Rancang Bangun Sistem Pengirim Notifikasi Massal Menggunakan Metode Publish Subscribe Berbasis Protokol XMPP

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Mochammad Irsyad
NIM: 125150200111096

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2017

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

PENGESAHAN

RANCANG BANGUN SISTEM PENGIRIM NOTIFIKASI MASAL MENGGUNAKAN METODE PUBLISH SUBSCRIBE BERBASIS PROTOKOL XMPP

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Mochammad Irsyad
NIM: 125150200111096

Skripsi ini telah diuji dan dinyatakan lulus pada
1 Februari 2017
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Adhitya Bhawiyuga, S.Kom, M.S	Ari Kusyanti, S.T, M.Sc
NIK: 201405 890720 1 001	NIK: 201102 831228 2 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 27 Januari 2017

Mochammad Irsyad
NIM: 125150200111096

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT atas anugrah serta limpahan rahmat-nya sehingga penulis dapat menyelesaikan skripsi dengan judul "Rancang Bangun Sistem Pengirim Notifikasi Masal Menggunakan Metode Publish Subscribe Berbasis Protokol XMPP". Skripsi ini disusun sebagai syarat memperoleh gelar sarjan pada Fakultas Program Studi Teknik Informatika, Jurusan Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.

Dalam proses penyelesaian skripsi ini Penulis mendapatkan banyak bantuan, baik bantuan moral maupun materil dari berbagai pihak. Oleh karena itu, Penulis mengucapkan banyak terima kasih kepada :

1. Adhitya Bhawiyuga S.Kom, M.S selaku Pembimbing satu yang telah memberikan bimbingan, arahan, ilmu dan masukan dalam penyelesaian skripsi ini.
2. Ari Kusyanti S.T, M.Sc selaku Pembimbing dua yang juga telah memberikan bimbingan, arahan, ilmu dan masukan dalam penyelesaian skripsi ini.
3. Ahmad Afif Supianto, S.Si, M.Kom selaku Dosen Pembimbing Akademik yang telah memberi dukungan moril maupun masukan selama masa perkuliahan.
4. Bapak dan Ibu dosen yang telah mendidik dan memberikan ilmu selama Penulis menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya.
5. Segenap karyawan di Fakultas Ilmu Komputer Universitas Brawijaya yang membantu Penulis dalam urusan - urusan pengerjaan skripsi ini.
6. Ayah, Ibu, Pakde dan Bude saya yang mendukung Penulis dengan segala usahanya seperti memberikan doa, materi, dukungan moral, semangat hidup, dan tauladan yang semata - mata untuk keberhasilan Penulis. Selain itu, seluruh keluarga besar yang juga telah memberikan semangat dan doa kepada Penulis.
7. Teman - teman di Universitas Brawijaya yaitu Sapta Oryza Putra, Gagah Istaid Billah, Mochammad Syaifullah Ferryansyah, Muhammad Abduh, Mohammad Torriq, Almas Avicena, Zilfikri Y. Rachmat, Okvio Akbar Karuniawan, Tri Rosita, Dwi Kurnia Ayuningrum yang telah membantu, memberi semangat dan dukungan dalam mengerjakan tugas akhir ini.
8. Semua pihak yang tidak dapat saya sebutkan satu persatu, yang telah membantu dan terlibat baik secara langsung maupun tidak langsung dalam penulisan skripsi ini.

Penulis menyadari bahwa skripsi ini tidak lepas dari kesalahan dan kekurangan. Oleh karena itu, Penulis bersedia menerima kritik dan saran yang membangun untuk memperbaiki diri. Penulis berharap semoga skripsi ini dapat memberi manfaat.

Malang, 27 Januari 2017

Penulis
mochammadirsyad94@gmail.com

ABSTRAK

Informasi mengenai kegiatan atau peristiwa yang terjadi pada suatu kota diperlukan bagi masyarakat, khususnya bagi mereka yang memang memiliki kepentingan terhadap kota tersebut. Salah satu solusi untuk menyelesaikan permasalahan tersebut adalah dengan membangun komunikasi yang baik antar pemerintah dengan masyarakat kota. Oleh karena itu diperlukan adanya pembangunan sistem yang dapat menjadi sarana untuk mewujudkan solusi tersebut. Sebuah sistem pengirim notifikasi dapat menjadi pilihan sebagai sarana bagi pemerintah menyampaikan informasi atau berita mengenai kejadia di sebuah kota kepada masyarakatnya. Dalam tahap perancangan sistem, salah satu yang dibutuhkan ialah model arsitektural dari sistem yang ingin dibangun. Model arsitektural menjelaskan bagaimana komunikasi antar element dilakukan, tersusun atas metode dan protokol. Untuk membangun sebuah sistem yang menyediakan layanan pengiriman pesan untuk banyak pengguna dengan memperhatikan kecepatan dan ketersedian pesan sampai kepada penerimanya. Maka metode yang dapat menyelesaikan masalah tersebut ialah metode *publish - subscribe*. *Publish - subscribe* merupakan metode komunikasi berdasarkan kejadian dan mempunyai keunggulan dalam hal skalabilitas, *time decoupling, space decoupling, dan synchronization decoupling*. Protokol dalam pengiriman pesan yang dapat mendukung metode dari publish - subscribe ialah MQTT dan XMPP. Untuk itu peneliti melakukan *preliminary research* untuk membandingkan kedua protokol tersebut. Dari kesimpulan *preliminary research* yang didapat ialah MQTT mempunyai lebih sedikit *delay* dibanding XMPP, namun protokol yang dipilih untuk diterapkan pada sistem ialah protokol XMPP, dikarenakan kebutuhan dari sistem yang membutuhkan beberapa fitur namun hanya disediakan oleh XMPP, seperti pendafataran pengguna, login pengguna dan *retrieve* pesan. Hasil dari penelitian ini ialah pembangunan sebuah sistem pengirim notifikasi masal dengan menerapkan metode dari *publish - subscribe* berbasis protokol XMPP, harapannya dapat membantu pemerintah sebuah kota untuk menyampaikan informasi mengenai kota tersebut kepada masyarakatnya. Tujuan dari penelitian yaitu untuk memilih metode dan protokol yang diperlukan berdasarkan kebutuhan dari sistem yang ingin dibangun dan penelitian ini juga ditujukan agar dapat melihat kinerja dari sistem sehingga dapat menjadi acuan dari peneliti lain dalam mengerjakan penelitiannya atau membantu seseorang yang ingin mengembangkan sebuah sistem pengirim notifikasi.

Kata Kunci : *Pendorong Notifikasi, Publish - Subscribe, Protokol MQTT dan XMPP*

repository.ub.ac.id

ABSTRACT

The information about events that happen in a city are required for society, especially those who have interest with the city. One of the solutions to solve this problem is to establish a good communication between the government and the society. Therefore, it's necessary to develop a system that can be used as a means to achieve such a solution. A push notification system could become an option as a means for the government to convey information or news about events in a town to its community. In the system's design phases, it's necessary to have the architectural model of the system to be built. The architectural model explains how the communication between elements performed and it's composed of a method and a protocol. To build a system that provides a messaging service for many users with the speed and availability of the message that match the system requirements, publish – subscribe method is required to achieve such system. Publish - subscribe is a event based communication method that have advantages in terms of scalability, time decoupling, space decoupling and synchronization decoupling. The message delivery protocols that can support the method of publish - subscribe is MQTT and XMPP. Thus, the researcher did a preliminary research to compare both protocols. The obtained conclusion of the preliminary research is that eventhough MQTT has less delay than XMPP but, the protocol that has to be applied to the system is the XMPP protocol, because the one that managed to fulfill the system requirements is XMPP, such as register users, login users and retrievel message. The results of this research is the development of the mass notification sender system by using the method of publish - subscribe protocol based on XMPP, with the expectation to help the government of a city to convey the information about the town to its community. The purpose of this study was to choose the method and protocol required by the needs of the system to be built and this study is also intended to be able to see the performance of the system so that it can become the reference of other researchers in doing future research or it can help someone who want to develop a push notification system.

***Keywords** : Push Notification, Publish-Subscribe, Protocol MQTT and XMPP*

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR DIAGRAM

BAB 1
PENDAHULUAN

1.1 Latar belakang

Informasi mengenai kegiatan atau peristiwa yang terjadi pada suatu kota diperlukan bagi masyarakat, khususnya bagi mereka yang memang memiliki kepentingan terhadap kota tersebut. Adanya kegiatan ataupun peristiwa yang terjadi secara mendadak pada suatu wilayah seperti acara pernikahan, konser, perbaikan jalan, matinya lampu lalu lintas hingga kecelakan merupakan hal - hal yang sering terjadi dan dapat mengganggu aktifitas keseharian dari warganya. Diambil dari dua portal berita yang berjudul, Malang Macet (Roy, 2015) dan Dua Kampus Gelar Wisuda, Kawasan Dinoyo Kota Malang Macet Parah (Lia, 2016), berita tersebut menjelaskan mengenai kemacetan di Kota Malang yang disebabkan oleh adanya kegiatan yang terjadi pada beberapa daerah. Tidak banyak masyarakat yang mengetahui akan kegiatan yang sedang terjadi, sehingga membuat mereka yang beraktifitas melalui jalan tersebut terjebak macet. Hal - hal seperti ini dapat dihindarkan jika saja masyarakat mendapatkan berita atau informasi terlebih dahulu mengenai kegiatan yang akan berlangsung. Sehingga kegiatan pun dapat berjalan dengan lancar dan tidak ada aktifitas dari masyarakat yang terganggu.

Salah satu solusi untuk menyelesaikan permasalahan tersebut adalah dengan membangun komunikasi yang baik antar pemerintah dengan masyarakat kota. Oleh karena itu diperlukan adanya pembangunan sistem yang dapat menjadi sarana untuk mewujudkan solusi tersebut. Sebuah sistem pengirim notifikasi dapat menjadi pilihan sebagai sarana bagi pemerintah menyampaikan informasi atau berita mengenai sebuah kota kepada masyarakatnya. Masyarakat juga akan mendapatkan keuntungan dari berita atau informasi yang diterima secara cepat dan akurat dari pemerintahan kotanya.

Push notification system adalah sebuah sistem yang menyediakan layanan pengiriman pesan, membuat penggunanya dapat mengirim pesan kepada satu atau lebih pengguna lainnya. Beberapa contoh penggunaan sistem pengirim notifikasi ialah seperti aplikasi chating dan sistem sensor gunung merapi. Dalam membangun sebuah sistem diperlukan adanya tahap - tahap seperti melakukan analisis kebutuhan dari sistem, merancang arsitektur dari sistem, melakukan implementasi kemudian dilakukan pengujian pada sistem yang telah dibangun. Dalam tahap perancangan sistem, salah satu yang dibutuhkan ialah model arsitektural dari sistem yang ingin dibangun. Model arsitektural menjelaskan bagaimana komunikasi antar element dilakukan, tersusun atas metode dan protokol (Colouris et al, 2012). Untuk membangun sebuah sistem yang menyediakan layanan pengiriman pesan untuk banyak pengguna dengan memperhatikan kecepatan dan ketersedian pesan sampai kepada penerimanya. Maka metode yang dapat menyelesaikan permasalahan tersebut berdasarkan metode yang telah ada ialah metode *publish – subscribe*. *Publish - subscribe*

arsitektur adalah sebuah paradigma grup komunikasi dimana pengkonsumsi informasi disebut *subscriber* dan sumber disebut *publisher* dengan ketertarikan yang sama saling terhubung oleh *broker* (Pongthawornkamol et all, 2007). *Subscriber* adalah penerima dari informasi, dan mereka sebelumnya telah mengungkapkan ketertarikan dalam satu atau lebih kelas dari informasi, sehingga hanya pesan berisi informasi yang menarik pilihan mereka yang akan diterima. *Broker* bertindak sebagai manajer, menjadi perantara dalam pengiriman pesan dan mengontrol tujuan pesan dengan melihat pengguna yang berlangganan topik. Topik – topik tersebut dibuat oleh *publisher* dan setiap pesan yang dikirim oleh *publisher* akan diterima jika pengguna lain berlangganan kepada topik mereka. *Publish - subscribe* dipilih karena beberapa keunggulan dari metode tersebut, yaitu dalam hal skalabilitas, *time decoupling*, *space decoupling* dan *synchronization decoupling* (Eugster, 2003). Salah satu contohnya ialah dapat dilihat dari insfrakstruktur google yang menggunakan sistem ini sebagai insfrastruktur dari sistem penyebaran informasi, yang mana membutuhkan kemampuan untuk penyebaran informasi dalam skala besar dan cepat (G, 2010). Selain itu, *Publish - subscribe* merupakan metode komunikasi berdasarkan kejadian (Faruk et al, 2015), yang berarti mempunyai cara kerja yang sama dengan cara kerja pada sistem yang dibangun.

Permasalahan kedua yang harus diselesaikan adalah bagaimana cara memilih protokol yang diterapkan pada sistem, ada banyak protokol yang telah tersedia dan telah disahkan dalam fungsinya yaitu mengatur jalannya komunikasi di internet. Beberapa contoh protokol komunikasi secara real-time yang telah diakui keberadaannya oleh IETF dan mendukung metode *publish - subscribe* ialah protokol XMPP dan MQTT. XMPP (*Extensible Mesagging and Presence Protokol*) merupakan protokol yang telah terstandarisasi dan dapat bebas digunakan untuk melakukan komunikasi real-time berbasis text, suara maupun video dengan teknologi *extensible markup language*. Sedangkan MQTT (*Message Queue Telemetry Transport*) adalah protokol yang dibuat khusus mendukung metode *publish subscribe*, protokol ini dikenal dengan pengiriman pesan yang mempunyai ukuran paket data dengan *low overhead* yang kecil (minimum 2 bytes) sehingga berefek pada konsumsi catu daya yang juga cukup kecil. Untuk itu peneliti melakukan *preliminary research* untuk membandingkan kedua protokol tersebut. Dari kesimpulan *preliminary research* yang didapat ialah MQTT mempunyai lebih sedikit *delay* dibanding XMPP. Namun protokol yang dipilih untuk diterapkan pada sistem ialah protokol XMPP, dikarenakan kebutuhan dari sistem yang membutuhkan beberapa fitur namun hanya disediakan oleh XMPP, seperti *register pengguna*, *login pengguna* dan *retrieve message*.

Untuk penggunaan perangkat yang digunakan pada pembangunan sistem yaitu pada sisi server berupa aplikasi server Ejabberd, dipilih karena Ejabberd menggunakan protokol XMPP untuk berkomunikasi dan mendukung banyak fitur yang disediakan oleh XMPP. Keunggulan lainnya ialah dalam hal kekuatan pada sistem aplikasinya, dan dapat digunakan dengan penggunaan berskala besar (ProcessOne, 2012). Aplikasi server Ejabberd dapat digunakan pada banyak

sistem operasi seperti windows, linux dan juga macinotsh. Perangkat yang digunakan pada sisi klien berupa telpon pintar berbasis android dikarenakan android adalah sistem operasi untuk *smartphone* yang paling banyak digunakan saat ini. Untuk menghubungkan protokol XMPP dengan aplikasi android berbasis java maka diperlukan library pendukung yang bernama Smack. Ada dua tipe dari aplikasi klient yang dibuat yaitu aplikasi untuk pengiriman dan menerima pesan.

Hasil dari penelitian ini ialah sebuah sistem pengirim notifikasi masal dengan menerapkan metode dari *publish - subscribe* berbasis protokol XMPP, harapannya dapat membantu pemerintah sebuah kota untuk menyampaikan informasi mengenai kota tersebut kepada masyarakatnya. Tujuan dari penelitian yaitu untuk memilih protokol yang cocok dengan kebutuhan sebuah sistem yang ingin dibangun dan penelitian ini juga ditujukan agar dapat melihat kinerja dari sistem yang dibangun sehingga harapannya dapat menjadi acuan dari peneliti lain dalam mengerjakan penelitiannya atau membantu seseorang yang ingin mengembangkan sebuah sistem pengirim notifikasi.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah diuraikan di atas, dapat dirumuskan beberapa masalah sebagai berikut :

1. Apa saja komponen yang digunakan pada pembangunan sistem?
2. Apa saja fungsi yang disediakan oleh sistem?
3. Bagaimana mengimplementasikan sistem pengiriman notifikasi dengan menggunakan metode *publish - subscribe* berbasis protokol XMPP?
4. Bagaimana agar pesan dapat diterima oleh seluruh pengguna yang berlangganan?
5. Bagaimana mengukur kinerja dari sistem yang dibangun?

1.3 Tujuan

Tujuan yang ingin dicapai dari pembuatan skripsi ini yaitu :

1. Untuk merancang sistem pengiriman notifikasi masal menggunakan metode *Publish Subscribe* berbasis protokol XMPP.
2. Untuk mengimplementasikan sistem pengiriman notifikasi masal menggunakan metode *Publish - Subscribe* berbasis protokol XMPP.
3. Untuk mengirimkan pesan keseluruh pengguna yang berlangganan
4. Untuk megukur kinerja dari sistem yang dibangun.

1.4 Manfaat

Penelitian pada skripsi ini diharapkan dapat memberikan manfaat dan berguna bagi pembaca dan penulis. Adapun manfaat yang penulis harapkan adalah:

1.4.1 Bagi Penulis

1. Menambah ilmu dan pengalaman dalam merancang dan membangun sebuah sistem pengirim notifikasi
2. Memahami lebih dalam mengenai komunikasi di internet khususnya mengenai metode *Publish - subscribe* dan protokol XMPP sebagai protokol real - time komunikasi.

1.4.2 Bagi Pembaca

1. Solusi dalam pembangunan sebuah sistem yang berfungsi sebagai layanan penyedia informasi kepada masyarakat oleh pemerintah sebuah kota.
2. Mendapatkan wawasan mengenai metode *publish subscribe* yang didukung oleh protokol XMPP dan diterapkan pada perangkat android.
3. Dapat menjadi referensi atau acuan untuk penelitian selanjutnya atau pengembangan sebuah sistem pengirim notifikasi

1.5 Batasan masalah

Agar permasalahan yang dirumuskan dapat lebih terfokus, maka pada penelitian ini dibatasi dalam hal :

1. Sistem yang dibuat hanya dapat mengirimkan pesan melalui aplikasi pengirim dan hanya akan diterima oleh aplikasi penerima, sedangkan antar penerima tidak dapat saling terhubung satu dengan yang lainnya.
2. Sistem akan digunakan oleh aplikasi yang telah dibuat dengan android sebagai sistem operasinya.
3. Aplikasi hanya dapat mengirim dan menerima data berupa text dan data tersebut dibungkus dengan format json, berisikan informasi - informasi berdasarkan form pengisian yang telah disediakan.
4. Pengujian delay dengan banyak pengguna tidak dilakukan menggunakan aplikasi asli yang diterapkan pada android, namun menggunakan aplikasi baru yang dijalankan dengan command prompt pada sebuah komputer.

1.6 Sistematika penulisan

Sistematika penulisan skripsi ini sebagai berikut :

BAB 1 : Pendahuluan

Pada bab ini penulis memaparkan latar belakang mengapa penulis memilih topik ini, rumusan masalah tentang implementasi sistem pengirim notifikasi, batasan masalah untuk menfokuskan masalah penelitian, tujuan dan manfaat adanya penelitian ini, serta sistematika penulisan laporan skripsi.

BAB 2 : Landasan Kepustakan

Bab ini berisi tentang dasar teori yang digunakan untuk mendukung pengerjaan skripsi ini. Teori-teori tersebut antara lain membahas tentang metode *Publish – Subscribe*, protokol yang mendukung metode *Publish – Subscribe* yaitu *MQTT* dan *XMPP*, serta teori mengenai pengujian *delay* pada sistem.

BAB 3 : Metodologi Penelitian dan Perancangan

Bab ini membahas tentang metode yang digunakan dalam penelitian serta penulisan laporan penelitian. Metode tersebut terdiri dari studi literatur, analisa kebutuhan, perancangan sistem, implementasi, pengujian sistem dan analisa hasil pengujian, serta penarikan kesimpulan.

