POZNAN UNIVERSITY OF TECHNOLOGY

Filip Waligórski

Rozgłaszanie danych w grafach dużej skali

Praca magisterska

Promotor: dr Anna Kobusińska

Poznań, 2017

Spis treści

Rozdział 1

Wstęp

Rozdział 2

Istniejące rozwiązania

W tym rozdziale zaprezentowano istniejące komunikatory dla dwóch osób oraz komunikatory grupowe. Ich cele i funkcjonalność są zbliżone choć realizują je z różnymi założeniami oraz bazując na różnych architekturach i koncepcjach. Poniżej opisane zostały wybrane rozwiązania z naciskiem na cechy wyróżniające je spośród konkurencyjnych aplikacji.

Facebook Messenger [1]

Jest to jeden z najpopularniejszych obecnie komunikatorów. Oferuje zarówno czat dla 2 osób jak i grupowy. Wspiera wysyłanie wszelkich multimediów i plików oraz dostarczanie wiadomości pod nieobecność nadawcy. Dostępny jest na najszerszej gamie platform — jako aplikacja webowa, mobilna oraz desktopowa, czym wyróżnia się na tle konkurencji. Architektonicznie Messenger polega na „centralnym serwerze" przekazującym wiadomości. Cudzysłów wynika z faktu, że pod pojęciem „serwer" kryje się ogromna infrastruktura złożona z wielu maszyn, którą firma musi utrzymywać. Wadami tego komunikatora jest choćby brak wsparcia szyfrowania wiadomości czy fakt, że nie jest to oprogramowanie open-source.

Bleep [2]

Bleep jest komunikatorem zaprojektowanym przez firmę rozwijającą protokół BitTorrent. Do dyspozycji użytkowników oddano aplikację mobilną oraz aplikację desktopową (brak aplikacji webowej). Kod źródłowy aplikacji nie został udostępniony (nie jest to open-source), co oznacza, że użytkownicy nie mogą upewnić się, czy aplikacja działa i została zaimplementowana zgodnie z założeniami. Bleep oferuje rozmowy dla dwóch osób, a w planach twórców jest zaimplementowanie komunikacji grupowej. Wiadomości są szyfrowane przed wysłaniem na urządzeniu nadawcy i odszyfrowywane na urządzeniu odbiorcy — szyfrowanie end to end.

Jednak najważniejszą cechą wyróżniającą ten komunikator jest jego architektura - brak centralnego serwera pośredniczącego w przekazywaniu wiadomości. Komunikaty przesyłane są bezpośrednio między urządzeniami, jeśli oba są dostępne w momencie wysyłania, a w przeciwnym przypadku wiadomość umieszczana jest w DHT (Distributed Hash Table) i przechowywana do czasu odebrania jej. Specjalny mechanizm dba o to, by wiadomość nie

zniknęła z DHT wcześniej. Dane o koncie użytkownika oraz klucze szyfrujące pozostają lokalnie na urządzeniu.

Signal [3]

Twórcy aplikacji Signal skupili się przede wszystkim na bezpieczeństwie i prywatności użytkowników. Wiadomości są szyfrowane na urządzeniach więc pomimo faktu, że architektura zakłada obecność centralnego serwera, wiadomości przechowywane na nim nie mogą zostać odczytane przez osoby trzecie. Kod źródłowy jest dostępny publicznie co oznacza, że każdy może sprawdzić zgodność implementacji z oferowanymi założeniami. Podobnie jak w przypadku aplikacji Bleep, dostępne są natywne aplikacje mobilna i desktopowa. Możliwe jest prowadzenie rozmowy grupowej pomimo zastosowania szyfrowania wiadomości — treść zostaje zaszyfrowana symetrycznie (jedna wersja dla wszystkich odbiorców niezależnie od ich liczby), a następnie sam klucz jest szyfrowany zgodnie z oczekiwaniami każdego z odbiorców z osobna. Dzięki temu mechanizmowi uniknięto sytuacji, w której nadawca musiałby przygotować n wersji całej, potencjalnie dużej wiadomości dla n odbiorców.

Podobne rozwiązania: Wire, WhatsApp, Telegram, Allo

Darkwire [4]

Darkwire to aplikacja open-source oferująca komunikator grupowy z dostępem poprzez stronę internetową (aplikacja webowa). W przeciwieństwie do większości rozwiązań użytkownik nie musi tworzyć konta by skorzystać z programu. W celu skomunikowania się z użytkownikami należy wymienić między nimi identyfikator czatu (link do konkretnego pokoju) za pośrednictwem innego kanału komunikacyjnego (np. e-mail). Takie rozwiązanie zakłada, że identyfikator nie zostanie odgadnięty przez osoby trzecie — w przeciwnym przypadku będą one mogły odczytać wysyłane wiadomości. Architektura zakłada istnienie centralnego serwera uczestniczącego w przekazywaniu wiadomości. Z racji faktu, że aplikacja ma otwarte źródła, każdy może uruchomić swój własny serwer. Komunikaty są szyfrowane na urządzeniu (w przeglądarce) przed wysłaniem, zatem serwer nie zna treści wiadomości. Centralny serwer przesyła wiadomości tylko do tych uczestników, którzy są dostępni w momencie nadania wiadomości (brak wsparcia dla odbierania starszych wiadomości czy wysyłania wiadomości do użytkowników niedostępnych w danej chwili).

