YAYASAN AKRAB PEKANBARU
Jurnal AKRAB JUARA
Volume 7 Nomor 2 Edisi Mei 2022 (300-310)

28

IMPLEMENTASI MESSAGING SYSTEM BERBASIS PROTOKOL AMQP UNTUK SISTEM CONTROL NETWORK BERBASIS ANDROID

Rahmat Tri Yunandar, Muhammad Fahmi
Sistem Informasi Fakultas Teknologi Informasi Universitas Bina Sarana Informatika, Sistem Informasi Universitas Nusa Mandiri
(Naskah diterima: 1 Maret 2022, disetujui: 28 April 2022)

Abstract

The development of technology is currently very fast, fast and dynamic with a change that is accompanied by the need for supporting devices that surround it. One example of this technology is a network router network which is a group of devices that can provide internet services that share resources or data with each other. To support daily activities, a quality, stable and secure internet network is needed. Often there are problems that occur in the technology, including activities sourced from users that can interfere with continuity, stability of internet connection access that exceeds the maximum resource bandwidth capacity or interference due to attacks from outside (external factors) or even due to damage to the hardware side and others. . This needs to be followed up quickly and responsively by a server admin. So we need an additional architectural alternative that can overcome this situation so that it can be conveyed quickly to the Admin server. In this study, an application was developed that allows network server administrators to receive direct messages in the form of notifications on the android screen containing a log of events that occurred on the proxy router with segment parameter issues that exist when problems occur on the server or on the mikrotik router. The design of this system applies a communication protocol, namely AMQP (Advanced Message Queueing Protocol), through the RabbitMQ message broker which is an intermediary that functions to regulate messages sent by the sender (sender/producer/publisher) to the recipient (receiver/consumer/subscriber). . The development method used in this research is the Rapid Application Development (RAD) method which has advantages in the process of making the system which is relatively fast but also has good results. As a configuration simulation of this system, it takes a proxy router device that is connected to local and public networks to an available server. The results of this study were able to create a Messaging System application using the AMQP communication protocol based on the android platform which was tested functionally through simulations on the system and network. And it is concluded that this research is considered to be the right answer.
Keywords*:* *Network, proxy router, AMQP protocol, android*

Abstrak

Perkembangan teknologi saat ini begitu sangat pesat, cepat dan dinamis dengan sebuah perubahan yang ada diiringi kebutuhan perangkat-perangkat pendukung yang melingkupinya.

Salah satu contoh teknologi tersebut adalah jaringan *router network* yang merupakan sekelompok piranti yang dapat menyediakan layanan internet yang saling berbagi sumber daya atau data. Untuk menunjang kegiatan aktifitas sehari-hari dibutuhkan jaringan internet yang berkualitas, stabil dan aman. Sering kali terdapat permasalahan yang terjadi pada teknologi tersebut diantaranya aktifitas bersumber dari user yang dapat mengganggu keberlangsungan, kestabilan akses koneksi internet yang melebihi *resource* maksimum kapasitas *bandwidth* atau gangguan karena serangan dari luar (faktor eksternal) atau bahkan karena kerusakan pada sisi hardware dan lain-lain. Hal ini perlu ditindak lanjuti dengan cepat dan tanggap penanganannya oleh seorang Admin server. Sehingga diperlukan sebuah alternatif arsitektur tambahan yang dapat mengatasi situasi tersebut bisa tersampaikan dengan cepat ke Admin server. Pada penelitian kali ini dikembangkan sebuah aplikasi yang memungkinkan Administrator server jaringan dapat menerima langsung pesan berupa notifikasi pada layar android berisikan *log event* yang terjadi pada router mikrotik dengan segment parameter issue yang ada ketika terjadi permasalahan pada server ataupun di router mikrotik. Perancangan sistem ini menerapkan sebuah protokol komunikasi yaitu AMQP (*Advanced Message Queueing Protocol*), melalui *message broker* RabbitMQ yang merupakan sebuah perantara yang berfungsi mengatur antar pesan yang dikirim oleh pengirim pengirim (sender/producer/publisher) kepada penerima (receiver/consumer/subscriber). Metode pengembangan yang digunakan penelitian ini melalui metode Rapid ApplicationDevelopment (RAD) yang memiliki kelebihan pada proses pembuatan sistem yang terbilang cepat akn tetapi juga memiliki hasil yang baik. Sebagai simulasi konfigurasi dari sistem ini dibutuhkan device router mikrotik yang terhubung pada jaringan lokal dan public ke sebuah server yang tersedia. Hasil dari penelitian ini mampu menciptakan suatu *Messaging System* dengan menggunakan protokol komunikasi AMQP berbasis *platform* android yang diuji secara fungsionalitas melalui simulasi pada jaringan. Dan disimpulkan penelitian kali ini dianggap menjadi jawaban yang tepat.
Kata Kunci: Jaringan, router mikrotik, protocol AMQP, android.

