Specyfikacja protokołu zdalnych kolejek RQP

Krzysztof Choromański
kc219408@students.mimuw.edu.pl
19 kwietnia, 2008

Spis treści:

1. Cele

Protokół zdalnych kolejek RQP ma za zadanie rozszerzyć możliwość korzystania ze zwykłych uniksowych kolejek komunikatów, poprzez wysyłanie ich przez sieć do kolejek zdalnych. Istotną cechą protokołu jest to, że istnieje wyróżniona grupa komunikatów, które w dalszej części dokumentu nazywamy „pewnymi”. Nie będą one usuwane z kolejki transmisyjnej dopóki

agent wysyłający nie dostanie potwierdzenia ich odebrania. Protokół umożliwia także przechowywanie pewnych (np: uszkodzonych) komunikatów w specjalnej strukturze DEAD.LETTER.Q wraz z opisem przyczyny znalezienia się ich w specjalnej kolejce.

2. Założenia

Protokół zdalnych kolejek RQP działa w warstwie aplikacji. Jako warstwy transportu wykorzystywane są protokoły:
- UDP/IP (do rozgłoszenia komunikatu informującego o chęci przesłania danych do kolejki zdalnej)
-TCP/IP

Dany odbiorca komunikatu jest jednoznacznie identyfikowany przez port, na którym będzie odbierał komunikaty oraz swój adres IP.
Komunikacja pomiędzy maszynami przebiega bez udziału centralnego serwera. Maszyny uczestniczące w komunikacji wysyłają komunikaty bezpośrednio do siebie.
Agent wysyłający korzysta z możliwości rozgłoszenia chęci wstawienia komunikatu z danymi do kolejki o określonym kluczu, którą umożliwia UDP/IP. Z tej warstwy transportowej korzystają także agenci odbierający powyższe zapytania.
Dalsza komunikacja przebiega już bez udziału UDP/IP, za to przy wsparciu TCP/IP.
Do rozgłoszenia komunikatu, służącego do odnalezienia maszyny zawierającej kolejkę o odpowiednim kluczu zastosowano warstwę UDP/IP, ze względu na to, że protokół TCP/IP nie umożliwia rozgłaszania. W dalszej fazie komunikacji wybrano jednak warstwę TCP/IP z kilku powodów. Warstwa ta zapewnia niezawodność dostarczenia danych z możliwością reemisji segmentu w przypadku nieotrzymania potwierdzenia. Zapewnia też zdecydowanie lepszą efektywność (m.in. algorytm opóźnionych skumulowanych potwierdzeń).
Protokół zdalnych kolejek RQP korzysta z protokołu usługowego DNS w celu przekształcenia nazw maszyn

(które pojawią się m.in. w specjalnym pliku .rqprc) na odpowiadające im adresy IP.

W dalszej części dokumentu będziemy odwoływać się do następujących nazw portów, na których będą odbierane poszczególne komunikaty:

-request_port

port przyjmujący komunikaty-zapytania o kolejkę z określonym kluczem

-findreceiver_port

port przyjmujący komunikaty potwierdzające gotowość do odebrania danych ze zdalnej kolejki

-accept_port

port, na którym potencjalny odbiorca danych ze zdalnej maszyny czeka na potwierdzenie od nadawcy

-endtransmission_port

port, na którym agent wysyłający czeka na potwierdzenie odebrania danych przez agenta odbierającego (jeśli przesłany był komunikat pewny)

-data_port

port, na który będzie wysyłany komunikat z danymi

Znaczenie i dokładny opis wymienionych wyżej komunikatów zostanie wyczerpująco przedstawiony w dalszej części dokumentu.

3. Format komunikatów

Uwaga1:

W dalszej części dokumentu agentem wysyłającym będziemy nazywać agenta, który ma za zadanie wysłać komunikat z danymi do kolejki zdalnej. Agent odbierający będzie agentem maszyny, która ma wstawić komunikat z danymi do swojej lokalnej kolejki uniksowej.

Uwaga2:

W poniższych strukturach wprowadzono pole **address**, które zawsze oznacza adres IP agenta, który wysyła komunikat o formacie danej struktury. Takie określenie formatu struktur pozwala na wykorzystanie ich także w wersji protokołu, które nie korzystają z TCP/IP jako warstwy transportu. W przedstawionej tu specyfikacji protokołu zdalnych kolejek RQP wszędzie tam gdzie wykorzystywana jest warstwa transportu TCP/IP pole **address** można pominąć, ponieważ ta warstwa sama dostarcza informacji o adresie IP nadawcy wiadomości.

Przyjęto, że porządek oktetów w liczbach jest sieciowy. Liczby całkowite ze znakiem oraz liczby zmiennoprzecinkowe nie występują.

Przejdziemy teraz do zdefiniowania poszczególnych struktur, służących do budowania określonych komunikatów.

