Design and Implementation of Network Systems, Database, and GUI For General Users on Earthquake and Tsunami Detectors Systems Network Decision Support System

Christoporus Deo Putratama[1, a], Ary Setijadi Prihatmanto [2,b], Aciek Ida Wuryandari [3,c]
[1]*Teknik Elektro, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Bandung, 40132, Indonesia*
[2,3]*Laboratorium Sistem Kendali dan Komputer, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 40132, Indonesia*
[a]Email: christoporus.deo@students.itb.ac.id, chris.dp.41@gmail.com
[b]Email: asetijadi@lskk.ee.itb.ac.id
[c]Email: aciek@lskk.ee.itb.ac.id

***Abstract*—Earthquake Catcher Network (ECN) is the result of the development of earthquake and tsunami network detector and decision support system by using the movement sensors and location locating sensors that being integrated by the Internet network as a decision tools for evacuation. ECN is used as one of the solutions of cheap Earthquake and Tsunami Early Warning System. ECN consists of three parts, namely earthquake sensors, network systems and databases, and earthquake processing software. ECN can detect vibrations that exist in the soil surface. Data of accelerations are being combined with sensor location data and then will be transmitted over the internet network. The delivery protocol must be able to distinguish between sensors and can also transmit seismic data to both database servers and earthquake processing software, as well as to GUI software for general use. The results of data processing will be compared with data processing conducted by BMKG, and USGS. In its implementation, an earthquake sensor module will transmit seismic data in GeoJSON format. Data transmission is done by using over the internet network to RabbitMQ Messaging Server and then performed data consumption by MySQL based server software. Network testing is done by sending a timestamp to RabbitMQ then the data is consumed using a program. This program will give a second timestamp then both marks will be sent to the database. The desired result is the same second time data so as to prove that the data transmission from the earthquake sensor to the desired customer succeeds.**

***Index Terms*— Databases, Data Seismic, Earthquake Catcher Network, GUI.**

I. INTRODUCTION

Earthquake Catcher Network is a seismic detector network technology developed from previously created by Stanford University with the title Quake Catcher Network (QCN) [6]. Cheap earthquake detector technology and integrated with the Internet network allows researchers to obtain seismic data more accurately and also can improve the performance of earthquake dissemination. In this paper, the authors will focus on network systems and databases that serve as a bridge for earthquake sensors to be able to transmit seismic data required by researchers and can also be processed by software for common users. This network system involves connection from the sensor to the messaging server and also the database server.

II. DESIGN

A. Messaging Server in RabbitMQ

In this project, the communication protocol used is based Advanced Messaging Queuing Protocol (AMQP) [7][9]. This protocol is chosen because it has some features that match the system. AMQP is implemented by a RabbitMQ so it will use RabbitMQ with AMQP method as a system communication protocol. RabbitMQ itself is a message broker, where a message sent from the sender to the destination will be facilitated by RabbitMQ. Message protocols can be modified so that:

1) Message delivery can be sorted in a line queue;
2) Publishing messages to all parts of the network system;
3) Subscribe messages by the user to the broker;
4) Connecting and filtering messages;
5) Selection of messages to be obtained;
6) Remotely control the network by remote method.

These features can be beneficial because not all data obtained is important information and most likely the system will carry a message that can be affected by distortion due to bugs in the system. In addition, it is easier to control a large network system with this AMQP message protocol.
The block diagram of the RabbitMQ communication protocol system is as follows:

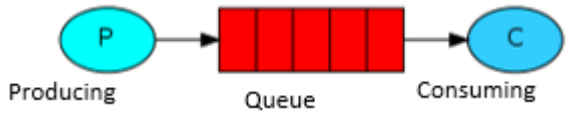


Fig. 1. The simplest RabbitMQ topology schematic server messaging.

At Network Systems Detectors Earthquake and Tsunami Decision Support System, connection topology to the Messaging Server RabbitMQ is as follows:

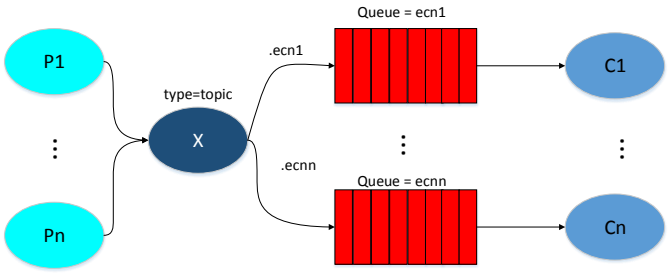


Fig. 2. schematic of network topology used on ECN.

P1 ... Pn is the first sensor to-n, then the sensor will send a message to the exchange with type: topic. These messages will then be sent to the queue from first to n to be stored in accordance with the routing path of each (.ecn) key. C1 to Cn is consumer that is client that need data from RabbitMQ. This consumer can be either a direct client or a database server.

B. *MySQL Server Database*

Data storage are in the form of a MySQL database [2] [4]. The MySQL database is created to store seismic data or parameters required for earthquake analysis. The use of MySQL is commonly used in implementing the database so that the design of the server on this project is using MySQL. Therefore, columns corresponding to seismic data will be created.

These data include:

a. Sensor + Timestamp name
b. Time Zone
c. Delivery time interval
d. Latitude
e. Longitude
f. Acceleration of x axis
g. Acceleration of y axis
h. Acceleration of z axis

Sensor name along with timestamp will become unique data so that it can be used as primary key for filling query in MySQL database. The design of the program to be able to store messages to the Database is as follows:

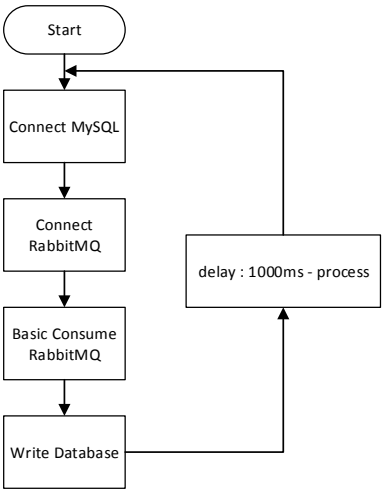


Fig 3. Flowchart of the message conversion program from RabbitMQ to MySQL

C. *Graphical User Interface for General Users*

Users can monitor the earthquake sensors by using a GUI that can display the acceleration motion graph of the sensor and also the location of the sensor is placed. The displays that will be made are as follows [3]:

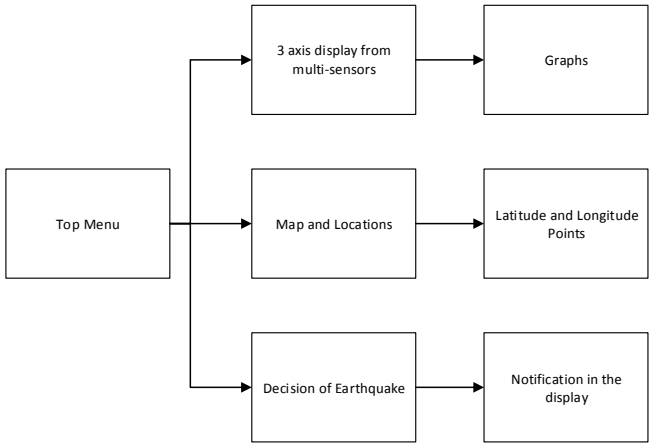


Fig. 4. Displays to be created on the GUI for General users

III. Implementation and Testing

Message delivery is done from the earthquake sensor to the messaging server. The protocol used by the sensor to send the message is MQTT, while the message is received by the RabbitMQ messaging server with the AMQP protocol. This can be done using the format of sending messages in the form of files with JSON extension so that the two protocols can communicate with each other. Then the messages that already stored in RabbitMQ will be consumed using the C# console program to then be entered into the MySQL database.

But the consumer in this system not only to be included in the database, but will also be consumed by a GUI program for general users who want to monitor data on earthquake sensors.

A. *Implementation and Testing of RabbitMQ Messaging Server*

RabbitMQ is implemented using the AMQP protocol as below:

amqp: // username: password @ server: port / host

The URL above is a parameter that should be added to the connection configuration of sending and retrieving data. Then, in addition to setting the server protocol as described above, the next setting will be explained as follows:

1. Queue → ecn + sensor number
   Naming queue is released but must be unique and specially allocated to a sensor.
2. Exchange → amq.topic
   Amq.topic is a default exchange with type = topic that can be used in the implementation process
3. Routing key → exchange + queue
   The routing key is meant to connect the exchange to a particular queue. It aims to sort messages according to the type and name of the sensor.
4. Durable → true

Durable aims to hold messages so as not to be lost when the server is off.

The implementation for the above parameters is as follows (by using C # programming language):

```
channel.QueueDeclare(queue: "ecn",
                     durable: true,
                     exclusive: false,
                     autoDelete: false,
                     arguments: null);
channel.QueueBind(queue: "ecn",
                  exchange: "amq.topic",
                  routingKey: "amq.topic.ecn"
                  );
```

After then, we tested by sending a message to the server. The sensor sends a message every 1 second to Messaging Server with a set WiFi network. This message is a geospatial information message with JSON format. The tester debugs the sensor by reading serial communication to know what the sensor is sending.

Here is the data generated by the sensor when successfully sending a message along with the debugging view of the generated message:

Fig. 5. Results of seismic data transmission from earthquake sensors to messaging server

Then the contents of the messages sent are as follows:

{ "pointTime": "2000-0-0T0:0:0Z","timeZone":"Asia/Jakarta","interval":"1000","clientID":"ECN-1","geojson" :{"geometry":{"type":"Point","coordinates": [-6.89,107.61]},"properties":{"name":"ITB"}},"accelerations": [{"x": -2.0661,"y": 4.8328,"z": 5.2528},{"x": -2.0859,"y": 4.8904,"z": 5.3096},{"x": -2.1068,"y": 4.9404,"z": 5.3676},{"x": -2.1281,"y": 4.9897,"z": 5.4209},{"x": -2.1516,"y": 5.0375,"z": 5.4726},{"x": -2.1710,"y": 5.0862,"z": 5.5196},{"x": -2.1899,"y": 5.1320,"z": 5.5701},{"x": -2.2104,"y": 5.1750,"z": 5.6138},{"x": -2.2282,"y": 5.2156,"z": 5.6570},{"x": -2.2458,"y": 5.2601,"z": 5.7015},{"x": -2.2648,"y": 5.3003,"z": 5.7426},{"x": -2.2820,"y": 5.3374,"z": 5.7857},{"x": -2.2963,"y": 5.3774,"z": 5.8231},{"x": -2.3130,"y": 5.4122,"z": 5.8600},{"x": -2.3297,"y": 5.4444,"z": 5.8985},{"x": -2.3458,"y": 5.4768,"z": 5.9321},{"x": -2.3593,"y": 5.5074,"z": 5.9641},{"x": -2.3726,"y": 5.5376,"z": 5.9999},{"x": -2.3879,"y": 5.5690,"z": 6.0290},{"x": -2.3978,"y": 5.5987,"z": 6.0641},{"x": -2.4110,"y": 5.6320,"z": 6.0913},{"x": -2.4203,"y": 5.6601,"z": 6.1175},{"x": -2.4339,"y": 5.6857,"z": 6.1457},{"x": -2.4411,"y": 5.7115,"z": 6.1718},{"x": -2.4504,"y": 5.7361,"z": 6.1995},{"x": -2.4614,"y": 5.7576,"z": 6.2212},{"x": -2.4707,"y": 5.7819,"z": 6.2485},{"x": -2.4815,"y": 5.8033,"z": 6.2715},{"x": -2.4923,"y": 5.8249,"z": 6.2971},{"x": -2.5006,"y": 5.8451,"z": 6.3182},{"x": -2.5088,"y": 5.8637,"z": 6.3422},{"x": -2.5156,"y": 5.8826,"z": 6.3636},{"x": -2.5221,"y": 5.9009,"z": 6.3845},{"x": -2.5288,"y": 5.9186,"z": 6.4063},{"x": -2.5368,"y": 5.9352,"z": 6.4273},{"x": -2.5464,"y": 5.9518,"z": 6.4467},{"x": -2.5534,"y": 5.9692,"z": 6.4634},{"x": -2.5586,"y": 5.9802,"z": 6.4793},{"x": -2.5656,"y": 5.9951,"z": 6.4893},{"x": -2.5717,"y": 6.0085,"z": 6.5053}]}

Fig. 6. Results of seismic data consumption of the messaging server

As the results above, messages sent by the sensor have an appropriate format and valid data. The validity of this message will be checked later by the consumer. In the implementation of the AMQP protocol on earthquake sensors, some configurations are made in order to convert from MQTT to AMQP. The earthquake sensor used is Node MCU Amica with AMQP message delivery protocol through internet network. The configuration is as follows:

mqtt_server	Variabel penampung alamat server yang akan diakses
server_topic	Variabel penampung nama exchange yang digunakan
mqtt_clientID	Variabel nama klien sensor gempa
mqtt_user	Variabel penampung user dan vhost dari messaging server
mqtt_password	Variabel penampung password messaging server

When performing data consumption from the server, a program must be created first. Programs created can use C # programming languages, as well as Python. The created program should be able to JSON deserialize or parse the data according to the message format used.