I. PENDAHULUAN

Perkembangan teknologi saat ini begitu sangat pesat, cepat dan dinamis dengan sebuah perubahan yang ada diiringi kebutuhan perangkat-perangkat pendukung yang melingkupinya. Berdasarkan (Rizfan, 2019) tentang perkembangan jaringan, dari tahun 1940 sampai dengan saat ini cukup siginifikan pesat. Diawali dari perkembangan jaringan dengan *batch processing* tahun 1940, pengembangan TSS (*Time Sharing System*) tahun 1950, *Distribution Processing* tahun 1960, penggunaan TCP/IP untuk standarisasi kode unik pada tahun 1980, *World Wide Web* di tahun 1990, hingga tahun 2000. Mengikuti arah teknologi tersebut adalah jaringan *router network* yang merupakan sekelompok piranti yang dapat menyediakan layanan internet yang saling berbagi sumber daya ataupun data. Untuk menunjang kegiatan

aktifitas sehari-hari dibutuhkan jaringan internet yang berkualitas, stabil dan aman. Namun sering kali terdapat permasalahan yang terjadi pada teknologi tersebut diantaranya disebabkan aktifitas bersumber dari pengguna yang dapat mengganggu keberlangsungan, kestabilan akses koneksi internet yang melebihi *resource* maksimum kapasitas *bandwidth* mengakibatkan internet lambat, kemudian gangguan karena serangan dari luar (faktor eksternal), isu lainnya sering diklaim komunikasi *timeout, data traffic* padat hingga karena kerusakan pada sisi hardware itu sendiri.

Tentu hal ini perlu ditindak lanjuti dengan cepat dan tanggap penanganannya oleh seorang admin server. Permasalahan inilah yang dihadapi, dimana kondisi jaringan harus selalu terpantau dalam keadaan aman dan stabil sehingga tidak mengganggu pengguna yang sedang melakukan aktifitasnya. Hal lainnya, seorang admin server tidak selalu berada pada ruangan teknisi atau didekat dengan area lokasi jaringan untuk memantau kondisi secara *real-time*. Karena itu diperlukan sebuah alternatif arsitektur tambahan yang dapat mengatasi situasi tersebut bisa tersampaikan dengan cepat dan dapat membantu kinerja Admin server.

Penelitian sebelumnya yang dilakukan (R. Pradikta, A. Affandi, dan E. Setijadi, 2013) mengambil judul "*Rancang Bangun Aplikasi Monitoring Jaringan dengan Menggunakan Simple Network Management Protocol*" membahas mengenai perancangan aplikasi monitoring jaringan yang dapat digunakan sebagai perantara pengambilan dan pengolahan nilai SNMP yang dapat tersimpan di database sehingga dapat ditampilkan berupa laporan informasi tentang kondisi jaringan perangkat dan trafik pada transport TCP.

Penelitian lainnya yang dilakukan (Santoso, Primananda, Amron, 2019) dengan judul "*Implementasi Sistem Pemantauan Suhu dan Kelembapan Udara Berbasis Protokol AMQP*" menjabarkan tentang pengembangan sistem pemantauan suhu dan kelembapan udara. Memberikan hasil bahwa implementasi protokol AMQP pada sistem tersebut berhasil dilakukan dan dari hasil pengujian menunjukkan bahwa sistem yang dibangun mampu mengirimkan komunikasi berbagi data antar setiap node yang ada.

