Studia Informatica Pomerania nr 2/2017 (44) | www.wnus.edu.pl/si | DOI: 10.18276/si.2017.44-06 | 63–72

MODEL SYNCHRONIZACJI DANYCH SYSTEMÓW MOBILNYCH DOSTĘPNYCH W RAMACH USŁUG *CLOUD COMPUTING*

dr Grzegorz Wojarnik

Uniwersytet Szczeciński
e-mail:grzegorz.wojarnik@usz.edu.pl

Słowa kluczowe

cloud computing, synchronizacja danych, urządzenia mobilne, komunikacja mobilna

Abstrakt

Synchronizacja danych, zwłaszcza w dobie nabierającej coraz większego znaczenia technologii *cloud computing*, odgrywa coraz większą rolę. Równocześnie można zaobserwować coraz większą ilość urządzeń mobilnych, która sięga obecnie zawrotnej liczby miliardów urządzeń na całym świecie. Z uwagi na potrzeby użytkowników, które ewoluują w kierunku absorbcji różnorodnych informacji potrzebnych w pracy, rozrywce, życiu prywatnym, coraz bardziej istotny jawi się problem zapewnienia zgodności danych obecnych na urządzeniu użytkownika z danymi dostępnymi w ramach różnych usług oferowanych w modelu *cloud computing*. W artykule podjęto próbę opisania różnorodnych aspektów wymiany danych na styku urządzenia mobilne – usługi *cloud computing*. Przedstawiono: wyzwania komunikacji mobilnej, kryteria jakości wymiany danych mobilnych, wymieniono modele i zasady synchronizacji danych. W dalszej części przedstawiono modelową koncepcję synchronizacji danych w modelu *cloud computing*.

Wprowadzenie

Prawie każda aplikacja mobilna musi synchronizować dane z serwerem lub zapleczem (którym coraz częściej są serwisy w ramach usług *cloud computing*), ale trudno znaleźć wzorce, algorytmy, strategie lub przykładowy kod, aby pomóc programistom wdrożyć optymalną strategię synchronizacji danych rozwijanych przez nich aplikacji mobilnych.

Synchronizacja danych może być definiowana jako wymiana rekordów między dwiema różnymi bazami danych lub spójne przechowywanie replikowanych kopii zestawu danych (Sethia, Mehta, Chowdhary, Bhatt, Bhatnagar, 2014). W kontekście aplikacji mobilnych jest to system, który ustala ruch danych między urządzeniem mobilnym a bazą danych po stronie serwera (Sedivy, Barina, Morozan, Sandu, 2012). Synchronizacja bazy danych może być wykonana w ramach procedury jednokierunkowej lub dwukierunkowej, może także odbywać się w trybie czasu rzeczywistego (synchronicznie) lub okresowo (asynchronicznie) (Zhenyu, Zhang, Zunfeng, 2010). Tryb asynchroniczny jest ze swojej natury okresowy i najbardziej korzystny dla środowiska, w którym klienci mogą rozłączyć się z siecią i nie stracić żadnego ze zmienionych obiektów po ponownym połączeniu. Takie podejście zapewnia najskuteczniejsze wykorzystanie pasma służącego do transmisji danych.

W obszarze komunikacji taki proces wymiany danych, gdzie kierunek ich pobierania (serwer wysyła dane do klienta) jest uważany za znacznie bardziej obciążony w zakresie wykorzystania przepustowości niż kierunek wysyłania danych (klient wysyła dane na serwer), charakteryzuje się wyraźną asymetrią (Barbará, 1999). Ponieważ zazwyczaj nie ma możliwości transmisji danych z bardzo dużą prędkością (należy zawsze przyjmować założenie, że łącze nie jest idealne i nie działa w sposób optymalny), ten model został uznany za pasujący do systemu mobilnego. Ponadto te same urządzenia pracujące w modelu klient–serwer oczywiście mają potencjał do bardzo szybkiej wymiany danych.