3.1 Komunikaty wymieniane w procesie zestawiania

Komunikaty tego typu korzystają z następujących struktur danych:

3.1.1 Struktura question_msg

```
struct question_msg
{
    uint32 address;
    uint 32 key;
}
```

Znaczenie poszczególnych pól:

adress – adres IP maszyny, która wysyła komunikat

key – klucz kolejki docelowej, do której ma trafić komunikat z danymi z lokalnej kolejki nadawcy

3.1.2 Struktura accept_sender_msg

```
struct accept_sender_msg
{
    uint32 address;
    uint16 accept_port;
    uint32 key;
    uint16 data_port;
}
```

Znaczenie poszczególnych pól:

address – adres IP maszyny, która wysyła komunikat
accept_port – port, na którym maszyna zgłaszająca chęć odebrania komunikatu z danymi od nadawcy będzie oczekiwać na potwierdzenie chęci wysłania komunikatu z danymi

key – klucz kolejki oferowanej przez maszynę odbierającą, do której ma trafić komunikat z danymi od agenta wysyłającego

data_port – port, na którym agent odbierający będzie oczekiwał na komunikat z danymi

3.1.3 Struktura accept_receiver_msg

```
struct accept_receiver_msg
{
    uint32 address;
    uint16 endtransmission_port;
    uint32 id;
    uint8 msg_type;
}
```

Znaczenie poszczególnych pól struktury:

address - adres IP maszyny, która wysyła komunikat

endtransmission_port – port, na którym nadawca będzie czekał na potwierdzenie odbioru całego komunikatu z danymi przez agenta odbierającego (to pole jest istotne tylko dla komunikatów pewnych)

id – identyfikator komunikatu z danymi, który będzie wysyłany przez agenta wysyłającego

Uwaga:

Zakładamy, że identyfikator będzie jednoznacznie identyfikował przesyłany komunikat z danymi. Będzie on tworzony na podstawie adresu IP maszyny nadawcy oraz jednoznacznego identyfikatora komunikatu w obrębie maszyny nadawcy. (Identyfikatory w obrębie maszyny mogą być tworzone

np.: przy pomocy odrębne kolejki z możliwymi identyfikatorami, która zapewnia atomowość operacji wstawienia identyfikatora oraz wyjęcia identyfikatora.)

msg_type – typ komunikatu z danymi, który agent wysyłający chce przesłać na dalszym etapie komunikacji:

LOW_QUALITY – komunikat niepewny
HIGH_QUALITY – komunikat pewny

3.2 Komunikaty z danymi do kolejek na maszynach zdalnych

Te komunikaty korzystają z następującej struktury:

3.2.1 Struktura data_msg

```
struct data_msg
{
  uint32 address;
  uint32 key;
  uint32 message_size;
  octet [message_size] data;
}
```

Znaczenie poszczególnych pól struktury:

address – adres IP maszyny, która wysyła komunikat

key – klucz kolejki, do której ma trafić
komunikat od agenta wysyłającego
message_size – rozmiar tablicy **data**
data – tablica bajtowa zawierająca same dane, które
mają być przesłane z lokalnej kolejki
do zdalnej kolejki uniksowej

3.3 Komunikaty potwierdzenia odbioru

Komunikaty należące do tej grupy korzystają z następującej struktury danych:

3.3.1 Struktura endtransmission_msg

```
struct endtransmission_msg
{
  uint32 address;
  uint32 id;
  uint8 result;
}
```

Znaczenie poszczególnych pól struktury:

address – adres IP maszyny, która wysyła komunikat
id – identyfikator komunikatu z danymi, który był
odbierany przez agenta odbierającego

result – pole wskazujące czy odebranie komunikatu z
danymi zostało zakończone sukcesem:

FAILURE – odebranie zakończone porażką
SUCCESS – odebranie zakończone sukcesem

Uwaga:

Przedstawione poniżej struktury są wykorzystywane tylko w kolejkach specjalnych (kolejce transmisyjnej agenta wysyłającego i kolejce DEAD.LETTER.Q). Nie są one wykorzystywane

bezpośrednio w komunikacji pomiędzy agentem wysyłającym i odbierającym. Ponieważ łączą się one ściśle z implementowanym protokołem, z celu podania pełnego opisu protokołu, zostały one pokrótce opisane.
Uwaga ta odnosi się także do analogicznej sekcji w podpunkcie: *Opis komunikatów*.

3.4 Komunikaty kolejki transmisyjnej agenta wysyłającego

Komunikaty w kolejce transmisyjnej agenta wysyłającego wykorzystują następującą strukturę:

3.4.1 Struktura t_msg

```
 struct t_msg
{
  uint8 msg_type;
  uint32 id;
  uint32 key;
  uint32 message_size;
  octet[message_size] data;
}
```

Znaczenie poszczególnych pól struktury:

msg_type – typ komunikatu z danymi, który nadawca chce przesłać na dalszym etapie komunikacji:

LOW_QUALITY – komunikat niepewny
HIGH_QUALITY– komunikat pewny

id – identyfikator komunikatu z danymi, który będzie wysyłany przez agenta wysyłającego

key – klucz kolejki docelowej, do której ma trafić
komunikat z danymi z lokalnej kolejki nadawcy

message_size – rozmiar tablicy **data**

data – tablica bajtowa zawierająca same dane, które
mają być przesłane z lokalnej kolejki
do zdalnej kolejki uniksowej

3.5 Komunikaty kolejki DEAD.LETTER.Q

Komunikat z pewnych przyczyn (uszkodzenie lub brak kolejki docelowej, do której miał zostać wysłany) może trafić do specjalnej kolejki DEAD.LETTER.Q. Będzie on tam przechowywany wraz z opisem przyczyny trafienia do specjalnej kolejki. Komunikaty są tam przechowywane w postaci następującej struktury:

3.5.1 Struktura garbage_msg

```
struct garbage_msg
{
  uint32 message_size;
  octet [message_size] data;
  uint32 reason;
}
```

Znaczenie poszczególnych pól struktury:

message_size – rozmiar tablicy **data**

data – tablica bajtowa zawierająca same dane, które
miały być przesłane z lokalnej kolejki
do zdalnej kolejki uniksowej

reason – przyczyna, z powodu której komunikat
trafił do kolejki specjalnej:

INJURED – komunikat uszkodzony
PARTIALLY_RECEIVED – komunikat, który nie został w całości odebrany
ZOMBIE – komunikat, dla którego w trakcie odbierania przestała istnieć kolejka docelowa

4. Opis komunikatów

Komunikaty protokołu zdalnych kolejek RQP są podzielone na następujące grupy:

INTRODUCTORY_MSG := QUESTION_MSG |
ACCEPT_SENDER_MSG |
ACCEPT_RECEIVER_MSG

UNIX_QUEUE_MSG := DATA_MSG

END_SESSION_MSG := END_TRANSMISSION_MSG

QUEUED_MSG := T_MSG | GARBAGE_MSG

Poniżej zostanie podany opis poszczególnych grup komunikatów. Ponieważ formaty poszczególnych komunikatów są takie same jak formaty odpowiadającym im struktur, zaś pola tych struktur zostały opisane wcześniej, nie będą one dokładnie omawiane w tym punkcie. W przypadku niektórych komunikatów znaczenie pewnych pól zostanie przypomniane w kontekście celu komunikatu.

4.1 INTRODUCTORY_MSG

Komunikaty tej grupy są wymieniane między agentami kanałowymi w celu znalezienia kolejki o odpowiednim kluczu, której następnie przesłane zostaną dane.

4.1.1 QUESTION_MSG

address	key
uint32	uint32

Ten komunikat jest rozgłaszany przez agenta wysyłającego do agentów odbierających na wszystkich maszynach (odczytanych z pliku .rqprc) za pomocą warstwy UDP/IP. Komunikat informuje wszystkie maszyny, do których został wysłany o chęci przekazania przez agenta wysyłającego komunikatu z danymi do kolejki o kluczu **key**. Komunikat ma format struktury question_msg i jest wysyłany do portu *request_port*, którego numer jest ustalony dla wszystkich maszyn korzystających z protokołu. Oczekiwanie na odpowiedź nastąpi na porcie *findreceiver_port* o ustalonym dla wszystkich maszyn numerze. Jeśli do czasu **T_WAIT1** agent wysyłający nie otrzyma potwierdzenia od żadnego agenta odbierającego, to agent wysyłający pozostawia komunikat w kolejce (na ostatniej pozycji) i przerywa połączenie TCP na porcie *findreceiver_port*. Spróbuje wysłać ten komunikat ponownie po rozpatrzeniu innych komunikatów, które znajdują się w kolejce transmisyjnej.

Wszystkie pozostały komunikaty będą korzystały z warstwy transportu TCP/IP dlatego nie będzie to za każdym razem zaznaczane.

4.1.2 ACCEPT_SENDER_MSG

address	accept_port	key	data_port
uint32	uint16	uint32	uint16

Komunikat ma format struktury accept_sender_msg.
Jest potwierdzeniem otrzymania przez agenta odbierającego komunikatu QUESTION_MSG od agenta wysyłającego oraz oznacza gotowość do przyjęcia komunikatu z danymi, ponieważ istnieje kolejka docelowa o kluczu **key** na maszynie odbiorcy.
Accept_port jest numerem portu, na którym agent odbierający będzie oczekiwał na potwierdzenie chęci przesłania komunikatu z danymi od agenta wysyłającego.
Jeśli do czasu **T_WAIT2** agent odbierający nie otrzyma od agenta wysyłającego potwierdzenia chęci wysłania komunikatu z danymi to nasłuchiwanie na tym porcie zostaje zakończone.
Data_port to port, na którym agent wysyłający będzie oczekiwał na komunikat z danymi po odebraniu komunikatu na porcie *accept_port*.
Uwaga:

Przed wysłaniem tego komunikatu agent odbierający musi zarezerwować dla siebie kolejkę, do której ma zostać wstawiony komunikat z danymi. Jeśli jednak nie ma kolejki o danym kluczu, która nie jest zarezerwowana, agent odbierający nie wysyła agentowi wysyłającemu komunikatu ACCEPT_SENDER_MSG.

4.1.3 ACCEPT_RECEIVER_MSG

address	endtransmission_port	id	msg_type
uint32	uint16	uint32	uint8

Komunikat ma format struktury accept_receiver_message.

Jest wysyłany przez agenta wysyłającego do agenta odbierającego i oznacza potwierdzenie, ze dana maszyna została wybrana na odbiorcę komunikatu z danymi.
Pole **id** jest identyfikatorem komunikatu z danymi, który będzie przesyłany. Pole **endtransmission_port** jest portem, na którym agent wysyłający będzie oczekiwać na potwierdzenie odebrania całego komunikatu z danymi przez agenta odbierającego. Ponieważ tylko komunikaty pewne wymagają potwierdzenia, to pole ma istotną wartość różną od ZERO_ID tylko w przypadku gdy wysyłany komunikat jest pewny. Gdy wysyłany jest komunikat niepewny to pole ma wartość ZERO_ID.
Pole **msg_type** informuje o tym, czy komunikat z danymi, który będzie przesłany jest pewny czy niepewny.