B. Implementasi dan Pengujian Database MySQL

In order to enter messages into MySQL, then first the data in the consume by using a program. The program to be created is a C # program that aims to do a consume on RabbitMQ and get messages. Then this message is sorted out according to the existing column in MySQL. The algorithm of the program is as follows:

```
con.ConnectionString = "server=" + server_host + ";user id=" + server_name + ";password=" + server_password + ";database=earthquake";

con.Open();
```

The above code is used to connect from a program to a MySQL server. Then to do the command to write data in the MySQL table, created the code as follows:

```
command_add.CommandText = "INSERT INTO store_id (point_time, time_zone_id, interval_id, client_id, lattitude_id, longitude_id) VALUES('" + data[0] + "', '" + data[1] + "', '" + data[2] + "', '" + data[3] + "', '" + data[4] + "', '" + data[5] + "')";

command_add.ExecuteNonQuery();
```

The above code using the programming language used is C #.

C. Testing the Network System Latency

Network systems on the ECN must be reliable to run. Therefore, the authenticity of the actual data must also be taken into account. Testing is done by sending a Timestamp message from a producer then match it with the timestamp from producer with consumer timestamp.

Messages in the Messaging Server are consumed by the C # console program and are combined with the consumer Timestamp. Both time stamps are then stored into the MySQL

Database and exported the data for further processing. The processing results are as follows:

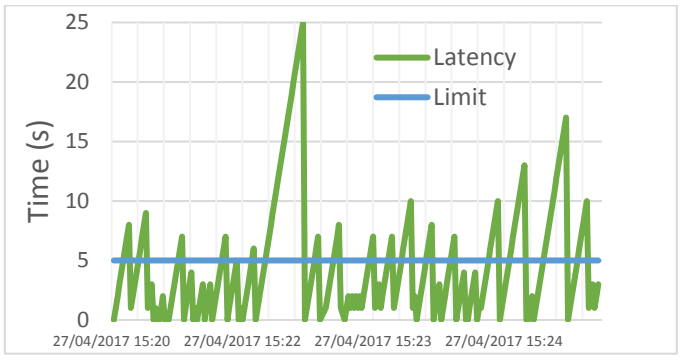


Fig. 7. graphs of data latency, where latency is the timeliness of lines of data to the server, and the line of the tolerance limit is the limit of data to be late

This means there are some pieces of data that is not real time because it is too late to get to the server processing. After processed the data, obtained the following results:

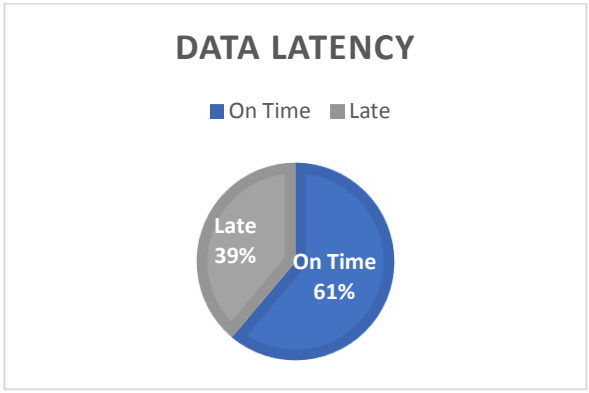


Fig. 8. The percentage of possible data experiences latency that cannot be tolerated

There is a 61% chance that the data transmitted is real-time data when through a network system.

D. GUI Implementation for Common Users

In accordance with the site planning design in the previous section, the GUI has been designed to have the following design [1] [8].

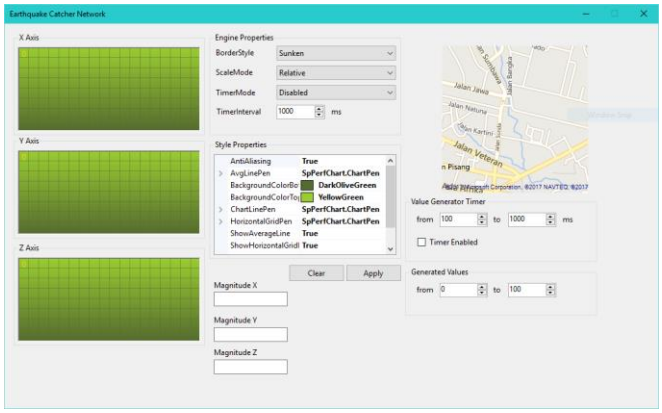


Fig. 9. GUI view for general users who can observe data acquisition on earthquake sensors

This view is the initial view of the graphical user interface of this earthquake detector system. Only data from one sensor is entered into the process that is behind this view. The data is an acceleration in the direction of x, y, and z, and position data displayed through the map in the top right corner. The sensor will be marked with a blue pin on the map. If more than one sensor is connected, there will be many blue pins on the map. Other buttons and bars are used to adjust the acceleration data graphic shown on the green box on the left. In addition, the magnitude of the vibrations on each axis is also shown quantitatively at the bottom.

The map is provided by Bing for free, via the gMap.Net project, a package of maps and GPS tailored for the .NET platform, including C #. The program will receive location data from each sensor in the form of latitude and longitude, and the program will then create a blue pin on the map whose location corresponds to the latitude and longitude received. Pin can be made on the map with an unlimited amount. Users can also shift the map so that other parts of the earth are visible on the screen, and also do enlarging or downsizing.

Graphical displays that represent accelerated data sent by the sensor can be arranged as follows. The three options at the top serve to adjust the speed of the graph. For example, some set the speed of the graph shifted to the left, some of which set the graph scale to scale up automatically, and also set the type of the graphic border, whether absolute or moving following the current chart value.

