JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 29–32
Received: 19 September 2022
Accepted: 7 November 2022

Comparative analysis of transfer protocols asynchronous messages on systems queuing

Analiza porównawcza protokołów przesyłania wiadomości asynchronicznych w systemach kolejkowych

Grzegorz Derlatka*, Piotr Kopniak

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
This article presents an analysis of the performance of two protocols supported by queuing systems, i.e. MQTT (MQ Telemetry Transport) and AMQP (Advanced Message Queuing Protocol). This analysis was performed using two message brokers - ActiveMQ and RabbitMQ. The time of sending the message was analyzed, determined on the basis of the time of sending and receiving the message for both protocols in both of the above-mentioned queuing systems. The tests were carried out using proprietary applications written in Java and the Spring application framework.

Keywords: message broker; asynchronous communication; AMQP protocol; MQTT protocol

Streszczenie
W tym artykule została przedstawiona analiza wydajności dwóch protokołów obsługiwanych przez systemy kolejkowe, tj. protokołu MQTT (ang. MQ Telemetry Transport) oraz AMQP (ang. Advanced Message Queuing Protocol). Analiza ta została przeprowadzona z użyciem dwóch brokerów wiadomości - ActiveMQ oraz RabbitMQ. Analizie został poddany czas przesłania wiadomości wyznaczony na podstawie czasu wysłania i odebrania komunikatu dla obu protokołów w obu przytoczonych systemach kolejkowych. Testy zostały przeprowadzone przy pomocy własnych aplikacji napisanych w języku Java oraz szkielecie aplikacji Spring.

Słowa kluczowe: system kolejkowy; broker komunikatów; komunikacja asynchroniczna; protokół AMQP; protokół MQTT

*Corresponding author
Email address: **grzegorz.derlatka@pollub.edu.pl** (G. Derlatka)

1. Wstęp

System kolejkowy to oprogramowanie, które umożliwia aplikacjom, systemom i usługom wzajemną komunikację oraz wymianę informacji. Broker komunikatów, bo również pod taką nazwą funkcjonuje, realizuje taką funkcję dzięki tłumaczeniu komunikatów pomiędzy formalnymi protokołami komunikacyjnymi. Dzięki temu usługi współzależne mogą ze sobą bezpośrednio „rozmawiać", nawet jeżeli zostały napisane w różnych językach lub wdrożone na różnych platformach [1].

System kolejkowy opiera się na komunikacji asynchronicznej. Polega ona na wysyłaniu danych przez nadawcę wiadomości bez konieczności przesłania informacji zwrotnej, co oznacza, że po drugiej stronie nie musi być odbiorcy w tym samym czasie. Taki rodzaj komunikacji nie wymaga współistnienia rozmawiających ze sobą procesów (bytów). Takie podejście sprawia, że poszczególne moduły, komponenty programu są luźniej powiązane, przez co podmiana ich jest łatwiejsza, bezpieczniejsza, a także awaria, któregoś z nich nie powoduje tylu błędów co w przypadku komunikacji synchronicznej.

Analizie zostały poddane dwa protokoły wykorzystywane w brokerach wiadomości – MQTT oraz AMQP.
Protokół MQTT jest lekkim protokołem transmisji danych, opartym o wzorzec publikacji /subskrypcji, pracujący na szczycie warstwy TCP (ang. Transmission Control Protocol) / IP (ang. Internet Protocol). Znajduje on zastosowanie głównie w systemach związanych z Internetem rzeczy. Natomiast protokół AMQP to protokół o otwartym standardzie, który umożliwia współdziałanie poprzez przesyłanie wiadomości między systemami, niezależnie od dostawcy lub używanej platformy brokera wiadomości. Dzięki standardowi AMQP do wymiany wiadomości można używać dowolnej biblioteki klienta zgodnej z AMQP oraz dowolnego brokera zgodnego z AMQP. Klienci wiadomości korzystający z AMQP są całkowicie niewiążący.

Motywacją do wykonania takiej analizy było bardzo częste oraz stale rosnące zastosowanie brokerów wiadomości w systemach komercyjnych.

W przedstawionej pracy zostały porównane protokoły systemów kolejkowych, ponieważ w dostępnej literaturze głównie porównywane są systemy kolejkowe same w sobie.

