

Opis protokołu obsługi zdalnych kolejek RQP (Remote Queues Protocol)

Michał Łyczek

14 kwietnia 2008

Spis treści

1	Streszczenie	3
2	Cele	3
3	Założenia	3
4	Format komunikatów	3
5	Opis wymienianych komunikatów	4
6	Opis stanów	5
6.1	MCA wysyłający	5
6.1.1	SENDER_IDLE_STATE	5
6.1.2	SENDER_AWAITING_QUEUE_REQUEST_STATE	6
6.1.3	SENDER_AWAITING_QUEUE_STATE	6
6.1.4	SENDER_AWAITING_MESSAGE_STATE	7
6.2	MCA odbierający	7
6.2.1	RECEIVER_IDLE_STATE	7
7	Numery	8

1 Streszczenie

Niniejszy dokument jest opisem protokołu obsługi zdalnych kolejek RQP. Protokół ten rozszerza możliwość korzystania ze zwykłych uniksowych kolejek komunikatów, umożliwiając zapisywanie danych przez sieć do kolejek na systemach zdalnych.

Struktura dokumentu:

- Opis celów protokołu
- Opis celów założeń
- Opis formatu komunikatów wymienianych przez protokół
- Opis wymienianych komunikatów
- Opis stanów protokołu
- Zestawienie używanych stałych

2 Cele

Celem protokołu jest udostępnienie przez sieć uniksowych kolejek komunikatów na systemach zdalnych. Protokół ma umożliwiać przesyłanie danych do wielu kolejek znajdujących się na różnych komputerach w sieci.

3 Założenia

Protokół działa w warstwach 5, 6 i 7 modelu OSI. Protokół korzysta z protokołu IP do dostępu do sieci. Protokół korzysta z protokołu transportowego TCP. W komunikacji będzie wykorzystywany port o numerze RQP_PORT.

4 Format komunikatów

Oktety są przesyłane przez sieć w porządku sieciowym. Podstawowy typ komunikatów zawiera tylko liczbę reprezentującą dany komunikat:

```
MSG_T {
    uint4 msg_id;
}
```

Typ dotyczący kolejek posiada dodatkową liczbę identyfikującą kolejkę:

```
MSG_QUEUE_T {
    uint4 msg_id;
    uint32 queue_id;
}
```

Typ służący do przesyłania wiadomości posiada dodatkowe pola do identyfikacji i przechowywania wiadomości:

```
MSG_BODY_T {
    uint4 msg_id;
    uint32 queue_id;
    uint4 type;
    uint32 body_length;
    octet [body_length] body;
}
```

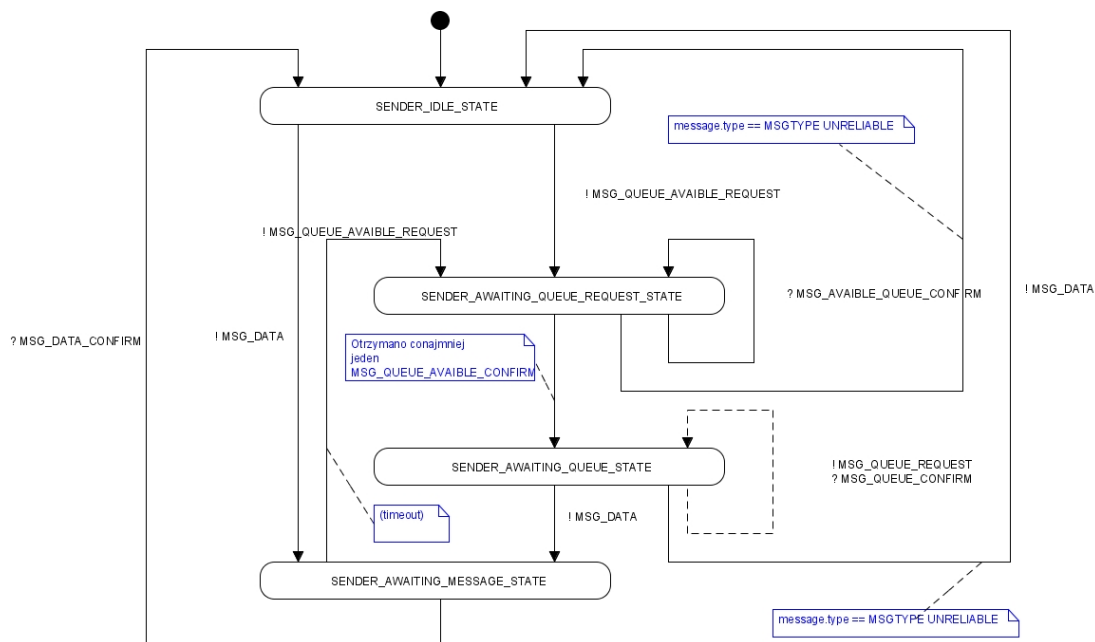
5 Opis wymienianych komunikatów

Protokół wymienia między sobą następujące komunikaty (w nawiasach podane są argumenty, a po dwukropku typ komunikatu). Numer porządkowy to jednocześnie `msg_id` przydzielone komunikatowi.