Dari permasalahan dan kebutuhan yang telah dijelaskan diatas menjadi dasar ide penelitian kali ini dikembangkan sebuah aplikasi yang memungkinkan Administrator server jaringan dapat menerima langsung pesan

berupa notifikasi pada layar android berisikan *log event* yang terjadi pada router mikrotik dengan segment parameter issue yang ada ketika terjadi permasalahan pada server ataupun di router mikrotik. Perancangan sistem ini menerapkan sebuah protokol komunikasi yaitu AMQP (*Advanced Message Queueing Protocol*), melalui message broker RabbitMQ yang merupakan sebuah perantara yang berfungsi mengatur antar pesan yang dikirim oleh pengirim (*sender/producer/publisher*) kepada penerima (*receiver/consumer/subscriber*). Metode pengembangan yang digunakan penelitian ini melalui metode Rapid Application Development (RAD) yang memiliki kelebihan pada proses pembuatan sistem yang terbilang cepat akn tetapi juga memiliki hasil yang baik. Sebagai simulasi konfigurasi dari sistem ini dibutuhkan *device router mikrotik* yang terhubung pada jaringan lokal dan *public* ke sebuah server. Hasil dari penelitian ini mampu menciptakan suatu *Messaging System* dengan menggunakan protokol komunikasi AMQP berbasis *platform* android yang diuji secara fungsionalitas melalui simulasi pada jaringan.

II. METODE PENELITIAN

2.1 Metodologi Pengembangan Sistem

Rapid Application Development (RAD) adalah model proses pengembangan perangkat lunak yang bersifat incremental terutama untuk waktu pengerjaan yang pendek (Sukamto & Shalahudin, 2016). RAD merupakan model proses perangkat lunak yang menekankan pada fokus pengembangan yang singkat, dan adaptasi versi yang cepat dengan menggunakan kontruksi komponen di sistem (Putri, dkk, 2018).

Gambar 1. Tahapan Pengembangan *RAD*

RAD terbagi tiga langkah tahapan terstruktur dan saling kebergantungan satu dengan lainnya, yaitu:

1. Pererencanaan, pada tahapan di proses ini dilakukan identifikasi kebutuhan pada proses sistem yang akan dibangun. Kemudian juga dilakukan identifikasi isu permasalahan utama serta membentuk alternatif solusi yang akan ditawarkan. Pada tahapan ini pula digambarkan arsitektur dari sistem yang akan dikembangkan.
2. *Workshop* desain, dalam proses ini dilakukan pembahasan secara detail bagaimana sistem dikembangkan. Dimana proses ini

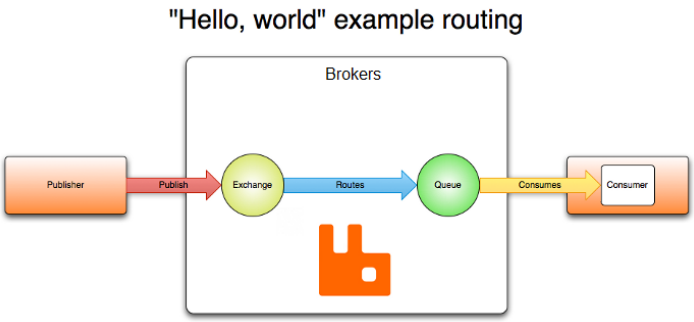
dilakukan pemodelan sistem yang akan dikembangkan, pemodelan tersebut mencakup pemodelan tingkah laku, interaksi dan juga struktur dari sistem.

3. Implementasi, tahapan proses ini melakukan eksekusi dan implementasi langsung dari semua pekerjaan yang telah dilakukan ke dalam bentuk sistem siap digunakan.

2.2 Protocol AMQP

Standar protokol untuk pesan-berorientasi *Message Oriented Middleware* (MOM). MOM sendiri mendukung kualitas layanan tinggi, komunikasi antar program secara asinkron melalui antar dua atau lebih komponen, atau antar dua atau lebih aplikasi. AMQP menentukan protokol, sehingga pesan dapat dipertukarkan di antara komponen perangkat lunak yang disediakan oleh vendor yang berbeda tanpa memerlukan gateway atau adaptor. AMPQ memiliki keunggulan sendiri yaitu efisien, portabel, *multichannel* dan *safety*. Protokol *biner* menjadikan nilai otentikasi dan enkripsi dengan cara *SASL* ataupun *TLS*, serta bergantung pada protokol transport layaknya TCP (Rouse, 2018).