Serwer synchronizacji (SS) jest zlokalizowany między serwerem bazy danych a bazami danych umiejscowionymi na urządzeniach mobilnych. Układ taki umożliwia synchronizację danych, jak również zarządzanie pewnymi istotnymi informacjami, które można wykorzystać do efektywnej synchronizacji. Pula połączeń jest używana w celu zminimalizowania obciążenia dostępem po stronie serwera na serwerze SS, w którym są ustanowione zasady synchronizacji. Aby urządzenie mobilne przeprowadzało synchronizację, poszczególne zestawy narzędzi są przypisywane do każdego urządzenia, aby uzyskać dostęp do serwera SS za pośrednictwem sieci (gdy jest dostępna np. sieć przewodowa). W tej architekturze należy założyć, że połączenie może nie zawsze działać idealnie z urządzeniami mobilnymi, ponieważ mobilność jest jedną z cech urządzeń przenośnych, a użytkownicy przechowują swoje dane lokalnie w pamięci urządzenia mobilnego. Niektóre rozwiązania synchronizacji danych przyjęły powyższe podejście w rozwiązywaniu problemów związanych z wymianą danych między bazą danych urządzeń mobilnych i serwerem (Balakumar, Sakthidevi, 2012; Domingos i in., 2014). Jednakże badania (takie jak np. Alhaj i in., 2013), proponowały nieco inne podejście, zakładające połączenie bezprzewodowe między urządzeniem przenośnym a pośrednim (serwer synchronizacji). Dzięki temu serwer synchronizacji może uzyskać dostęp przez urządzenie mobilne za pośrednictwem połączenia bezprzewodowego. W powyższej architekturze wszystkie niezbędne dane znajdują się na serwerze, w którym znajduje się relacyjna baza danych. Z drugiej strony dane na serwerze są duplikowane w mobilnych bazach danych i na końcu synchronizuje się serwer, aby dokonać synchronizacji opartej na wcześniej zdefiniowanym algorytmie. Architektura ta jest skuteczna biorąc pod uwagę fakt, że urządzenia przenośne nie mogą utrzymywać połączenia z serwerem

i wymagać, aby bazy danych były przechowywane w trybie offline. Dane zapisane w trybie offline mogą być zsynchronizowane, gdy sieć jest dostępna.

Czasami zdarza się, że aplikacje mobilne, a nawet cały system, ma zostać przeniesiony z jednego urządzenia na inne, dlatego konieczne jest sklonowanie stanów aplikacji i danych w systemie magazynowania. Najistotniejszym w takim przypadku sposobem na zsynchronizowanie danych między dwoma środowiskami jest utrzymywanie identycznych danych dla wszystkich plików w ich systemach plików. W związku z tym dane muszą być aktualizowane, gdy tylko jedno ze środowisk zmodyfikuje dowolny plik, co pociąga za sobą ruch w sieci, a aplikacje muszą poczekać na zakończenie każdej operacji synchronizacji, jeśli jest używany ścisły protokół synchronizacji (Bernstein, Hadzilacos, Goodman, 1987), co może powodować długie opóźnienie w sieci komórkowej. Właśnie odpowiednie strategie synchronizacji pozwalają radzić sobie z tym problemem.

Aby poradzić sobie z powyższym problemem, należy najpierw podzielić pamięć danych na trzy kategorie: obraz systemu, dane całego systemu i dane aplikacji. Kiedy środowisko wirtualne jest zainicjowane, jego system plików jest ładowany z obrazem systemowym i pakietami aplikacji, które znajdują się na standardowym obrazie środowiska operacyjnego. Synchronizowanie obrazów systemu i pakietów aplikacji odbywa się rzadko i nie powinno być problemem dla środowiska wirtualnego w chmurze, ponieważ może pobrać obraz z sieci o dużej przepustowości. Dane dotyczące całego systemu odnoszą się do tych plików, które rejestrują informacje w całej systemie i/lub mogą wpływać na działanie systemu i aplikacji. Biblioteki są przykładami danych systemowych. Modyfikowanie bibliotek może mieć wpływ na wiele aplikacji działających w systemie. Kiedy środowisko fizyczne zmienia dane swojego systemu, często wymagane jest zatrzymanie poszczególnych aplikacji, na które mogą mieć wpływ, a czasem wymaga ponownego uruchomienia systemu. Podobnie jego odpowiednik środowiska wirtualnego powinien postępować zgodnie z tą samą procedurą.

