Encryption-Decryption Module, Server, and Scheduling Algorithm for FMCS

Shah Dehan Lazuardi[1], Arief Syaichu R.[2], Arif Sasongko[3], Agung Darmawan[4]

Electrical Engineering Field Study-STEI

Institut Teknologi Bandung

Ganeca 10, Bandung, 40135, Jawa Barat, Indonesia

[1]dehanlazuardi@students.itb.ac.id, [2]Arief@stei.itb.ac.id, [3]asasongko@gmail.com, [4]agungdar68@yahoo.com

Abstrac— **Scheduling is become main problem of public transportation and bus. Public transportation and bus operate in unfixed schedule depends on traffic condition. Scheduling problem can be solved by using scheduling system in Fleet Monitoring & Controlling System for Guided Bus. Scheduling system keeps each fleet on their schedule. Scheduling system needs position as input and calculate the speed of each fleet so they will stick to their schedule. The encrypted Data will be sent to the server them from each fleet by using MQTT protocols then the encrypted Data will be decrypted in GUI and parsed so the position data can be calculated by scheduling algorithm. In the system testing, encrypted data can be sent to server than those data can be decrypted in GUI and scheduling algorithm can keep fleet on their schedule.**

Keywords - decrypt, encrypt, FMCS, MQTT, Schedulling.

I. INDTRODUCTION

Indonesia is one of the fourth most densely populated countries in the world, after China, India and the United States. The population of Indonesia, 257 million people, is directly proportional to the use of private and public vehicles. The use of private vehicles is increasing every year. Based on data released by the Central Bureau of Statistics (BPS) of DKI Jakarta Province 2015 related to transportation statistics, the number of vehicles passing through DKI Jakarta Province continues to increase during 2010-2014 period, with an average increase of 9.93% per year. The highest contributor to the percentage increase was motorcycle vehicles with an average annual increase of 10.54%, followed by an increase in passenger car percentage of 8.75%. In contrast to the growth in the number of public transport which in 2013-2014 only increased by 1.74%.

Increasing the number of vehicles not in imbangin with services provided. The services provided by public transport are less satisfactory, making public transport users turn to private transportation. The city, bus and bajaj transport have a weak appeal to mobilize the community to move modes of transportation. Especially buses and city transport waiting for full passengers sometimes take a long time. Not infrequently buses and urban transport violate traffic signs such as stop where it is prohibited to stop. This is exacerbated by the lack of good urban arrangement and traffic system makes public transportation uncomfortable to use because it does not have a fixed schedule.

From these problems the community needs a transportation system that has a fixed scheduling, the system can monitor and give orders on the fleet to be on schedule so that people can use public transport comfortably without wasted time.

II. MQTT AND AES

Here are some supporting theories for design and implement subsystems of FMCS.

A. *Message Queuing Telemetry Transport (MQTT)*

MQTT protocol is a special protocol designed for communication machine to machine or simply for communication with devices or machines that do not have a special address, designed for areas / locations with limited network resources.

MQTT protocol has the ability to publish and subscribe so that it can be used for two-way communication either between servers or with other devices. There are three types of QoS Level in MQTT.

- QoS level 0
  Message will be sent without any acknowledge or commonly called fire and forget, there is no guarantee the message will be received by the client and the sender also do not know whether the message was received or not.
- QoS level 1
  Messages will be guaranteed to the client / revceiver at least once if using QoS level 1, but if there are constraints in the network & timeout, messages may be sent more than once. The sender will retain the message until it receives a PUBACK response from the recipient, if the sender does not receive PUBACK within a certain time, the message will be sent again.
- QoS level 2
  The highest QoS in MQTT is level 2, the message will be guaranteed received by recipient only once. This QoS level is the safest method of sending messages but also the slowest because it is waiting response from sender and receiver. The sender can also tell whether a message has been sent or not.