BAB 4 : Perancangan

Bab ini membahas tentang bagaimana sistem dibangun agar mampu menyediakan akses layanan *push notification* pada masyarakat sebuah kota. Sistem tersebut terdiri atas metode *Publish Subscribe* yang didukung oleh protokol XMPP untuk diterapkan pada aplikasi antarmuka pengguna berbasis android yang akan digunakan oleh masyarakat pada sebuah kota agar dapat menerima pemberitahuan mengenai informasi dari pemerintah kota tersebut. Bab ini juga berisi paparan kebutuhan fungsional dan rancangan aplikasi yang berperan sebagai antarmuka dengan pengguna.

BAB 5 : Implementasi

Bab ini membahas hasil rancangan dari analisa kebutuhan serta implementasi dari rancangan sistem dan aplikasi yang digunakan sebagai pengirim dan penerima data dari sistem.

BAB 6 : Pengujian dan Analisis

Bab ini berisi tentang proses dan hasil pengujian terhadap aplikasi dan sistem yang diimplementasikan. Pengujian yang dilakukan meliputi apakah sistem sudah dapat berjalan dengan baik dan menangani resiko koneksi yang buruk pada pengguna, resiko pesan yang diterima ketika sebelumnya pengguna offline, resiko pesan tidak sampai kepada penerima, pengukuran seberapa besar

kinerja pada sisi server dan klien, pengukuran seberapa besar delay yang terjadi saat pengiriman pesan berskala besar, serta apakah aplikasi sudah memenuhi kebutuhan fungsionalitas yang dibutuhkan.

BAB 2
LANDASAN KEPUSTAKAAN

Bab ini akan memberikan bahasan mengenai penelitian sebelumnya yang terkait dengan pembangunan sebuah sistem pengirim notifikasi secara *real – time* dan kajian pustaka yang mendukung pengerjaan penelitian ini. Kajian pustaka tersebut meliputi metode yang digunakan yaitu publish subscribe, protokol yang mendukung publish – subscribe yaitu XMPP, Ejabberd sebagai perangkat lunak untuk mendukung penggunaan protokol XMPP, library yang digunakan sebagai penghubung protocol XMPP dengan pemograman berbasis java dan kajian lainnya yang mendukung penelitian ini.

2.1 Kajian Pustaka

Kajian pustaka pada penelitian ini membahas tentang penelitian terkait penerapan sebuah informasi yang dikirimkan melalui notifikasi untuk tujuan pelaporan kereta api pada perangkat android. Yang mana judul dari penelitian sebelumnya adalah "Sistem Pelaporan dan Informasi Posisi Kereta Api Berbasis Global Positioning System (GPS) Pada Device Berbasis Android" (Adhitya, 2011). Penelitian ini berisi bahasan tentang perancangan dan implementasi sistem pelaporan informasi kereta api dengan menggunakan mekanisme web service untuk mengirimkan datanya kepada web server lalu disimpan pada database server yang mana nantinya data tersebut dapat diolah dan selanjutnya ditampilkan oleh aplikasi web dan mobile yang telah dibuat. Selain itu sistem ini juga melakukan integrasi dengan penggunaan dari teknologi *sms gateway* untuk dapat memberikan notifikasi kepada penggunanya. Sistem tersebut terdiri dari *client application* yang merupakan aplikasi yang berjalan di sisi *client* terdiri dari aplikasi berbasis android dan web yang berfungsi untuk menampilkan lokasi dari kereta, *service requester* yang merupakan *web service* yang menjalankan *request* dari *client*, dan *Web service* pada *server* yang menyediakan fungsi-fungsi pada *server* untuk melakukan pengolahan data dan penyimpanan data pada database server. Penelitian ini terdiri dari empat aplikasi yaitu aplikasi klien untuk mengupdate posisi kereta menggunakan GPS pada sebuah mobile, lalu aplikasi tersebut mengirimkan data lokasi kepada sebuah web service yang disediakan untuk menyimpan data lokasi tersebut kepada sebuah database, kemudian aplikasi kedua yaitu aplikasi mobile dan aplikasi ke tiga yaitu aplikasi web dapat meminta data tersebut kepada web service yang telah disediakan dengan teknologi XML-RPC agar dua aplikasi tersebut dapat menampilkan data tersebut dalam bentuk sebuah map, terakhir adalah aplikasi ke empat yang berguna untuk melakukan pengolahan data pada web service yang nantinya akan mengirimakan pesan berupa informasi mengenai pelaporan terhadap poisisi kereta menggunakan *sms gateway*. Penelitan ini bertujuan untuk mengirimkan lokasi dari sebuah kereta api kepada penggunanya dikarenakan masalah kecelakan yang sering terjadi pada jalur kereta api, dan sistem ini sudah dapat

memenuhi kebutuhan dari pengguna untuk dapat mengakses peta dan menampilkan posisi kereta api.

2.2 Teknologi Pengiriman

Teknologi pengiriman data pada jaringan internet dilihat dari sudut pandang pengguna pertama yang melakukan permintaan untuk berkomunikasi dapat dibagi menjadi dua tipe yaitu :

2.2.1 Teknologi Menarik atau Pull Technology

Pull technology adalah gaya dari komunikasi pada jaringan dimana yang melakukan permintaan pertama dari data adalah klien dan di respon oleh server. Banyak klien meminta data atau informasi dari server terpusat, seperti contohnya permintaan terhadap halaman HTTP sebuah website.

Metode yang umum digunakan dalam pull technology ialah model *klient-server*.

2.2.2 Teknologi Mendorong atau Push Technology

Push technology atau server push menggambarkan cara dari internet berkomunikasi dimana permintaan untuk sebuah transaksi di mulai oleh *publisher* atau server pusat.

Layanan push seringkali didasarkan pada preferensi informasi. Hal seperti ini disebut sebuah model *publish-subscribe*. Seorang klien "*subscribe*" yang disediakan oleh server, setiap kali konten baru tersedia oleh salah satu siaran, server mendorong informasi kepada klien.

2.2.2.1 *Publish – Subscribe*

Publish – subscribe arsitektur adalah sebuah paradigma grup komunikasi dimana pengkonsumsi informasi disebut *subscriber* dan sumber disebut *publisher* dengan ketertarikan yang sama saling terhubung oleh *publish - subscribe broker* (Pongthawornkamol et al, 2007). Dalam sistem *publish - subscribe*, *subscriber* menentukan karakteristik dari kertarikan mereka disebut *topic* pada *publish - subscribe* broker. Dengan informasi yang diterima dari *publishers*, *publish – subscribe broker* lalu menyaring dan mengirimkan informasi kepada *subscriber* yang memiliki kesamaan kertarikan dengan *topic* atau konten dari informasi. *Publish - subscribe* arsitektur menawarkan beberapa skalabilitas properti termasuk *time*, *space* dan synchronization decoupling (Eugster, 2003). Hal tersebut mengijinkan informasi untuk disebarluaskan dari *publishers* ke *subscribers* secara *asynchronously* tanpa perlu *subscriber* atau *publishers* menyadari titik akhir komunikasi lainnya. Sifat - sifat tersebut memungkinkan sistem *publish - subscribe* untuk menjadi tulang punggung komunikasi dalam skala besar dan jaringan yang dinamis.

2.3 Protokol Komunikasi

Dalam melakukan komunikasi dimanapun kita berada, sebuah aturan dibutuhkan bertujuan agar dua objek yang saling berkomunikasi dapat saling mengerti satu dengan yang lainnya. Seperti halnya sebuah bahasa yang digunakan untuk berbicara, jika seseorang tersebut berada pada suatu negara, maka agar dapat melakukan komunikasi dengan orang lainya, dibuat lah sebuah aturan bahasa untuk mempermudah orang yang ingin berkomunikasi satu dengan yang lainnya. Sama seperti aturan komunikasi yang terjadi pada internet, dibutuhkannya sebuah aturan atau protokol yang telah saling disepakati oleh kedua objek yang akan berkomunikasi. Seperti yang telah dijelaskan, Protokol komunikasi adalah suatu sistem aturan yang memungkinkan dua atau lebih entitas dari sistem komunikasi untuk mengirimkan informasi melalui segala jenis variasi dari kuantitas fisik. Ini adalah aturan atau standar yang mendefinisikan sintaks, semantik dan sinkronisasi dari komunikasi dan kemungkinan metode pemulihan kesalahan. Protokol dapat digunakan oleh hardware, software, atau kombinasi keduanya.

Sistem berkomunikasi menggunakan format yang didefinisikan dengan baik untuk bertukar pesan. Setiap pesan memiliki makna yang tepat dimaksudkan untuk mendapatkan respon dari berbagai kemungkinan tanggapan sebelum ditentukan untuk situasi tertentu. Perilaku tertentu biasanya independen tentang bagaimana itu harus dilaksanakan. Protokol komunikasi harus disepakati oleh pihak-pihak yang terlibat. Untuk mencapai kesepakatan, protokol dapat dikembangkan menjadi standar teknis.

Mendapatkan data di dalam jaringan hanya sebagian dari masalah yang dihadapi oleh protokol. Data yang diterima harus dievaluasi sepanjang percakapan, sehingga protokol harus menentukan aturan yang menjelaskan percakapan tersebut. Pesan yang dikirimkan dan diterima dalam sebuah system komunikasi dilakukan untuk membangun sebuah komunikasi. Oleh karena itu protokol harus menentukan aturan yang mengatur transmisi. Dalam penelitian ini, ada dua jenis protokol yang akan digunakan yaitu :

2.3.1 Protokol XMPP

Mobile media sharing dapat dibagi kedalam 2 masalah umum: *file transfer*, *content repository management* dan *mobile collaboration* (Sollner, 2009). Dari proyek mengenai *mobile* sebelumnya telah menunjukan bahwa XMPP adalah sebuah protokol untuk *messaging* dan *presence*, cocok untuk tujuan kolaborasi bergerak.

The Extensible Messenging and Presence Protocol (XMPP) adalah sebuah implementasi dari model *Internet Engineering Task Force* (IETF) untuk pendekatan *real-time instant messaging* dan *presence.* XMPP berkembang menjadi protokol yang digunakan dalam bidang pesan berorientasi perangkat tengah sehingga dapat digunakan oleh lebih banyak *domain* (Sollner, 2009). Mekanisme keamanan yang di bangun oleh XMPP menjadi salah satu kunci

kekuatan dari protokol tersebut. Selain itu, ancaman keamanan tertentu dapat diatasi dengan melakukan pemeriksaan identitas otomatis pengguna yang terhubung menggunakan protokol *dialback*. Banyak implementasi dari XMPP yang telah ada saat ini bersifat terbuka.

Asal usul dari XMPP terletak pada proyek Jabber yang dibentuk pada tahun 1998. IETF membentuk sebuah group pekerja yang diberi nama XMPP pada tahun 2002. Dan menghasilkan empat spesifikasi (RFC 3920, RFC 3921, RFC 3922, RFC 3923) yang telah di setujui oleh IESG sebagai standar yang dianjurkan pada 2004. Pada tahun 2011, RFC 3920 dan RFC 3921 digantikan oleh RFC 6120 dan RFC 6121, lalu RFC 6122 menentukan format alamat XMPP. Pada 2015, RFC 6122 digantikan oleh RFC 7622. Selain protokol inti standar di IETF, para yayasan standar XMPP (sebelumnya Jabber Software Foundation) aktif dalam mengembangkan ekstensi XMPP terbuka.

Dengan sifatnya yang bebas dan terbuka, XMPP dirancang untuk menjadi protokol yang *extensible*, yang berarti menurunkan sehingga penggunaannya menjadi lebih luas. Banyak rancangan dari sebuah protokol komunikasi baru yang dibuat oleh para pengembang diturunkan dari protokol XMPP, seperti protokol untuk sistem *publish - subscribe*, *VoIP*, video, pengiriman file, *game*, *Internet of Things* (IOT) aplikasi seperti *grid* cerdas dan layanan jaringan sosial. Sehingga siapa pun dapat mengimplementasikan layanan XMPP dan beroperasi dengan implementasi organisasi lain. Karena XMPP adalah sebuah protokol terbuka, implementasi dapat dikembangkan menggunakan lisensi perangkat lunak, meskipun banyak (server, klien, dan implementasi library) didistribusikan sebagai perangkat lunak bebas dan open source, tetapi implementasi perangkat bebas dan perangkat lunak yang komersial juga masih banyak.

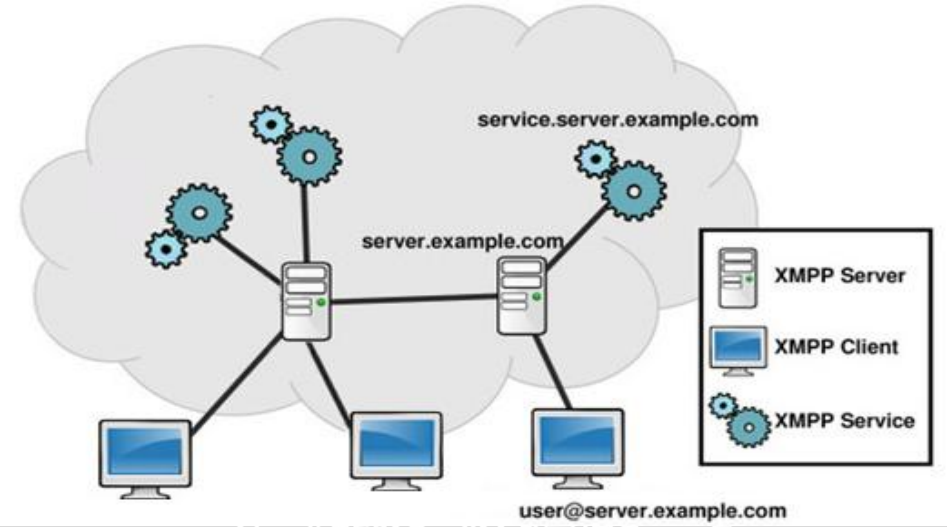
Dalam setiap hal pasti memiliki kelebihan dan kekurangan. Untuk protokol XMPP ada beberapa keuntungan jika digunakan sebagai bagian dari sistem yang dibuat, yaitu:

1. *Decentralization*, arsitektur dari jaringan XMPP mirip dengan arsitektur email. Siapapun dapat menjalankan server XMPP yang mereke masing - masing dan tidak ada server terpusat.
2. *Open standards*, IETF (Internet Engineering Task Force) meresmikan XMPP sebagai instant messaging and presence technology yang diakui dibawah nama XMPP. Tidak ada royalty atau membutuhkan izin untuk mengimplementasikan spesifikasi ini.
3. *History*, XMPP technology telah digunakan sejak 1999. Banyak implementasi dari standart XMPP yang tersedia untuk klien, server, component dan library program.
4. *Security*, XMPP server dapat diisolasi, seperti untuk sebuah intranet perushaan dan authentication keamanan (SASL) dan enkripsi (TLS) telah dibangun didalam spesifikasi utama dari XMPP.

5. *Flexibility*, fungsionalitas khusus dapat dibangun di atas dari XMPP. Untuk menjaga interoperability, extension umum telah diatur oleh XMPP Standards Foundation .

Feature dari XMPP seperti *federation across domain*, *publish-subscribe*, *authentication* dan *security for mobile endpoint* digunakan untuk mengimplementasikan *Internet of Things*.

Gambaran umum dari cara kerja menggunakan protokol XMPP dapat dilihat pada Gambar 2.1.



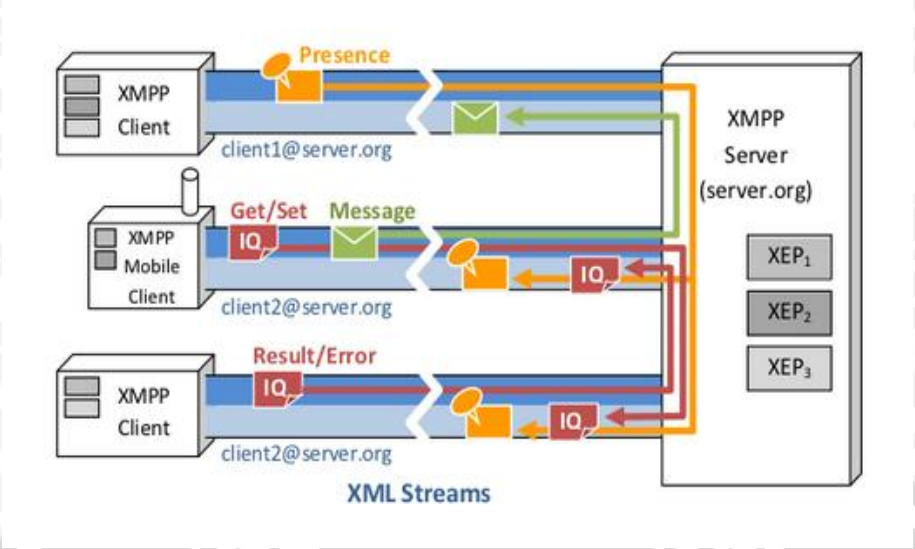
Gambar 2.1 Arsitektur Protokol XMPP

(Sollner, 2009)

Pada gambaran arsitektur sistem yang menggunakan protokol XMPP sebagai cara sistem tersebut berkomunikasi, maka sedikitnya ada dua komponen yang digunakan yaitu serve dank klien. Server yang menerapkan protokol dari XMPP akan memiliki fitur berupa layanan yang telah diatur penggunaanya oleh protokol tersebut. Layanan seperti *chat* atau *publish – subscribe*, dan lainnya yang akan dijalankan pada sebuah server sehingga klien dapat menggunakannya. Server dengan memiliki satu buah domain dapat menjalankan lebih dari satu layanan yang diimplementasikan. Layanan tersebut akan dibedakan berdasarkan nama dari *subdomain.* Seperti *domain* server.exampe.com maka nama dari *subdomain* layanannya ialah *chat.server.example.com* dan *pubsub.server.example.com*. Untuk email dari pengguna, XMPP menggunakan format nama dan ditambah nama domain pada bagian belakang, seperti pengguna@server.example.com.

2.3.1.1 Format XMPP

XMPP penggunas berkomunikasi melalui sebuah alamat XMPP yang mirip dengan alamat sebuah email. Alamat XMPP terdiri dari sebuah *usernam* dan XMPP server domain. Contohnya *username@domain*.



Gambar 2.2 Skenario Message pada Arsitektur XMPP

Format dari pesan XMPP dapat dilihat pada Gambar 2.2, didasarkan pada XML. Ada tiga type dari pesan: XML Stream, XML Stanza, dan Stream Authentication. XML Stream adalah pembungkus dari XML Stanza, digunakan untuk menukarkan element XML antara dua entitas di internet. Stream Authentication berguna menghubungkan ke TLS (*Transport Layer Security*), SASL(*Simple Authentication dan Security Layer*) *authentication* dan *encryption*. Semua pesan dikirimkan antara klien dan server atau antara server dan server yang terhubung melalui XML Stream. XML Stream diawali dengan tag pembuka *<stream>* dan diakhir dengan close tag *</stream>.* XML Stanza dimulai dengan anak tanda *<message/>, <presence/>* dan *<iq/>.*

Info Query

Info Query atau *<iq/>* adalah sebuah mekanisme *request - response*, membangun dasar dialog *query* antara dua XMPP entitas. IQ Stanza dapat memberikan informasi berupa type status sepert *get* (permintaan terhadap sebuah informasi), *set* (memberikan informasi), *result* (hasil dari sebuah permintaan yang berisi informasi), or *error* (hasil dari permintaan yang menyatakan kegagalan). Contoh dari standza IQ :

<iq from =’romeo@domain . com/ yard’ to=’juliet@domain .com/balcony’
id=’m940AE76’ type =’get’>
<query xmlns =’http: //jabber . org/protocol/disco # info’/>
</iq >

Dan penerima normalnya akan membalas dengan hasil IQ yang bersesuaian (id atribut yang sama).