Dane aplikacji odnoszą się do plików należących do aplikacji, a synchronizacja danych dotyczących aplikacji może odbywać się na podstawie każdego żądania podczas migracji aplikacji. Dlatego stosowana może być na żądanie polityka „leniwa" (*lazy*) (Gray, Lorie, Putzolu, Traiger, 1994), aby zsynchronizować dane na żądanie aplikacji bez ciągłego aktualizowania danych aplikacji przez protokół spójności. Uruchamianie wspólnej aplikacji, która działa jednocześnie na wielu urządzeniach, wymaga od twórcy aplikacji zaprojektowania specjalnie dla niej interfejsów transmisji danych oraz synchronizacji.

Wyzwania komunikacji mobilnej

W związku ze specyfiką komunikacji mobilnej, podczas projektowania rozwiązania dotyczącego synchronizacji mobilnej, zwłaszcza biorąc pod uwagę przetwarzanie danych w modelu *cloud* computing, należy zwrócić uwagę na najważniejsze wyzwania (Dinh, Lee, Niyato, Wang, 2013) do których należą:

– niska przepustowość, gdzie szerokość dostępnego pasma jest jednym z największych problemów dla mobilnej chmury obliczeniowej, ponieważ zasób radiowy dla sieci bezprzewodowych jest znacznie rzadszy w porównaniu z tradycyjnymi sieciami przewodowymi,
– dostępność usługi, która staje się ważniejszą kwestią dla mobilnej chmury obliczeniowej niż np. w środowisku klient–serwer z sieciami przewodowymi; użytkownicy urządzeń przenośnych mogą nie być w stanie połączyć się z chmurą z powodu przeciążenia ruchu, awarii sieci bądź braku sygnału.
– heterogeniczność, gdzie urządzenie mobilne w chmurze będzie stosowane w sieciach o wysokiej różnorodności w zakresie interfejsów sieci bezprzewodowych; różne węzły mobilne mają dostęp do chmury za pośrednictwem różnych technologii dostępu radiowego, takich jak WCDMA, GPRS, WiMAX, CDMA2000 i WLAN, w związku z tym pojawia się problem z obsługą połączeń bezprzewodowych, spełniających wymagania mobilnej chmury obliczeniowej (np. ciągła łączność, skalowalność połączeń bezprzewodowych na żądanie i efektywność energetyczna urządzeń przenośnych).

Aby zapewnić możliwość wymiany danych pomiędzy urządzeniami mobilnym a usługami dostępnymi w ramach usług *cloud computing*, można wyróżnić następujące wzorce wymiany danych (Data Interchange…), które pozwalają sprostać wymaganiom współczesnych systemów przetwarzania rozproszonego:

1. Asynchroniczna synchronizacja danych – z danymi wzorcowymi aplikacja mobilna synchronizuje się z serwerami zaplecza bez blokowania interfejsów użytkownika. Proces ten odbywa się w ten sposób, że aplikacja ta działa z lokalnymi danymi lub pamięcią podręczną w trybie ciągłym, jednak prezentowane dane w urządzeniach mogą być niezgodne z serwerem zaplecza podczas procesu synchronizacji.
2. Synchroniczna synchronizacja danych – jest to wzorzec samego mechanizmu wymiany i uzgadniania danych. Gdy rozpocznie się proces synchronizacji, aplikacja mobilna musi poczekać, aż zakończy się synchronizacja. Oczywiście wątek interfejsu użytkownika jest również blokowany. Zgodnie z tym wzorcem użytkownik ma pewność, że zawsze pracuje na danych zgodnych z serwerem.
3. Częściowe składowanie danych – ten wzorzec pozwala aplikacji na zapisanie danych tylko w razie potrzeby, dlatego też urządzenia przenośne zmniejszają zużycie pamięci lokalnej, a także optymalizują przepustowość sieci.
4. Przechowywanie danych kompletnych – w tym schemacie aplikacja mobilna pobiera wszystkie dane z serwerów, nawet jeśli nie potrzebuje ich od razu. Dzięki temu aplikacja ta ma dużą dostępność do danych, nawet jeśli połączenie jest niedostępne. Takie podejście może jednak marnować dużą ilość lokalnej przestrzeni przewidzianej na przechowywanie danych i szerokości pasma.
5. Pełny transfer danych – aby zminimalizować konflikty, pełny wzorzec transferu wymaga przesłania i pobrania wszystkich danych dla każdego żądania synchronizacji, w związku z tym ruch duplikujących danych powoduje znaczne zużycie dostępnego pasma na transfer danych.