Test results from the GUI that have been made are as follows:

Fig. 10. GUI display of message delivery results from messaging server to GUI

From the picture above, seismic data are successfully consumed from the messaging server to the GUI and then displayed in accordance with the needs. Movement data of the earthquake sensor is shown on the left of the image where there is a graph. The earthquake sensor location data is displayed on the right side of the image where there is a map of the sensor location.

IV. CONCLUSION

The earthquake detector system requires a network system to store its data to a database and can also display a GUI that can conduct surveillance of seismic data acquisition so that it can be seen whether there is seismic vibration on the ground surface that can lead to earthquake. The use of RabbitMQ aims to bridge the movement of data from earthquake sensors with a large number to get to every consumer in need, be it a database server or data processing software earthquake. RabbitMQ settings are customized on the software according to the programming language used. MySQL database settings are also done so that the software can instantly send and save the seismic data with responsiveness. After performing the design and implementation, the experiment is done and can be taken some conclusions. Network system using RabbitMQ can be implemented and tested properly and in accordance with the desired. But in the process of data consumption, there is still delay in data acquisition so that it can lead to loss in terms of data accuracy.

REFERENCES

[1] "Designing A User Interface (Visual C#)". Msdn.microsoft.com. N.p., 2017. Web. 9 May 2017.

[2] DuBois, Paul. Mysql, Fourth Edition. 1st ed. Addison-Wesley Professional, 2008. Print.

[3] "Graphical User Interface". En.wikipedia.org. N.p., 2017. Web. 9 May 2017.

[4] "Mysql :: Mysql 5.7 Reference Manual :: 1.3.1 What Is Mysql?". Dev.mysql.com. N.p., 2017. Web. 9 May 2017.

[5] Perkins, Benjamin, Jacob Vibe Hammer, and Jon D Reid. Beginning C# 6 Programming with Visual Studio® 2015. 1st ed. Indianapolis, IN: Wrox, A Wiley Brand, 2016. Print.

[6] "QCN | The Quake-Catcher Network". Qcn.stanford.edu. N.p., 2017. Web. 12 May 2017.

[7] "Rabbitmq - Messaging That Just Works". Rabbitmq.com. N.p., 2017. Web. 9 May 2017.

[8] "Simple Performance Chart - Codeproject". Codeproject.com. N.p., 2017. Web. 9 May 2017.

[9] Videla, Alvaro, and Jason J. W Williams. Rabbitmq In Action: Distributed Messaging For Everyone. 1st ed. Manning Publications, 2012. Print.

Perancangan dan Implementasi Sistem Jaringan, Basis Data, dan GUI Untuk Pengguna Umum Pada Sistem Jaringan Detektor Gempa Dan Tsunami Decision Support System

Christoporus Deo Putratama[1, a], Ary Setijadi Prihatmanto [2,b], Aciek Ida Wuryandari [3,c]
[1]*Teknik Elektro, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Bandung, 40132, Indonesia*
[2,3]*Laboratorium Sistem Kendali dan Komputer, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 40132, Indonesia*
[a]Email: christoporus.deo@students.itb.ac.id, chris.dp.41@gmail.com
[b]Email: asetijadi@lskk.ee.itb.ac.id
[c]Email: aciek@lskk.ee.itb.ac.id

***Abstrak*—*Earthquake Catcher Network (ECN*) merupakan hasil pengembangan dari sistem jaringan detektor gempa dan tsunami decision support system dengan menggunakan sensor pergerakan dan sensor penentukan lokasi yang terintegrasi jaringan internet sebagai alat bantu keputusan untuk melakukan evakuasi. ECN digunakan sebagai salah satu solusi untuk Sistem Peringatan Awal Gempa dan Tsunami yang murah. ECN tediri dari tiga bagian, yaitu sensor gempa, sistem jaringan dan basis data, dan perangkat lunak pengolahan gempa. ECN dapat mendeteksi getaran yang ada di permukaan tanah. Data percepatan ini kemudian digabungkan dengan data lokasi sensor lalu dikirimkan melalui jaringan internet. Protokol pengiriman yang dilakukan harus dapat membedakan antar sensor dan juga dapat mengirimkan data seismik menuju server basis data maupun perangkat lunak pengolahan gempa, maupun menuju perangkat lunak GUI untuk penggunan umum. Pada implementasinya, sebuah modul sensor gempa akan mengirimkan data seismik melewati sebuah jaringan internet Pengiriman data dilakukan dengan menggunakan melalui jaringan internet menuju Messaging Server RabbitMQ dan kemudian dilakukan konsumsi data oleh perangkat lunak server berbasis MySQL. Pengujian jaringan dilakukan dengan mengirimkan sebuah tanda waktu menuju RabbitMQ kemudian data tersebut dikonsumsi dengan menggunakan sebuah program. Program ini akan memberikan tanda waktu kedua lalu kedua tanda tsb akan dikirimkan ke basis data. Hasil yang diinginkan adalah data kedua waktu ini sama sehingga membuktikan bahwa pengiriman data dari sensor gempa menuju konsumen yang diinginkan berhasil.**

***Kata kunci*— Basis Data, Data Seismik, *Earthquake Catcher Network*, Jaringan, GUI**

I. PENDAHULUAN

Earthquake Catcher Network (ECN) merupakan pengembangan teknologi detektor gempa dalam jaringan yang dibuat dari teknologi yang sebelumnya dibuat oleh Stanford University dengan judul *Quake Catcher Network* (QCN)[6]. Teknologi detektor gempa yang murah dan diintegrasikan dengan jaringan internet memungkinkan peneliti untuk memperoleh data seismik dengan lebih akurat dan juga dapat meningkatkan performa diseminasi gempa. Dalam tulisan ini, penulis akan berfokus pada sistem jaringan dan basis data yang menjadi jembatan bagi sensor gempa untuk dapat mengirimkan data seismik yang dibutuhkan oleh peneliti dan juga dapat diolah sebagai perangkat lunak untuk pengguna umum. Sistem jaringan ini melibatkan koneksi dari sensor menuju *server* pengirim pesan dan juga *server* basis data.