Setidaknya ada empat komponen utama pada arsitektur AMQP yaitu, *Publisher*, sipembuat pesan lalu dikirim ke *broker*. *Exchange*, mengatur cara pendistirbusian pesan ke antrian (*queue*). *Queue*, kemudian mendistribusi pesan ke *consumer*. *Consumers* yang menerima pesan dari *queue* ataupun *customers* sendiri yang *listener* langsung. Dan pada *Brokers* yang berperan sebagai jembatan antara *Producer* dan *Consumers* dalam komunikasi data (Marsh, 2009). Adapun ilustrasi cara kerja dari protokol AMQP ditampilkan pada gambar berikut.



Gambar 2. Arsitektur AMQP

Pada AMQP terdapat beberapa bentuk penempatan dan peletakan sumber data Exchange ke Queue yaitu:

1. Direct Exchange

   Pesan akan di route langsung ke *queue* yang ada terdaftar pada *Key Message*. Sehingga ketika *key message* diterima maka pesan akan langsung dikirim ke *queue*. Jika ada 2 *queue* atau lebih memiliki *key message* yang sama, maka pesan akan didistribusikan menerapkan Algoritma Round roubin.

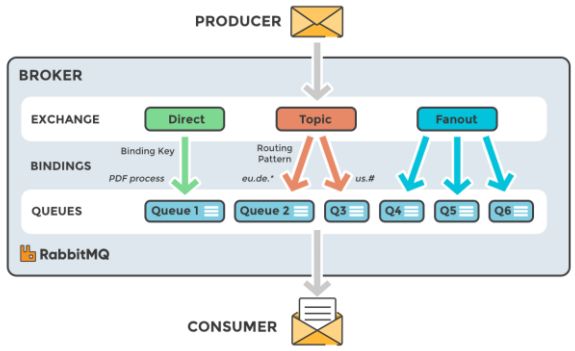
2. Fanout Exchange

   Pesan tidak bergantung pada key ataupun pattern. Pesan yang diterima menyalin kemudian dikirimkan ke setiap queue dan akan langsung di publish kesemua queue yang terdapat pada broker.

3. Topic Exchange

   Pesan akan di route berdasarkan pattern dan key yang ada dan bersifat publish subscribe, sehingga exchange mendistribusikan pesan berdasarkan routing key yang sama.

Adapun ilustrasi cara kerja data Exchange ke Queue ditampilkan pada gambar berikut.

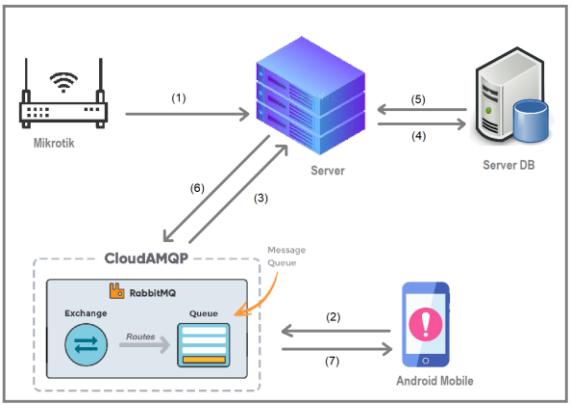


Gambar 3. Ilustrasi data Exchange ke Queue

3. ANALISA DAN DESAIN

3.1 Arsitektur Sistem

Pada bagian ini akan digambarkan bagaimana arsitektur dari jalannya sistem yang dikembangkan seperti penjelesan gambar berikut.