4.2 UNIX_QUEUE_MSG

Komunikaty tej grupy zawierają dane, które mają zostać przesłane do kolejek zdalnych. W tej grupie mamy tylko komunikaty DATA_MSG:

4.2.1 DATA_MSG

address	key	message_size	data
uint32	uint32	uint32	octet[message_size]

Ten komunikat ma format struktury data_msg.
Jest wysyłany przez agenta wysyłającego do agenta odbierającego i zawiera dane, które powinny trafić bezpośrednio do kolejki zdalnej (pole **data**).
Jeśli komunikat ten nie zostanie odebrany w całości na specjalnym porcie *data_port* w czasie **T_WAIT3** od momentu rozpoczęcia nasłuchiwania na tym porcie, to połączenie między agentami kanałowymi zostanie przerwane przez agenta odbierającego.
Jeśli odebrany został pewny komunikat z danymi o identyfikatorze, który już posiadał odebrany kiedyś

komunikat z danymi, to nowoodebrany komunikat z danymi nie jest wstawiany do kolejki docelowej. Agent odbierający poinformuje później agenta wysyłającego o tym, że odebranie zakończyło się niepowodzeniem (ten mechanizm zapobiega sytuacji wstawienia wielokrotnie tego samego komunikatu do tej samej kolejki).

4.3 END_SESSION_MSG

Komunikaty w tej grupie mają na celu zakończenie połączenia między agentami kanałowymi w sytuacji gdy celem połączenia było przekazanie komunikatu pewnego. Do tej grupy należą komunikaty tylko jednego rodzaju:

4.3.1 END_TRANSMISSION_MSG

address	id	result
uint32	uint32	uint8

Ten komunikat ma format struktury end_transmission_msg. Jest wysyłany przez agenta odbierającego, który chce potwierdzić odebranie całego komunikatu pewnego, bądź stwierdzić, że odbieranie zakończyło się niepowodzeniem (np.: z powodu usunięcia kolejki docelowej, w której miał się znaleźć komunikat pewny). Komunikat zostanie odebrany przez agenta wysyłającego na specjalnym porcie *endtransmission_port*. Odebranie tego komunikatu kończy sesję połączenia między agentami kanałowymi, która miała na celu przekazanie do kolejki zdalnej komunikatu pewnego. Jeśli w czasie **T_WAIT4** od momentu rozpoczęcia nasłuchiwania na porcie *endtransmission_port* nie nastąpi odebranie komunikatu (z odpowiednim polem **id**), komunikat pewny pozostanie w kolejce transmisyjnej (zostanie wstawiony na jej koniec), a agent wysyłający zajmie się wysyłaniem następnego komunikatu z kolejki transmisyjnej, przerywając nasłuchiwanie na porcie *endtransmission_port*.

4.4 QUEUED_MSG

Ta grupa komunikatów składa się z komunikatów, które są przechowywane w dodatkowych kolejkach agentów kanałowych służących do poprawnego działania komunikacji opartej na protokole zdalnych kolejek RQP.
Są to:
- kolejka transmisyjna agenta wysyłającego (T)
- kolejka DEAD.LETTER.Q

4.4.1 T_MSG

msg_type	id	key	message_size	data
uint8	uint32	uint32	uint32	octet[message_size]

Ten komunikat ma format struktury t_msg. Przechowuje informacje o żądaniach klienta-nadawcy dotyczących kolejek na maszynach zdalnych. Pole **id** to identyfikator komunikatu z danymi. Pole to jest równe ZERO_ID jeśli ten komunikat opisuje chęć wysłania wiadomości niepewnej (wówczas wartość tego pola nie jest istotna dla dalszego przebiegu komunikacji) oraz przechowuje wartość rzeczywistego identyfikatora dla wiadomości pewnej (wymagającej potwierdzenia). Pole **data** zawiera dane, które mają być przesłane do kolejki zdalnej. Pole **key** oznacza klucz kolejki docelowej, do której ma być wysłana wiadomość.
Nadawca wysyłając wiadomość za pomocą funkcji sendMessage określa rodzaj wiadomości (pewna/niepewna).
Tę informację zgodnie z tym, co było powiedziane przy omawianiu struktury t_msg, przechowuje pole **msg_type**.

4.4.2 GARBAGE_MSG

message_size	data	reason
uint32	octet[message_size]	uint32

Ten komunikat ma format struktury garbage_msg.
Opisuje on komunikaty, które z pewnych powodów nie mogły zostać wstawione do kolejek docelowych lub były uszkodzone i dlatego nie mogły zostać wysłane. Opis poszczególnych pól jest identyczny jak dla struktury garbage_msg.

