

# Systemy rozproszone - raport z zadań z zajęć (RabbitMQ)

Mateusz Pawłowicz 305391

## 1 Zadanie 1

### 1.1 Zadanie 1a - niezawodność

#### 1.1.1 Który sposób jest bardziej niezawodny?

1. Po potwierdzeniu wiadomości, wiadomość jest traktowana jako poprawnie dostarczona. Zatem w przypadku kiedy potwierdzamy od razu po dostarczeniu, w przypadku resetu w trakcie przetwarzania nie dojdzie do retransmisji, zatem dane nie zostaną poprawnie przetworzone w tym przypadku. W przypadku resetu po przetwarzaniu nie ma raczej problemu - reset nastąpił po wykonaniu zadania.
2. W przypadku potwierdzania po przetworzeniu wiadomości, reset podczas przetwarzania nie jest problemem - od razu po włączeniu na nowo konsumenta otrzymuje on niedostarczoną wiadomość, w związku z czym zaczyna on ją przetwarzać (nie zgubiliśmy wiadomości, która była przetworzona 'w połowie'). Mechanizm ten działa również dla większej ilości niedostarczonych wiadomości. W przypadku resetu po przetwarzaniu nie dochodzi do retransmisji, jednak zadanie zostało wykonane więc nie ma tutaj żadnego problemu z tym.

Zatem potwierdzanie **po** przetworzeniu daje lepszą niezawodność w sensie działania (patrzac na sam fakt dostarczenia wiadomości do konsumenta to następuje to w obydwu warunkach).

#### 1.1.2 Brak wysyłania potwierdzeń

W przypadku braku wysyłania potwierdzeń, konsument poprawnie przetworzy każdą wysłaną do niego wiadomość. Jednak ze względu na brak potwierdzeń, każde ponowne uruchomienie konsumenta sprawia, że na początku otrzymuje on wszystkie niepotwierdzone wcześniej wiadomości i je przetwarza. Dzieje się tak do czasu potwierdzenia wcześniej niedostarczonych wiadomości (czyli do złamania zasady braku potwierdzania).

## 1.2 Zadanie 1b - load-balancing

### 1.2.1 Przed włączeniem usługi QoS

Przed włączeniem usługi QoS, ze względu na politykę round-robin, wiadomości na zmianę były przekazywane do konsumentów (każdy konsument dostawał co drugą). Efektem tego było nierównomierne rozłożenie czasu pracy (jeden konsument dostawał same małe liczby, drugi same duże).

Poniżej zrzut ekranu prezentujący uzyskany wynik:

Z1 CONSUMER	Z1 CONSUMER
Waiting for messages...	Waiting for messages...
Received: 1	Received: 5
Done with message: 1	Done with message: 5
Received: 1	Received: 5
Done with message: 1	Done with message: 5
Received: 1	Received: 5
Done with message: 1	Done with message: 5
Received: 1	Received: 5
Done with message: 1	Done with message: 5
Received: 1	Received: 5
Done with message: 1	Done with message: 5

Rysunek 1: Działanie komunikacji w zadaniu 1b przed włączeniem usługi QoS

### 1.2.2 Po włączeniu usługi QoS

Włączenie usługi pozwalało kolejkom mieć jedną nie potwierdzoną wiadomość (stąd wymóg potwierdzania po przetworzeniu danych), w efekcie czego rozdział wiadomości przestał stać się round-robin.

W naszym przypadku uzyskaliśmy dość równomierny rozkład czasu działania między konsumentami:

Z1 CONSUMER	Z1 CONSUMER
Waiting for messages...	Waiting for messages...
Received: 1	Received: 5
Done with message: 1	Done with message: 5
Received: 1	Received: 1
Done with message: 1	Done with message: 1
Received: 5	Received: 5
Done with message: 5	Done with message: 5
Received: 1	Received: 1
Done with message: 1	Done with message: 1
Received: 5	Received: 5
Done with message: 5	Done with message: 5

Rysunek 2: Działanie komunikacji w zadaniu 1b przed włączeniem usługi QoS

## 2 Zadanie 2 - routing oraz exchange'e

### 2.1 Routing direct

#### 2.1.1 Zaproponowany przykład

1. Klucz konsumenta 1: 'wiolinowy'
2. Klucz konsumenta 2: 'do domu'
3. Komunikacja producenta:
  - (a) Klucz: 'wiolinowy', wiadomość: 'osemka' (odbiera konsument 1, konsument 2 nie)
  - (b) Klucz: 'do domu', wiadomość: 'zgubiony' (odbiera konsument 2, konsument 1 nie)
  - (c) Klucz: 'imbusowy', wiadomość: 'znaleziony' (żaden z konsumentów nie odbiera - klucz się nie zgadza)

#### 2.1.2 Demonstracja działania

Powyższe przewidywania sprawdziły się w praktyce, co widać na poniższych zrzutach ekranów:



```
Z2 CONSUMER
Enter key:
wiolinowy
created queue: amq.gen-YRde01jV26TNFnJeaE04aQ with key: wiolinowy
Waiting for messages...
Received: osemka

Z2 CONSUMER
Enter key:
do domu
created queue: amq.gen-W-wTIdAEfHLNUI0Ad-uaTw with key: do domu
Waiting for messages...
Received: zgubiony

Z2 PRODUCER
Enter key:
wiolinowy
Enter message:
osemka
Sent: osemka
Enter key:
do domu
Enter message:
zgubiony
Sent: zgubiony
Enter key:
imbusowy
Enter message:
znaleziony
Sent: znaleziony
```

Rysunek 3: Wyniki działania zadania 2 dla routingu direct

### 2.2 Routing topic

#### 2.2.1 Zaproponowany przykład

1. Klucz konsumenta 1: '#.do.\*' (dowolny początek, w środku '.do.' i potem jedno słowo po kropce)
2. Klucz konsumenta 2: 'imbusowy.#' (zawiera prefiks 'imbusowy.')

### 3. Komunikacja producenta:

- (a) Klucz: 'imbusowy.do.drzwi', wiadomość: 'nie pasuje' (oba konsumenci odbierają - klucz dopasowuje się do obydwu kolejek)
- (b) Klucz: 'do.skrzynki', wiadomość: 'na strychu' (konsument 1 odbiera, konsument 2 nie)
- (c) Klucz: 'imbusowy.maly', wiadomość: 'malutki' (konsument 2 odbiera, konsument 1 nie)
- (d) Klucz: 'duzy.metalowy', wiadomość: 'sie zgubil' (żaden z konsumentów nie odbiera)

#### 2.2.2 Demonstracja działania

Ponownie, zakładany rezultat został otrzymany, co widać poniżej:



```
Z2 CONSUMER
Enter key:
# do.*
created queue: amq.gen-XXf7WsMpNAK2arsVgAYwmQ with key: #.do.*
Waiting for messages...
Received: nie pasuje
Received: na strychu

Z2 CONSUMER
Enter key:
imbusowy.#
created queue: amq.gen-zyR4oLDAcNXpiavYt0wCYA with key: imbusowy.#
Waiting for messages...
Received: nie pasuje
Received: malutki

Z2 PRODUCER
Enter key:
imbusowy.do.drzwi
Enter message:
nie pasuje
Sent: nie pasuje
Enter key:
do.skrzynki
Enter message:
na strychu
Sent: na strychu
Enter key:
imbusowy.maly
Enter message:
malutki
Sent: malutki
Enter key:
duzy.metalowy
Enter message:
sie zgubil
Sent: sie zgubil
```

Rysunek 4: Wyniki działania zadania 2 dla routingu topic