B. *Advanced Encryption Standard (AES)*

Advanced Encryption Standard (AES) is a cryptographic algorithm that can be used to secure sensitive data. The AES algorithm is a symmetric chipertext block that can encipher and decipher information. Encryption converts data to text that can't be read called ciphertext; Otherwise the decipher is conert ciphertext into its original form that we know as plaintext. The AES algorithm uses cryptographic keys 128, 192, and 256 bits to encrypt and decrypt data on blocks of 128 bits.

III. Design and Implementation

In the FMCS the author designs and implements three parts: information security, server implementation, and scheduling algorithm. The specifications of the three parts are

- Server spesification
  a. Server could forward the received message.
  b. the server latency is less than 4 ms.
- Scheduling Algortithm specification
  a. Scheduling algorithm could maintain fleet according to schedule
- Encryption-Decryption modul specification
  a. Message between fleet and control station can only be read only by them.

A. Server

The server is located on Control Station. The GUI will access the server using the local network while the fleet hardware will access the server over the internet network using a particular protocol.

1. Server's Computer Design

The assumption used in the design of this server is that there are 40 fleets that access the server every 0.7 seconds. According benchmark done by Scalagent[1] that used following specification as server's computer:

- Intel Core 2 Duo CPU E8400 3.00GHz
- 4 GB RAM
- Gigabit switch

CPU usage when receiving 2000 message / second is 4% with latency 4 ms. Then the computer specs

- Intel i5-4300M 2.5 Ghz
- 8 GB RAM
- FS1104 fiberhome router

Dcan handle load as big as the assumptio.

2. Communication Protocol Design

In the design of server communication protocol used assumptions:

- The network used by the fleet is unstable
- Limited fleet hardware computing capability
-

Jenis	MQTT	HTTP
Transport	TCP	TCP
Messaging	Publish/Subscribe	Request/Response
Protocol	28 bytes	491 bytes
QoS	v	x
tipe	Asychronous	Synchronous
Resources Utilization	low	high
Latency	Millisecond [1]	second[4]

As description above MQTT is selected as server's protocol.

3. Server's Impelementation

There are several steps to install MQTT server:

- Download the Mosquitto installer and install on the server computer.
- After that install OpenSSL and copy libeay32.dll and ssleay32.dll from the openSSL directory to the mosquitto directory.
- Copy pthreadVC2.dll to mosquito directory, at this time server has been installed.

To run the server click the mosquito icon on the directory where the server is installed.

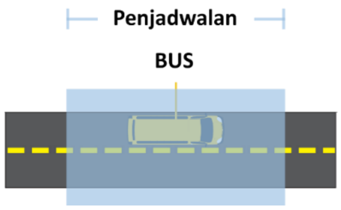
Once the server is installed do port forwarding on the router corresponds to the port used for the MQTT server so that the server can be accessed via the internet network.

B. Sheduling Algorithm

The scheduling algorithm maintain fleet scheduling. This algorithm aims to spread the fleet evenly on the route lane so passengers do not have to wait for the fleet for too long.

1. Scheduling Design

The scheduling algorithm will determine the position of the fleet scheduling. Scheduling is a range of positions if the fleet is located in the range then the fleet is on schedule.



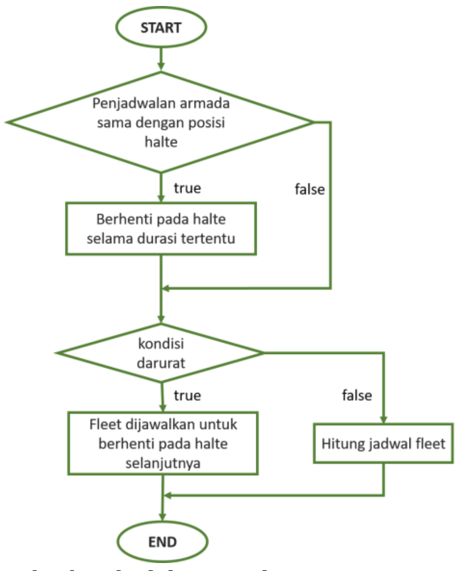
If there is a fleet that does not match the position specified then this algorithm will calculate the difference of the fleet and send the command in the form of speed to be followed by the fleet driver to match scheduling position.