5. Opis stanów

W tym punkcie zostaną opisane wszystkie stany, w jakich może się znajdować agent kanałowy (wysyłający lub odbierający). Zostaną też przedstawione stany, w jakich może znajdować się komunikat (rozumiany jako komunikat z danymi) wysyłany do kolejki zdalnej

5.1 Agent wysyłający

Wyróżniono następujące stany:

(a) WAIT_FOR_RECEIVER_STATE
(b) WAIT_FOR_CONFIRMATION_STATE

Opis poszczególnych stanów:

(a) WAIT_FOR_RECEIVER_STATE

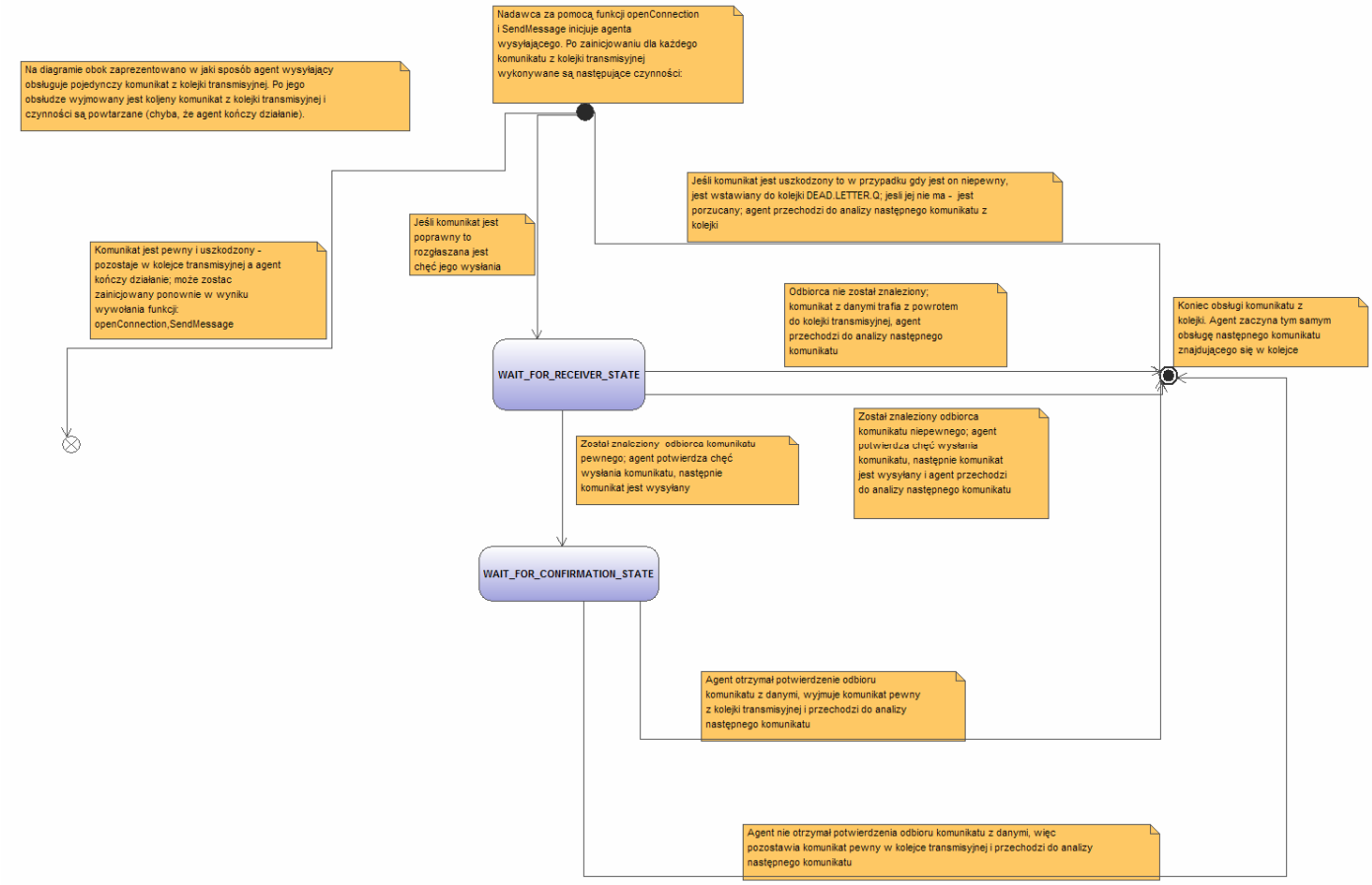
W tym stanie agent wysyłający, po wcześniejszym wysłaniu komunikatu z zapytaniem o maszynę z kolejką o odpowiednim kluczu, czeka na zgłoszenie się potencjalnego odbiorcy wiadomości. W tym stanie agent przebywa nie dłużej niż czas T_WAIT1. Jeśli do czasu T_WAIT1 agent nie otrzyma zgłoszenia od agenta odbierającego, to jeśli próbuje on wysłać komunikat niepewny, jest on z powrotem wstawiany do kolejki transmisyjnej agenta wysyłającego (wcześniej ten komunikat był przez niego wyjęty z kolejki).
Jeśli agent próbuje wysłać komunikat pewny, to nie musi on być wstawiany do kolejki transmisyjnej po niezgłoszeniu się odbiorcy, ponieważ nie był on wcześniej wyjęty przez agenta wysyłającego.

Następnie, ponieważ nie został znaleziony odbiorca komunikatu, agent wysyłający analizuje następny komunikat z kolejki.
Jeśli do czasu T_WAIT1 zgłosi się agent odbierający, agent wysyłający potwierdza jego wybór, wysyłając specjalny komunikat **ACCEPT_RECEIVER_MSG**. Potem następuje wysłanie komunikatu z danymi do kolejki zdalnej.
Jeśli wysyłany komunikat był niepewny, to po jego wysłaniu agent wysyłający rozpoczyna analizę następnego komunikatu z kolejki. W przeciwnym przypadku przechodzi do stanu WAIT_FOR_CONFIRMATION_STATE.