2. Przegląd literatury

W pierwszej pracy autorstwa J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate oraz P. Manzoni [2] zbadano protokoły AMQP i MQTT w takich przypadkach jak utrata wiadomości, opóźnienie, odchylenie od ustalonych, okresowych charakterystyk sygnału. Analiza została przeprowadzona przy użyciu aplikacji testowej, która nazywa się Amqperf, została opracowana w celu wygenerowania obciążenia dla systemu kolejkowania wiadomości u producenta. Amqperf korzysta z brokera wiadomości RabbitMQ, która jest implementacją protokołu AMQP oraz biblioteki Paho, która zawiera implementację protokołu MQTT. Badania wykazały, że użycie protokołu AMQP jest lepsze w przypadku budowania niezawodnych, skalowalnych i zaawansowanych infrastruktura przesyłania wiadomości w klastrze przez sieć WLAN.

W następnej pracy autorstwa N. Q. Uy oraz V. H. Nam [3] badano skuteczność użycia protokołów MQTT i AMQP w oparciu o eksperymenty, symulacje środowiska transmisji danych z różnym opóźnieniem i współczynnikiem utraty pakietów. Badaniu zostały poddane czasy, w którym pakiety poruszały się w środowisku z opóźnieniem oraz wskaźnikiem strat dla obu protokołów. Do przeprowadzenia eksperymentu zostały użyte dwa komputery z aplikacjami napisanymi w języku Python oraz bazą danych MySQL do zapisywania wyników wysyłania i odbierania wiadomości. Z przeprowadzonych badań wynika, że protokół MQTT przynosi pewne korzyści w przypadku bardzo nisko zasilających urządzeń, jest bardzo skuteczny jeśli skupiamy się głównie na wydajności, wymaga mniej zasobów. AMQP to bardziej zaawansowany protokół, bardziej niezawodny i zapewniający lepsze bezpieczeństwo.

W kolejnym badaniu autorstwa Mishra B., Mishra B. i Kertesz A. [4] poddano ocenie wydajność kilku implementacji brokerów dla protokołu MQTT przy użyciu testów obciążeniowych oraz analizy ich powiązań z projektem systemu. Analiza wyników testu odbywała się za pomocą trzech różnych metryk: obciążenie procesora, opóźnienie i szybkość wiadomości. Badania były prowadzone przy użyciu lokalnego komputera oraz Google Cloud Platform. Wyniki wykazały, że Mosquitto przewyższa inne rozważane rozwiązania w większości metryk. Jednak ActiveMQ jest najbardziej wydajny pod względem skalowalności dzięki wielowątkowej implementacji, podczas gdy Bevywise ma obiecujące wyniki dla scenariuszy z ograniczonymi zasobami.

Ostatnia praca autorstwa Kaciuczyk T., Korga T. oraz Smołka J. [5] dotyczyła badań wydajności wybranych brokerów komunikatów: Apache ActiveMQ, RabbitMQ oraz Apache Kafka. Analizie został poddany czas przesłania wiadomości wyznaczony na podstawie czasu wysłania i odebrania komunikatu. Testy zostały przeprowadzone za pomocą autorskich aplikacji klienckich napisanych w języku Java. Badania wykazały, że najbardziej wydajną kolejką wiadomości jest Apache Kafka.

Na podstawie przytoczonej literatury można stwierdzić, że podobne badania zostały już przeprowadzone. Jednak w tej pracy badane są głównie protokoły, nie systemy kolejkowe. Analizują one wydajność, tj. czas odbierania i wysyłania wiadomości dwóch protokołów wiadomości - AMQP oraz MQTT w dwóch systemach kolejkowych ActiveMQ i RabbitMQ.

Ponadto, na podstawie przeprowadzonej analizy literatury postawiono tezę: „Protokół MQTT jest wydajniejszy, tj. czas przesyłania wiadomości jest krótszy”

3. Aplikacje opracowane na potrzeby badań

Aplikacje do pomiarów czasów zostały napisane w języku Java oraz szkielecie aplikacji Spring. Programy rejestrują czasy wysłania i odebrania wiadomości oraz wyświetlają wyniki w konsoli. Takie testy zostały wykonane dla protokołu AMQP oraz MQTT przy wykorzystaniu systemów kolejkowych – ActiveMQ oraz RabbitMQ. Stąd, zostały stworzone odpowiednio aplikacje dla odbiorcy i wydawcy wiadomości w różnych modelach kolejek. Przesyłane wiadomości były w różnej liczbie jednocześnie. Na podstawie uzyskanych danych zostały wyznaczone średnie czasy przesyłu. Badania zostały przeprowadzone na komputerze o parametrach:

- procesor Intel Core i5-7200U CPU@2.5Ghz,
- 8GB pamięci RAM, dysk twardy typu SSD o pojemności 256 GB,
- system operacyjny Ubuntu 18.04.