6. Transfer znaczników czasu (*timestamps*) – w tym wzorcu używane są znaczniki czasowe jako identyfikatory w celu aktualizowania niezsynchronizowanych danych w miarę potrzeb, dzięki czemu oszczędzana jest przepustowość, gdyż nie są przesyłane dane nadmiarowe.
7. Transfer matematyczny – ten schemat wymiany danych jest podobny do transferu opartego na znacznikach czasu. Poza prostym porównaniem w znaczniku czasu, istnieje kilka istotnych obliczeń matematycznych w celu wykrycia niezsynchronizowanych danych.

W sprawnym przetwarzaniu danych w komunikacji mobilnej, opartej zwłaszcza na przetwarzaniu danych w ramach usług *cloud computing*, istotne są kryteria, od których zależy jakość komunikacji mobilnej, a w konsekwencji jakość procesu synchronizacji danych oraz jakość danych dostępnych w ramach aplikacji mobilnych użytkownika. Poniżej omówiono te kryteria (za: Rehman Khan, Othman, Madani, Khan, 2013).

Świadomość kontekstu modelu aplikacji, która odwołuje się do jej świadomości na temat podmiotów i parametrów mogące wpływać na decyzje obciążenia obliczeniowego. Zasadniczo bardzo ważne jest, aby model aplikacji był świadomy kontekstu, ponieważ statyczne obciążenie nie zawsze jest korzystne, a mogą wystąpić przypadki, gdy wydajność aplikacji ulegnie degradacji wraz ze wzrostem obciążenia obliczeniowego.

W mobilnym przetwarzaniu danych w chmurze **latencja** określana jest jako czas związany z obciążeniem obliczeniowym i odbieraniem wyników z pobliskiej infrastruktury lub chmury, czasami określany jako czas odpowiedzi. Latencja zależy od wielu czynników, takich jak rozmiar kodu, rozmiar wprowadzania danych, lokalizacja wymaganych danych, schemat obciążenia i granulacja, przepustowość sieci, opóźnienie wykonania i wynikowy rozmiar danych.

W modelach aplikacji **wykorzystanie przepustowości** odnosi się do ilości danych/kodu wysyłanego lub pobieranego podczas wymiany danych. Dlatego też, jeśli obciążenie to wymaga dodatkowo dużej ilości danych, które mają zostać przeniesione, mogą wystąpić większe opóźnienia w odpowiedzi systemu. Alternatywnie, jeśli dane są wcześniej pobierane z chmury w celu zmniejszenia opóźnienia, jest wymagana synchronizacja danych.

Ogólność modelu aplikacji, która odnosi się do jego wsparcia dla szeregu zastosowań. W praktyce istnieje wiele typów aplikacji o różnych wymaganiach i zachowaniu zasobów.

Prywatność – dzięki zaawansowanym technologiom urządzeń przenośnych, np. takie czujniki jak GPS stały się tanie i są dostępne w prawie wszystkich najnowszych smartfonach. Wiele nowych aplikacji wymaga lokalizacji użytkownika do dostarczania usług opartych na lokalizacji. Usługi te są wywoływane przez użytkownika, aby uzyskać informacje o lokalizacji lub zlecić usługodawcy dostarczanie reklam opartych na lokalizacji. Stąd istotna jest możliwość określania przez użytkownika, w jakim zakresie aplikacja mobilna będzie dzieliła się z usługami w chmurze.

Kompleksowość – aplikacje opracowane dla platform mobilnych w chmurze muszą być w stanie działać zarówno w trybie online, jak i offline. Ponadto muszą korzystać z minimalnej szerokości pasma ze znacznym opóźnieniem. W związku z tym niektóre modele dzielą aplikacje

na składniki łatwiejsze do rozliczenia, które mogą być przenoszone do chmury przy minimalnym zapotrzebowaniu na pasmo.

Bezpieczeństwo jest jednym z najbardziej widocznych „wąskich gardeł" w przyjmowaniu usług *cloud computing* przez użytkowników. Technologia *cloud computing* znosi wiele kwestii związanych z bezpieczeństwem, na przykład kontrolę dostępu do danych, transfer danych w infrastrukturze dystrybucyjnej, integralność danych, dostępność usług i bezpieczną komunikację. Ponadto mobilność powoduje dodatkowe kwestie związane z bezpieczeństwem, które sprawiają, że mobilne zabezpieczenia w chmurze są trudniejsze niż w systemach, nad którymi pełną kontrolę posiada użytkownik. Kolejnym problemem związanym z bezpieczeństwem jest zapewnienie dostępności zasobów dla niezaufanych użytkowników. Na przykład ktoś może używać praktycznie nieograniczonych zasobów w ramach przedsiębiorstwa i powodować problemy dla innych użytkowników działających w tej samej chmurze i w konsekwencji – dla dostawców usług w chmurze.