1. `MSG_DATA : MSG_BODY_T`
Komunikat z danymi do wstawienia do kolejki adresata.
2. `MSG_DATA_CONFIRM : MSG_T`
Potwierdzenie wstawienia wiadomości do kolejki nadawcy komunikatu.
3. `MSG_QUEUE_AVAILABLE_REQUEST : MSG_QUEUE_T`
Monit o dostępność kolejki `queue_id` na potrzeby nadawcy komunikatu.
4. `MSG_QUEUE_AVAILABLE_CONFIRM : MSG_QUEUE_T`
Potwierdzenie dostępności kolejki `queue_id` na potrzeby nadawcy komunikatu.
5. `MSG_QUEUE_REQUEST : MSG_QUEUE_T`
Monit o zarezerwowanie kolejki `queue_id` na potrzeby nadawcy komunikatu.
6. `MSG_QUEUE_CONFIRM : MSG_QUEUE_T`
Potwierdzenie zarezerwowania kolejki `queue_id` dla nadawcy komunikatu.

6 Opis stanów

6.1 MCA wysyłający



6.1.1 SENDER_IDLE.STATE

Stan początkowy dla nadawcy. Przejście do innego stanu jest możliwe w przypadku obecności wiadomości (`message : MSG_BODY_T`) w kolejce transmisyjnej. Akceptowane komunikaty:

1. !MSG_DATA

(Tylko jeśli znany jest odbiorca związany z kolejką `message.queue_id`)

Przekazanie wiadomości do kolejki adresata komunikatu. Jeśli:

- `message.type == MSGTYPE_RELIABLE`, to przejście do stanu `SENDER_AWAITING_CONFIRMATION`.
- `message.type == MSGTYPE_UNRELIABLE`, to usunięcie wiadomości z kolejki transmisyjnej i przejście do stanu `SENDER_IDLE.STATE`.

2. !MSG_QUEUE_AVAILABLE_REQUEST

Wysłanie zapytania o dostępność kolejki (`message.queue_id`) do wszystkich MCA odbierających. Przejście do stanu `SENDER_AWAITING_QUEUE.STATE`.

6.1.2 SENDER_AWAITING_QUEUE_REQUEST_STATE

Stan, w którym znajduje się protokół po wysłaniu zapytania (MSG_QUEUE_AVAILABLE_REQUEST) o dostępność kolejki. Akceptowane komunikaty:

1. ?MSG_QUEUE_AVAILABLE_CONFIRM

Nadawca komunikatu potwierdza dostępność kolejki. Nadawca komunikatu jest dodawany do listy potencjalnych odbiorców danej kolejki (message.queue_id).

2. (timeout)

Jeśli lista potencjalnych odbiorców danej kolejki (message.queue_id) nie jest pusta, to przejście do stanu SENDER_AWAITING_QUEUE_STATE. w przypadku gdy lista jest pusta, to gdy:

- message.type == MSGTYPE_RELIABLE przerwanie działania i zgłoszenie błędu.
- message.type == MSGTYPE_UNRELIABLE wiadomość jest dodawana do kolejki DEAD_LETTER_Q i przejście do stanu SENDER_IDLE.

6.1.3 SENDER_AWAITING_QUEUE_STATE

Stan osiągany po otrzymaniu co najmniej jednego komunikatu MSG_QUEUE_AVAILABLE_CONFIRM, czyli gdy lista udostępnionych wolnych kolejek nie jest pusta. Akceptowane komunikaty:

1. !MSG_QUEUE_REQUEST

Komunikat wysyłany do pierwszego odbiorcy z listy udostępnionych wolnych kolejek. Oczekiwanie na komunikat MSG_QUEUE_CONFIRM. w przypadku braku komunikatu w określonym czasie usunięcie odbiorcy z listy i powtórzenie operacji wysłania komunikatu do kolejnego. Jeżeli nastąpi opróżnienie listy przed otrzymaniem MSG_QUEUE_CONFIRM to, gdy:

- message.type == MSGTYPE_RELIABLE przerwanie działania i zgłoszenie błędu.
- message.type == MSGTYPE_UNRELIABLE wiadomość jest dodawana do kolejki DEAD_LETTER_Q i przejście do stanu SENDER_IDLE.