(b) WAIT_FOR_CONFIRMATION_STATE

W tym stanie agent wysyłający czeka na potwierdzenie od agenta odbierającego, że ten odebrał komunikat pewny. Agent wysyłający przebywa w tym stanie nie dłużej niż czas T_WAIT4. Jeśli do tego czasu agent wysyłający nie otrzyma potwierdzenia to komunikat pewny nie jest wyjmowany z kolejki transmisyjnej agenta wysyłającego, a ten rozpoczyna analizę następnego komunikatu z kolejki
(komunikat pewny jest wstawiany na koniec kolejki). Jeśli do czasu T_WAIT4 agent otrzyma potwierdzenie odbioru komunikatu, wyjmuje on z kolejki transmisyjnej komunikat pewny, którego potwierdzenie odbioru otrzymał, a następnie rozpoczyna analizę następnego komunikatu z kolejki transmisyjnej.

Poniżej w formie graficznej przedstawione zostały stany agenta wysyłającego i możliwe przejścia między nimi:



5.2 Agent odbierający

Agent odbierający składa się z wielu wątków wykonujących opisane wcześniej czynności agenta odbierającego. Poniżej zostały przedstawione możliwe stany tych wątków.

Wyróżniono następujące stany:

(a) WAIT_FOR_SENDER_STATE
(b) WAIT_FOR_APPROVAL_STATE
(c) WAIT_FOR_MESSAGE_STATE

Opis poszczególnych stanów:

(a) WAIT_FOR_SENDER_STATE

W tym stanie wyróżniony wątek agenta odbierającego cały czas czeka na prośby od agentów wysyłających o przyjęcie komunikatu z danymi do lokalnej kolejki o odpowiednim kluczu.
Po odebraniu komunikatu z prośbą, w wątku sprawdzane jest czy maszyna agenta odbierającego udostępnia kolejkę o żądanym kluczu. Jeśli kolejka o żądanym kluczu jest aktualnie udostępniona, to wysyłane jest potwierdzenie możliwości odbioru komunikatu z danymi. Następnie tworzony jest wątek agenta odbierającego (nazywany dalej wątkiem odbierającym), którego zadaniem jest przejęcie obsługi nadawcy. (O tym wątku jest mowa w opisie kolejnych stanów). Wątek znajduję się początkowo w stanie WAIT_FOR_APPROVAL_STATE.

(b) WAIT_FOR_APPROVAL_STATE

W tym stanie wątek odbierający czeka na otrzymanie potwierdzenia, że agent wysyłający wybrał agenta odbierającego na odbiorcę komunikatu z danymi. Jeśli w ciągu czasu T_WAIT2 potwierdzenie nie zostanie uzyskane, wątek kończy działanie. W przeciwnym przypadku przechodzi do stanu WAIT_FOR_MESSAGE_STATE.

(c) WAIT_FOR_MESSAGE_STATE

W tym stanie wątek odbierający czeka na/odbiera dane do lokalnej kolejki maszyny agenta odbierającego. Jeśli przez czas T_WAIT3 nie zostanie otrzymany komunikat z danymi, wątek odbierający kończy połączenie na porcie data_port z agentem wysyłającym i kończy działanie. W przeciwnym wypadku możliwe są następujące scenariusze:

- komunikat z danymi został co prawda odebrany, ale jest uszkodzony (np.: nie został odebrany w całości)

Wówczas jeśli po stronie agenta odbierającego znajduje się kolejka DEAD.LETTER.Q, komunikat z danymi, wraz z informacją o przyczynach umieszczenia go w kolejce DEAD.LETTER.Q, jest tam wstawiany.
Jeśli maszyna, na której został uruchomiony agent

odbierający nie udostępnia kolejki specjalnej DEAD.LETTER.Q to komunikat z danymi jest ignorowany. W obu przypadkach wątek odbierający wysyła agentowi wysyłającemu komunikat END_TRANSMISSION_MSG z polem **result** ustawionym na FAILURE i kończy działanie