II. PERANCANGAN

A. *Messaging Server Pada RabbitMQ*

Pada proyek ini, protokol komunikasi yang digunakan adalah berbasis Advanced Messaging Queueing Protocol (AMQP) [7][9]. Protokol ini dipilih karena mempunyai beberapa fitur yang sesuai dengan sistem yang digunakan. AMQP diterapkan oleh sebuah *vendor* komunikasi RabbitMQ sehingga akan digunakan RabbitMQ dengan metode AMQP sebagai protokol komunikasi sistem. RabbitMQ sendiri merupakan *broker* pesan, dimana sebuah pesan yang dikirimkan dari pengirim menuju tujuan akan difasilitasi oleh RabbitMQ. Protokol pesan dapat dimodifikasi sehingga:

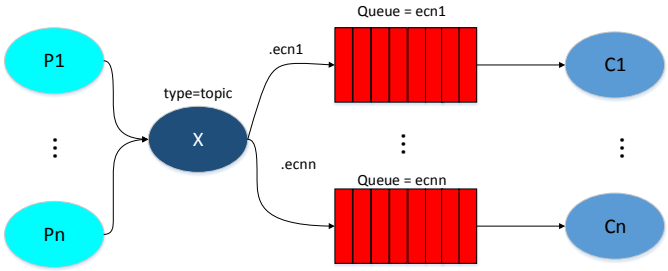
1) Pengiriman pesan dapat diurutkan dalam sebuah antrian baris;
2) Melakukan penerbitan pesan ke seluruh bagian sistem jaringan;
3) Melakukan langganan pesan oleh pengguna pada broker;
4) Melakukan penghubungan dan penyaringan pesan;
5) Seleksi pesan yang akan diperoleh;
6) Kendali jarak jauh pada jaringan dengan metode remote.

Fitur-fitur diatas dapat menguntungkan karena tidak semua data yang diperoleh merupakan informasi yang penting dan besar kemungkinan pada sistem bawa sebuah pesan dapat terkena distorsi akibat bug pada sistem. Selain itu mudah untuk mengendalikan sebuah sistem jaringan yang besar dengan protokol pesan AMQP ini.

Diagram blok dari sistem protokol komunikasi RabbitMQ adalah sebagai berikut:

Gmbr. 1. skematik topologi messaging server di RabbitMQ yang paling sederhana.

Pada Sistem Jaringan Detektor Gempa dan Tsunami *Decision Support System*, topologi koneksi ke *Messaging Server* RabbitMQ adalah sebagai berikut:



Gmbr. 2. skematik topologi jaringan yang digunakan pada ECN.

Dimana P1…Pn adalah sensor pertama hingga ke -n, kemudian *sensor* akan mengirimkan pesan menuju *exchange* dengan *type: topic*. Pesan-pesan ini kemudian akan dikirimkan menuju *queue* dari yang pertama hingga ke-n untuk disimpan sesuai dengan jalur *routing key* .ecn masing-masing. C1 hingga Cn merupakan *consumer* yaitu klien yang membutuhkan data dari RabbitMQ. *Consumer* ini dapat berupa klien langsung ataupun sebuah server basis data.

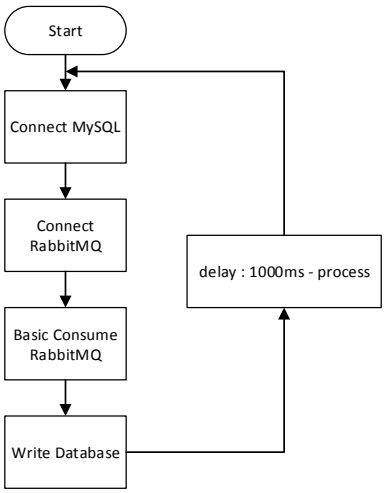
B. *Database Server MySQL*

Pada *server* terdapat media penyimpan data berupa *database* MySQL [2][4]. *Database* MySQL dibuat untuk menyimpan data GeoJSON atau parameter-parameter yang dibutuhkan untuk analisa gempa. Penggunaan MySQL umum digunakan dalam mengimplementasikan *database* sehingga desain *server* pada proyek ini menggunakan MySQL. Oleh karena itu, akan dibuat kolom-kolom yang sesuai dengan data GeoJSON.

Data-data tersebut meliputi:

a. Nama sensor + Timestamp
b. Time Zone
c. Interval waktu pengiriman
d. Garis Lintang
e. Garis Bujur
f. Percepatan sumbu x
g. Percepatan sumbu y
h. Percepatan sumbu z

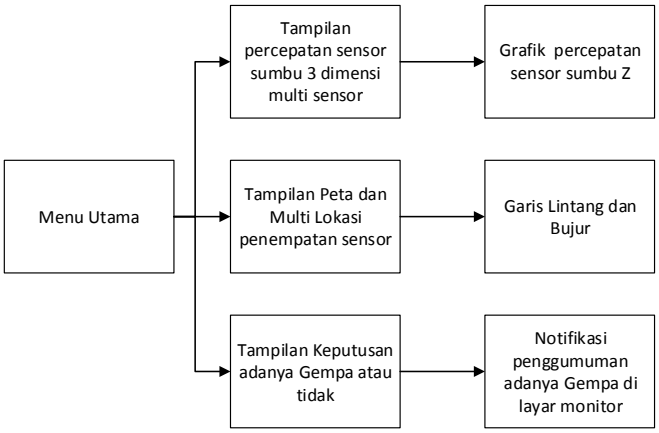
Nama sensor beserta *timestamp* akan menjadi data yang unik sehingga bisa dijadikan *primary key* untuk pengisian *query* pada *database* MySQL. Desain program untuk dapat melakukan penyimpanan pesan menuju *Database* adalah sebagai berikut:



Gmbr 3. *Flowchart* dari program konversi pesan dari RabbitMQ menuju MySQL

C. *Graphical User Interface untuk Pengguna Umum*

Pengguna dapat melakukan *monitoring* pada sensor gempa dengan menggunakan GUI yang dapat menampilkan grafik gerakan akselerasi sensor dan juga letak sensor diletakkan. Tampilan yang akan dibuat adalah sebagai berikut[3]:



Gmbr. 4. Tampilan yang akan dibuat pada GUI untuk pengguna Umum

III. IMPLEMENTASI DAN PENGUJIAN

Pengiriman pesan dilakukan dari *sensor* gempa menuju *messaging server*. Protokol yang digunakan oleh sensor untuk mengirim pesan adalah MQTT, sedangkan pesan diterima oleh *messaging server* RabbitMQ dengan *protocol* AMQP. Hal ini dapat dilakukan dengan menggunakan *format* pengiriman pesan berupa *file* dengan ekstensi JSON sehingga kedua protokol tersebut dapat saling berkomunikasi satu sama lain. Kemudian pesan yang sudah tersimpan dalam RabbitMQ akan di *consume* menggunakan *console* C# untuk kemudian dimasukkan ke dalam *database* MySQL.