Abstrakcje programistyczne – platformy chmury obsługują różne interfejsy API, modele danych, języki zapytań i modele kosztów. Podobnie smartfony działają w różnych systemach operacyjnych, które mają zmienne wymagania sprzętowe i programowe. Dlatego heterogeniczność w smartfonach i platformach *cloud computing* sprawia, że mobilne aplikacje stają się skomplikowane. Jednak niejednorodności powstają ze względu na brak standardów, a czasami konstrukcje programistyczne są tworzone przez dostawców w ten sposób, aby uwzględniać zachowanie użytkowników.

Skalowalność jest jedną z najważniejszych cech *cloud computing*, dlatego mobilne modele aplikacji w chmurze muszą wspierać rozwój aplikacji, które można skalibrować w chmurze, aby sprostać nieprzewidywalnym wymaganiom użytkowników. Co więcej, modele aplikacji muszą ulepszyć obsługę oferowanych funkcji w celu szybkiego wprowadzania nowych typów aplikacji.

Zasoby uruchomieniowe – aplikacje mobilne w chmurze są wykonywane na dwa sposoby. W pierwszym przypadku uruchamiane są na pobliskiej infrastrukturze, która działa jak wirtualna chmura, na przykład komputery osobiste, laptopy i serwery. W drugim przypadku aplikacje działają w prawdziwej chmurze, na przykład Amazon AWS, Google Cloud i Microsoft Azure. Dlatego mobilne modele aplikacji w chmurze powinny dysponować możliwością obsługi zarówno dostępnej infrastruktury lokalnej, jak i chmury, ale również obydwu tych rodzajów infrastruktury.

Platforma jest podstawową technologią oprogramowania smartfonów, na których oparte są modele aplikacji. Smartfony produkowane przez różnych producentów mogą być zgrupowane razem w oparciu o systemy operacyjne działające na urządzeniach. Znane systemy operacyjne smartfonów to: Android, iOS, Windows Phone.

Wzorce mechanizmów synchronizacji danych odpowiadają na pytanie: „Kiedy aplikacja ma synchronizować dane między urządzeniem a systemem zdalnym (takim jak serwer chmury)?" Ten problem jest często pomijany w projektowaniu aplikacji mobilnych, ale nie istnieje jedno rozsądne rozwiązanie. Twórcy aplikacji mobilnych muszą wziąć pod uwagę ograniczenia wielu czynników, w tym dostępność sieci, wymagania dotyczące aktualności danych i projekt

interfejsu użytkownika. Wykorzystanie wzorców mechanizmów synchronizacji danych można rozpatrywać z dwóch perspektyw: przesyłania i pobierania danych. Przesyłanie danych jest wysłaniem danych z aplikacji mobilnej do zdalnego systemu, podczas gdy pobieranie danych jest przesyłaniem danych z systemu zdalnego do aplikacji mobilnej. W obu przypadkach sukces lub niepowodzenie operacji powinny być przekazane użytkownikowi bezpośrednio (z powiadomieniem lub dialogiem) lub pośrednio (w dzienniku lub oddzielnej części aplikacji) z odpowiednimi informacjami o błędach, jeśli wystąpi awaria (Mccormick, Schmidt, 2012).

Założenia do modelu synchronizacji danych w architekturze cloud computing

Należy założyć, że to aplikacja mobilna (urządzenie) zawsze inicjuje synchronizację, a nie serwer. Podejście takie jest spowodowane faktem, że aplikacja najlepiej wie, kiedy jest właściwy moment na synchronizację danych, np. inicjacja operacji synchronizacji zależy od bieżącego stanu połączenia z serwerem lub od działania użytkownika aplikacji.