If there is an emergency condition on one of the fleets, this algorithm will stop the scheduling of the fleet at the nearest stop in front of the fleet. When the emergency has been fixed then the scheduling will recalculate the scheduling distance of each fleet so that the fleet will be spread evenly.

The scheduling algorithm design is.

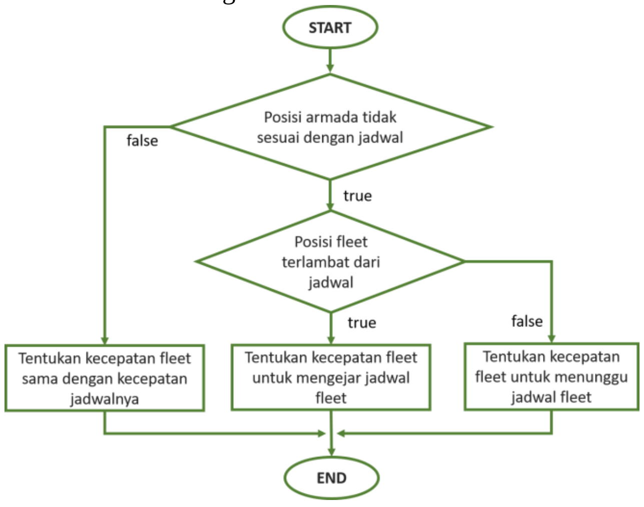
- Scheduling procedure

  This procedure will set the schedule for fleet every second.


START
Penjadwalan armada sama dengan posisi halte
true
false
Berhenti pada halte selama durasi tertentu
kondisi darurat
true
false
Fleet dijawalkan untuk berhenti pada halte selanjutnya
Hitung jadwal fleet
END

- Check Schedule Procedure

  This procedure will check the position of each fleet against its schedule.


START
Posisi armada tidak sesuai dengan jadwal
false
true
Posisi fleet terlambat dari jadwal
true
false
Tentukan kecepatan fleet sama dengan kecepatan jadwalnya
Tentukan kecepatan fleet untuk mengejar jadwal fleet
Tentukan kecepatan fleet untuk menunggu jadwal fleet
END

- Emergency Procedure

  This procedure is called when one fleet is in an emergency condition and the fleet driver presses the emergency button on the fleet hardware. This procedure will determine the nearest stop so in an emergency situation fleet stops at the nearest stop.

- Calculate schedule procedure

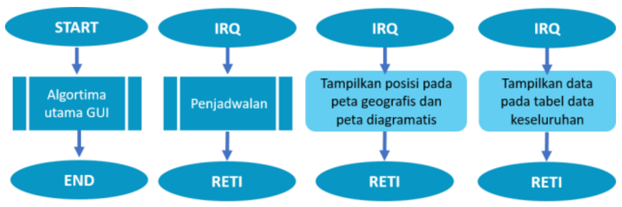
  This procedure is called after an emergency situation can be overcome. This procedure determines the position of the new scheduling array.

- Check scjeduling procedure

  This procedure works when the emergency is over. This procedure will keep the scheduling position from resting.

2. Scheduling algorithm implementation

The main GUI algorithm has flowchart as described below.


START
Algortima utama GUI
END
IRQ
Penjadwalan
RETI
IRQ
Tampilkan posisi pada peta geografis dan peta diagramatis
RETI
IRQ
Tampilkan data pada tabel data keseluruhan
RETI

The implementation of scheduling algorithm is as follows.

- Scheduling procedure

```
If posisi penjadwalan = posisi halte
        counter durasi halte++
        If counter durasi halte > durasi berhenti
                posisi penjadwalan ++
Else
        If kondisi darurat = true
                posisi penjadwalan ++ sampai pada halte terdekat di depan armada
        else
                posisi penjadwalan ++
```

This procedure is called through an interrupt every 200ms so that the scheduling position moves at a speed of 54km/s.