Namun *consumer* yang ada dalam sistem ini bukan hanya untuk dimasukkan ke dalam database, tetapi akan di konsumsi pula oleh sebuah *program* GUI bagi pengguna umum yang ingin melakukan *monitoring* data pada sensor gempa.

A. Implementasi dan Pengujian Messaging Server RabbitMQ

RabbitMQ diimplementasikan dengan menggunakan protokol AMQP seperti dibawah ini:

```
amqp://username:password@server:port/host
```

URL diatas merupakan merupakan parameter yang harus ditambahkan pada proses koneksi pengiriman dan pengambilan data. Kemudian selain mengatur protokol server seperti yang dijelaskan diatas, pengaturan berikutnya akan dijelaskan sebagai berikut:

1) *queue* → ecn + nomer sensor
   penamaan queue dibebaskan namun harus unik dan khusus dialokasikan untuk sebuah *sensor.*
2) *exchange* → amq.topic
   amq.topic merupakan sebuah *exchange default* dengan *type = topic* yang dapat dipakai dalam proses implementasi
3) *routing key → exchange + queue*
   routing key dimaksudkan untuk menghubungkan exchange menuju queue tertentu. Hal ini bertujuan untuk memilah pesan sesuai dengan jenis dan nama sensornya.
4) *durable → true*
   durable bertujuan untuk menahan pesan supaya tidak hilang ketika *server* mati.

Sehingga implementasi untuk parameter-parameter diatas adalah sebagai berikut (dengan menggunakan bahasa pemograman C#):

```
channel.QueueDeclare(queue: "ecn",
                     durable: true,
                     exclusive: false,
                     autoDelete: false,
                     arguments: null);
channel.QueueBind(queue: "ecn",
                  exchange: "amq.topic",
                  routingKey: "amq.topic.ecn"
                  );
```

Kemudian dilakukan pengujian dengan mengirimkan sebuah pesan menuju *server* tersebut. Sensor mengirimkan sebuah pesan setiap 1 detik menuju *Messaging Server* dengan jaringan WiFi yang sudah di *set*. Pesan ini merupakan pesan informasi geospasial dengan *format* JSON. Penguji melakukan *debugging* pada *sensor* dengan membaca komunikasi *serial* sehingga mengetahui apa yang dikirimkan oleh sensor.

Berikut ini data yang dihasilkan oleh *sensor* saat berhasil mengirimkan sebuah pesan beserta tampilan *debugging* dari pesan yang dihasilkan:

Gmbr. 5. Hasil pengiriman data seismik dari sensor gempa menuju messaging server

Kemudian isi pesan yang dikirimkan adalah sebagai berikut:

{ "pointTime": "2000-0-0T0:0:0Z","timeZone":"Asia/Jakarta","interval":"1000","clientID":"ECN-1","geojson" :{"geometry":{"type":"Point","coordinates": [-6.89,107.61]},"properties":{"name":"ITB"}},"accelerations": [{"x": -2.0661,"y": 4.8328,"z": 5.2528},{"x": -2.0859,"y": 4.8904,"z": 5.3096},{"x": -2.1068,"y": 4.9404,"z": 5.3676},{"x": -2.1281,"y": 4.9897,"z": 5.4209},{"x": -2.1516,"y": 5.0375,"z": 5.4726},{"x": -2.1710,"y": 5.0862,"z": 5.5196},{"x": -2.1899,"y": 5.1320,"z": 5.5701},{"x": -2.2104,"y": 5.1750,"z": 5.6138},{"x": -2.2282,"y": 5.2156,"z": 5.6570},{"x": -2.2458,"y": 5.2601,"z": 5.7015},{"x": -2.2648,"y": 5.3003,"z": 5.7426},{"x": -2.2820,"y": 5.3374,"z": 5.7857},{"x": -2.2963,"y": 5.3774,"z": 5.8231},{"x": -2.3130,"y": 5.4122,"z": 5.8600},{"x": -2.3297,"y": 5.4444,"z": 5.8985},{"x": -2.3458,"y": 5.4768,"z": 5.9321},{"x": -2.3593,"y": 5.5074,"z": 5.9641},{"x": -2.3726,"y": 5.5376,"z": 5.9999},{"x": -2.3879,"y": 5.5690,"z": 6.0290},{"x": -2.3978,"y": 5.5987,"z": 6.0641},{"x": -2.4110,"y": 5.6320,"z": 6.0913},{"x": -2.4203,"y": 5.6601,"z": 6.1175},{"x": -2.4339,"y": 5.6857,"z": 6.1457},{"x": -2.4411,"y": 5.7115,"z": 6.1718},{"x": -2.4504,"y": 5.7361,"z": 6.1995},{"x": -2.4614,"y": 5.7576,"z": 6.2212},{"x": -2.4707,"y": 5.7819,"z": 6.2485},{"x": -2.4815,"y": 5.8033,"z": 6.2715},{"x": -2.4923,"y": 5.8249,"z": 6.2971},{"x": -2.5006,"y": 5.8451,"z": 6.3182},{"x": -2.5088,"y": 5.8637,"z": 6.3422},{"x": -2.5156,"y": 5.8826,"z": 6.3636},{"x": -2.5221,"y": 5.9009,"z": 6.3845},{"x": -2.5288,"y": 5.9186,"z": 6.4063},{"x": -2.5368,"y": 5.9352,"z": 6.4273},{"x": -2.5464,"y": 5.9518,"z": 6.4467},{"x": -2.5534,"y": 5.9692,"z": 6.4634},{"x": -2.5586,"y": 5.9802,"z": 6.4793},{"x": -2.5656,"y": 5.9951,"z": 6.4893},{"x": -2.5717,"y": 6.0085,"z": 6.5053}]}

Gmbr. 6. Hasil konsumsi data seismik dari messaging server

Pesan yang terkirim oleh sensor memiliki *format* yang sesuai dan data yang *valid*. Validitas pesan ini akan dicek kemudian oleh konsumer.