<iq from =’juliet@domain.com /balcony’ to=’romeo@domain.com/yard’ id=’m940AE76’ type =’result’>
<query xmlns =’http: // jabber . org/protocol /disco#info’>
</query >
</iq >

Presence

Informasi *presence* atau kehadiran umumnya dianggap sebagai *multicast information* dari pengguna pada jaringan XMPP. Secara khusus, *multicast* berarti pengiriman kesemua alamat yang telah berlangganan kehadiran terbaru pengguna dengan memiliki pengguna tertentu. Dalam scenario pesan instant, sebuah *presence standza* akan berisi semua informasi tentang konteks pengguna saat ini seperti *availability* (*online, offline, away, busy*, dll), status pesan pengguna, lokasi, atau data lain yang masih terkait. Contoh dari *standza presence* :

<presence from =’juliet@domain .com/ balcony’ type =’available’/>

Meesage

Sebuah pesan dapat dilhat sebagaiman mendorong data dari satu entitas dari jaringan XMPP ke entitas lain. Dalam skenario pesan singkat, ini biasanya berhubungan dengan sebuah pesan yang dikirim ke pengguna lain dengan bagian dari tag *body* mirip dengan sebuah email. Tapi pesan dapat digunakan juga dalam sekenario lain, contohnya ketika memberikan notifikasi mengenai sebuah event kepada sebuah entitas di XMPP, hanya untuk entitas yang telah berlangganan. Contoh *standza message* ialah :

<message from =’juliet@domain.com/balcony’ to=’romeo@domain.com/yard’ id=’m940AE74’>
<body> Hai </body>
</ message >

Perhatikan, bagaimana nomor unik acak yang ada dari pengirim dan penerima digambarkan menjadi atribut dari tag message. *Payload* sebenarnya dari sebuah tag message adalah tidak didapat dari text biasa tapi bisa lebih berupa konten *wellformed XML.*

2.3.1.2 XMPP Extension

Disisi lain dari apa yang telah dijelaskan, komunitas XMPP mengembangkan spesifikasi lebih lanjut untuk standarisasi komunikasi XMPP di berbagai area seperti *file sharing, collaborative drawing, event publication/subscription, avatars* dan banyak lagi. Setiap standar yang di terbitkan akan dipanggil sebagai

XMPP Extension Protocol (XEP) yang berjalan melalui proses standarisasi yang disepakati bersama. XEP-0060 dan XEP-0248 menjelaskan mengenai model dari *publish – subscribe* pada protokol turunan XMPP.

Publish Subscribe (XEP-0060)

Spesifikasi XEP-0060 menjelaskan mengenai sebuah protokol turunan XMPP yang berisikan fungsi umum dari metode *publish-subscribe*. Protokol memungkinkan entitas XMPP untuk membuat *nodes* (topik) dan mengirimkan atau mem*publish* informasi ke sebuah nodes (topik) pada sebuah layanan *publish-subscribe*. Tidak hanya itu namun entitas XMPP juga dapat berlanganan atau men*subscribe* sebuah nodes (topik) untuk mendapat sebuah notifikasi jika ada sebuah *event.* Event tersebut dapat berisi sebuah data text atau hanya data kosongan yang berguna sebagai sinyal.

Oleh karena itu *publish-subscribe* menganut *classic Observer design pattern* dan dapat berfungsi sebagai dasar untuk berbagai macam aplikasi, termasuk *news feed, content syndication, rich presence, geolocation, workflow systems, network management systems,* dan banyak aplikasi lain yang membutuhkan standart notifikasi berbasis *event*.

Publish Subscribe Lanjutan (XEP-0248)

Spesifikasi XEP-0248 juga menjelaskan mengenai sebuah protokol turunan XMPP mengenai metode dari *publish-subscribe*, namun aturan ini merupakan aturan tambahan dari XEP-0060. Perbedaan antara keduanya dapat dilihat pada sebuah node yang dijalankan, pada XEP-0060 hanya mengatur mengenai left node sedangkan pada XEP-0248 mengatur mengenai collection node. Left node merupakan satu buah node, sedangkan collection node dapat terkumpul dari left node yang didalamnya terdapat node, dst.

2.3.2 Protokol MQTT

MQTT (Message Queuing Telemetry Transport) adalah sebuah protokol pesan ringan yang didasarkan pada sistem *publish subscribe* yang digunakan diatas protokol TCP/IP. Dr Andi Standford dari IBM dan Arlen Niper dari Arcom(Eurotech) adalah pencipta dari protokol ini pada tahun 1999. MQTT merupakan protokol yang digunakan sebagai penghubung dari *machine to machine* (M2M) berbasis bebas dan terbuka. Hal ini dirancang untuk koneksi dengan lokasi terpencil di mana *small code foootpring* dibutuhkan atau keterbatasan *bandwidth,* dengan latency tinggi atau berjalan pada jaringan yang tidak dapat diandalkan. MQTT juga sangat ideal digunakan dalam keadaan perangkat yang terhubung dalam sebuah sistem membutuhkah daya baterai yang kecil.

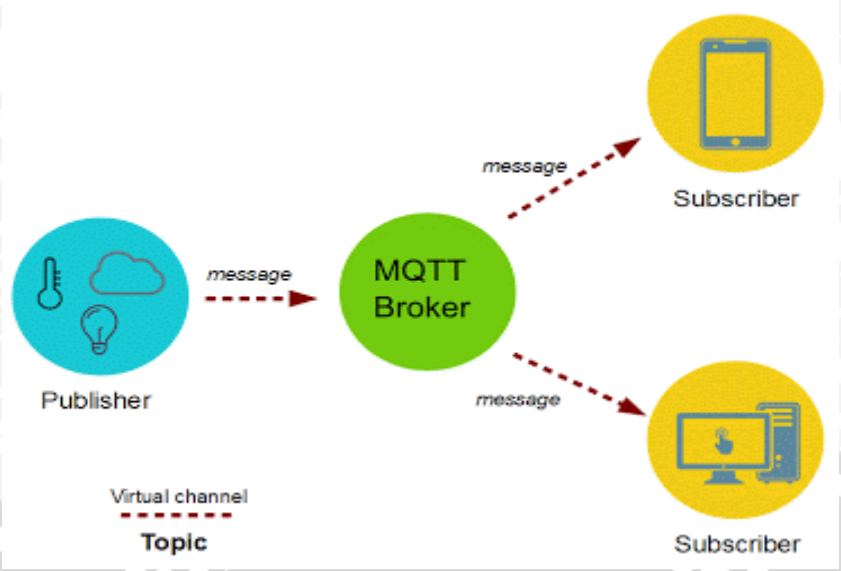
Protokol MQTT terkenal akan kegunaannya dalam mendukung model pengiriman data yang bersifat *publish subscribe*. Pola pesan dari *publish - subscribe* membutuhkan broker. Broker bertanggung jawab untuk mendistribusikan pesan kepada klien yang tertarik berdasarkan pada topik pesan. *Subscriber* adalah penerima dari informasi, dan mereka sebelumnya telah mengungkapkan ketertarikan dalam satu atau lebih kelas dari informasi, sehingga hanya pesan berisi informasi yang menarik pilihan mereka yang akan diterima. Topik – topik tersebut dibuat oleh *publisher* dan setiap pesan yang dikirim oleh *publisher* akan diterima jika pengguna lain berlangganan kepada topik mereka.

- MQTT Klien
  - *Publisher*, klien yang bertugas menjadi sumber dari informasi. Umumnya sebuah sensor.
  - *Subscriber*, klien yang bertugas sebagai penerima informasi. Umumnya komputer dan telepon pintar.
- MQTT Broker

  Ada beberapa broker MQTT tersedia seperti ActiveMQ, Apollo, HiveMQ, IBM Message Sight, JoramMQ, mosquitto, RabbitMQ, Solace Pesan Router, dan VerneMQ. Mereka berbeda dalam fitur dan beberapa dari mereka mengimplementasikan fitur tambahan di atas fungsi MQTT standar.
- MQTT methods

  MQTT mendefinisikan metode untuk menunjukkan tindakan yang diinginkan dalam mengidentifikasi sumber daya. Apa yang sumber daya ingin tampilkan, apakah data yang sudah ada atau data yang dihasilkan secara dinamis, tergantung pada pelaksanaan server. Seringkali, sumber daya menyesuaikan dengan file atau output dari sebuah eksekusi yang berada di server.
  - Connect, menunggu untuk koneksi yang akan dibentuk dengan server.
  - Disconnect, menunggu klien MQTT untuk menyelesaikan sebuah pekerjaan yang harus dilakukan dan setelah itu TCP/IP session disconect
  - *Subscribe*, menunggu selesainya metode berlangganan atau berhenti berlangganan.
  - *Unsubscribe*, permintaan kepada server untuk klien berhenti berlangganan dari satu atau lebih topik.
  - *Publish*, segera kembali ke thread aplikasi setelah melewati permintaan ke klien MQTT.

Gambaran umum dari cara kerja menggunakan protokol MQTT dapat dilihat pada Gambar 2.3. Dapat dilihat bahwa komponen yang digunakan oleh MQTT ada tiga yaitu *publisher*, *subscriber* dan broker. Mirip seperti komponen pada sebuah metode *publish – subscribe* dikarena MQTT adalah protokol yang khusus dikembangkan untuk mengimplementasikan metode tersebut.



Gambar 2.3 Arsitektur Protokol MQTT

Untuk menjamin sebuah pesan agar terkirim kepada klien, MQTT mempunyai 3 level *Quality of Service* (QoS), yaitu:

1. QoS 0 : Pesan hanya akan terkirim sekali, pengirim tidak perduli apakah pesan sampai atau tidak kepada penerima
2. QoS 1 : Pesan hanya akan dikirim sekali, namun pengirim akan menunggu jawaban dari penerima apakah pesan diterima atau tidak, jika penerima tidak memberikan sinyal pesan diterima kepada pengirim maka pesan akan dikirim ulang dengan adanya jarak waktu pengiriman dan pesan yang pertama akan ditimpa oleh pesan kedua, begitu seterusnya.
3. QoS 2 : Mirip dengan QoS 1 namun pesan pertama tidak akan ditimpa dengan pesan kedua melainkan dibuat sebuah antrian pesan.

2.4 Quality of Service (QoS)

Quality of service adalah kemampuan jaringan untuk menyediakan layanan yang lebih baik lagi bagi trafik yang melewatinya.(A.Kevin, 2011) QoS terdiri dari delay, jitter dan bandwidth. Delay adalah waktu tunda suatu paket yang diakibatkan oleh proses transmisi dari suatu titik ke titik lain yang menjadi tujuan. Jitter merupakan variasi delay yang terjadi pada jaringan IP, dan bandwith adalah lebar frekuensi yang digunakan oleh sinyal dalam mengirimkan paket data.

Delay

Delay adalah waktu tunda suatu paket yang diakibatkan oleh proses transmisi dari suatu titik ke titik lain yang menjadi tujuan. Delay dalam jaringan dapat digolongkan sebagai berikut :

1. Paketisasi Delay, Delay yang disebabkan oleh waktu yang diperlukan untuk proses pembentukan paket IP dari pengguna, yang harus melalui

proses enkapsulasi 7 lapisan OSI atau 4 lapisan TCP/IP. Delay ini hanya terjadi 1 kali di sumber informasi

2. Queueing Delay, Delay ini disebabkan oleh waktu proses yang diperlukan oleh router di dalam menangani transmisi paket di sepanjang jaringan. Umumnya delay ini sangat kecil, kurang lebih sekitar 100ms.
3. Delay Propagasi, Proses perjalanan informasi selama media transmisi, misalnya coaxial atau tembaga, menyebabkan delay yang disebut delay propagasi.

Umumnya delay didapatkan tidak hanya dalam sekali percobaan, sehingga diperlukannya rumus untuk mendapatkan rata - rata delay :

$$\bar{} Delay = \sum \quad Delay \div \sum \quad Paket\ diterima \text{ (1)}$$

Kategori *Latensi*	Besar *delay (ms)*	Indeks
Sangat Bagus	< 150 ms	4
Bagus	150 ms s/d 300 ms	3
Sedang	300 ms s/d 450 ms	2
Jelek	> 450 ms	1

Gambar 2.4 Tabel Kategori Delay

(Tiphon, 1999)

Pada Gambar 2.4 adalah standarisasi yang telah dikeluarkan oleh Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) mengenai kelayakan dari delay pada sebuah protokol komunikasi *realtime*

BAB 3
METODOLOGI PENELITIAN

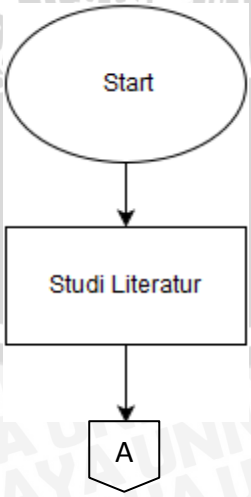
3.1 Tinjauan umum

Pesan singkat ialah sebuah aplikasi yang dibuat bertujuan untuk mengirimkan sebuah pesan berupa text kepada penerimanya. Pesan singkat dapat dikategorikan sebagai sistem notifikasi. Di zaman yang maju seperti sekarang ini, sebuah sistem notifikasi sudah menjadi layanan penting dalam jaringan komputer. Banyak aplikasi berbasis notifikasi yang telah dibuat pada perangkat komputer yang kita gunakan. Dibutuhkannya metode dan protokol dalam sebuah sistem untuk dapat saling terhubung antar komputer agar data terkirim dan diterima. Metode dan jenis – jenis dari protokol notifikasi sudah beragam, masing - masing dibuat sesuai kebutuhan yang berbeda.

Penelitian ini bertipe implementasi. Dimana proses implementasi sistem dimulai dari pengumpulan data melalui studi literatur, analisis kebutuhan, perancangan, kemudian implementasi dan pengujian lalu diakhiri dengan anilisis hasil pengujian yang setelah itu dapat diambil kesimpulan dan saran. Hasil dari penelitian ini ialah sebuah sistem pengirim notifikasi yang dapat digunakan untuk mengirimkan pesan kepada setiap penggunanya yang telah berlangganan dari sebuah topik informasi.

3.2 Strategi dan rancangan penelitian

Metodologi penelitian berisi tentang tata cara yang digunakan untuk melakukan penelitian atau dapat juga diartikan tindakan yang akan digunakan untuk teknik pemilihan rancangan di dalam penelitian ini. Adapun tahapan penelitian yang ditunjukkan pada diagram 3.1.



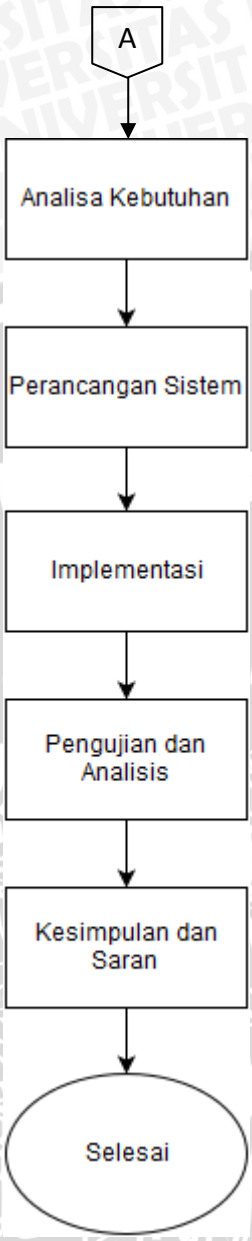


Diagram 3.1 Tahapan Penelitian

3.3 Studi Literatur

Dalam pembuatan penelitian ini, penulis membutuhkan literatur sebagai acuan untuk merealisasikan implementasi proses pengiriman notifikasi masal dengan metode *publish - subscribe* menggunakan protokol XMPP dalam pertukaran datanya. Informasi dan pustaka yang memiliki keterkaitan dengan penelitian ini didapat dari jurnal, buku, dan dosen. Adapun teori - teori pendukung tersebut meliputi :

1. *Push Technology*

2. Protokol *Real-Time* Komunikasi
3. *Publish - Subscribe*
4. XMPP khususnya XEP-0060
5. Ejabberd server dan Smack library
6. QoS(*Quality of Service*) jaringan

3.4 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk menentukan apa saja yang mampu dilakukan oleh sistem dan aplikasi yang akan dirancang. Sistem yang dimaksud adalah sistem pengirim notifikasi yang dibangun dengan menerapkan metode *publish - subscribe* yang didukung oleh protokol XMPP. Sementara itu, aplikasi yang dimaksud adalah aplikasi dari pengguna yang betugas sebagai pengirim dan penerima pesan. Kebutuhan fungsional sistem antara lain adalah :

- Memiliki metode dan protokol dari sistem yang dibangun
- Memiliki perangkat untuk pembangunan sistem yang mendukung mengenai metode dan protokol yang digunakan pada sistem
- Memiliki perangkat yang akan digunakan untuk pengujian sistem

3.4.1 Analisis Kebutuhan Perangkat Lunak

Spesifikasi kebutuhan perangkat lunak yang digunakan pada sistem adalah sebagai berikut :

1. Aplikasi server Ejabberd versi 16.06 sebagai server yang akan berfungsi sebagai perantara antar pengguna.
2. Sistem operasi Windows dan Android yang digunakan pada sistem.
3. Library Smack sebagai library untuk menggunakan layanan dari protokol XMPP dengan pemograman berbasis java.
4. Android Studio untuk membuat program aplikasi berbasis java dan xml.
5. Perfomance Monitor sebagai aplikasi untuk mencatat dan mengawasi kinerja dari sistem pada sisi server.

3.4.2 Analisis Kebutuhan Perangkat Keras

Perangkat keras yang dibutuhkan adalah perangkat keras yang akan digunakan untuk instalasi perangkat lunak, sehingga kebutuhannya mengikuti kebutuhan dari perangkat lunak. Spesifikasi kebutuhan perangkat keras yang digunakan sebagai berikut.

1. Laptop
    - CPU internal Core I3
    - RAM 4 GB
2. Telepon pintar
    - RAM 1 GB
    - Memory Internal 8GB

3.5 Perancangan

Perancangan yang dilakukan ialah merancang arsitektur dari sistem yang akan dibangun, seperti hubungan antar komponen pada sistem dan komponen apa saja yang digunakan. Kemudian memaparkan fungsi - fungsi apa saja yang akan dibangun pada sistem.

3.6 Implementasi

Implementasi dilakukan untuk menerapkan perancangan yang telah dibuat kepada perangkat aslinya.

3.6.1 Ruang Lingkup Penelitian

Ruang lingkup dari penelitian ini yaitu pembuatan sistem notifikasi dengan menggunakan metode *publish subscribe* berbasis protokol XMPP. Notifikasi yang akan dikirim dilakukan oleh klien yang bertugas sebagai *publisher*, sebelum dilakukan pengiriman, device akan melakukan pengecheckan terlebih dahulu terhadap ketersedian internet dan apakah pengguna telah tersambung keserver. Setelah itu barulah pesan akan dikirim ke broker yang sebelumnya sudah di jalankan pada komputer peneliti. Tidak hanya sampai di broker, namun alamat yang dikirimkan oleh *publisher* ditujukan sampai ke sebuah node. Di node tersebutlah pesan yang telah diterima akan diproses terlebih dahulu, dan selanjutnya pesan akan dikirim ke semua daftar *subscriber* pada node tersebut. Klien yang bertugas menjadi *subscriber* dapat masuk dalam daftar pada node tersebut dengan cara menjalankan fungsi *subscribe* yang terdapat pada aplikasi *subscriber*. Jika *broker* melihat bahwa *subscriber* dengan email tersebut layak untuk diterima menjadi bagian dari pengguna node tersebut, maka broker akan mengirimkan balasan berupa tanda terima anggota. Sejak saat itu *subscriber* akan selalu dikirimkan pesan oleh sebuah node yang ia masuki.

