ma przesłać do kolejek zdalnych. Wiadomości niepoprawne, z którymi nie wiadomo co zrobić (np. bez klucza kolejki) trafiają do stworzonej wcześniej kolejki DEAD.LETTER.Q. Pozostałe komunikaty są natomiast przeznaczone do wysłania. Każdy MCA wysyłający posiada w swojej pamięci strukturę przechowującą informacje, na którą zdalną maszynę ma zostać wysłany komunikat o danym kluczu kolejki. Jeśli jednak dla danego komunikatu MCA odbierający, do którego ma on trafić, nie jest zdefiniowany to MCA wysyłający wysyła po kolei, do wszystkich maszyn z listy, zapytanie czy udostępniają one zdalną kolejkę o tym samym kluczu co klucz komunikatu. Jeśli od któregoś MCA nadejdzie odpowiedź twierdząca, to agent wysyłający przerywa odpytywanie i zapisuje do swojej pamięci dane tego MCA i w przyszłości dla komunikatów o tym kluczu korzysta z tego właśnie MCA. Jeśli natomiast żaden z MCA nie odpowie twierdząco to: jeśli wiadomość należała to typu niepewnych, to trafia do kolejki

DEAD.LETTER.Q, jeśli natomiast do typu pewnych to MCA wysyłający przerywa działanie, a komunikat trafia z powrotem do kolejki komunikatów oczekujących na wysłanie. Jeśli wszystkie powyższe czynności zakończą się sukcesem, to komunikat dostaje swój unikalny klucz (identyfikujący agenta wysyłającego i komunikat). W tym momencie następuje wysłanie komunikatu do odpowiedniego MCA odbierającego. W przypadku wiadomości niepewnych następuje krótkie oczekiwanie na potwierdzenie, które jeśli nie zakończy się sukcesem, to następuje zwiększenie licznika nieudanych prób. Jeśli przekroczy on pewną wartość (np. 3), to następuje wykasowanie ze struktury w pamięci danego MCA odbierającego jako domyślnego dla kolejki o danym kluczu. W przypadku wiadomości oznaczonych jako pewne, nadejście komunikatu potwierdzającego powoduje zakończenie operacji wysyłania sukcesem, natomiast jego brak skutkuje wykasowaniem ze struktury w pamięci danego MCA jako domyślnego i powtórzenie od początku wszystkich czynności dla danego komunikatu.

Z kolei MCA odbierający czeka na nadejście komunikatów od agentów wysyłających. Dowiaduje się on (korzystając ze standardowych funkcji związanych z kolejkami uniksowymi), do jakich kolejek lokalnych ma dostęp (prawo do zapisu) i te kolejki może zgłosić jako udostępnione odpytującym go agentom wysyłającym. Gdy natomiast otrzyma wiadomość od agenta wysyłającego to w przypadku, gdy jest ona niepoprawna (np. MCA nie ma już dostępu do kolejki o danym kluczu), to trafia ona do lokalnej kolejki DEAD.LETTER.Q, a w przeciwnym wypadku wysyłane jest potwierdzenie odbioru wiadomości.

4 Format i opis komunikatów

Przez sieć bedą przesyłane liczby w sieciowym porządku oktetów.

4.1 Komunikat wiadomości

Komunikat wiadomości służy do przesyłania wiadomości z kolejki MCA wysyłającego do kolejki MCA odbierającego.

MESSAGE {
`uint8 message_code`
`uint8 message_type`
`uint64 key`

uint64 queue_key
uint64 message_length
octet[message_length] message
}
Poszczególne pola:
message_code - kod komunikatu (w tym przypadku 1)
message_type - typ wiadomości (SURE_MSG lub UNSURE_MSG)
key - unikalny identyfikator dla pary MCA wysyłający, komunikat
queue_key - klucz kolejki do której ma trafić komunikat
message_length - długość właściwej wiadomości (bez nagłówka)
message - właściwa wiadomość

4.2 Potwierdzenie odbioru komunikatu

Komunikat potwierdzający odbiór komunikatu przez MCA odbierającego

DELIVERED_YES {
uint8 message_code
uint8 data
}
Poszczególne pola:
message_code - kod komunikatu (w tym przypadku 2)
data - zarezerwowane do użytku w przyszłości

4.3 Błąd odbioru komunikatu

Komunikat o błędzie po stronie MCA odbierającego

DELIVERED_ERROR {
uint8 message_code
uint8 data
}
Poszczególne pola:
message_code - kod komunikatu (w tym przypadku 3)
data - zarezerwowane do użytku w przyszłości

4.4 Zapytanie o posiadanie kolejki o danym kluczu

Komunikat wysyłany przez MCA odbierającego w celu dowiedzenia się, który z MCA wysyłających dysponuje kolejką o danym kluczu.