Pada implementasi protokol AMQP pada sensor gempa, dilakukan beberapa konfigurasi agar dapat melakukan konversi dari MQTT menuju AMQP. Sensor gempa yang digunakan adalah *Node MCU Amica* dengan protokol pengiriman pesan AMQP melalui jaringan internet. Konfigurasinya adalah sebagai berikut:

mqtt_server	Variabel penampung alamat server yang akan diakses
server_topic	Variabel penampung nama exchange yang digunakan
mqtt_clientID	Variabel nama klien sensor gempa
mqtt_user	Variabel penampung user dan vhost dari messaging server
mqtt_password	Variabel penampung password messaging server

Apabila melakukan konsumsi data dari server, sebuah program harus dapat dibuat terlebih dahulu. *Program* yang dibuat dapat menggunakan bahasa pemograman C#, maupun Python. Program yang dibuat harus dapat melakukan deserialisasi JSON atau *parsing* data sesuai dengan format pesan yang digunakan.

B. *Implementasi dan Pengujian Database MySQL*

Supaya dapat memasukkan pesan kedalam MySQL, maka terlebih dahulu data di *consume* dengan menggunakan sebuah *program*. *Program* yang akan dibuat adalah *program* C# yang bertujuan untuk melakukan *consume* pada RabbitMQ dan mendapatkan pesan. Lalu pesan ini dipilah-pilah sesuai dengan kolom yang ada pada MySQL. Algoritma dari program tersebut adalah sebagai berikut:

```
con.ConnectionString = "server=" + server_host + ";user id=" + server_name + ";password=" + server_password + ";database=earthquake";

con.Open();
```

Kode di atas digunakan untuk melakukan proses koneksi dari sebuah program menuju *server* MySQL. Kemudian untuk melakukan perintah untuk menulis data pada tabel MySQL, dibuat kode seperti berikut:

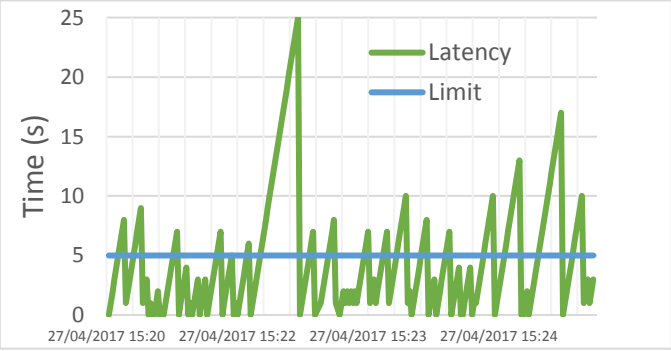
```
command_add.CommandText = "INSERT INTO store_id (point_time, time_zone_id, interval_id, client_id, lattitude_id, longitude_id) VALUES('" + data[0] + "', '" + data[1] + "', '" + data[2] + "', '" + data[3] + "', '" + data[4] + "', '" + data[5] + "')";

command_add.ExecuteNonQuery();
```

Kode diatas menggunakan bahasa pemograman yang digunakan adalah C#.

C. *Pengujian Latency Sistem Jaringan*

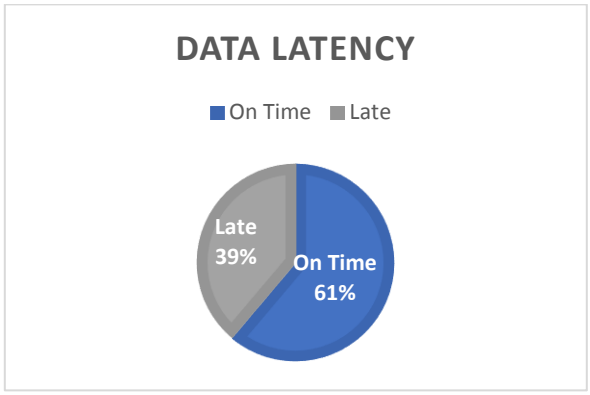
Sistem jaringan pada ECN harus dapat berjalan dengan handal. Oleh karena itu, keaslian data secara aktual juga harus diperhitungkan. Pengujian dilakukan dengan mengirimkan sebuah pesan *Timestamp* dari sebuah producer kemudian mencocokkan *Timestamp* dari produser dengan *Timestamp* konsumer.

Pesan di dalam *Messaging Serve*r di *consume* oleh *console* C# dan digabungkan dengan *Timestamp consumer*. Kedua time stamp ini kemudian disimpan ke dalam Database MySQL dan di *export* datanya untuk pengolahan lebih lanjut. Hasil pengolahannya adalah sebagai berikut:



Gmbr. 7. grafik latency data, dimana garis *latency* adalah ketepatan waktu data sampai ke server, dan garis *limit* adalah batas toleransi data untuk tidak terlambat

Artinya terdapat beberapa buah data yang tidak *real time* karena terlambat untuk sampai ke pengolahan server. Setelah diolah datanya, didapatkan hasil sebagai berikut:

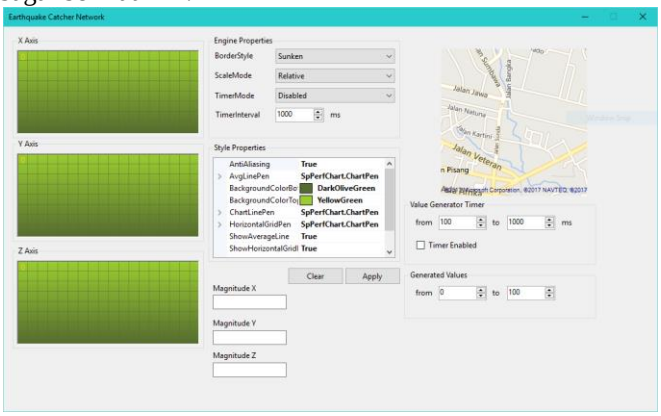


Gmbr. 8. persentase kemungkinan data mengalami latency yang tidak dapat ditoleransi

Terdapat peluang sebesar 61% bahwa data yang dikirimkan adalah data *real-time* ketika melalui sistem jaringan.