4. Metoda badawcza

Średnie czasy wysyłania wiadomości zostały obliczone na podstawie 10 otrzymanych wyników dla różnych konfiguracji, aby ograniczyć ryzyko błędnych danych przez opóźnienia komputera. Testowano czas dla 1, 10, 1000 oraz 10000 wiadomości jednocześnie. Badania prowadzono dla wiadomości trwałych, czyli takich, które są przechowywane w systemie kolejkowym. Dla komunikatów powyżej jednego mierzono czas od momentu wysłania pierwszej wiadomości do momentu odebrania ostatniej wiadomości. Przesyłana wiadomość była stałej wielkości i wynosiła 10 kB.

Tabela 1: Scenariusz nr 1

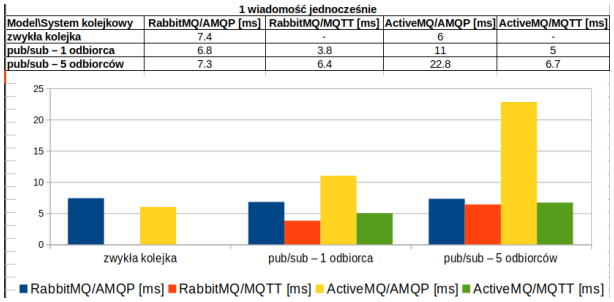
Nazwa: Badanie czasu wysłania wiadomości		
Cel badań: Ustalenie czasów wysyłania wiadomości dla różnych wielkości		
Warunki początkowe: Działające aplikacje, uruchomione systemy kolejkowe		
Warunki końcowe: Dane dot. czasów wysłania wiadomości zostaną wyświetlone oraz zapisane		
Liczba użytkowników biorących udział w badaniu: 1		
Kolejne czynności		
Lp.	**Opis czynności**	**Oczekiwany wynik**
1	Użytkownik uruchamia aplikację oraz system kolejkowy	W aplikacji zostają wyświetlone informacje o jej uruchomieniu oraz poprawnym połączeniu z systemem kolejkowym
2	Aplikacja zaczyna wysyłać wiadomości	W aplikacji zostają wyświetlone czasy wysłania wiadomości

Tabela 2: Scenariusz nr 2

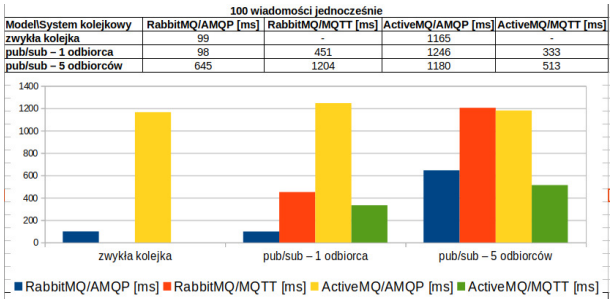
Nazwa: Badanie czasu odbierania wiadomości		
Cel badań: Ustalenie czasów odbierania wiadomości dla różnych wielkości		
Warunki początkowe: Dane dot. czasów odbierania wiadomości zostaną wyświetlone oraz zapisane		
Warunki końcowe: Dane dot. czasów wysłania wiadomości zostaną wyświetlone oraz zapisane		
Liczba użytkowników biorących udział w badaniu: 1		
Kolejne czynności		
Lp.	**Opis czynności**	**Oczekiwany wynik**
1	Użytkownik uruchamia aplikacje oraz system kolejkowy	W aplikacji zostają wyświetlone informacje o jej uruchomieniu oraz poprawnym połączeniu z systemem kolejkowym
2	Aplikacja, po jej uruchomieniu, zaczyna wysyłać wiadomości	W aplikacji nr 1 zostają wyświetlone daty wysłania wiadomości
3	Aplikacja nr 2 odbiera wiadomości wysłane przez aplikację nr 1	W aplikacji nr 2 zostają wyświetlone informacje o odebraniu wiadomości oraz daty odebrania

5. Wyniki

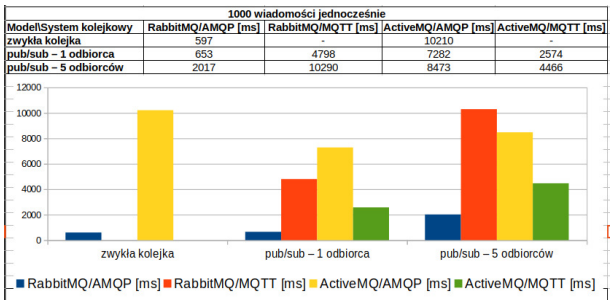
Pierwszym badaniom zostało poddane wysyłanie pojedynczych wiadomości w modelu zwykłej kolejki oraz w modelu publikacji i subskrypcji. W modelu publikacji i subskrypcji mierzono czasy dla jednego oraz pięciu subskrybentów. Natomiast z racji tego, że protokół MQTT obsługuje tylko model publikacji i subskrypcji, w badaniach nie uwzględniono zwykłej kolejki. Wynik badania przedstawia Rys. 1. Pozostałe badania wykonano analogicznie do poprzedniego, z tym, że wysyłano różne ilości wiadomości jednocześnie. Wyniki przedstawiają kolejno Rys. 2, Rys. 3 oraz Rys. 4.