Gambar 4. Arsitektur Sistem

1. Device Mikrotik mengirim log event secara terus menerus ke server yang kemudian selanjutnya di parsing berdasarkan parameter tertentu.
2. Device Mobile berupa OS Android pada aplikasi yang diinstall dan dijalankan. Untuk berkomunikasi dengan *server Broker* middleware pengiriman pesan, running pertama kali aplikasi akan menginisiasi RabbitMQ Client.
3. RabbitMQ sebagai *broker messages* yang menerapkan protokol AMQP, mengirim data user berupa profiling dan autentikasi perangkat yang terdeteksi konek dan listen ke broker.
4. Server kemudian menjalankan perintah insert ke sebuah server database setiap aktivitas yang ada di *broker message*. Pada tahap ini dilakukan oleh siklus sistem yang bertujuan untuk selanjutanya data dapat

dengan mudah diakses dan dipelajari secara historical sebagai bahan evaluasi dikemudian waktu.

5. Server memperoleh (*retrive*) data *message* dan *profiling* dari database yang dikelola.
6. Pada tahap ini juga ketika terdapat *log event errors* yang dikirim oleh mikrotik ke server, maka server akan memparsing pesan, men-*trigger* dan meneruskan menuju *Broker Messages* sebagai media *producers* yang selanjutnya akan dikirim ke *customers*.
7. Pesan yang diterima oleh *Broker Messages* akan dilakukan proses penentuan pesan (*Exchanges*) ke *queue* dan selanjutnya akan dikirim ke *customers* / device mobile android berupa notifikasi.

3.2 Implementasi RabbitMQ

Untuk menerapkan Protokol AMQP (Advanced Message Queueing Protocol) sebagai *process messaging* dibutuhkan *Message Brokers*. *Message Brokers* adalah apps service mengatur antar pesan yang dikirim oleh pengirim (*sender/producer/publisher*) kepada penerima (*receiver/consumer/subscriber*) dalam bentuk proses komunikasi *asyncronous*. Pesan serentak dikirim oleh *producers* dan menerima pesan oleh *consumers* secara Bersamaan. Berikut adalah persiapan hingga penerapannya.

1. Installasi dan konfigurasi RabbitMQ di server di lingkungan Ubuntu 20.04

a. Tambahkan Repository Signing Key

```
wget -O- https://www.rabbitmq.com/rabbitmq-release-signing-key.asc | sudo apt-key add -
```

b. Insialisasi RabbitMQ repository

```
echo "deb https://dl.bintray.com/rabbitmq-erlang/debian focal erlang-22.x" | sudo tee /etc/apt/sources.list.d/rabbitmq.list
```

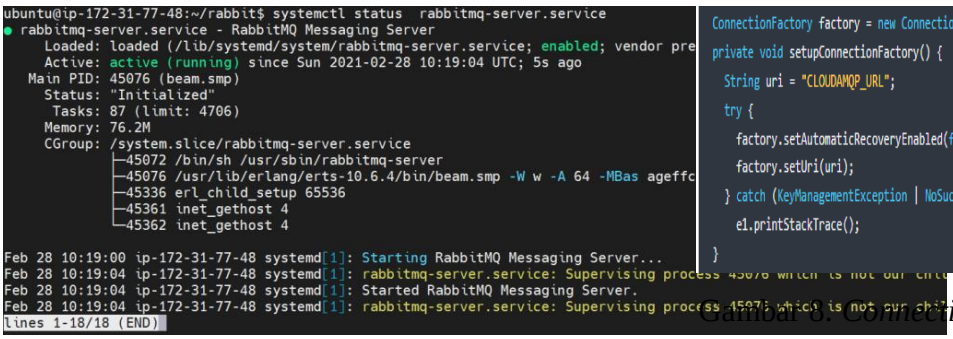
c. Installasi RabbitMQ Server

```
sudo apt install rabbitmq-server -y
```

d. Melihat Status RabbitMQ

```
systemctl status rabbitmq-server.service
```

Berikut ini adalah gambar dari status proses installasi pada server.