D. *Implementasi GUI untuk Pengguna Umum*

Sesuai dengan perecanaan perancangan pada subbab sebelumnya, maka GUI yang sudah dirancang memiliki desain sebagai berikut [1][8]:



Gmbr. 9. tampilan GUI untuk pengguna umum yang dapat mengamati akuisisi data pada sensor gempa

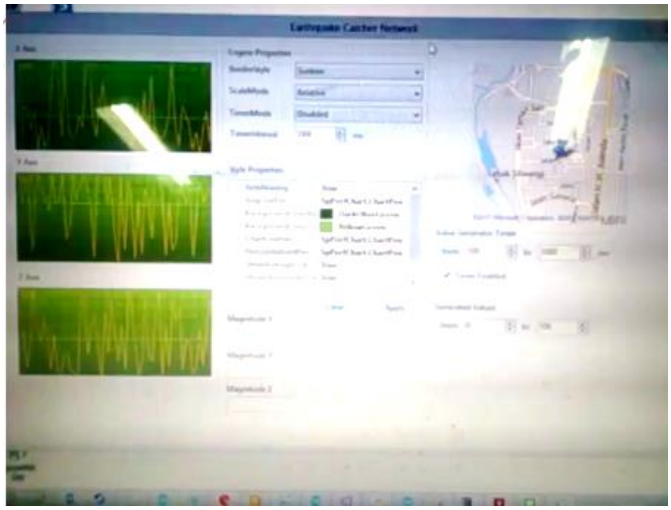
Tampilan ini adalah tampilan awal dari antarmuka pengguna grafis sistem detektor gempa ini. Hanya data dari satu sensor yang dimasukkan ke dalam proses yang ada dibalik tampilan ini. Data tersebut berupa percepatan ke arah x,y,dan z, serta data posisi yang ditampilkan melalui peta di pojok kanan atas. Sensor akan ditandai dengan pin berwarna biru di peta tersebut. Bila ada lebih dari satu sensor yang terhubung, maka akan ada banyak pin biru di peta. Tombol dan bar lainnya digunakan untuk melakukan pengaturan terhadap grafik data percepatan yang ditampilkan pada kotak hijau di bagian kiri. Selain itu, magnituda dari getaran pada masing-masing sumbu juga ditampilkan secara kuantitatif di bagian bawah.

Peta tersebut disediakan oleh Bing secara gratis, melalui proyek gMap.Net, yaitu paket peta dan gps yang khusus dibuat untuk *platform* .NET, termasuk didalamnya C#. Program akan

menerima data lokasi dari masing-masing sensor berupa lintang dan bujur, dan program lalu akan membuat sebuah pin biru di peta yang lokasinya sesuai dengan lintang dan bujur yang diterima. Pin dapat dibuat pada peta dengan jumlah yang tidak terbatas. Pengguna juga dapat menggeser peta sehingga bagian lain dari bumi yang terlihat pada layar, dan juga melakukan pembesaran atau pengecilan.

Tampilan grafik yang merepresentasikan data percepatan yang dikirim sensor bisa diatur sebagai berikut. Tiga pilihan yang terletak pada bagian atas berfungsi untuk mengatur kecepatan dari grafik. Sebagai contoh, ada yang mengatur kecepatan grafik bergeser ke kiri, ada yang mengatur pembesaran/pengecilan skala grafik secara otomatis, dan juga mengatur jenis dari perbatasan grafik, apakah absolut atau bergerak mengikuti nilai grafik yang sekarang.

Hasil pengujian dari GUI yang sudah dibuat adalah sebagai berikut:



Gmbr. 10. tampilan GUI hasil pengujian pengiriman pesan dari messaging server menuju GUI

Dari gambar diatas, data seismik berhasil dikonsumsi dari *messaging server* menuju GUI lalu ditampilkan sesuai dengan kebutuhan. Data pergerakan sensor gempa ditampilkan pada bagian kiri gambar dimana terdapat grafik. Data lokasi sensor gempa ditampilkan pada bagian kanan gambar dimana terdapat peta lokasi sensor.

IV. KESIMPULAN

Sistem detektor gempa membutuhkan sebuah sistem jaringan untuk menyimpan datanya ke sebuah basis data dan juga dapat menampilkan GUI yang dapat melakukan pengawasan terhadap akuisisi data seismik sehingga dapat dilihat bahwa apakah terjadi getaran seismik pada permukaan tanah yang dapat mengakibatkan gempa. Penggunaan RabbitMQ bertujuan untuk menjembatani perpindahan data dari sensor gempa dengan jumlah yang banyak hingga dapat sampai ke setiap konsumen yang membutuhkan, baik itu sebuah *server* basis data maupun perangkat lunak pengolahan data gempa. Pengaturan RabbitMQ disesuaikan pada perangkat lunak sesuai dengan bahasa pemograman yang digunakan. Pengaturan basis data MySQL juga dilakukan agar perangkat lunak dapat langsung mengirim dan menyimpan data seismik dengan responsif. Setelah melakukan perancangan dan implementasi, dilakukan uji coba dan bisa diambil beberapa kesimpulan. Sistem jaringan menggunakan RabbitMQ dapat diimplementasikan dan diuji dengan baik dan sesuai dengan yang diinginkan. Namun pada proses konsumsi data, masih terdapat keterlambatan pemerolehan data sehingga dapat mengakibatkan kerugian dalam segi ketepatan data.

REFERENCES

[1] "Designing A User Interface (Visual C#)". Msdn.microsoft.com. N.p., 2017. Web. 9 May 2017.

[2] DuBois, Paul. Mysql, Fourth Edition. 1st ed. Addison-Wesley Professional, 2008. Print.

[3] "Graphical User Interface". En.wikipedia.org. N.p., 2017. Web. 9 May 2017.

[4] "Mysql :: Mysql 5.7 Reference Manual :: 1.3.1 What Is Mysql?". Dev.mysql.com. N.p., 2017. Web. 9 May 2017.

[5] Perkins, Benjamin, Jacob Vibe Hammer, and Jon D Reid. Beginning C# 6 Programming with Visual Studio® 2015. 1st ed. Indianapolis, IN: Wrox, A Wiley Brand, 2016. Print.

[6] "QCN | The Quake-Catcher Network". Qcn.stanford.edu. N.p., 2017. Web. 12 May 2017.

[7] "Rabbitmq - Messaging That Just Works". Rabbitmq.com. N.p., 2017. Web. 9 May 2017.

[8] "Simple Performance Chart - Codeproject". Codeproject.com. N.p., 2017. Web. 9 May 2017.

[9] Videla, Alvaro, and Jason J. W Williams. Rabbitmq In Action: Distributed Messaging For Everyone. 1st ed. Manning Publications, 2012. Print.