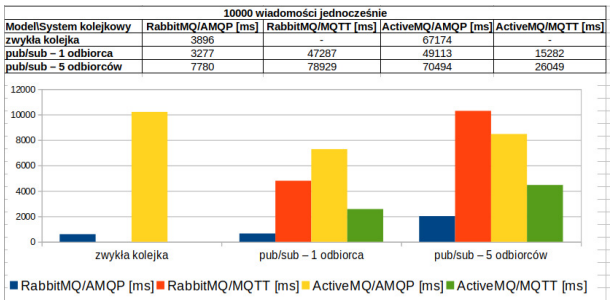
Rysunek 1: Czas przesłania pojedynczej wiadomości dla przedstawionych modeli, protokołów oraz kolejek.



Rysunek 2: Czas przesłania 100 wiadomości dla przedstawionych modeli, protokołów oraz kolejek.



Rysunek 3: Czas przesłania 1000 wiadomości dla przedstawionych modeli, protokołów oraz kolejek.



Rysunek 4: Czas przesłania 10000 wiadomości dla przedstawionych modeli, protokołów oraz kolejek.

6. Dyskusja

W modelu zwykłej kolejki przeprowadzone badania zostały z wykorzystaniem tylko protokołu AMQP. Zmierzone czasy wykazały, że lepiej poradził sobie system RabbitMQ. Dla pojedynczej wiadomości nie była to zauważalna różnica, gdyż wyniki różniły się o zaledwie 1.4 ms. Natomiast dla większej ilości komunikatów czasy różniły się o rzędy wielkości.

W modelu publikacji i subskrypcji można zauważyć, że RabbitMQ przy użyciu protokołu AMQP radzi sobie najlepiej, zarówno dla jednego odbiorcy jak i pięciu. Najgorzej w tym zestawieniu wypada message broker ActiveMQ przy użyciu protokołu AMQP.

Analizując dane pod względem tylko protokołów można stwierdzić, że generalnie lepsze czasy tj. krótsze osiąga protokół MQTT.

Czas przesyłania pojedynczej wiadomości drastycznie spada w przypadku wysyłania wielu wiadomości. Może to być spowodowane długim czasem łączenia się aplikacji z systemem kolejkowym czy też optymalizacjami message brokera podczas wysyłania tych samych wiadomości.

Porównując badania z przeglądem literatury można stwierdzić, że przeprowadzona analiza prowadzi do podobnych wniosków. Przykładowo, badania autorstwa N. Q. Uy oraz V. H. Nam [3] wykazały również, że protokół MQTT jest wydajniejszy. Zatem, postawiona teza: ”Protokół MQTT jest wydajniejszy, tj. czas przesyłania wiadomości jest krótszy” została udowodniona i potwierdzona.

7. Podsumowanie

W powyżej pracy porównano czas przesyłania wiadomości dla dwóch protokołów używanych
w systemach kolejkowych - AMQP oraz MQTT. Analiza została przeprowadzona przy użyciu dwóch brokerów wiadomości - RabbitMQ oraz ActiveMQ. Analiza wykazała, że wydajniejszym (osiągającym krótsze czasy przesyłania wiadomości) systemem jest RabbitMQ.
W wyniku porównania bezpośrednio protokołów wykazano, że wydajniejszym protokołem jest MQTT. Stwierdzono to poprzez obliczenie średnich czasów nie biorąc pod uwagę brokera wiadomości, a sam protokół.

Literatura

[1] Zasada działania, cechy systemu kolejkowego, https://www.ibm.com/cloud/learn/message-brokers, [12.09.2022].

[2] J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, P. Manzoni, A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks, 12th Annual IEEE Consumer Communications and Networking Conference 12 (2015) 931-936.

[3] N. Q. Uy, V. H. Nam, A comparison of AMQP and MQTT protocols for Internet of Things, 6th NAFOSTED Conference on Information and Computer Science (2019) 292-297, https://doi.org/10.1109/NICS48868.2019.9023812.

[4] B. Mishra, B. Mishra, A. Kertesz, Stress testing mqtt brokers: A comparative analysis of performance measurements, Energies 14 (2021) 5817-5837.

[5] T. Kaciuczyk, T. Korga, J. Smołka, Functional and performance analysis of selected message brokers in a distributed application, Journal of Computer Sciences Institute 14 (2020) 19-25.