Po stronie aplikacji mobilnej powinien znaleźć się mechanizm obsługi ewentualnych konfliktów. Przyczyną takiej funkcjonalności jest fakt, że w wielu przypadkach konflikt wymaga informowania użytkownika. Nie oznacza to, że aktualna wersja obiektu podlegająca synchronizacji po stronie klienta ma pierwszeństwo! Która wersja zostanie określona jako priorytetowa (serwer lub klient), zależy od strategii rozwiązywania konfliktów wybranej dla danego projektu (Nelissen, 2014).

Aplikacja mobilna zawsze w pierwszej kolejności powinna pobierać dane z serwera, a dopiero następnie powinny być przesyłane nowe lub zmienione dane z aplikacji na serwer.

W związku z faktem, że serwer nie jest świadomy stanu synchronizacji klientów, aplikacja po stronie klienta musi śledzić, które dane zostały już zsynchronizowane, a które nie. Dzieje się tak, ponieważ aplikacje mobilne (klienci) mogą być usuwane, ponownie instalowane itd. Natomiast serwer nie powinien śledzić całej tej działalności. Istotne też jest zaimplementowanie spójności transakcji. Oznacza to, że działanie aplikacji mobilnej powinno zagwarantować pełną spójność danych. Można to rozwiązać poprzez opracowanie koncepcji blokowania i wprowadzenie mechanizmu sprawdzającego, czy faktycznie dane wysłane na serwer zostały zapisane (chociażby poprzez takie zaprojektowanie obiektu *response*, zwracającego dane z API serwera, który będzie zawierał atrybut lub grupę atrybutów informujących o sukcesie zapisu danych po stronie serwera).

Aby powyższe założenia mogły być zrealizowane, muszą być spełnione następujące warunki:

- obiekty podlegające synchronizacji powinny posiadać unikalne identyfikatory (najlepiej oparte o GUID),
- obiekty podlegające synchronizacji powinny posiadać atrybut typu *timestamp*, który będzie aktualizowany podczas każdej operacji modyfikacji obiektu,
- zegary po stronie aplikacji mobilnej oraz po stronie serwera nie muszą być zsynchronizowane,

– każda operacja aktualizacji danych po stronie serwera zwróci klientowi dla każdego synchronizowanego obiektu czas modyfikacji, który zostanie zapisany po stronie aplikacji klienckiej dla danego obiektu,
– każda zmiana obiektu po stronie aplikacji klienckiej spowoduje oznaczenie takiego obiektu jako przeznaczonego do synchronizacji,
– API serwera posiada mechanizm umożliwiający zwrócenie zestawu danych zmienionych po określonym czasie.

W ramach modelu synchronizacji danych należy wyróżnić następujące scenariusze algorytmu synchronizacji:

1. Pobranie danych z serwera od ostatniej synchronizacji
   a) pobranie wszystkich nowych obiektów z serwera,
   b) ustalenie ostatniej daty modyfikacji dla odebranych obiektów i zapisanie jej jako czasu ostatniej synchronizacji,
   c) dodanie/aktualizacja pobranych obiektów w lokalnej bazie danych aplikacji klienckiej,
   d) jeśli któryś z pobranych obiektów jest równocześnie oznaczony jako zmodyfikowany i niezsynchronizowany, to poinformowanie o takiej niezgodności użytkownika, który powinien podjąć decyzję, czy ustali dane z serwera jako aktualne, czy pozostawi zmieniony obiekt w dalszym ciągu jako przeznaczony do synchronizacji.
2. Wysłanie zmienionych/dodanych danych na serwer
   a) po stronie serwera dla wszystkich obiektów zostanie ustawiony ten sam czas synchronizacji, który zostanie zwrócony aplikacji klienckiej,
   b) we wszystkich obiektach wysłanych do synchronizacji zostanie ustawiony atrybuty czasu modyfikacji na ten, który został zwrócony przez serwer,
   c) o ile powyższa operacja zostanie zakończona sukcesem, wszystkie obiekty wysłane do synchronizacji zostaną oznaczone jako zsynchronizowane,
   d) zapisanie daty z serwera jako daty ostatniej synchronizacji po stronie aplikacji klienckiej.

Podsumowanie

Problem synchronizacji danych jest na tyle szeroki, że wymaga obszernych studiów i badań, które pozwoliłyby na optymalne wykorzystanie urządzeń powszechnego użytku, jakimi są urządzenia mobilne, oraz użycie danych dostępnych w bardzo różnorodnych obecnie urządzeniach mobilnych. W związku z tym w artykule podjęto próbę systematyki tego zagadnienia oraz zaproponowano założenia dla modelu synchronizacji danych, które zostaną zapewne rozwinięte przez autora w dalszej pracy badawczej.