W przeciwnym przypadku jeżeli otrzyma MSG_QUEUE_CONFIRM przed opróżnieniem listy to, gdy:

- message.type == MSGTYPE_RELIABLE, to wysłanie MSG_DATA i przejście do stanu SENDER_AWAITING_MESSAGE_STATE
- message.type == MSGTYPE_UNRELIABLE, to przejście do stanu SENDER_IDLE_STATE

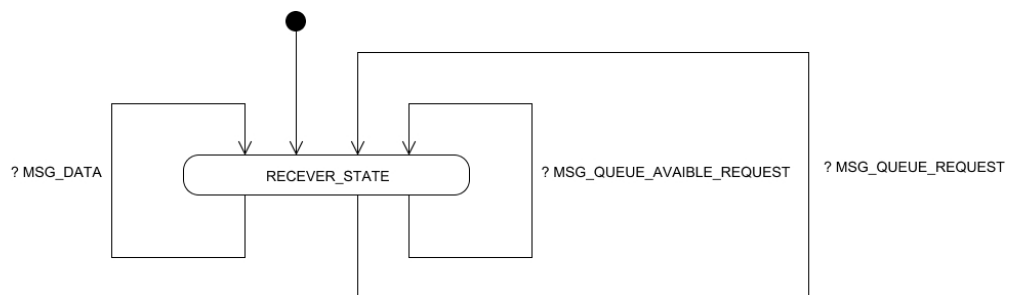
6.1.4 SENDER_AWAITING_MESSAGE_STATE

Stan, w którym znajduje się protokół po wysłaniu wiadomości typu `MSGTYPE_RELIABLE` oczekując na potwierdzenie zapisania wiadomości. Akceptowane komunikaty:

1. `?MSG_DATA_CONFIRM`
Potwierdzenie zapisania wiadomości w zdalnej kolejce. Usunięcie wiadomości z kolejki transmisyjnej i przejście do stanu `SENDER_IDLE`.
2. `(timeout)`
W przypadku, gdy przez określony czas nie dojdzie potwierdzenie zapisania wiadomości wysłanie `MSG_QUEUE_AVAILABLE_REQUEST` do wszystkich MCA odbierających i przejście do stanu `SENDER_AWAITING_QUEUE_REQUEST_STATE`.

6.2 MCA odbierający

6.2.1 RECEIVER_IDLE_STATE



Stan początkowy dla odbiorcy. Akceptowane komunikaty:

1. `?MSG_DATA`
Komunikat z wiadomością do wstawienia do kolejki. W przypadku pomyślnego wstawienia wiadomości do kolejki `queue_id`:
 - `MSG_DATA.type == MSGTYPE_RELIABLE` wysłanie komunikatu `MSG_DATA_CONFIRM` i przejście do stanu `RECEIVER_IDLE_STATE`.
 - `MSG_DATA.type == MSGTYPE_UNRELIABLE` przejście do stanu `RECEIVER_IDLE_STATE`.

W przypadku niepowodzenia operacji dodania wiadomości do kolejki przejście do stanu `RECEIVER_IDLE_STATE`.
2. `?MSG_QUEUE_AVAILABLE_REQUEST`
Pytanie o dostępność kolejki. Jeżeli kolejka jest dostępna to wysłanie komunikatu `MSG_QUEUE_AVAILABLE_CONFIRM`. Przejście do stanu `RECEIVER_IDLE_STATE`.

3. ?MSG_QUEUE_REQUEST

Pytanie o zarezerwowanie kolejki. Jeżeli kolejka jest dostępna to przypisanie kolejki do nadawcy oraz wysłanie komunikatu MSG_QUEUE_CONFIRM. Przejście do stanu RECEIVER_IDLE_STATE.

7 Numery

```
RQP_PORT = 11311
MSGTYPE_RELIABLE = 0
MSGTYPE_UNRELIABLE = 1
MSG_DATA.msg_id = 1
MSG_DATA_CONFIRM.msg_id = 2
MSG_QUEUE_AVAILABLE_REQUEST.msg_id = 3
MSG_QUEUE_AVAILABLE_CONFIRM.msg_id = 4
MSG_QUEUE_REQUEST.msg_id = 5
MSG_QUEUE_CONFIRM.msg_id = 6
```