Subscriber juga akan selalu mendapatkan message walaupun sebelumnya *subscriber* berada pada kondisi *offline*. Message akan diterima oleh *subscriber* ketika *subscriber* telah kembali online. Hal itu dikarenakan semua node pada broker telah dikonfigurasi untuk menyimpan semua message yang ada, sehingga ketika *subscriber* kembali online, *subscriber* akan melakukan permintaan pesan *offline* kepada broker, yang ditujukan kesemua node yang dirinya telah menjadi anggota. Semua pesan yang diterima oleh *subscriber* akan disimpan pada database di android yang telah peneliti buat. Sehingga jika dalam keadaan tidak terhubung ke serverpun pesan masih dapat dibuka.

3.7 Pengujian

Setelah semua implementasi dirasa selesai oleh peneliti, maka lanjut kepada tahap pengujian. Pengujian yang akan dilakukan oleh peneliti yaitu pengujian fungsional, lalu pengujian terhadap QoS jaringan yang digunakan, pengujian scalability untuk melihat peforma dari server saat berjalan, kemudian terakhir

pengujian availability untuk memastikan bahwa pesan dapat diterima oleh seluruh pengguna yang jumlahnya sebanyak 900 penguna.

3.7.1 Pengujian Fungsional

Pengujian fungsional dilakukan dengan menggunakan pengujian black box yang berguna untuk melihat apakah fungsi - fungsi dari aplikasi sudah dapat dijalankan dan sesuai dengan yang diinginkan.

3.7.2 Pengujian QoS Jaringan

Pengujian ini bertujuan untuk melihat delay yang terdapat pada jaringan yang digunakan oleh aplikasi pada sistem yang dibangun. Langkah yang dilakukan dalam melakukan pengujian delay yaitu dengan membuat program sederhana yang bertugas untuk mengirimkan dan menerima data. Program pengirim akan mencatat waktu saat pengiriman dilakukan dan program penerima akan mencatat berapa waktu saat pesan diterima. Peneliti akan mengambil selisih dari waktu pengiriman dan waktu pesan diterima sehingga mendapatkan sebuah waktu delay.

3.7.3 Pengujian Availability

Pada pengujian ini, yang akan dilihat ialah keberadaan dari pesan. Apakah pesan tersebut akan selalu didapat oleh pengguna yang bersama - sama sedang menggunakan aplikasi. Sehingga terlihat berapa persen message yang benar - benar sampai kepada penerima jika server digunakan sebanyak 900 pengguna.

3.7.4 Pengujian Scalability

Pengujian ini bertujuan untuk melihat kinerja dan kebutuhan pemakaian dari sisi klient dan server. Penggunaan memory dan processor sebagai parameter yang akan dimonitor kinerjanya pada sisi server sedangkan pada sisi klien dilakukan monitor terhadap konsumsi baterai dan konsumsi pemakaian akses internet untuk mengakses data.

3.8 Penarikan Kesimpulan dan Saran

Kesimpulan dan sarah akan menjadi tahapan terakhir dalam penelitian skripsi, tahapan ini akan dilakukan setelah pengujian selesai. Kesimpulan akan diberikan untuk merangkum seluruh hasil dari penelitian yang didapat. Saran akan diberikan jika peneliti merasa masih ada yang harus dikembangkan dari penelitian ini. Kedua hal ini akan menjadi referensi sebagai penelitian selanjutnya.

BAB 4
PERANCANGAN SISTEM

Sesuai latar belakang dan rumusan masalah yang telah disebutkan sebelumnya, sistem yang akan dibangun adalah sistem yang mampu membantu pengguna untuk mengirimkan pesan ke pada pengguna lainnya dengan menggunakan metode *publish - subscribe* yang didukung oleh protokol XMPP dalam format komunikasinya. Dua hal tersebut termasuk dalam kebutuhan umum dari sistem jika ingin membangun sebuah sistem pada jaringan (George et al, 2012).

Kemacetan yang disebabkan oleh adanya kegiatan yang sedang berlangsung dapat dihindari dengan adanya pemberitahuan dari pemerintah kepada masyarakatnya. Untuk itu dibutuhkan sistem pengirim notifikasi yang dapat menjadi sarana dalam mencapai hal tersebut. Sistem pengirim notifikasi yang akan dibuat akan digunakan untuk mengirimkan informasi - informasi mengenai kegiatan yang akan atau sedang terjadi pada suatu kota dari pemerintah kepada masyarakat.

Sistem ini akan menggunakan aplikasi berbasis android pada sisi penggunanya. Aplikasi dari pengguna yang dibuat oleh peneliti terbagi menjadi dua yaitu aplikasi yang bertugas untuk mengirimkan pesan sebagai sumber informasi yang akan digunakan oleh pemerintah disebut aplikasi *publisher* dan aplikasi yang bertugas untuk menerima pesan digunakan oleh masyarakat luas disebut aplikasi *subscriber*. Untuk menghubungkan antara pengguna yang terdapat pada sistem maka diperlukan sebuah server. Ejabberd digunakan sebagai aplikasi server bertugas seperti fungsi dari sebuah broker yaitu menerima dan menyaring pesan berdasarkan daftar dari pengguna yang berlangganan topik tertentu kemudian mengirimkannya kepada setiap penerima.

Aplikasi pengirim akan menampilkan daftar dari topik berupa kota - kota yang ada di Indonesia, topik tersebut akan digunakan sebagai tujuan dari pesan yang dikirim. Hal tersebut juga tersedia pada aplikasi penerima berguna untuk memilih topik apa saja yang diinginkan oleh penerima, sehingga penerima hanya akan mendapatkan notifikasi berisi informasi dari sebuah topik yang dipilih.

4.1 Arsitektur Sistem Notifikasi

Perancangan awal ialah merancang arsitektur dari sistem yang dibangun, seperti hubungan antar komponen pada sistem dan komponen apa saja yang akan digunakan.

Transmisi data pada penelitian ini dilakukan dengan menggunakan metode *publish - subscribe* untuk mendistribusikan informasi antar berbeda penerima, sehingga tidak perlu membuat metode baru dalam mengirimkan pesan. *Publish - subscribe* arsitektur adalah sebuah paradigma grup komunikasi dimana pengkonsumsi informasi disebut *subscriber* dan sumber disebut *publisher*

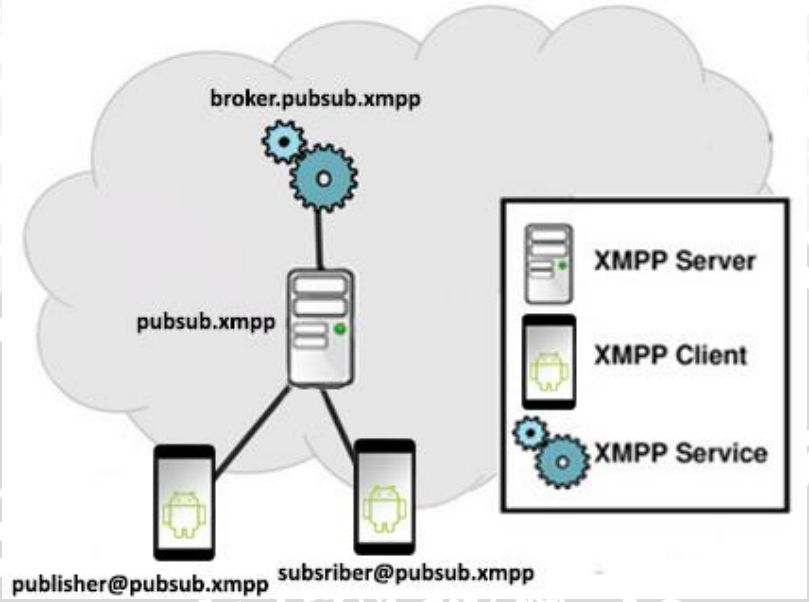
dengan ketertarikan yang sama saling terhubung oleh broker (Pongthawornkamol at all, 2007). *Publish - subscribe* dipilih karena beberapa keunggulan dari metode tersebut, yaitu scalability, *time decoupling*, *space decoupling* dan synchronization decoupling (Eugster, 2003). Salah satu contohnya ialah dapat dilihat dari insfrakstruktur google yang menggunakan sistem ini sebagai insfrastruktur dari sistem penyebaran informasi, yang mana membutuhkan kemampuan untuk penyebaran informasi dalam skala besar dan cepat (Li, 2010). Selain itu, *Publish Subscribe* merupakan metode komunikasi berdasarkan kejadian (Fajar at all, 2015), yang berarti mempunyai cara kerja yang sama dengan cara kerja pada sistem yang dibangun. *Subscriber* adalah penerima dari informasi, dan mereka sebelumnya telah mengungkapkan ketertarikan dalam satu atau lebih kelas dari informasi, sehingga hanya pesan berisi informasi yang menarik pilihan mereka yang akan diterima. *Broker* bertindak sebagai manajer, menjadi perantara dalam pengiriman pesan dan mengontrol tujuan pesan dengan melihat pengguna yang berlangganan topik. Topik – topik tersebut dibuat oleh *publisher* dan akan diterima jika pengguna lain berlangganan kepada topik mereka. Format topik diatur oleh sistem sehingga akan mudah untuk melakukan komunikasi jika terdapat perangkat baru yang menggunakan.

Protokol XMPP digunakan dalam mendukung penggunaan metode *publish - subscribe* dikarenakan banyaknya fitur yang tersedia. Beberapa fitur digunakan pada sistem pengirim notifikasi ini seperti *register pengguna* berguna untuk pendaftaran pengguna baru, *login pengguna* berguna sebagai proses pengecekan bila pengguna ingin menggunakan layanan sistem dan *retrieve message* yang dibutuhkan untuk mendapatkan kembali pesan yang tersimpan pada *broker*. Fitur tersebut dibutuhkan oleh sistem dikarenakan beberapa hal sebagai berikut :

1. Sistem yang ingin dibuat bertujuan untuk mengirimkan pesan mengenai sebuah kota dari pemerintah kepada masyarakat. Tidak semua petugas yang ada pada pemerintahan dapat melakukan hal tersebut, dikarenakan informasi yang dikirimkan seharusnya merupakan informasi yang benar - benar terjadi. Sehingga dibutuhkan beberapa orang yang dapat memegang tanggung jawab tersebut. Pengecekan pengguna diperlukan pada aplikasi pengirim agar hanya orang yang diberi tanggung jawab yang dapat menggunakannya.
2. Fungsi pendaftaran dan pengecekan pengguna berlaku pada aplikasi penerima. Fungsi tersebut bertujuan agar setiap pengguna dari aplikasi ini dapat mendaftarkan dirinya pada sistem, hal tesebut juga akan berguna ketika seorang pengguna ingin berpindah perangkat, aplikasi akan secara otomatis melakukan proses *subscribe* jika sejarah dari pengguna pernah berlangganan pada topik - topik tertentu.
3. Pengambilan pesan kembali pada aplikasi penerima berguna sebagai penanganan ketika adanya buruknya koneksi yang menyebabkan pengguna terputus dari server sehingga untuk memastikan pesan yang dikirim sampai

kepada penerima maka dibutuhkan fungsi dari pengambilan pesan ketika pengguna kembali terhubung ke server.

Arsitektur dari sistem yang akan dibangun dapat dilihat pada Gambar 4.1.

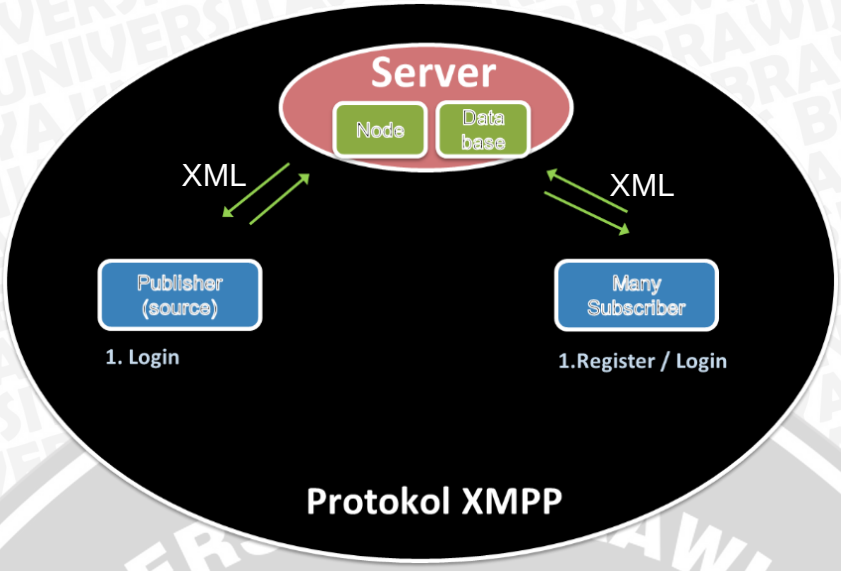


Gambar 4.1 Perancangan Sistem pada Arsitektur XMPP

Android akan digunakan sebagai perangkat pada sisi pengguna (pengirim dan penerima). Aplikasi yang digunakan sebagai pengirim dan penerima pesan ialah aplikasi yang dibuat oleh peneliti dengan menggunakan library dari Smack sebagai penyedia fungsi dari protokol XMPP pada pemograman berbasis Java. Aplikasi yang telah terinstall pada android akan terhubung ke server dengan ranah *pubsub.xmpp*.

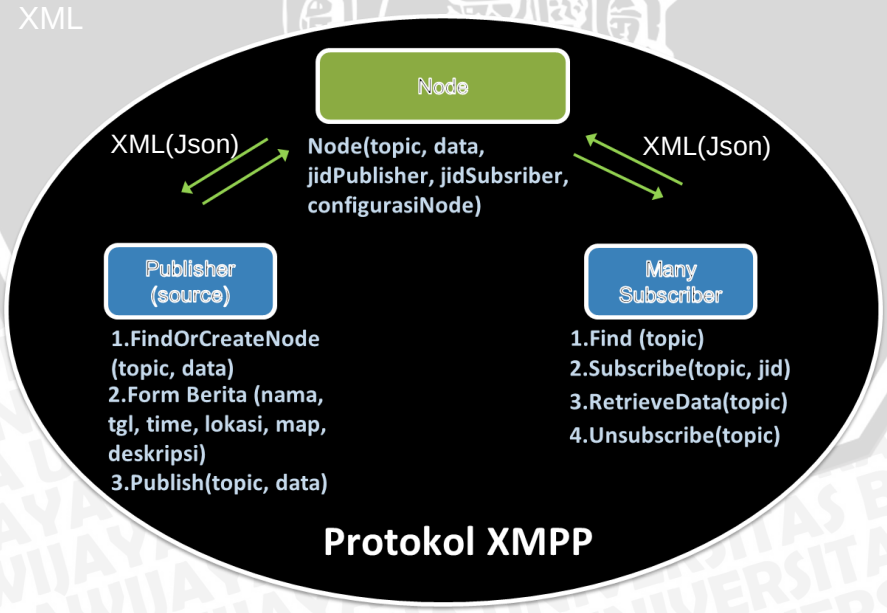
Server yang digunakan ialah Ejaberd, yang mana mengimplementasikan layanan dari *publish - subscribe* untuk dapat mengirimkan pesan kepada setiap pengguna. Ejabberd adalah aplikasi server yang bebas untuk digunakan, dimodifikasi dan disebarluaskan, dengan Erlang sebagai bahasa programannya. Ejabberd memiliki keunggulan dalam hal kekuatan pada sistem aplikasinya, lalu dapat digunakan dengan penggunaan berskala besar dan yang terpenting ialah dapat mendukung semua fitur yang disediakan oleh protokol XMPP (ProcessOne, 2012). Ejabberd menggunakan port 5280 sebagai port yang digunakan untuk mengakses halaman dari web admin oleh pengguna dan port 5222 untuk pengguna jika ingin menggunakan fitur yang disediakan oleh Ejabberd. Ejaberd dapat digunakan pada sebuah komputer dengan sistem operasi Windows, Linux dan Macintos.

Untuk perancangan arsitektur yang lebih detail, dapat dilihat pada gambar 4.2 :



Gambar 4.2 Sistem Sebelum Login

Aktor pada sistem terdiri dari klien yang bertugas sebagai *publisher* atau *subscriber* dan server. Pada Gambar 4.2 adalah awal mula klien terhubung keserver. Persyaratan awal untuk menggunakan sistem notifikasi ini ialah klien diharuskan mendaftarkan sebuah *email* dan *password* untuk proses pengecekan. Setelah proses pengecekan berjalan dan hasil pengecekan menyatakan pengguna telah terdaftar di server, maka pengguna akan dapat menggunakan aplikasi sistem notifikasi seperti pada gambar 4.3.



Gambar 4.3 Sistem Setelah Login

Berbeda dari Gambar 4.2 yang mana Gambar 4.3 memperlihatkan penggunaan format dari pengiriman data yang berbeda. Untuk format dari data, disimpan dengan menggunakan format JSON, lalu format JSON tersebut yang akan dibungkus dengan format XML dikarenakan protokol XMPP hanya mendukung pertukaran data menggunakan format XML. Namun peneliti tetap menggunakan JSON dikarenakan size yang dihasilkan jika data pesan dibungkus dengan format JSON akan lebih kecil dibandingkan dengan format XML, oleh karena itu untuk mengurangi pemakaian size dari data yang dikirim peneliti mengggunakan JSON sebagai pembungkus pesannya.

Kemudian untuk konfigurasi pada sebuah *node* hanya dapat dilakukan oleh klien yang membuatnya, yang mana dalam hal ini ialah *publisher*. Sedangkan untuk *subscriber* tidak dapat membuat sebuah *node* melainkan hanya menggunakan *node* yang telah disediakan. Node akan diberi nama sesuai topik yang disediakan oleh *publisher*, dan *subscriber* akan berlangganan salah satu atau semua dari topik tersebut.

Fungsi – fungsi yang disediakan pada sistem, juga termasuk dalam kebutuhan dari sistem yaitu terdiri dari :

1. *Connect,* proses awal untuk terhubung ke server.
2. *Register,* proses pendaftaran pengguna.
3. *Login,* proses pengecekan pengguna agar dapat mengakses sistem.
4. *Create or Find Node*, proses membuat atau mencari node berdasarkan nama dari topik yang ingin diinginkan oleh pengguna.
5. *Publish*, proses pengiriman pesan dari aplikasi *publisher* kepada *node* sebagai perantaranya. Parameter untuk melakukan proses pengiriman ialah nama topik dan data pesan yang berisi nama kegiatan, jam, tanggal, lokasi pada sebuah map, detail lokasi secara penjelasan kata – kata, deskripsi kegiatan yang berlangsung, waktu pengiriman pesan dan *email* pengirim.
6. *Subscribe,* proses pengajuan untuk berlangganan mengenai sebuah topik kepada *node* yang tersedia pada sistem, lalu *node* akan memasukan nya ke dalam daftar dari pengguna yang berlangganan sehingga klien akan menerima pesan berdasarkan topik tersebut.
7. *Unsubcribe,* proses berhenti berlangganan suatu topik kepada *node*, *node* akan menghapus nama pengguna tersebut dari daftar pengguna yang berlangganan topik tersebut, sehingga klien tidak akan menerima notifikasi dari sistem mengenai topik tersebut.
8. *Retrieve Message,* proses pengambilan pesan yang terdapat pada sebuah node. Dilakukan ketika awal dari aplikasi terhubung ke server.
9. *Reconnect,* hampir sama seperti proses *connect,* hal ini terjadi untuk penanganan jika terjadi kesalahan pada jaringan seperti hilangya akses internet pada perangkat pengguna yang menyebabkan terputusnya hubungan antara klien terhadap server. Namun proses ini tidak membutuhkan pengguna untuk melakukan proses *login.*

4.2 Perancangan Klien

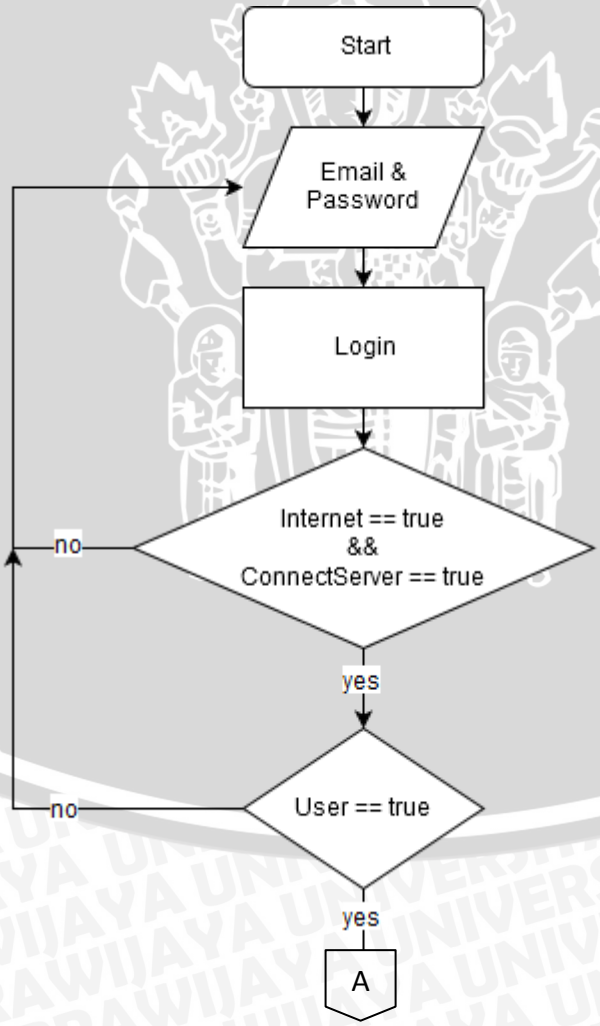
Perancangan yang dilakukan pada sisi klien terbagi menjadi dua aplikasi dan untuk tiap aplikasinya terdiri dari tahap menggambar alur kerja atau flowchart dan merancang tampilan antar muka aplikasi.