Literatura

Alhaj, T.A., Taha, M.M., Alim, F.M. (2013). Synchronization Wireless Algorithm Based on Message Digest (SWAMD) For Mobile Device Database (s. 259–262). *International Conference on Computing, Electronics and Electrical Engineering Synchronization*.

Balakumar, V., Sakthidevi, I. (2012). An Efficient Database Synchronization Algorithm for Mobile Devices Based on Secured Message Digest (s. 937–942). *International Conference on Computing, Electronics and Electrical Technology*.

Barbará. D. (1999). Mobile Computing and Databases – A Survey. *IEEE Transactions on Knowledge and Data Engineering*, *11* (1), 108–117.

Bernstein, P.A., Hadzilacos, V., Goodman, N. (1987). *Concurrency Control and Recovery in Database Systems*. Boston: Addison-Wesley.

Dinh, H.T., Lee, C., Niyato, D., Wang, P. (2013). A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches. *Wireless Communications and Mobile Computing*, *13*, 1587–1611. DOI: 10.1002/wcm.1203.

Data Interchange Models. Pobrano z: https://ecommons.usask.ca/xmlui/bitstream/handle/10388/ETD-2014-09-1765/FU-THESIS.pdf?sequence=3&isAllowed=y (4.04.2017).

Gray, J.N., Lorie, R.A., Putzolu, G.R., Traiger, I.L. (1994). Granularity of Locks and Degrees of Consistency in a Shared Data Base. W: J.M. Hellerstein, M. Stonebraker (red), *Readings in Database Systems*, 181–208. Morgan Kaufmann Publishers Inc.

McCormick, Z., Schmidt, D.C. (2012). *Data Synchronization Patterns in Mobile Application Design*. Proceedings of the Pattern Languages of Programs (PLoP) 2012 Conference, October 19–21, Tucson, Arizona.

Nelissen, N. (2014). *AppSync.org: Open-source Patterns and Code for Data Synchronization in Mobile Apps*. Pobrano z: https://www.slideshare.net/nikonelissen/appsyncorg-opensource-patterns-and-code-for-data-synchronization-in-mobile-apps (28.04.2017).

Rehman Khan, A.U., Othman, M., Madani, A.A., Khan, S.U. (2013). A Survey of Mobile Cloud Computing Application Models. *IEEE Communications Surveys & Tutorials*, *16*, 393–413. Pobrano z: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.402.1725 (29.04.2017).

Sedivy, J., Barina, T., Morozan, I., Sandu, A. (2012). *MCSync – Distributed , Decentralized Database for Mobile Devices*. IEEE International Conference on Cloud Computing in Emerging Markets (CCEM).

Domingos, J., Simões, N., Pereira, P., Silva, C., Marcelino, L. (2014). *Database Synchronization Model for Mobile Devices* (s. 1–7). 9th Iberian Conference on Information Systems and Technologies (CISTI).

Sethia, D., Mehta, S., Chowdhary, A., Bhatt, K., Bhatnagar, S. (2014). *MRDMS-Mobile Replicated Database Management Synchronization* (s. 624–631). International Conference on Signal Processing and Integrated Networks (SPIN).

Zhenyu, L., Zhang, C., Zunfeng, L. (2010). *Optimization of Heterogeneous Databases Data Synchronization in WAN by Virtual Log Compression* (s. 98–101). Second International Conference on Future Networks.

MODEL OF MOBILE DATA SYNCHRONIZATION WITHIN CLOUD COMPUTING SERVICES

KEYWORDS cloud computing, data synchronization, mobile devices, mobile communication

ABSTRACT

Data synchronization especially in the era of ever-increasing importance of cloud computing is playing an increasingly important role. At the same time, a growing number of mobile devices are now reaching billions of devices around the world. Due to the needs of users who evolve to absorb the variety of information they need in their work, entertainment and personal life, there is a growing concern that the data present on the user's device will be compatible with the data available under the various services offered in the cloud computing model. Therefore, the article attempts to describe the various aspects of data exchange at the interface of mobile devices – cloud computing services. The following are presented: mobile communication challenges, mobile data exchange quality criteria, models and rules for data synchronization. The model for data synchronization in the cloud computing model is presented also.