Friends [5]

Ten niszowy projekt open-source oferuje aplikację desktopową i umożliwia czat grupowy. Szyfrowanie wiadomości nie zostało do tej pory zrealizowane, ale jest jednym z punktów przyszłego rozwoju. Głównym celem twórców było stworzenie programu niezależnego od centralnego serwera oraz umożliwiającego rozmowę przy użyciu alternatywnych kanałów komunikacyjnych (np. poprzez Bluetooth) w sytuacji gdy połączenie internetowe jest niedostępne. Aplikacja wykorzystuje algorytm plotkowania (gossiping) oraz replikuje wiadomości przy użyciu drzewa skrótów (hash tree, Merkle DAG, DAG - Directed Acyclic Graph). Dzięki temu wiadomości w czacie mogą zostać połączone nawet w przypadku,

gdy ktoś nadał komunikaty będąc odłączonym od sieci — mechanizm podobny do łączenia zmian w repozytorium kodu. Gwarantuje to ostateczną spójność — przykładowy scenariusz dla 3 użytkowników:

1. Wiadomość wysłana przez użytkownika A została odebrana przez użytkownika B, który dołączył ją do swojego drzewa wiadomości.
2. Użytkownik A stał się niedostępny.
3. Użytkownik C stał się dostępny i odebrał od użytkownika B zmienioną wersję drzewa i w ten sposób dowiedział się o wiadomości wysłanej przez użytkownika A pomimo faktu, że ten jest w tej chwili niedostępny.

Tox [6]

Tox jest z założenia rozproszonym i szyfrowanym protokołem do wymiany wiadomości. Powstało kilkanaście implementacji klientów obsługujących go co pozwala na komunikowanie się z użytkownikami różnych aplikacji. Wśród zaimplementowanych aplikacji są programy na komputery stacjonarne oraz smartfony. Wiadomości są przesyłane bezpośrednio między nadawcą i odbiorcą dlatego obie strony muszą być dostępne jednocześnie. Brak wsparcia dostarczania wiadomości gdy jedna strona jest niedostępna to duża wada wszystkich aplikacji implementujących ten rodzaj transmisji P2P. Jednym z rozwiązań tego problemu zaproponowanym przez twórców protokołu jest skorzystanie z serwerów, którym użytkownik ufa i których zadaniem jest przekazywanie wiadomości do odbiorcy pod nieobecność nadawcy. Narusza to jednak założenie o rozproszeniu systemu (braku centralnych węzłów). Wsparcie dla komunikacji grupowej jest jednym z celów rozwoju protokołu.

ZeroChat, ZeroMail, BitMessage [7][8]

Przytoczone aplikacje realizują pomysły na komunikatory oparte o mechanizm podobny do transakcji kryptowalutowych. Wysłanie wiadomości wymaga obliczenia funkcji skrótu z zadanym prefiksem (proof of work) i umieszczenia bloku w łańcuchu (blockchain). Samo tylko wyliczenie funkcji skrótu powinno z definicji zająć około 4 minut [9], podczas gdy pozostałe komunikatory dążą do uzyskania czasu dostarczenia wiadomości bliskiego zeru (rozmowa w czasie rzeczywistym). Mimo tej znaczącej wady należy potraktować te projekty jako próbę stworzenia rozwiązania o innej architekturze niż dotychczas zaprezentowane (centralny serwer lub P2P). Być może w przyszłości wady uda się zminimalizować, a zalety architektury opartej o blockchain okażą się kluczowe.

Rozdział 3

Koncepcja

Rozdział 4

Architektura

Rozdział 5

Wyniki testów

Rozdział 6

Wnioski

Bibliografia

[1] https://pl-pl.messenger.com/

[2] http://www.bleep.pm/

[3] https://whispersystems.org/

[4] https://darkwire.io/

[5] http://moose-team.github.io/friends/

[6] https://tox.chat/

[7] https://zeronet.readthedocs.io/en/latest/using_zeronet/sample_sites/

[8] https://bitmessage.org/wiki/Main_Page

[9] https://bitmessage.org/bitmessage.pdf