Gambar 5. Status proses installasi RabbitMQ

2. Implementasi RabbitMQ Client di Android

Untuk menggunakan RabbitMQ Client pada aplikasi android, beberapa tahapan persiapan yang harus dilakukan adalah:

a. Tambahkan *depedency library* pada build.gradle

```
dependencies {
    ...
    compile files('libs/rabbitmq-client.jar')
    ...
}
```

Gambar 6. *depedency library*

b. Tambahkan *permission* di Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.cloudamqp.rabbitmq"
    android:versionCode="1"
    android:versionName="1.0">
    ......
    <uses-permission android:name="android.permission.INTERNET"></uses-permission>
</manifest>
```

Gambar 7. *permission* Android Manifest

c. Buat fungsi *ConnectionFactory* untuk merangkum parameter konfigurasi koneksi.

```
ConnectionFactory factory = new ConnectionFactory();
private void setupConnectionFactory() {
  String uri = "CLOUDAMQP_URL";
  try {
    factory.setAutomaticRecoveryEnabled(false);
    factory.setUri(uri);
  } catch (KeyManagementException | NoSuchAlgorithmException | URISyntaxException e1) {
    e1.printStackTrace();
  }
```

Gambar 8. *ConnectionFactory* konfigurasi koneksi

d. Buat fungsi *Subscriber Handler* untuk listner pesan yang masuk.

```
void subscribe(final Handler handler)
{
  subscribeThread = new Thread(new Runnable() {
    @Override
    public void run() {
      while(true) {
        try {
          Connection connection = factory.newConnection();
          Channel channel = connection.createChannel();
          channel.basicQos(1);
          DeclareOk q = channel.queueDeclare();
          channel.queueBind(q.getQueue(), "amq.fanout", "chat");
          QueueingConsumer consumer = new QueueingConsumer(channel);
          channel.basicConsume(q.getQueue(), true, consumer);