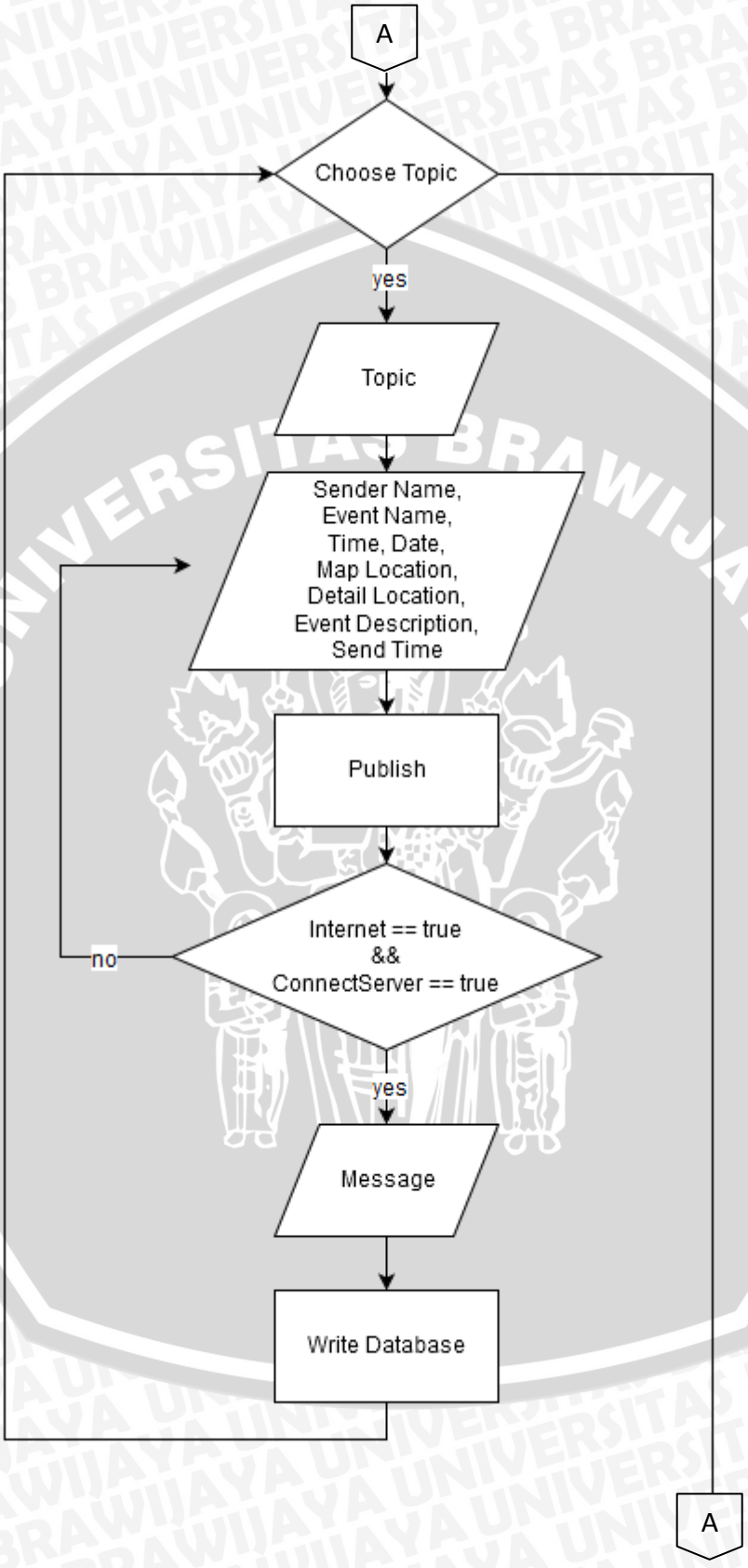
4.2.1 Flowchart Aplikasi

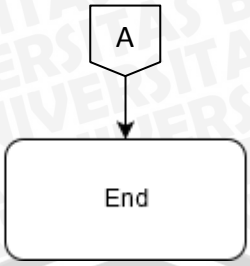
Flowchart pada pembahasan ini menggambarkan alur kerja dari aplikasi klien. Aplikasi yang dibuat pada pihak klien ada dua jenis, yaitu aplikasi untuk klien yang bertugas memberikan informasi atau berita. Kemudian aplikasi kedua yaitu aplikasi untuk klien yang bertugas sebagai *subscriber*, aplikasi ini berguna untuk menerima informasi atau berita yang dikirimkan dari aplikasi *publisher*.

4.2.1.1 *Pengirim atau Publisher*

Flowchart pada Gambar 4.4 menunjukan bagaimana penggunaan aplikasi *publisher*.




A
Choose Topic
yes
Topic
Sender Name,
Event Name,
Time, Date,
Map Location,
Detail Location,
Event Description,
Send Time
Publish
Internet == true
&&
ConnectServer == true
no
yes
Message
Write Database
A



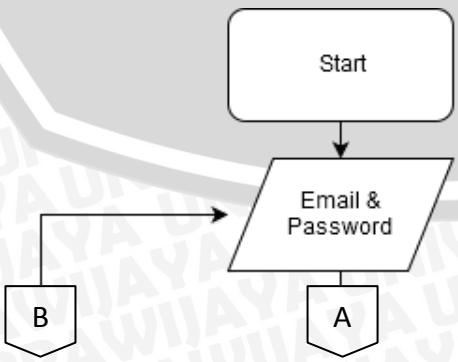
Gambar 4.4 Gambar Flowchart *Publisher*

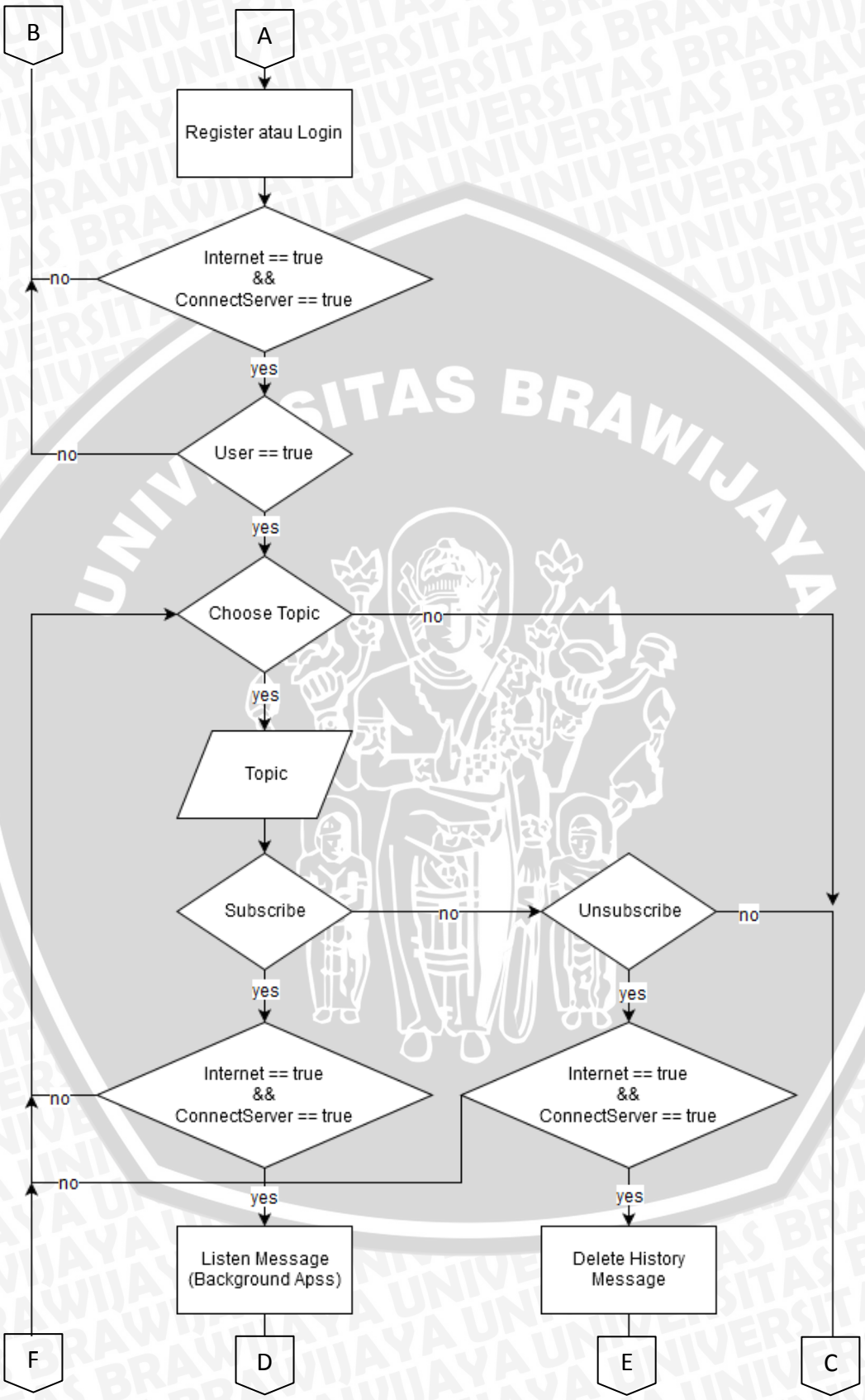
Dibawah ini prosedur kerja dari *flowchart* :

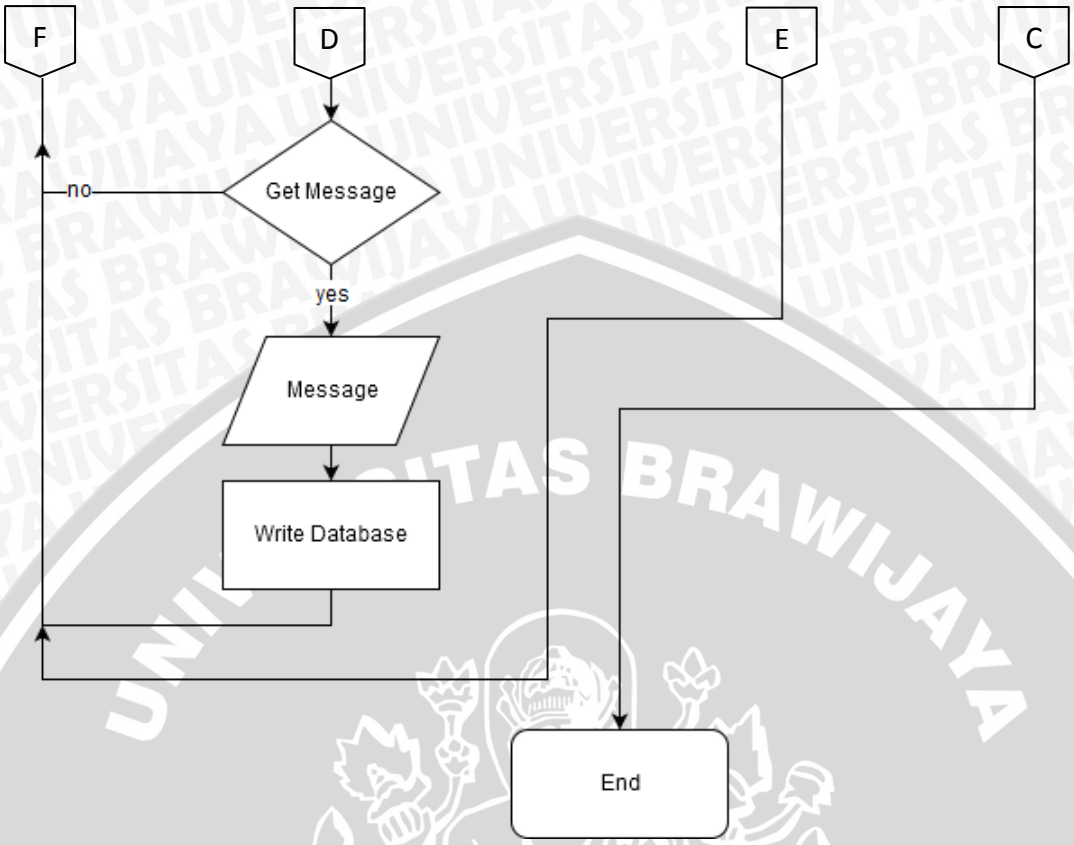
1. Memulai atau membuka aplikasi
2. Pengguna diminta memasukan email dan password yang telah didaftarkan. Bagi *publisher*, email dan password didaftarkan oleh admin langsung melalui backend server.
3. Jika gagal maka kembali ke halaman login, namun jika berhasil akan dibawa kehalaman pengisian awal aplikasi.
4. Halaman awal aplikasi berisi daftar dari kota, *publisher* yang ingin menyebarkan informasi mengenai suatu kota tinggal memilihnya lalu akan dibawa kehalaman pengisian form. Dimana pengisian form wajib dilakukan yang terdiri dari Nama kegiatan, tanggal dan jam kegiatan, lokasi dengan gambaran dari google maps, lalu lokasi detail seperti nomor rumah dll, dan terakhir deskripsi kegiatan.
5. Setelah semua form telah terisi, maka terakhir tinggal menekan tombol *publish*. Untuk dapat mengirimkan pesan ini, aplikasi akan mengecheck terlebih dahulu koneksi dari handphone pengguna dan mengecheck koneksi keserver apakah masih terhubung.
6. Dan jika pesan terkirim, aplikasi akan kembali ke halam awal.
7. Selesai

4.2.1.2 *Penerima atau Subscriber*

Flowchart pada gambar 4.5 menunjukan bagaimana penggunaan aplikasi subscriber.




B
A
Register atau Login
Internet == true
&&
ConnectServer == true
no
yes
User == true
no
yes
Choose Topic
no
yes
Topic
Subscribe
no
Unsubscribe
no
yes
yes
Internet == true
&&
ConnectServer == true
Internet == true
&&
ConnectServer == true
no
no
yes
yes
Listen Message
(Background Apss)
Delete History
Message
F
D
E
C



Gambar 4.5 Gambar Flowchart *Subscriber*

Dibawah ini prosedur kerja :

1. Memulai atau membuka aplikasi
2. Pengguna diminta memasukan email dan password yang telah didaftarkan lalu melakukan hubungan ke server untuk melakukan pengecekan pengguna. Pada saat ini pengguna telah terhubung ke server sehingga jika proses berhasil maka hubungan yang telah tersambung akan dijaga oleh layanan aplikasi dibelakang layer. Jika gagal maka kembali ke halaman login dengn hubungan terputus.
3. Jika sukses dalam pengecekan email dan password, maka akan masuk ke halaman awal aplikasi, hal ini bekerja bersamaan dengan jalannya layanan aplikasi yang menjaga koneksi dibelakang layer.
4. Halaman awal aplikasi berisi daftar dari kota, *subscriber* dapat memilih kota apa saja yang ingin dia ikuti beritanya. Ketika *subscriber* memilih kota, aplikasi akan mengirimkan permintaan *subscribe* kepada server atas nama topik kota yang dipilih.
5. Proses ini akan dijalankan dari antar muka aplikasi ke bagian layanan belakang layar aplikasi. Selain menjaga hubungan bagian layar belakang aplikasi juga berguna untuk menunggu pesan yang telah di*subscribe* oleh

pengguna sehingga aplikasi akan selalu mendapatkan pesan dari broker jika ada *publisher* yang mengirimkan pesan ke broker dengan topik yang sama.

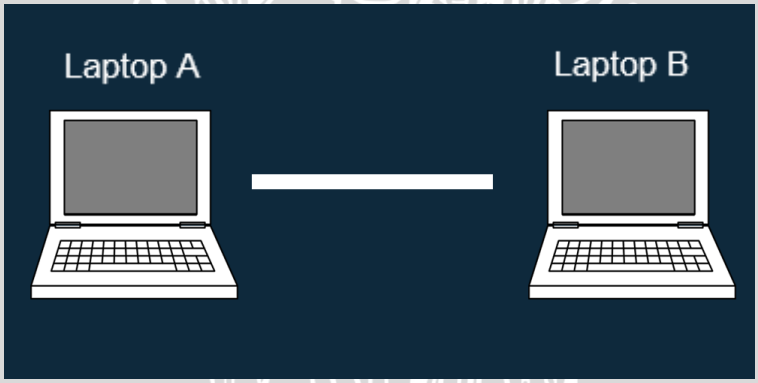
6. Jika *subscriber* ingin berhenti dari mendapatkan berita salah satu topik kota yang telah dipilihnya, *subscriber* dapat menghapusnya pada daftar *subscriber* yang berarti pengguna telah melakukan proses *unsubscribe*.
7. Selesai

4.3 Skenario Pengujian

Untuk melakukan pengujian mencari delay dari pengiriman pesan pada sistem serta untuk melihat kinerja server seperti CPU dan penggunaan memory maka diperlukan sebuah skenario untuk melakukan pengujian tersebut seperti yang terlihat pada gambar.

4.3.1 Pengujian Delay

Pengujian terbagi menjadi tiga skenario dalam melihat delay dari pengiriman pesan pada sistem yang dibangun. Pada Gambar 4.6 memperlihatkan perangkat yang digunakan pada pengujian delay. Yang mana laptop A akan digunakan sebagai server serta untuk menjalankan program pengiriman pesan, kemudian pada laptop B akan digunakan untuk menjalakan program penerima pesan. Pada program penerima pesan akan dijalankan sebanyak lima ratus kali dengan akun dari pengguna yang berbeda - beda.



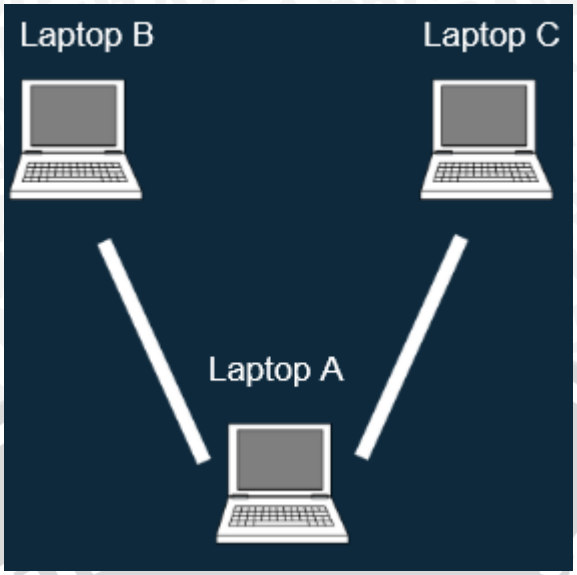
Gambar 4.6 Skenario Pengujian Delay

Skenario dalam menerima pesan akan dilakukan dengan cara yang berbeda - beda sebanyak tiga kali.