- Check Schedule Procedure

```
If posisi armada != posisi penjadwalan
        If posisi penjadwalan > posisi armada
                If |posisi penjadwalan - posisi armada| > 30 satuan array
                        fleet speed = 86.4 km/h
                else
                        fleet speed = 64.8 km/h
        else
                If |posisi penjadwalan - posisi armada| > 30 satuan array
                        fleet speed = 21.6 km/h
                else
                        fleet speed = 43.2 km/h
Else
        fleet speed = 54 km/h
```

This procedure is located on the main GUI algorithm. This procedure is called when the message from fleet is parsed.

- Emergency Procedure

```
If kondisi darurat = true
        for i=0 until i<jumlah halte
                if posisi armada < halte[i]
                        halte terdekat didepan armada = halte[i]
```

- Calculate schedule procedure

```
for i=1 until i<fleetsch2.length
        fleetsch2[i] += fleethsch2[i-1] + jarak
```

- Check scjeduling procedure

```
If sch2 != posisi penjadwalan
        If sch2 > posisi penjadwalan
```

```
                posisi penjadwalan++
        else
                posisi penjadwalan--
```

C. Encryption-Decryption Module

Encryption and decryption module aims to secure data when published to the internet. The design and implementation of encryption-decryption module is:

1. *Design of Encryption-Decryption Module*

There are three encryption algorithms that have advantages and disadvantages [2] shown in the table below.

NAME	*Avalance effect*	*throughput*
AES	93%	950 bits/sec
DES	75%	500 bits/sec
RSA	60%	100 bits/sec

Avalanche effect is a change in encryption that occurs when some of the messages are changed. The greater this value the better.

Throughput is the amount of data processed per second. The greater the throughput the faster the algorithm performs the encryption.

From the table above AES is choosen for the encryption and decryption module in FMCS.

2. *Implementation of Encryption-Decryption Module*

Implementation of AES is done on fleet hardware and also GUI uses AES-128 bit. In fleet hardware used library AES.h. Encryption is done using the following procedure.

```
do_aes_encrypt(plain, length, ciphertext, key, bits, iv);
```

The explanation of the parameters of the procedure is as follows

- plaintext, the variable containing the message to be encrypted.
- length, is the length of the message.
- ciphertext is an encrypted plaintext, the variable will be sent to the server.
- key is a particular string used as a "key" to encrypt and decrypt a message.
- bits are the number of bits of the key, in this implementation used 128 bit key.
- iv is the initial vector used in the encryption & decription process .

After the message is encrypted on fleet hardware. The message is sent to the server and the server forwards the message to the GUI. Decrypting messages in the GUI uses the following functions.

```
DecryptStringFromBytes_Aes(ciphertext, key, iv);
```

The function will generate a string containing the decrypted message. Key and iv that used in fleet hardware is used to decrypt the message. The message will be processed by the GUI to determine the command to be sent on the fleet hardware. The command must go through the encryption process on the GUI using the following functions.

```
EncryptStringToBytes_Aes(plaintext, key, iv);
```

The function will return a byte variable containing an encrypted message. The message should be decrypted on fleet hardware using the following procedure.

```
do_aes_decrypt(ciphertext, length, message, key, bits, iv);
```

The procedure will store the decrypted message on the message variable.

IV. Testing Result

A. Server

Server testing is done to determine the reliability of the server in receiving many data / connection at once. The assumption used in testing the server is the latency generated purely from the server.

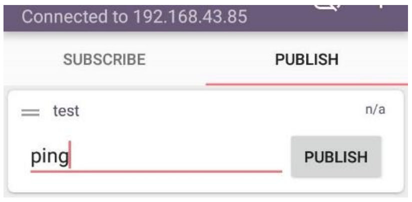
1. *Server Testing Procedure*

The server testing procedure is:

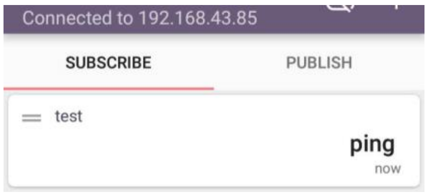
- Run server and connect to the local network.
- Run dummy data application and connect it with server on local network. This application will publishes 40 messages every 0.7 seconds on the server.
- Calculate server latency by counting the delays of received messages.