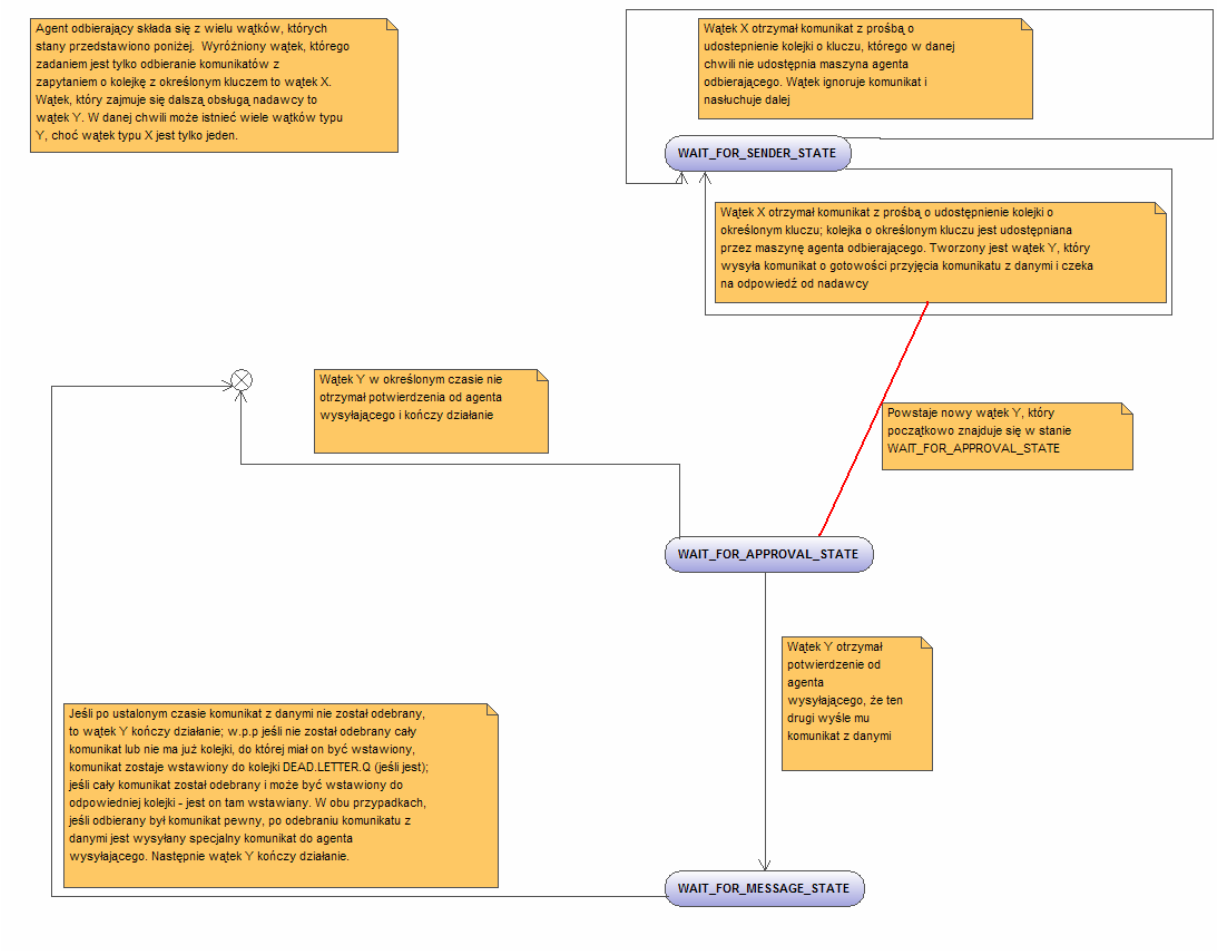
- **komunikat z danymi został co prawda odebrany, ale nie istnieje już kolejka, do której miał on zostać wstawiony**

  Wątek odbierający postępuje wówczas analogicznie jak w poprzedniej sytuacji.

- **komunikat z danymi został odebrany w całości i istnieje kolejka, do której miał on zostać wstawiony**

  W tej sytuacji dane z komunikatu zostają wstawione do właściwej kolejki lokalnej. Do kolejki DEAD.LETTER.Q żaden komunikat nie jest wstawiany. Wysyłany jest natomiast komunikat END_TRANSMISSION_MSG do agenta wysyłającego z polem **result** ustawionym na SUCCESS. Następnie watek odbierający kończy działanie.

Poniżej w formie graficznej przedstawione zostały stany wątków agenta odbierającego i możliwe przejścia między nimi:



5.3 Komunikat do kolejki zdalnej

Zanim dane z kolejki lokalnej trafią do kolejki zdalnej, dane te muszą znaleźć się w kilku stanach pośrednich. Opiszemy je dokładnie tutaj.

Wyróżniono następujące stany:

(a) BEFORE_TRANSMISSION_STATE
(b) DURING_TRANSMISSION_STATE
(c) FINAL_STATE

Opis poszczególnych stanów:

(a) BEFORE_TRANSMISSION_STATE

Komunikat przebywa w tym stanie gdy znajduje się w kolejce transmisyjnej agenta wysyłającego, ale nie jest w trakcie wysyłania przez agenta wysyłającego. Po wybraniu
przez agenta wysyłającego komunikat przechodzi do stanu DURING_TRANSMISSION_STATE, jeśli nie jest uszkodzony lub do stanu FINAL_STATE w.p.p

(b) DURING_TRANSMISSION_STATE

W tym stanie komunikat jest w fazie wysyłania przez agenta wysyłającego. Jeśli jest to komunikat niepewny to w tym stanie jego kopia po stronie agenta wysyłającego nie znajduje się już w kolejce transmisyjnej, w.p.p cały czas tam jest. Jeśli agent wysyłający otrzymał potwierdzenie odbioru komunikatu, to jego kopia po stronie agenta wysyłającego jest usuwana (jeśli komunikat był pewny to kopia jest usuwana z kolejki transmisyjnej). Komunikat po stronie odbiorcy jest już w stanie FINAL_STATE. Jeśli agent wysyłający nie otrzymał potwierdzenia od agenta odbierającego otrzymania komunikatu (pewnego), jego kopia po stronie nadawcy przechodzi do stanu BEFORE_TRANSMISSION_STATE
(jeśli odbiorca odebrał komunikat, to po jego stronie komunikat jest w stanie FINAL_STATE).
Jeśli komunikat był niepewny, to jego kopia po stronie nadawcy jest usuwana po wysłaniu
komunikatu (niezależnie od tego czy komunikat został odebrany i wstawiony do kolejki docelowej). Jeśli odbiorca odebrał komunikat, to po jego stronie komunikat znajdzie się w stanie FINAL_STATE.