          while (true) {
            QueueingConsumer.Delivery delivery = consumer.nextDelivery();
            String message = new String(delivery.getBody());
            Log.d("","[r] " + message);
            Message msg = handler.obtainMessage();
            Bundle bundle = new Bundle();
            bundle.putString("msg", message);
            msg.setData(bundle);
            handler.sendMessage(msg);
          }
        } catch (InterruptedException e) {
          break;
        } catch (Exception e1) {
          Log.d("", "Connection broken: " + e1.getClass().getName());
          try {
            Thread.sleep(5000); //sleep and then try again
          } catch (InterruptedException e) {
            break;
          }
        }
      }
    }
  });
  subscribeThread.start();
}
```

Gambar 9. Gambar *Subscriber Handler*

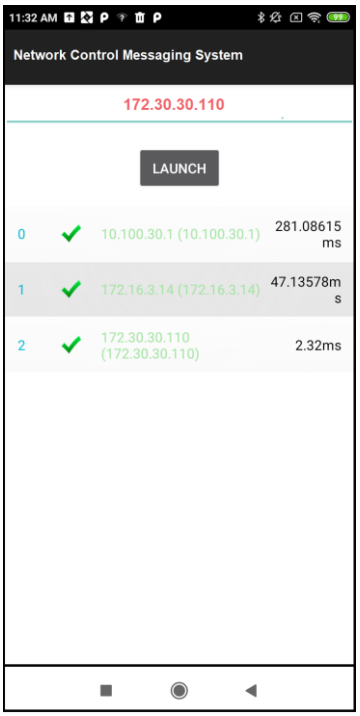
III. HASIL PENELITIAN

Hasil dan pembahasan uji coba pengujian fungsionalitas pada pengembangan sistem ini, antara lain:

1. Membuat simulasi konfigurasi *action & rules* pada router mikrotik untuk parameter limit bandwidth dengan cara menambahkan script pada menu system. Kemudian pada hasil eksekusinya berjalannya perintah script tersebut jika terdapat kondisi kapasitas bandwidth mencapai atau melebihi yang telah diatur, maka akan terbentuk log event yang kemudian oleh server diteruskan ke client berupa notifikasi melalui protocol amqp.

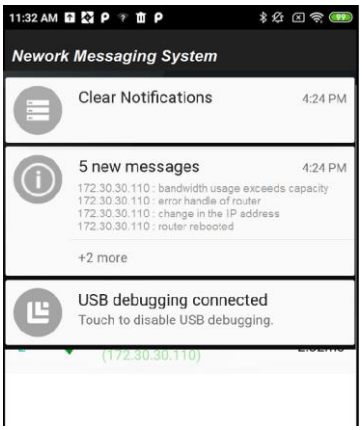
Gambar 10. *Script Limit Bandwidth*

2. Membuat konfigurasi *action & rules* pada router mikrotik untuk parameter perubahan IP dan *information error* melalui menambahkan fitur dengan nama "log" pada menu logging. Tentukan type adalah Remote dan remote address adalah alamat IP server yang telah ditentukan. Dan hasil dari prosesnya *action* ini yang akan menghubungan antara log dari router mikrotik ke server.
3. Jalankan aplikasi *Messaging System Notification* dan konfigurasi alamat server. Kemudian Menguji koneksi antara device android menuju server



Gambar 11. Aplikasi *Messaging System Notification*

4. Tampilan pesan notifikasi pada layar popup device android.



Gambar 12. Tampilan notifikasi layar android

IV. KESIMPULAN

Dari hasil pembuatan dan pengembangan sistem ini, dapat dirangkum beberapa kesimpulan antara lain:

1. Optimalisasi waktu dan penanganan *troubleshoot* Administrator Server bisa berjalan dengan baik dan meringankan pekerjaan admin server untuk mengkontrol jaringan server agar berjalan stabil.
2. Sistem dapat terintegrasi dan berjalan dengan maksimal pada device *Routerboard Mikrotik RB3011UIAS-RM*, dengan *OS Linux Ubuntu* 20.04 dan service *messages brokers* RabbitMQ. Dengan spesifikasi tersebut sistem juga dapat menjalankan simulasi disetiap parameter secara baik.
3. *Messaging System* berbasis Protocol AMQP dapat memfasilitasi integrasi tugas berupa *exchange messages* ke *queue* secara *asyncronous* kepada integrasi di sistem lainnya.
4. Proses *Messaging System* dengan Protocol AMQP dimulai dari *log event* router mikrotik yang dikirim ke server sebagai *Producers* kemudian memparsing pesan dan selanjutnya mengirim ke *brokers*. Terkahir *brokers* meneruskan ke *Consumers* pada perangkat mobile Android.

DAFTAR PUSTAKA

Marsh, G. A. S. D., 2009. Scaling Advanced Message Queuing Protocol (AMQP) Architecture with Broker Federation and InfiniBand. http://web.cse.ohio-state.edu/tech-report/2009/TR17.pdf

Putri, M.P., & Effendi, H., 2018, *Implementasi Metode Rapid Application Development Pada Website Service Guide "Waterfall Tour South Sumatera"*, Jurnal SISFOKOM, No. 2, Vol. 7, Hal. 130-136, Link: https://media.neliti.com/media/publications/265635-implementasi-metoderad-pada-website-ser-fa285f1d.pdf

Reza, P. Achmad, A. Eko, S. 2013. *Rancang Bangun Aplikasi Monitoring Jaringan Dengan Menggunakan Simple Network Management Protocol*. Jurnal Teknik ITS, Vol. 2 No. 1, Juni 2013. ISSN 2337-3539. Link:

https://ejurnal.its.ac.id/index.php/teknik/article/view/2265

Rifzan. 2019. *Mengenal Sejarah Perkembangan Jaringan Computer Secara Singkat*. https://www.robicomp.com/mengenal-sejarah-perkembangan-jaringan-computer-secarasingkat.html

Rouse. M,. 2018. Advanced Message Queuing Protocol (AMQP). Retrieved March 4, 2022. Link: https://whatis.techtarget.com/definition/Advanced-Message-Queuing-ProtocolAMQP

Santoso, E., Primananda, R., & Amron, K. *Implementasi Sistem Pemantauan Suhu dan Kelembapan Udara Berbasis Protokol AMQP*. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, vol. 3, no. 4, p. 3557-3562, jan. 2019. ISSN 2548-964X. Link: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/4973

Sukamto, R.A., & Shalahudin, M., 2016, *Rekayasa Perangkat Lunak*, Bandung: Informatika Bandung.

RabbitMQ Official. 2000. RabbitMQ for AMQP 0-9-1 Model Explained. Retrieved March 10, 2022, from https://www.rabbitmq.com/tutorials/amqp-concepts.html