2. *Test Results Server*

This test is to prove server that it can forward the received messages that sent to a certain topic like the picture below

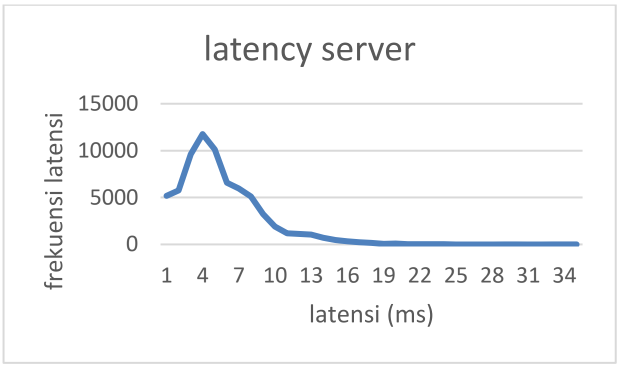

Connected to 192.168.43.85
SUBSCRIBE
PUBLISH
test
n/a
ping
PUBLISH

The message that sent the word "ping" on the topic "test". Afterwards it is observed whether the server can forward messages on the topic by subscribing on the same topic before. Can be seen in the picture below


Connected to 192.168.43.85
SUBSCRIBE
PUBLISH
test
ping
now

The "ping" message is received in the "test" topic. The server has successfully forwarded messages sent on the topic.

The next test is done for 20 minutes to see server latency. There are 70 thousand data latency obtained like the graph below.



the average of latency is 4.48 seconds.

B. *Scheduling*

This test is to evaluate the algorithm that had previously been designed is tested to specific case. The response is observed whether the algorithm can maintain the position of the fleet so that it will match the schedule. The first case is when the fleet is left behind by scheduling, the second case is when the fleet is in front of the scheduling, and the last case is when the fleet is in match with the scheduling.

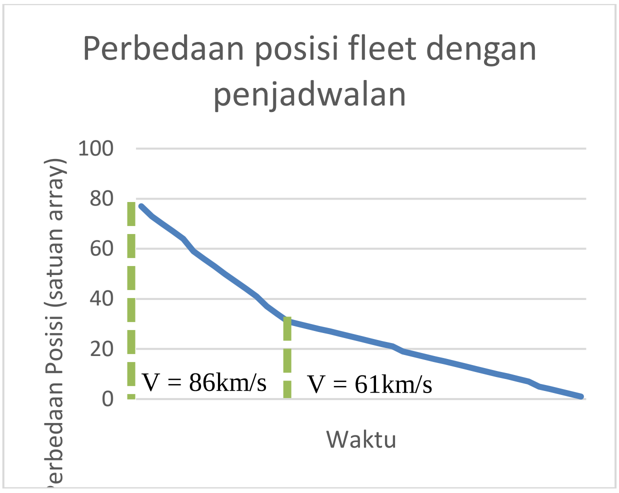
1. *Scheduling Testing Procedure*

This test is performed three times in accordance with the case in the test. The test procedure is:

- Run the server and connect to the internet network.
- Run the GUI and connect to the server over the local network.
- Set the fleet position to fit the case to be tested in the dummy data program.
- Run the Dummy Data app and connect it to the server over the internet.
- Record the screenshots and the time difference needed by the fleet to match the scheduling.