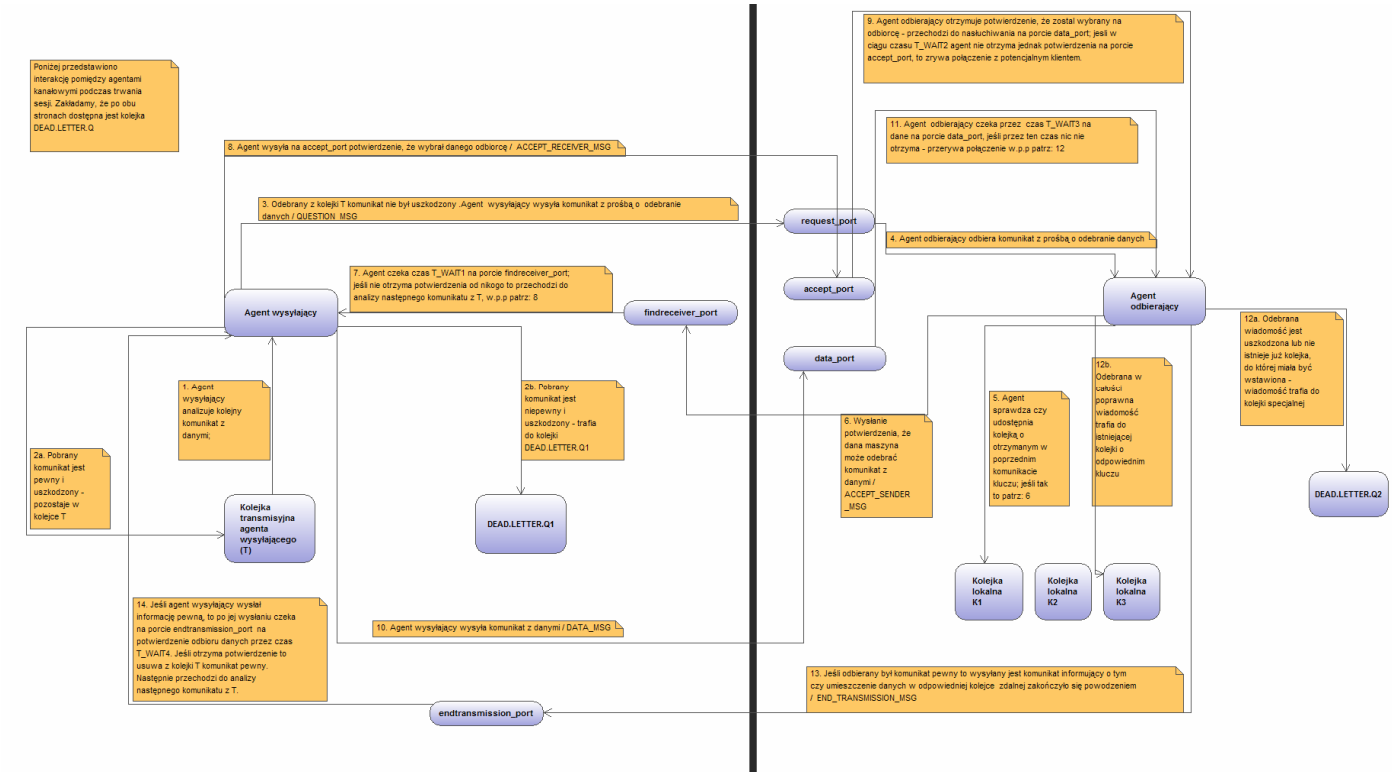
(c) FINAL_STATE

W tym stanie komunikat znajduje się już w kolejce docelowej, do której miał trafić z kolejki lokalnej lub w kolejce DEAD.LETTER.Q jeśli z jakichś powodów komunikat nie mógł zostać wysłany lub jego odbieranie zakończyło się niepowodzeniem.

Uwaga:

Opisując powyższe stany mówiąc o „ komunikacie po stronie nadawcy/odbiorcy” mieliśmy na myśli kopię komunikatu znajdującą się na maszynie nadawcy/odbiorcy. Komunikat z danymi może znajdować się w dwóch kopiach (na każdej maszyny w jednej kopii). Oczywiście stany w jakich komunikat występuje na każdej maszyn mogą być różne.
Czasami, gdy z kontekstu jasno wynikało, o której kopii komunikatu jest mowa (tej po stronie nadawcy lub odbiorcy) używaliśmy krótkiego określenia: „komunikat”.

Poniżej znajduje się rysunek przedstawiający schemat interakcji pomiędzy agentami kanałowymi przy wysyłaniu wiadomości z jednej maszyny na drugą.



6. Numery

Poniżej przedstawiono dokładne wartości pewnych stałych, których użyto w dokumencie:

1. HIGH_QUALITY = 1
2. LOW_QUALITY = 0
3. SUCCESS = 1
4. FAILURE = 0
5. INJURED = 0
6. PARTIALLY_RECEIVED = 1
7. ZOMBIE = 2
8. ZERO_ID = 0