QUEUE_ASK {
`uint8 message_code`
`uint8 data`
`uint64 queue_key`
}
Poszczególne pola:
`message_code` - kod komunikatu (w tym przypadku 4)
`data` - zarezerwowane do użytku w przyszłości
`queue_key` - klucz kolejki, o którą pytamy

4.5 Zgłoszenie posiadania kolejki

Komunikat wysyłany przez MCA odbierającego potwierdzający obsługę kolejki o danym kluczu

QUEUE_YES {
`uint8 message_code`
`uint8 data`
}
Poszczególne pola:
`message_code` - kod komunikatu (w tym przypadku 5)
`data` - zarezerwowane do użytku w przyszłości

4.6 Zgłoszenie nie posiadania kolejki

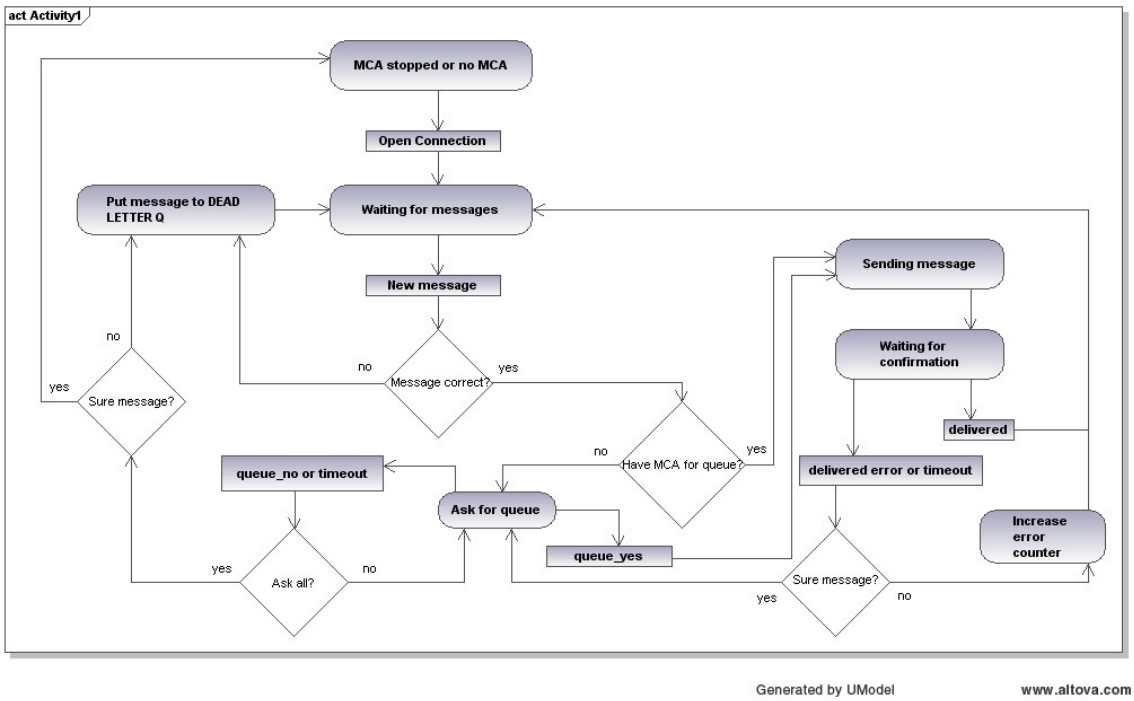
Komunikat wysyłany przez MCA odbierającego informujący o nieobsługiwaniu danej kolejki

QUEUE_NO {
`uint8 message_code`
`uint8 data`
}
Poszczególne pola:

`message_code` - kod komunikatu (w tym przypadku 6)
`data` - zarezerwowane do użytku w przyszłości

5 Opis stanów

5.1 MCA wysyłający



Stany MCA wysyłającego:

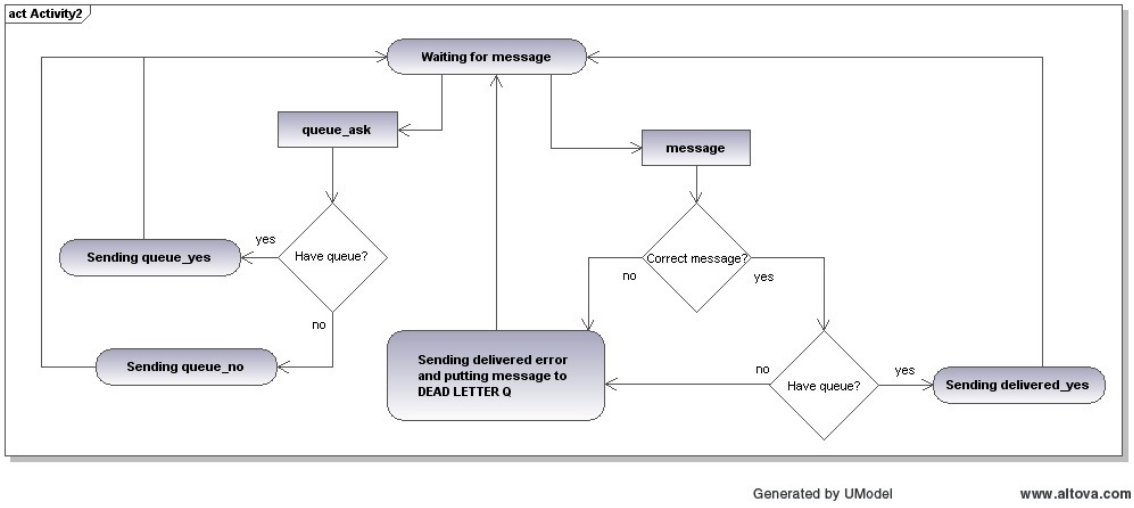
- MCA stopped or no MCA - agent wstrzymany lub brak agenta
- Waiting for message - oczekiwanie na pojawienie się jakiejś wiadomości w kolejce
- Sending message - wysyłanie wiadomości
- Waiting for confirmation - czekanie na potwierdzenie odebrania komunikatu
- Icrease error counter - zwiększanie licznika błędów odebrania dla danej kolejki i danego MCA odbierającego

- Put message to DEAD.LETTER.Q - wrzucanie wiadomości do kolejki DEAD.LETTER.Q
- Ask for queue - rozsyłanie do wszystkich MCA odbierających zapytania o obsługę kolejki o danym kluczu

Zdarzenia wpływające na zmianę stanu MCA wysyłającego:

- Open Connection - wywołanie przez użytkownika funkcji openConnection
- New message - istnienie w kolejce jakiejś wiadomości do wysłania
- Delivered - odebranie komunikatu typu DELIVERED_YES
- Delivered error or timeout - odebranie komunikatu typu DELIVERED_ERROR lub przekroczenie czasu oczekiwania na potwierdzenie
- Queue_yes - odebranie komunikatu typu QUEUE_YES
- Queue_no or timeout - odebranie komunikatu typu QUEUE_NO lub przekroczenie czasu oczekiwania na odpowiedź

5.2 MCA odbierający



Stany MCA odbierającego:

- Waiting for message - oczekiwanie na pojawienie się jakiegoś komunikatu
- Sending delivered_yes - wysyłanie komunikatu typu DELIVERED_YES
- Sending queue_yes - wysyłanie komunikatu typu QUEUE_YES
- Sending queue_no - wysyłanie komunikatu typu QUEUE_NO
- Sending delivered error and putting message to DEAD.LETTER.Q - wrzucanie wiadomości do kolejki DEAD.LETTER.Q i wysyłanie komunikatu typu DELIVERED_ERROR

Zdarzenia wpływające na zmianę stanu MCA wysyłającego:

- Message - odebranie komunikatu typu MESSAGE
- Queue_ask - odebranie komunikatu typu QUEUE_ASK

6 Podsumowanie używanych stałych

6.1 Typy komunikatów

MESSAGE = 1
DELIVERED_YES = 2
DELIVERED_ERROR = 3
QUEUE_ASK = 4
QUEUE_YES = 5
QUEUE_NO = 6

6.2 Inne stałe

SURE_MSG = 0
UNSURE_MSG = 1
ERROR_LIMIT = 3