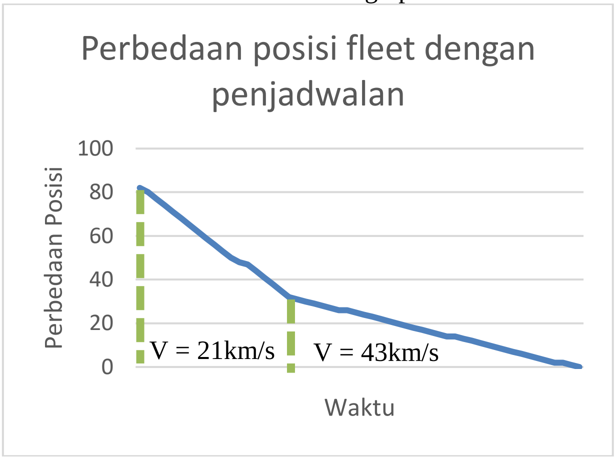
2. *Scheduling Testing Results*

The test results from the first case of the late fleet of schedule are as follows



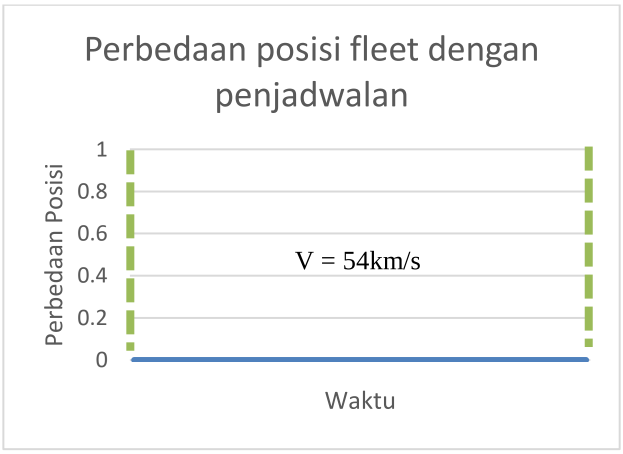
The chart above shows when the fleet is delayed from schedule scheduler algrotima can improve the fleet position to fit the schedule.

In the second case test that the fleet precedes the schedule obtained results like the graph below



Can be observed from the chart above when the fleet overtakes the scheduled schedule scheduler schedule can improve the fleet's position to fit its schedule.

In the third test case of the fleet in accordance with the schedule obtained results like the picture below.



The graph above shows when the fleet is match with schedule, the lagortihm will maintain the speed of the fleet same with scheduling speed.

C. Encryption-Decryption Module

Encryption and decryption module testing is to find out whether only the fleet hardware and control station (GUI) can read encrypted messages.

3. *Testing Procedure of Encryption-Decryption Module*

This test is done by following steps:

- Run the server and connect to the internet network.
- Run the GUI and connect to the server over the local network.
- Turn on fleet hardware.
- Observe the results of data received by GUI and client.

In this test fleet hardware encrypt the message and GUI decrypt the message.

4. *Testing Results of Encryption-Decryption Module*

The test results show the GUI decrypted message the same as the message before it is encrypted by fleet hardware if using the same key and iv. From the test results also showed when using a different iv or key then the decrypted message does not match the message before it is encrypted.

V. CONCLUSION

The MQTT server has been successfully implemented. Server capable of forwarding messages with latency 4.81 milisecond. This MQTT server can be accessed via the internet.

Scheduling algorithms can matain the fleet on schedule. If the fleet posistion isn't same with scheduling, the algorithm will give the command of the speed that must be followed by the fleet to match with the schedule.

Encryption and decryption modules have been implemented on fleet hardware and also GUI using AES algorithm. Fleet hardware and GUIs can read encrypted messages sent to each other using the same "key" the same.

REFERENSI

[1] Acalagent, Benchmark of MQTT servers, January 2015

[2] S.N.Ghosh. (2015, Sep.). Performance Analysis of AES, DES, RSA And AES-DES-RSA Hybrid Algorithm for Data Security. *Volume 2, Issue 5*. [Online]. 85, Available: https://www.academia.edu/15749187/Performance_Analysis_of_AES_DES_RSA_And_AES-DES-RSA_Hybrid_Algorithm_for_Data_Security

[3] Isode, M-Link & XMPP Performance Measurements over Satcom and Constrained IP Networks. Available: http://www.isode.com/whitepapers/xmpp-performance-constrained.html.