1. Sebanyak 500 pengguna berlangganan satu buah topik.
2. Sebanyak 500 pengguna berlangganan dua buah topik.
3. Sebanyak 250 pengguna akan berlangganan topik A dan 250 pengguna lainya akan berlangganan topik B.

4.3.2 Pengujian Scalibility

Untuk pengujian scalability yang akan diukur pada sisi server ialah penggunaan dari memori dan processor. Gambar 4.7 memperlihatkan skenario yang akan dilakukan pada pengujian untuk melihat performa dari server.



Gambar 4.7 Skenario Pengujian Performa

Skenario dilakukan dengan menghubungkan pengguna sebanyak - banyaknya, yaitu 900 pengguna hingga menyebabkan laptop B dan C yang digunakan sebagai program penerima pesan melakukan proses maksimal. Laptop B menjalankan program sebanyak 500 kali dengan setiap pengguna yang berbeda - beda kemudian laptop C menjalankan 400 kali program dengan pengguna yang berbeda - beda. Pada Laptop A digunakan sebagai server serta untuk menjalankan program pengiriman pesan.

4.3.3 Pengujian Availability

Skenario yang dilakukan untuk pengujian ini ialah dengan mengirimkan pesan kepada pengguna penerima yang sedang berada dalam kondisi *offline* dan melihat apakah pesan informasi tersebut tetap akan diterima oleh pengguna yang dikirimi pesan pada saat pengguna tersebut kembali pada kondisi *online*.

BAB 5
IMPLEMENTASI

Tahap implementasi merupakan tahap penerapan sistem yang telah selesai dirancang. Implementasi dilakukan setelah melewati tahap analisis dan perancangan. Dalam penelitian ini, peneliti membangun sistem dengan mengimplementasikan metode dari *publish - subscribe* yang didukung oleh protokol XMPP.

5.1 Implementasi Sistem

Perangkat yang digunakan pada server menggunakan aplikasi yang telah disediakan atau telah siap digunakan bernama Ejabberd, aplikasi server ini bersifat bebas dan terbuka namun ada beberapa feature yang jika ingin digunakan, pengguna harus membayar terlebh dahulu. Dan pada aplikasi server Ejabberd ini sudah mempunyai bawaan aplikasi penyimpanan data yang bernama Mnesia. Sehingga peneliti ini tidak perlu mencari database dan mengintegrasikannya dengan server. Aplikasi server ini menggunakan bahasa pemograman Erlang sebagai bahasa programnya. Keunggulan dari Ejabberd memiliki sistem yang kuat pada aplikasinya, lalu dapat digunakan dengan penggunaan berskala besar dan yang terpenting ialah dapat mendukung semua fitur yang disediakan oleh protokol XMPP (ProcessOne, 2012). *Ejabberd* menggunakan port 5280 sebagai port yang digunakan untuk mengakses halaman dari web admin oleh pengguna dan port 5222 untuk pengguna jika ingin menggunakan fitur yang disediakan oleh *Ejabberd*. Pada sisi klien peneliti mengimplementasikannya pada telpon pintar berbasis android dikarenakan android adalah sistem operasi untuk *smartphone* yang paling banyak digunakan saat ini. Untuk menghubungkan protokol XMPP dengan aplikasi android berbasis java maka diperlukan library pendukung yang bernama Smack. Ada dua tipe dari aplikasi klient yang dibuat pada sistem yaitu aplikasi untuk pengiriman dan menerima pesan.

Implementasi sistem berdasarkan hasil analisa sistem yang sudah dilakukan akan dibagi menjadi beberapa bagian seperti berikut :

1. Implementasi *connect*, *reconnect, disconnect*
2. Implementasi *register* dan *login*
3. Implementasi membuat dan menemukan node
4. Implementasi publish *message*
5. Implementasi subscribe dan unsubscribe *message*
6. Implementasi *Retrieve message*

5.1.1 Spesifikasi Perangkat Keras dan Perangkat Lunak

Perangkat keras yang digunakan akan berpengaruh terhadap kinerja dari pengguna perangkat lunak. Keduanya memiliki fungsi yang saling terhubung. Dan spesifikasi dari perangkat yang terlihat pada table 5.1 ialah spesifikasi minimum untuk digunakan

Tabel 5.1 Spesifikasi Perangkat yang digunakan

NO	JENIS PERANGKAT	KOMPONEN
1	Perangkat Keras	• Smartphone(RAM 1GB) • Laptop(Processor Intel Core I3, RAM 4GB)
2	Perangkat Lunak	• Sistem Operasi Android (MiUi v8) • Server :Ejabberd 16.06 (win 64) • Database Server : Mnesia • Database Klien : Sqlite • Bahasa Pemograman yang digunakan : Java, Erlang, XML, *Json* . • Android Studio • Library Smack(support XEP-0060)

5.1.2 Implementasi Register dan Login

Tampilan register dan login hanya dibedakan dengan adanya form telepon. Form inputan nomor telepon sebenarnya tidak memiliki fungsi yang penting hanya untuk pembeda antara tampilan login dan register.

Gambar 5.1 Tampilan Register

Gambar 5.1 merupakan antar muka dari halaman register, terdapat form pengisian email, password dan juga telepon. Kemudian pada Gambar 5.2 merupakan antar muka dari halaman login, hanya terdapat form pengisian email dan password.

Gambar 5.2 Tampilan Login

Sebagai langkah awal dalam pengimplementasian sebuah sistem pengiriman pesan berbasis android, diperlukannya sebuah mekanisme login untuk kedua aplikasi klien dan mekanisme register pada satu aplikasi klien. Sebelum dapat melakukan mekanisme register dan login, yang pertama dibutuhkan ialah komponen - komponen yang digunakan untuk menghubungkan ke server seperti port_address_server, ip_server, dan service_server.

```
1  public class Config {
2      int PORT = 5222;
3      String SERVER_ADDRESS = "192.168.43.228";
4      String SERVICE = "pubsub.xmpp";
5  }
```

Gambar 5.3 Implementasi Konfigurasi Variable

Potongan kode pada Gambar 5.3 ialah sebuah komponen yang digunakan untuk dapat terhubung keserver. Setelah itu dibawah ini adalah potong program yang berfungsi untuk melakukan mekanisme registrasi dan login. Namun sebelum melakukan registrasi dan login, klien membutuhkan untuk menginisialisasi koneksi terlebih dahulu dan menghubungkan koneksi ke server.

```
1  XMPPTCPConnectionConfiguration.Builder configBuilder;
2  XMPPTCPConnection connection;
3
4  configBuilder = XMPPTCPConnectionConfiguration.builder();
5  configBuilder.setSecurityMode(XMPPTCPConnectionConfiguration.Security
6  Mode.disabled);
7        configBuilder.setServiceName(config.SERVICE);
8        configBuilder.setPort(config.PORT);
9        configBuilder.setHost(config.SERVER_ADDRESS);
10       configBuilder.setDebuggerEnabled(true);
11
12       connection = new XMPPTCPConnection(configBuilder.build());
13       connection.setPacketReplyTimeout(10000);
14
15
16 SASLAuthentication.unBlackdaftarSASLMechanism("PLAIN");
17 SASLAuthentication.blackdaftarSASLMechanism("DIGEST-MD5");
18 SASLAuthentication.blackdaftarSASLMechanism("SCRAM-SHA-1");
19 SASLAuthentication.blackdaftarSASLMechanism("X-OAUTH2");
20
21 try {
22          connection.connect();
23          ReconnectionManager.getInstanceFor(connection).enableAutomatic
            Reconnection();
24 } catch (SmackException e) {
25          e.printStackTrace();
26 } catch (IOException e) {
27          e.printStackTrace();
28 } catch (XMPPException e) {
29          e.printStackTrace();
30 }
31
32
```

```
33  try {
34          connection.login(username, password);
35  } catch (XMPPException e) {
36          e.printStackTrace();
37  } catch (SmackException e) {
38          e.printStackTrace();
39  } catch (IOException e) {
40          e.printStackTrace();
41  }
42
43  AccountManager accountManager =
44  AccountManager.getInstance(connection);
45  accountManager.sensitiveOperationOverInsecureConnection(true);
46  try {
47          accountManager.createAccount(username, password);
48  } catch (SmackException.NoResponseException e) {
49          e.printStackTrace();
50  } catch (XMPPException.XMPPErrorException e) {
51          e.printStackTrace();
52
    } catch (SmackException.NotConnectedException e) {
          e.printStackTrace();
    }
```

Gambar 5.4 Implementasi Inisialisasi Hubungan ke Server

Pada Gambar 5.4, potongan kode pada baris ke 1 sampai 10 merupakan kode program untuk menginisialisasi koneksi yang ingin digunakan, seperti ke alamat mana koneksi akan dilakukan, port berapa dan apa nama domainnya. Untuk membungkus semua hal itu peneliti menggunakan kelas yang telah disediakan oleh library smack yaitu kelas *config builder,* secara gampangnya kelas ini lah yang akan menata inputan tersebut menjadi sebuah *url* yang digunakan menuju server.

Baris 15 sampai 18 merupakan kode program yang digunakan untuk menginisialisasikan security enkripsi apa yang akan digunakan ketika menjalin komunikasi dengan server. Dilihat dari potongan kode diatas, tersedia 4 macam type enkripsi yang dapat dilakukan, namun dikarenkan peneliti tidak terlalu

memikirkan untuk security datanya, sehingga peneliti memilih untuk menggunakan security enkripsi bertype plain text yang artinya komunikasi yang akan terjalin antara klien dan server hanya berupa text biasa yang dapat dibaca langsung tanpa ada proses enkripsi.

Baris 20 sampai 28 ialah potongan program untuk menjalin koneksi ke server dengan menggunakan fungsi *connect* yang telah disediakan oleh library. Disana juga terdapat fungsi yang mengaktifkan mekanisme automatis *reconnect* jika sambungkan keserver terputus dikarenakan beberapa sebab seperti hillangnya koneksi. Baris 31 sampai 39 merupakan program untuk melakukan login dengan mengirimkan nama pengguna atau *email* dan *password* yang telah terbuat atau terdaftar di server. Login hanya dapat dilakukan setelah koneksi telah tersambung dengan server. Pada baris 41 sampai 52 adalah potongan program untuk melakukan registrasi pengguna, proses ini hanya ada pada aplikasi klien yang bertugas sebagai *subscriber*.

5.1.3 Implementasi Membuat, Konfigurasi dan Menemukan Node

Baris 1 pada Gambar 5.5 merupakan inisalisasi dari variable LeafNode, variable ini digunakan untuk menyimpang nilai node yang akan dibuat, dikonfigurasikan dan selanjutnya menemukan node. Baris 3 digunakan untuk menginisialisasikan manager kelas pubsub yang mana inputan untuk kelas tersebut adalah sebuah koneksi yang terlah terhubung keserver. Kemudian baris kode dari 6 merupakan cara untuk mencari sebuha node apakah node tersebut telah ada atau belum dibuat berdasarkan nama dari node. Jika node sudah dibuat maka semua data tentang node yang didapat akan disimpan dalam variable node, sebaliknya jika node tidak ditemukan yang berarti node belum pernah dibuat maka kode pada baris ke 6 akan error dan menjalankan kode pada baris 8 sampai 23.

```
1   LeafNode myNode = null;
2
3   PubSubManager manager = new PubSubManager(connection);
4
5   try {
6           myNode = (LeafNode) manager.getNode(realTopicNode);
7   } catch (XMPPException e) {
8           ConfigureForm form = new ConfigureForm(DataForm.Type.submit);
9           form.setAccessModel(AccessModel.open);
10          form.setDeliverPayloads(true);
11          form.setNotifyRetract(false);
12          form.setPersistentItems(true);
```

```
13          form.setPublishModel(PublishModel.open);
14
15          try {
16             myNode = (LeafNode) manager.createNode(realTopicNode, form);
17          } catch (SmackException.NoResponseException e1) {
18             e1.printStackTrace();
19          } catch (XMPPException.XMPPErrorException e1) {
20             e1.printStackTrace();
21          } catch (SmackException.NotConnectedException e1) {
22             e1.printStackTrace();
23          }
24  } catch (SmackException.NotConnectedException e) {
25          e.printStackTrace();
26  } catch (SmackException.NoResponseException e) {
27          e.printStackTrace();
28  }
29
30
31
```

Gambar 5.5 Implementasi Membuat, Konfigurasi dan Menemukan Node

Pada baris 8 sampai 13 berguna sebagai inisialisi dan pengisian konfigurasi untuk sebuah node yang ingin dibuat. Contohnya seperti pengaturan mengenai siapa saja yang dapat mengakses node tersebut, karena program pada baris ke 9 menyatakan akses model sama dengan open yang artinya semua orang bebas mengakses, bebas mendapatkan data dari node tersebut, tidak perlu izin. Sama seperti akses mode yang open, public model yang open juga mengatur siapa saja yang dapat melakukan *publish message* pada node tersebut. Banyak lagi pengaturan lainnya seperti pengaturan mengenai apakah message atau item yang dikirim oleh *publisher* nantinya melewetai node tersebut hanya bersifat sementara atau data pesan akan disimpan didalam node, lalu pengaturan mengenai apakah seorang *publisher* yang mengirimkan message perlu di beritahu bahwa messagenya telah sampai di node, dan masih banyak yang lain. Setelah pengaturan telah di tentukan, selanjutnya pada baris ke 16 berfungsi untuk meminta server membuatkan sebuah node berdasarkan nama beserta mengirimkan pengaturan yang diinginkan.

5.1.4 Implementasi *Publish* Message

Gambar 5.6 Tampilan Form *Publish*

Tampilan pada gambar 5.6 merupakan tampilan dari form inputan ketika klien yang bertugas sebagai *publisher* ingin memasukan isi dari informasi yang akan di *publish* kepada *subscriber*nya. Form inputan pada tampilan tersebut terdiri dari judul kegiatan, tanggal kegiatan, waktu kegiatan, lokasi alamat yang didapat dari google maps, lokasi detail yang diberikan oleh informan dan terakhir deskripsi mengenai kejadian. Jika semua form telah terisi, maka pengguna dapat menekan tombol send atau *publish* untuk mengirimkannya kepada seluruh *subscriber* yang berlangganan berita seputar kota Malang.

```
1  try {
2  myNode.send(new PayloadItem(Jid+"/"+System.currentTimeMillis(), new
3  SimplePayload("Notification", "Notif:" + Jid + ":Publish", "<json >|" + isi +
4  "|</json >")));
5
6  } catch (SmackException.NoResponseException e) {
7          e.printStackTrace();
8  } catch (XMPPException.XMPPErrorException e) {
```

```
9           e.printStackTrace();
10  } catch (SmackException.NotConnectedException e) {
11          e.printStackTrace();
12  }
```

Gambar 5.7 Implementasi Publish Message

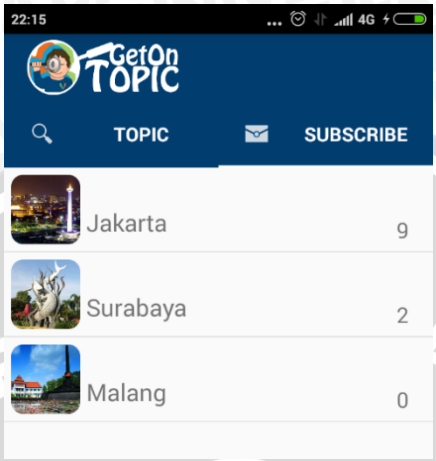
Potongan kode pada Gambar 5.7 merupakan program yang berfungsi untuk melakukan *publish* pesan kepada sebuah node. Sebelum melakukan pengiriman kepada sebuah node, tentunya untuk mencari sebuah node tersebut ada atau tidak dengan menjalankan fungsi menemukan node seperti yang telah ada pada potongan program 5.1.3. Setelah itu barulah dapat melakukan pengiriman pesan, untuk melakukan hal tersebut inputan yang diperlukan dalam melakukan pengiriman ialah data diharuskan telah di rubah kedalam kelas bernama *payload item,* inputan dari kelas tersebut adalah jid atau email dari pengguna, lalu parameter kedua ialah data yang telah dirubah formatnya menjadi format kelas bernama *simple payload.* Kelas simple payload membutuhkan inputan sebuah data yang ingin dikirimkan. Terlihat diatas bahwa data diberi tag *json* untuk menandakan bahwa data dikirim menggunakan format kelas dari *json* .

5.1.5 Implementasi *Subscribe* Message dan *Unsubscribe*

Tampilan pada gambar 5.8 merupakan tampilan yang terdiri dari daftar - daftar topik yang disediakan untuk pengguna yang ingin men*subscribe* sebuah informasi.

Gambar 5.8 Tampilan Daftar Topik (Home)

Jika pengguna ingin men*subscribe* salah satu topik maka pengguna hanya tinggal memilih salah satu dari gambar yang ada di daftar tersebut. Dan mengkonfirmasi persetujuan untuk melakukan *subscribe.*



Gambar 5.9 Tampilan Daftar Pesan

Tampilan pada gambar 5.9 merupakan daftar dari pesan yang masuk sekaligus daftar dari topik - topik yang pengguna *subscribe*. Untuk berhenti men*subscribe* sebuah topik, maka pengguna hanya tinggal menahan lebih lama dalam memencet salah satu daftar dari nama topik tersebut, sampai akhirnya ada pop window yang keluar berisikan pertanyaan untuk menghentikan *subscribe* atau *unsubscribe*. Jika pengguna menekan ya maka topik dan seluruh pesan yang tersimpan akan terhapus dari database handphone.

```
1  myNode.addItemEventDaftarener(new ItemEventDaftarener() {
2          @Override
3          public void handlePublishedItems(ItemPublishEvent items) {
4            if(items.isDelayed()){
5  //              //do nothing
6              }else{
7                String payloadItem = items.getItems().get(0).toString();
8                String splitPayload[] = payloadItem.split("\\|");
9                StringBuilder builder = new StringBuilder(splitPayload[2]);
10               String isi = builder.toString().replace("&quot;", "\"");
11               try {
12                 JSON Object json Isi = new JSON Object(isi);
```

```
13                  Notify(json Isi.getString("topic"), json Isi.getString("nama")+"
14  time(ms):"+ System.currentTimeMillis());
15                  databasePesan(json Isi);
16
17              } catch (JSON Exception e) {
18                  e.printStackTrace();
19              }
20          }
21
22      }
23
24  });
25
26  try {
27      myNode.subscribe(Jid);
28  } catch (SmackException.NoResponseException e) {
29      e.printStackTrace();
30  } catch (XMPPException.XMPPErrorException e) {
31      e.printStackTrace();
32  } catch (SmackException.NotConnectedException e) {
33      e.printStackTrace();
34  }
35
36  private void databasePesanReatrive(JSON Object isi) {
37      try {
38          dbTopic.addPesan(new                    Pesan(isi.getString("nama"),
39  isi.getString("tanggal"),       isi.getString("jam"),       isi.getString("lokasi"),
40  isi.getString("detaillokasi"),                  isi.getString("deskripsi"),
41  isi.getString("username"), isi.getString("map"), "new", isi.getString("time")),
42  isi.getString("topic"));
43          dbTopic.updateSubscribeTopic(new Topic(isi.getString("topic"), 1, 2,
44
```

```
45  0, null));
46          } catch (JSON Exception e) {
47              e.printStackTrace();
48          }
49      }
50  try {
51              myNode.unsubscribe(Jid);
52  } catch (SmackException.NoResponseException e) {
53              e.printStackTrace();
54  } catch (XMPPException.XMPPErrorException e) {
55              e.printStackTrace();
56  } catch (SmackException.NotConnectedException e) {
57              e.printStackTrace();
58  }
```

Gambar 5.10 Implementasi Subscribe dan Unsubscribe

Untuk melakukan *subscribe* terhadap sebuah node, pertama dilakukan ialah menemukan node yang diinginkan berdasarkan nama, untuk melakukan hal ini dapat dilihat pada potongan program Gambar 5.10 yang telah dijelaskan. Setelah node ditemukan, langkah selanjutnya ialah pada baris satu yaitu mengaktifkan fungsi dari daftarener terhadap pesan yang akan dating berdasarkan node yang dipilih. Jika node yang dipilih lebih dari satu maka daftarener yang akan diaktifkan pun sejumlah nodenya. Baris 4 sampai 15 merupakan aksi yang akan dilakukan jika adanya pesan yang diterima berdasarkan sebuah node. Pesan yang didapat masih dalam memiliki tag xml dikarenakan aplikasi ini dikirim menggunakan protokol yang didukung oleh bahasa xml untuk pertukaran datanya. Setelah proses pelepasan tag xml dilakukan, didapat pesan yang dibungku oleh format *json ,* hal ini ada dikarenakan peneliti menginginkan data di bungkus oleh format *json* , yang mana *json* merupakan format popular saat ini dalam pembungkusan data. Hal yang kemudian dilakukan ialah memasukan text kedalam database yang disediakan android yaitu sqlite. Potongan kode penulisan kedalam database ada pada baris ke 36 sampai 49. Setelah mengaktifkan fungsi daftarener, baris ke 26 sampai 34 berguna untuk melakukan *subscribe* terhadap suatu node yang memerlukan inputan berupa jid atau email dari pengguna.

Baris 51 sampai 59 merupakan potongan kode untuk melakukan *unsubscribe* yang membutuhkan inputan sebuah jid atau email dari pengguna.

5.1.6 Implementasi Retrieve Message

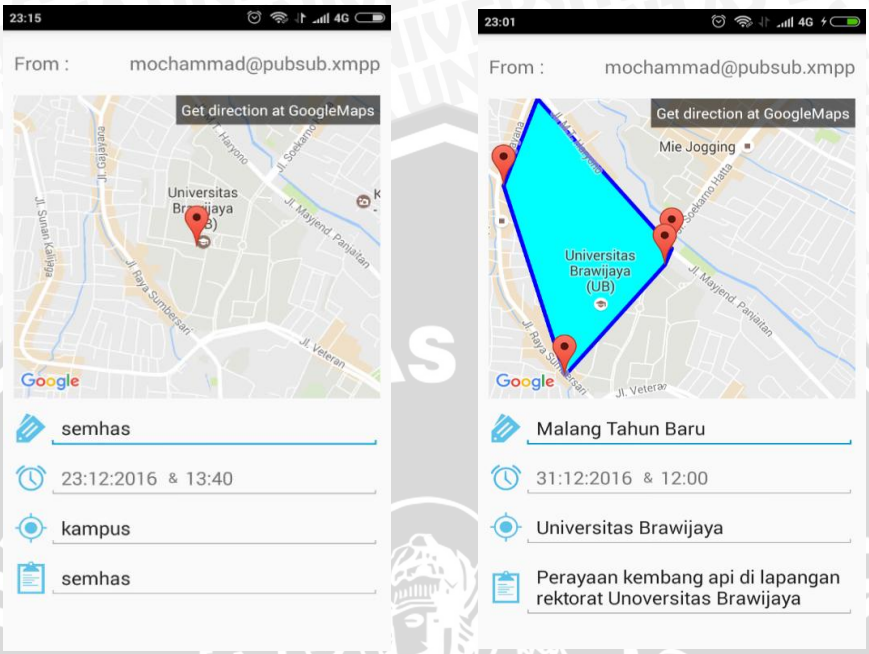
Retrieve berguna dalam beberapa kondisi, contoh salah satu kondisi ialah ketika pengguna baru saja melakukan *subscribe* terhadap sebuah topik. Aplikasi akan secara otomatis meminta history pesan yang ada disaat sebelum pengguna melakukan *subscribe* topik tersebut. Sehingga berita - berita yang masih memiliki waktu yang berdekatan masih dapat diterima oleh pengguna.

```
1       Daftar<? extends Item> items = null;
2   try {
3           items = myNode.getItems(10);
4   } catch (SmackException.NoResponseException e) {
5           e.printStackTrace();
6   } catch (XMPPException.XMPPErrorException e) {
7           e.printStackTrace();
8   } catch (SmackException.NotConnectedException e) {
9       e.printStackTrace();
10  }
11      for(int i=items.size()-1; i>=0; i--){
12          String payloadItem = items.get(i).toXML().toString();
13          String splitPayload[] = payloadItem.split("\\|");
14          StringBuilder builder = new StringBuilder(splitPayload[1]);
15          String isi = builder.toString().replace("&quot;", "\"");
16          try {
17              JSON Object json Isi = new JSON Object(isi);
18              databasePesanReatrive(json Isi);
19          } catch (JSON Exception e) {
20              e.printStackTrace();
21          }
22      }
```

Gambar 5.11 Implementasi Retrieve Pesan

Dapat dilihat pada Gambar 5.11. Baris 1 menginisialisai sebuah daftar yang nantinya akan digunakan untuk penyimpanan pesan yang diminta. Baris 3 menentukan jumlah dari pesan yang ingin diminta. Peneliti membataskannya 10 jumlah pesan, namun jika kondisi pesan yang di*publish* oleh *publisher* sangat banyak dengan frekuensi waktu yang berdekatan makan bisa saja retrieve pesan

yang dilakukan ditambah jumlahnya. Setelah pesan didapat baris ke 11 sampe 23 berfungsi untuk memasukan setiap pesan yang didapat kedalam database pesan.



Gambar 5.12 Tampilan Detail Pesan

Tampilan pada gambar 5.12 merupakan tampilan jika pesan yang diterima ingin dibuka menjadi lebih detail. Yang mana isi dari pesan tersebut ialah nama email dari pengirim, lokasi acara, judul acara, tanggal dan waktu acara, lokasi detail acara, dan terakhir ialah deskripsi dari acara. Pengguna juga dapat mengklik kata - kata *get direction at google maps* untuk mendapatkan rute nya pada aplikasi google maps sehingga memudahkan pengguna jika ingin menuju ketempat acara.

Gambar 5.12 juga memberikan contoh perbedaan dari pesan yang diterima jika pengirim mengirimkan tipe lokasi map berdasarkan point dan area. Selain kedua tipe tersebut masih terdapat tipe rute, mirip seperti tipe area namun garis - garis pada map lebih mengikuti alur dari jalan yang ada.

BAB 6
PENGUJIAN

Pada bab ini akan membahas mengenai hasil dari pengujian kinerja sistem dengan parameter keberhasilan yang menjadi tolak ukur pengujian yaitu sistem yang telah dikembangkan mampu mengirim pesan ke banyak pengguna lainnya. Dari sistem yang dikembangkan, ada empat pengujian yang akan dilakukan yaitu pengujian fungsional, pengujian QoS dengan parameter delay dan throughput, pengujian availability, dan pengujian scalability.

6.1 Pengujian Fungsional

Pengujian fungsional ini dilakukan untuk mengeetahui apakah sistem yang dikembangkan sudah sesuai dengan apa yang ada pada perancangan sistem. Dapat dilihat table 6.1 yang hasil pengujian fungsi - fungsi dari sistem.

Tabel 6.1 Hasil Pengujian Fungsional

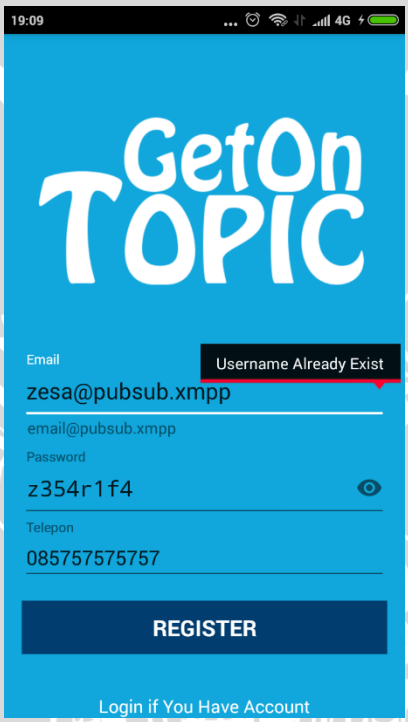
Kasus Uji	Nilai Masukan	Skenarion Pengujian	Hasil Yang Diharapkan	Hasil Uji
Register	Salah	Email Kosong	Tedapat pesan "Required Email"	Sukses
	Salah	Email Tidak Sesuai Format	Terdapat pesan "Invalid Email"	Sukses
	Salah	Email Sudah Terdaftar	Terdapat pesan "Email Already Exist"	Sukses
	Salah	Password Kosong atau Kurang Character	Terdapat pesan "Password Invalid"	Sukses
	Benar	Email dan password terisi sesuai format dan belum terdaftar	Email dan password akan terdaftar di server lalu aplikasi akan masuk kehalaman awal	Sukses
Login	Salah	Email Kosong	Tedapat pesan "Required Email"	Sukses

	Salah	Email Tidak Sesuai Format	Terdapat pesan "Invalid Email"	Sukses
	Salah	Password Kosong atau Kurang Character	Terdapat pesan "Password Invalid"	Sukses
	Salah	Email benar namun password salah atau sebaliknya	Terdapat pesan "Email dan Password not match"	Sukses
	Benar	Email dan password benar dan terdaftar	Server akan menganti status pengguna menjadi aktif dan aplikasi akan masuk kehalaman awal	Sukses
Publish, Subscriber & Unsubscribe	Benar	- Terdapat koneksi internet - Telah terhubung ke server dan telah login - Melakukan proses *publish* atau *subscribe* atau *unsubscribe*	Proses akan tercatat diserver. Jika *publish* maka setiap *subscriber* akan mendapatkan pesan. Jika *Subscribe* maka akan terdaftar sebagai *subscriber*. Dan jika *unsubscribe*, database pesan *subscribe* akan terhapus.	Sukses
Reconnect & Retrieve Message	Benar	- Telah terhubung dan login ke server - Koneksi ke server terputus dan tersambung lagi - Pengguna melihat pesan	- Pengguna dapat terhubung kembali ke server. - Pengguna mendapatkan pesan yang di*publish* ketika pengguna offline.	Sukses

6.1.1 Pengujian Register

Pengujian register dilakukan untuk melihat apakah aplikasi klien dapat melakukan proses registrasi klien baru. Setelah dilakukan pengujian, terlihat bahwa aplikasi berhasil menjalankan proses *registrasi*.

Pengujian juga dilakukan ketika klien memasukan sebuah *email* yang telah digunakan oleh klien sebelumnya.

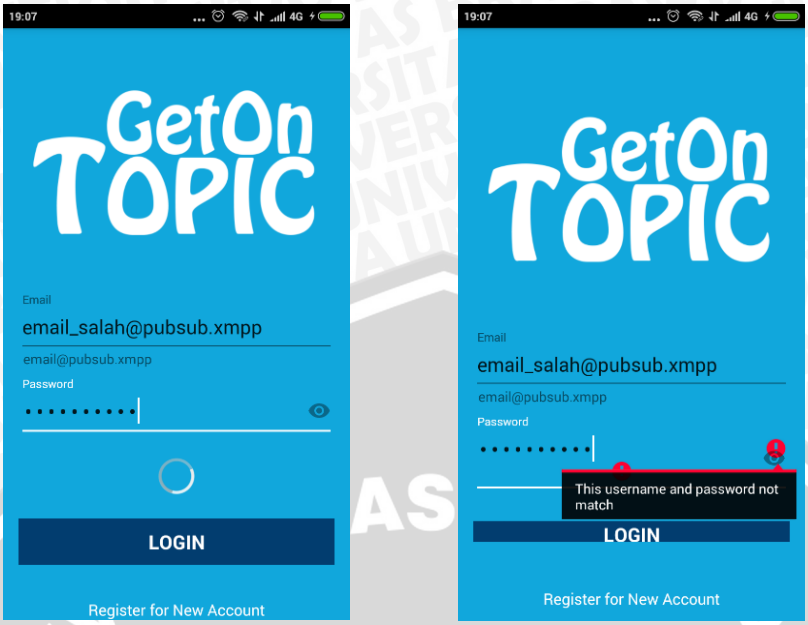


Gambar 6.1 Email Sama

Dari Gambar 6.1 terlihat bahwa proses registrasi gagal dikarenakan email yang ingin didaftarkan telah didaftarkan oleh pengguna lain.

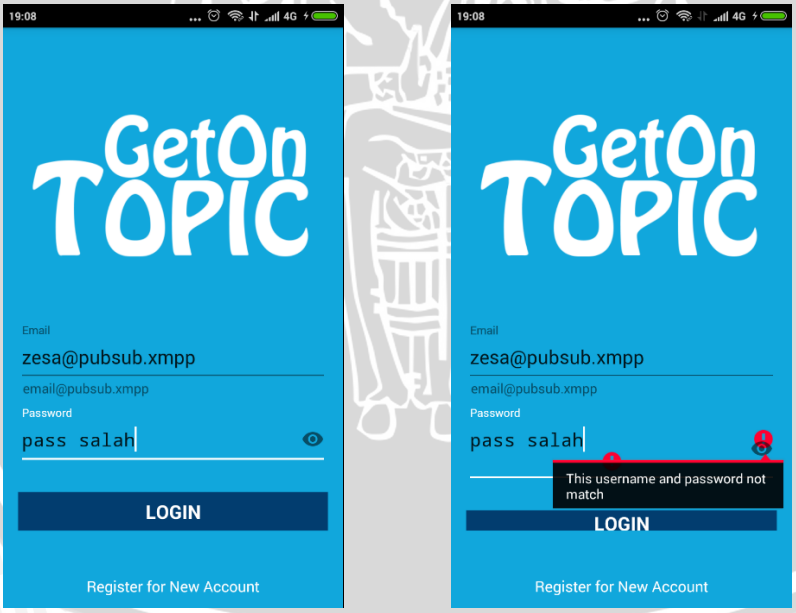
6.1.2 Pengujian Login

Proses login diuji dengan memasukan email dan password secara benar, terlihat dari gambar bahwa proses login berhasil. Selanjutnya proses login dilakukan dengan memasukan email yang salah, maka proses login akan gagal dengan memberikan keterangan bahwa email dan password tidak benar. Dapat dilihat pada gambar 6.2



Gambar 6.2 Login Email Salah

Gambar 6.3 merupakan pengujian terakhir yaitu pengujian proses login yang dilakukan dengan memasukan email benar namun password salah, proses login gagal dan memberikan keterangan email dan password tidak benar.

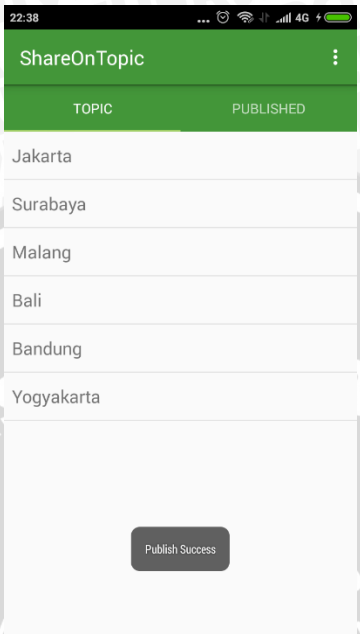


Gambar 6.3 Login Pass Salah

6.1.3 Pengujian *Publish*

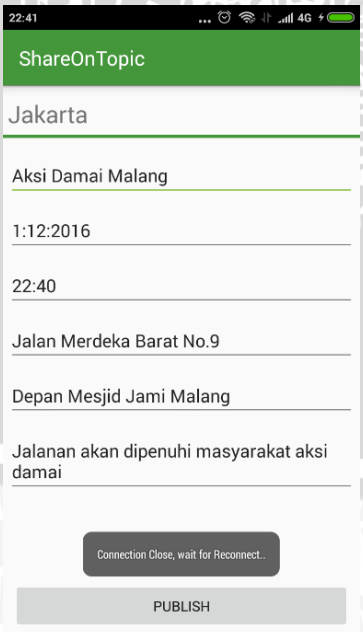
Pengujian *publish* dilakukan ketika *publisher* ingin mengirimkan sebuah informasi. Ketika *publisher* mempunyai koneksi dan *publisher* telah tersambung dan login dengan server, maka pesan akan dikirimkan dan secara otomatis

pengguna akan keluar dari halaman pesan menuju halaman awal. Hasil pengujian ketika pengiriman pesan sukses dapat dilihat pada gambar 6.4 dengan mengeluarkan sebuah pop up yang menyatakan pengiriman pesan berhasil.



Gambar 6.4 Berhasil *Publish*

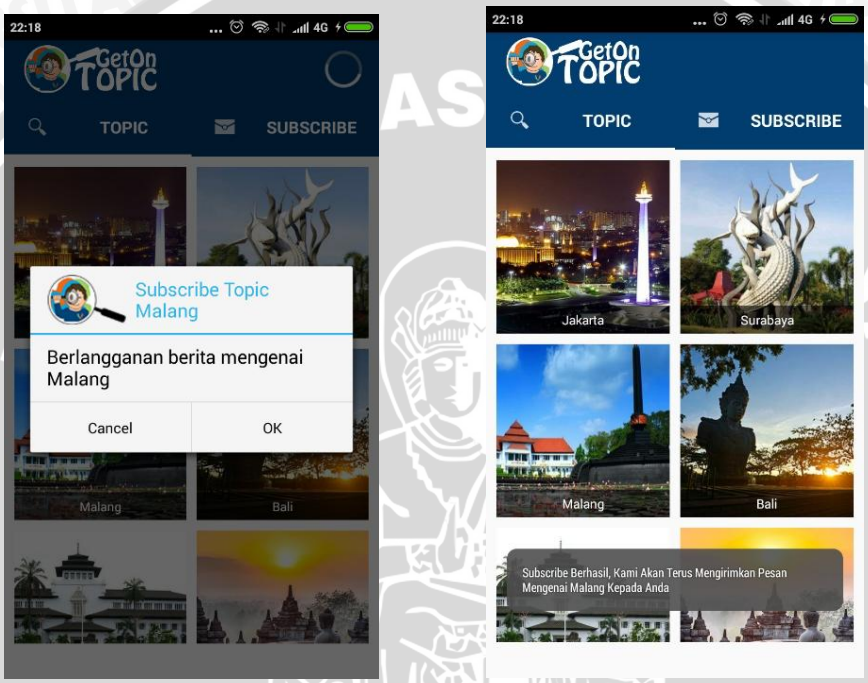
Namun dapat dilihat pada Gambar 6.5 yang mana *publisher* tidak akan dapat mengirimkan pesan ketika *publisher* tidak sedang terkoneksi dengan internet atau belum terhubung dengan server atau belum login. Sehingga pop akan memeberitahukan bahwa pesan gagal dikirim dikarenakan terputusnya hubungan keserver.



Gambar 6.5 Gagal *Publish*

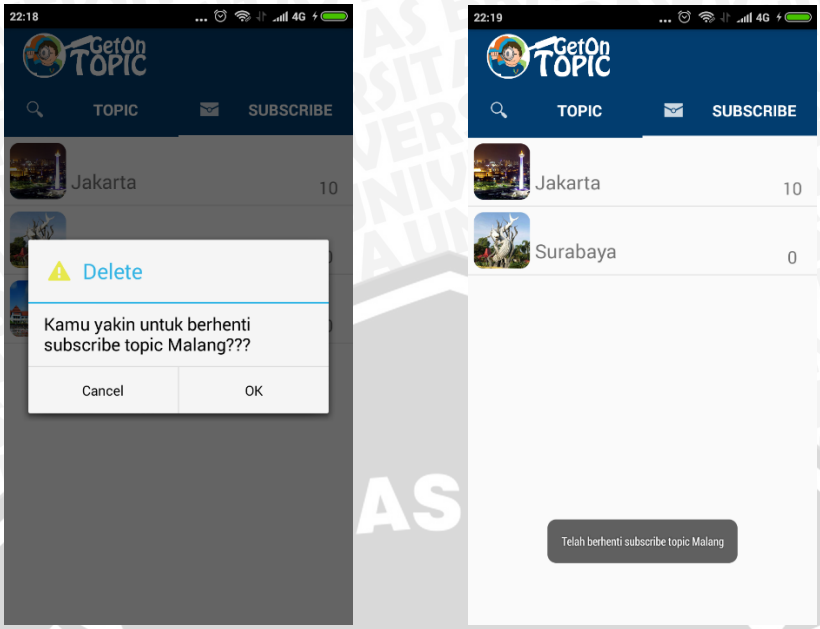
6.1.4 Pengujian *Subscribe* dan *Unsubscribe*

Tidak berbeda jauh dengan *publisher*, *subscriber* hanya dapat mens*ubscribe* sebuah topik ketika dalam keadaan mempunyai koneksi internet dan telah terhubung dan login ke server. Jika *subscriber* melakukannya disaat kondisi tersebut tidak terpenuhi maka aplikasi akan memberikan keterangan untuk melaukan nya lagi ketika telah terhubung. Gambar 6.6 merupakan pengujian untuk melakukan subscribe dengan benar sehingga proses berjalan dengan perancangan.



Gambar 6.6 *Subscribe*

Begitu juga yang terjadi ketika klien ingin melakukan *unsubscribe* topik. Dapat dilihat pada Gambar 6.7 mengenai proses unsubscribe dan menghasilkan pesan mengenai topik tersebut terhapus dari peyimpanan data pada telepon tersebut.



Gambar 6.7 *Unsubscribe*

6.2 Pengujian Delay

Pengujian ini bertujuan untuk melihat delay yang terdapat pada jaringan yang digunakan oleh aplikasi pada sistem yang dibangun. Langkah yang dilakukan dalam melakukan pengujian delay yaitu dengan membuat program sederhana yang bertugas untuk mengirimkan dan menerima data. Program pengirim akan mencatat waktu saat pengiriman dilakukan dan program penerima akan mencatat berapa waktu saat pesan diterima. Peneliti akan mengambil selisih dari waktu pengiriman dan waktu pesan diterima sehingga mendapatkan sebuah waktu delay.

Pengujian delay terbagi menjadi tiga skenario seperti yang telah direncanakan pada perancangan, yaitu :

1. Sebanyak 500 pengguna berlangganan satu buah topik.

Tabel 6.2 Rata - Rata Delay Skenario 1(ms)

Skenario 1				
Jumlah Pengguna	Delay 1	Delay 2	Delay 3	Rata - rata
500	684	703	664	684

2. Sebanyak 500 pengguna berlangganan dua buah topik.

Tabel 6.3 Rata - Rata Delay Skenario 2(ms)

Skenario 2				
Jumlah Pengguna	Delay 1	Delay 2	Delay 3	Rata - rata
500	690	665	694	683

3. Sebanyak 250 pengguna akan berlangganan topik A dan 250 pengguna lainya akan berlangganan topik B.

Tabel 6.4 Rata - Rata Delay Skenario 3(ms)

Skenario 3				
Jumlah Pengguna	Delay 1	Delay 2	Delay 3	Rata - rata
500	182	295	225	234

Seperti yang terlihat pada Table 6.2, 6.3 dan 6.4, pada table tersebut memperlihatkan rata - rata delay yang didapat dalam sistem yang telah dibangun menggunakan model *publish subscribe* berbasis protokol XMPP dengan 500 pengguna yang telah terhubung. Besarnya rata - rata delay pada skenario satu dan dua ialah 680ms dengan ketentuan 500 pengguna berlangganan satu atau dua topik yang sama. Hal tersebut mendapat indeks 1 atau buruk jika dihubungankan dengan parameter mengenai besarnya delay pengiriman pesan pada jaringan yang diatur oleh TIPHON. Namun dapat dilihat pada skenario tiga bahwa rata - rata delay yang didapat jauh lebih kecil dibandingkan skenario sebelumnya yaitu sebesar 234ms dan hal tersebut masuk dalam kategori baik oleh TIPHON, dengan jumlah dari pengguna yang terhubung sama - sama 500. Sehingga dapat diambil kesimpulan yaitu jumlah dari pengguna yang terhubung ke server tidak berpengaruh terhadap delay dari pesan yang dikirim, namun jumlah dari pengguna pada daftar list berlangganan topik yang akan mempengaruhi delay. Maksimal jumlah pengguna pada setiap daftar langganan topik agar delay pengiriman yang dihasilkan masuk kategori baik ialah sebanyak 250 user pada setiap topik dengan spesifikasi server seperti yang ada pada penelitian kali ini.

6.3 Pengujian Availabality

Pada pengujian ini, yang akan dilihat ialah keberadaan dari pesan. Apakah pesan tersebut akan selalu didapat oleh pengguna yang menggunakan aplikasi penerima pada sistem ini. Pengujian ini dilakukan agar pesan tetap akan sampai pada penggunanya walaupun kondisi jaringan sedang tidak stabil, sehingga menyebabkan pengguna sering terputus dari server.

Dengan mengkondisikkan pengguna yang akan menerima pesan telah terhubung ke server lalu pengguna tersebut memutuskan hubungan ke server untuk beberapa saat, dan kemudian menyambungkannya kembali. Dari hasil pengujian didapat bahwa pesan yang dikirim ketika pengguna tidak terhubung tetap akan diterima pada saat pengguna kembali terhubung dengan menggabungkan fungsi dari *reconnect* dan *retrieve* pesan.

6.4 Pengujian Scalability

Pengujian ini bertujuan untuk melihat kinerja dan kebutuhan pemakaian dari sisi klient dan server. Penggunaan memory dan processor sebagai parameter yang akan dimonitor kinerjanya pada sisi server sedangkan pada sisi klien dilakukan monitor terhadap konsumsi baterai dan konsumsi pemakaian akses internet untuk mengakses data.

Pada sisi server peneliti menggunakan aplikasi Performance Monitor. Performance Monitor ini adalah aplikasi yang bertujuan untuk memonitor penggunaan dari keseluruhan process yang berjalan pada sebuah sistem operasi. Adapun faktor peneliti menggunakan aplikasi tersebut ialah dikarenakan server yang digunakan pada sistem yang peneliti bangun menggunakan windows sebagai sistem operasinya dan aplikasi ini adalah aplikasi yang secara default telah terinstall pada sebuah komputer yang menggunakan sistem operasi windows. Sehingga untuk hasil tidak perlu di ragukan lagi karena pembuat dari aplikasi Performance Monitor adalah pembuat dari sistem operasinya.

Gambar diagram 2 dan 3 merupakan hasil dari pengujian penggunaan cpu dan memory pada sebuah komputer pada kondisi server mulai dinyalakan sampai server dimatikan.

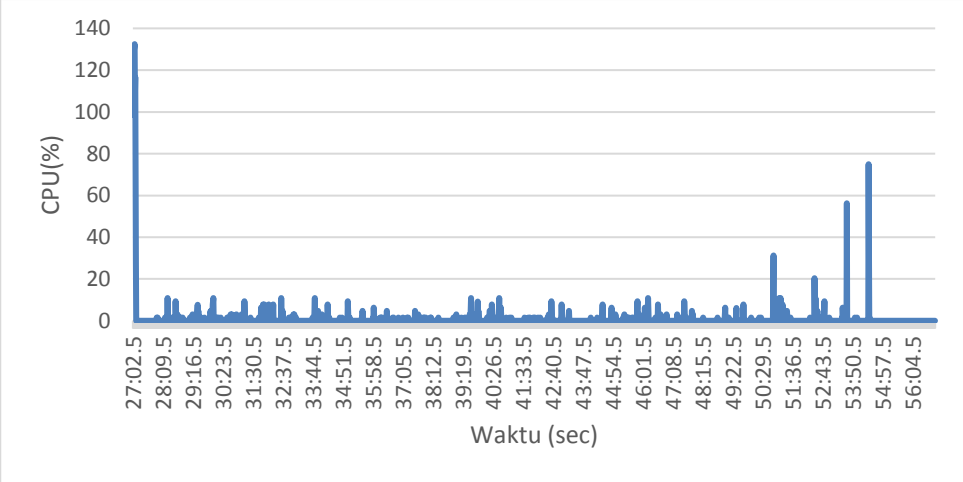


Diagram 6.1 Processor Server (%)

Pada Diagram 6.1 menjelaskan mengenai penggunaan CPU atau Procesor, rata - rata dari penggunaan processor ialah 6,1% selama 30 menit dengan proses yang terjadi terdiri dari awal server dihidupkan hingga server di matikan.

Penggunaan processor tertinggi ialah saat aplikasi server pertama kali dihidupkan yaitu sebesar 120%. Selain itu penggunaan processor saat proses pengiriman pesan dilakukan sebesar 60% kemudian naik menjadi 80% ketika semua klien diputuskan secara bersamaan.

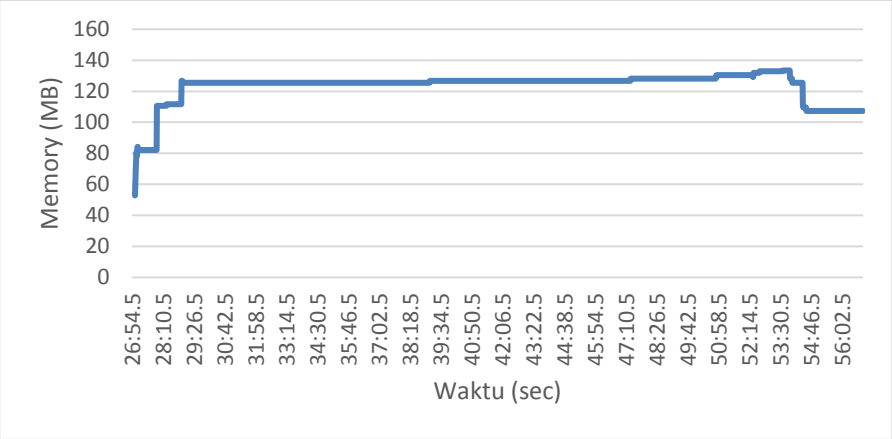
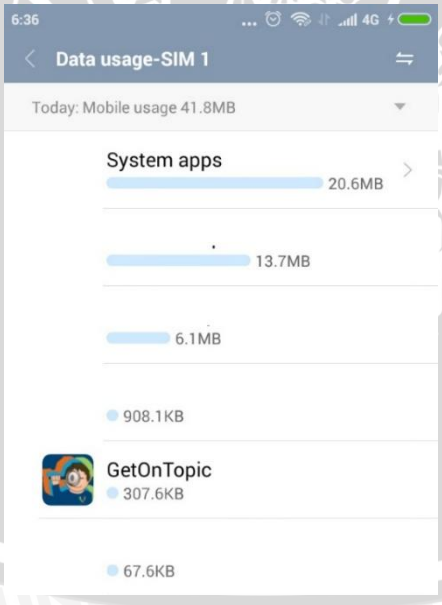


Diagram 6.2 Memory Usage (MB)

Lalu pada Diagram 6.2 mengenai penggunaan memory, terlihat bahwa memory terkecil terjadi pada saat server dihidupkan yaitu sebesar 50 MB dan penggunaan memory terbesar yaitu 136 MB berada pada saat pesan untuk terakhir kalinya di kirimkan. Yang mana pada percobaan ini dilakukan tiga kali pengiriman pesan. Pada menit ke empat tujuh, lima puluh dan lima tiga. Sehingga rata – rata dari penggunaan memory adalah sebesar 123 MB.

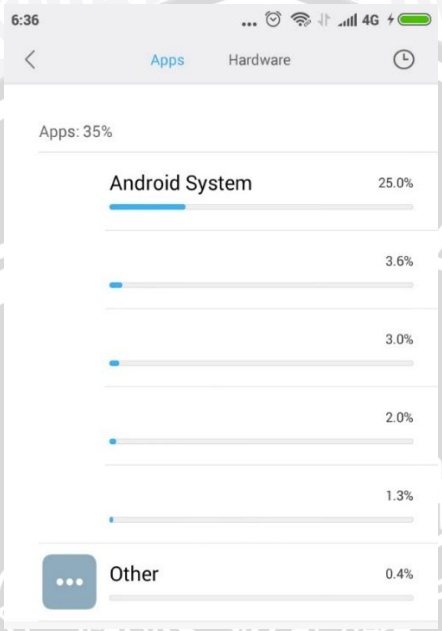


Gambar 6.8 Data Consumption

Sedangkan penggunaan pada sisi klien seperti yang terlihat pada Gambar 6.8 dan Gambar 6.9 yang mana pada penelitian ini klien menggunakan telepon pintar dan android sebagai sistem operasinya. Untuk konsumsi data yang

digunakan oleh aplikasi ketika aplikasi berada pada kondisi diam atau menunggu dengan terus terhubung ke server selama 12 jam.

Pada gambar 6.8 merupakan hasil dari total penggunaan data terhadap kouta internet ialah sebesar 300 KB, dengan menjalankan aplikasi dibelakang layar yang berguna untuk menjaga koneksi dan siap untuk menerima pesan.



Gambar 6.9 Processor Apps (%)

Sedangkan pada gambar 6.9 terlihat hasil dari processor yang digunakan ketika aplikasi berjalan dibelakang layar selama 12 jam yaitu tidak mencapai 1% yang menyebabkan persentase aplikasi tidak tampil pada daftar list mengenai penggunaan processor pada sistem android.

BAB 7
PENUTUP

7.1 Kesimpulan

Berdasarkan pembahasan dari bab - bab sebelumnya, serta dari hasil analisis hingga pengujian kinerja sistem yang telah dilakukan, maka dapat diambil beberapa kesimpulan sebagai berikut :

1. Komponen yang digunakan pada sistem pengirim notifikasi yang dibangun jika dilihat berdasarkan perangkat yang digunakan maka dapat dibagi menjadi dua yaitu klien dan server namun jika dilihat berdasarkan pekerjaannya didalam sistem maka dapat dibagi menjadi tiga yaitu *publisher* bertugas sebagai pengirim, *subscriber* sebagai penerima dan *broker* sebagai penyeleksi pesan yang nantinya akan dikirim kepada penerima yang telah terdaftar. Komponen – komponen tersebut saling terhubung untuk melakukan komunikasi dengan menerapkan metode publish - subscribe dimana format pesan diatur menggunakan protokol XMPP.
2. Ada 12 fungsi yang disediakan oleh sistem yaitu fungsi untuk terhubung ke server yang dilanjutkan dengan fungsi pengecekan dan pendaftaran pengguna, fungsi mendapatkan daftar topik yang berlangganan, fungsi membuat dan menemukan sebuah node, fungsi mengirim pesan, fungsi berlangganan dan berhenti berlangganan pesan, fungsi penanganan untuk kembali terhubung dikarenakan terdapat kondisi ketika terputusnya hubungan terhadap server, kemudian fungsi pengambilan pesan ketika telah terhubung kembali ke server berguna agar pesan yang tidak diterima oleh pengguna ketika dalam kondisi terputus tetap dapat diterima ketika kondisi telah kembali terhubung terakhir fungsi memutuskan hubungan terhadap server.
3. Dengan menginstall aplikasi server pada perangkat yang telah disediakan, secara umum server Ejabberd telah terkonfigurasi sehingga dapat langsung digunakan. Hanya tambahkan sedikit konfigurasi seperti memberikan nama domain dan perizinan terhadap IP klien untuk dapat mengakses port dari layanan. Implementasi juga dilakukan pada klient dengan membuat dua buah aplikasi android untuk mengirimkan dan menerima pesan.
4. Penanganan agar setiap pesan yang dikirim dapat sampai keseluruh penerima yaitu dengan memanggil fungsi dari pengambilan pesan kembali ketika penerima kembali aktif dari kondisi tidak aktif, lalu pesan akan dimasukan kedalam tempat penyimpanan yang terdapat pada aplikasi penerima dengan syarat pesan hanya akan disimpan jika status dari pesan belum pernah tersimpan pada tempat penyimpanan.

5. Pengujian kinerja dibagi menjadi dua yaitu server dan klien. Dengan menggunakan aplikasi bawaan dari windows bernama Performance Monitor untuk melihat kinerja dari penggunaan memory dan processor pada server. Dari pengamatan server didapat hasil mengenai rata - rata penggunaan memory sebesar 120 MB dengan maksimal memory 136 MB saat pengiriman pesan yang ketiga dan rata - rata penggunaan processor sebesar 6,1% dengan maksimal processor yang digunakan sebesar 120% saat proses awal server dihidupkan dan 80% ketika proses pengiriman pesan terjadi. Untuk pengamatan kinerja pada klien, digunakan aplikasi keamanan yang telah disediakan oleh *handphone* MI. Dari hasil pengamatan selama 12 jam didapat hasil bahwa rata - rata konsumsi data sebesar 300 KB dan rata - rata processor yang digunakan tidak mencapai 1%.
6. Jumlah dari pengguna yang terhubung ke server tidak berpengaruh terhadap delay dari pesan yang dikirim, namun jumlah dari pengguna pada daftar list berlangganan setiap topik yang akan mempengaruhi delay. Maksimal jumlah pengguna pada setiap daftar langganan topik agar delay pengiriman yang dihasilkan masuk kategori baik ialah sebanyak 250 user pada setiap topik dengan spesifikasi server seperti yang ada pada penelitian kali ini.

7.2 Saran

Adapun saran yang dapat peneliti berikan dari awal sampai selesainya penelitian ini ialah :

1. Untuk penelitian selanjutnya dapat mengimplementasikan feature security yang telah disediakan oleh Protkol XMPP dalam hal register, login, dan pengiriman pesan.
2. Implementasi dikembangkan tidak hanya untuk Android
3. Untuk pengujian dapat langsung diterapkan pada lebih banyak jenis device dan lebih banyak pengguna.

DAFTAR PUSTAKA

Kevin, A., 2011. That 'Internet of Things' Thing. RFID Journal, pp. 1-4.

Sollner, Benjamin., 2009. XMPP based Media Sharing for Mobile Collaboration with Android Phones, pp. 3-7.

Colouris, George. et al., 2012. DISTRIBUTED SYSTEMS Concepts and Design, 5th Edition. Addison Wesley Inc.

Eugster, P.T., Felber, P., Guerraoui, R. & Kermarrec, A.M., 2003. The Many Faces of Publish/Subscribe. ACM Comput. Surv., vol. 35, no. 2, pp. 114–131.

Pongthawornkamol, Thadpong., Nahrstedt, Klara., Wang, Guijun., 2007. The Analysis of Publish/Subscribe Systems over Mobile Wireless Ad Hoc Networks.

G. Li., 2010. Optimal and Robust Routing of Subscriptions for Unifying Access to the Past and the Future in Publish/Subscribe. PhD thesis.

Faruq, Fajar., Maulana., Alvianda, Ferdi., Sukarno, A.K., Mahatma, Yogi., Adila, W.N., 2015. Implementation Publish and Subscribe Mechanism In Portal Liputan6.com and Okezone.com, pp. 1-6.

Tiphon. 1999. ”Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) General aspects of Quality of Service (QoS)”, DTR/TIPHON-05006 (cb0010cs.PDF).

Process One. 1999. Ejabberd. [Program Komputer] Process One. Tersedia di : <https://www.process-one.net/en/company/> [Diakses 4 Januari 2016]

Lia. Radar Malang. 2016. Tersedia di : <http://radarmalang.co.id/dua-kampus-wisuda-kawasan-dinoyo-kota-malang-macet-parah-32506.htm> [Diakses 27 Februari 2016]

Roy, Rob. Kompasiana. 2016. Tersedia di : <http://www.kompasiana.com/roy.vansby/malang-macet_550a1657a33311af4d2e3c21> [Diakses 26 Juni 2016]