S-Que : M/M/c based Smart Queues Ticketing System

Case Study : Bank Queuing System

Aris Prawisudatama
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
soedomoto@gmail.com

Abstract*—This electronic document is a "live" template and already defines the components of your paper [title, text, heads, etc.] in its style sheet.** ****CRITICAL: Do Not Use Symbols, Special Characters, or Math in Paper Title or Abstract. **(***Abstract***)**

Keywords—component; formatting; style; styling; insert (key words)

I. INTRODUCTION

Antrian merupakan sebuah situasi yang umum dijumpai dalam kehidupan sehari-hari, seperti antrian pada pembelian tiket kereta api, layanan bank, pembuatan dokumen resmi, hingga rumah makan. Antrian, secara umum dapat digambarkan sebagai lamanya waktu tunggu hingga pelayanan diperoleh [1]. Antrian, bisa menjadi sesuatu yang menjengkelkan bagi mereka yang mempunyai waktu yang terbatas. Terkadang, antrian bisa terjadi seharian penuh, tanpa bisa diprediksi perkiraan waktu pelayanan akan diberikan. Dengan menggunakan teori antrian, sebuah model dirancang, sehingga panjang antrian dan lamanya waktu tunggu dapat diprediksi [2].

Teori antrian telah menjadi bahan penelitian sejak lama. Matematikawan Denmark, A.K. Erlang menggunakan special class dari gamma random variabel, sering disebut dengan Erlang-k random variable dalam penelitiannya tentang delay pada trafik telepon [3]. Sekarang, teori antrian telah banyak menjadi subject penelitian pada bidang engineering, production, biological engineering, communication, computer system design, and other fields [1].

Ada beberapa teori terkait dengan antrian, antara lain M/D/1, M/M/1, dan M/M/C. M/D/1 digunakan pada kasus kedatangan yang bersifat random, deterministic service, dan service channel tunggal. M/M/1 digunakan pada kasus kedatangan random, pelayanan acak, dan service channel tunggal. Sementara M/M/C digunakan pada kasus kedatangan random, pelayanan acak, dan beberapa service channel. Pada pelayanan sistem antrian pelayanan bank, terdapat lebih dari service channel yang dapat digunakan oleh pelanggan secara acak. Oleh karena itu, implementasi teori M/M/C lebih cocok untuk diterapkan.

II. THEORETICAL BACKGROUND

A. M/D/1

In queueing theory, an M/D/1 queue represents the queue length in a system having a single server, where arrivals are determined by a Poisson process and job service times are fixed (deterministic). Agner Krarup Erlang first published on this model in 1909, starting the subject of queueing theory [1]. An M/D/1 queue is a stochastic process whose state space is the set {0,1,2,3,...} where the value corresponds to the number of customers in the system, including any currently in service.

- Arrivals occur at rate λ according to a Poisson process and move the process from state i to i + 1.
- Service times are deterministic time D (serving at rate μ = 1/D).
- A single server serves customers one at a time from the front of the queue, according to a first-come, first-served discipline. When the service is complete the customer leaves the queue and the number of customers in the system reduces by one.
- The buffer is of infinite size, so there is no limit on the number of customers it can contain.

B. M/M/1

An M/M/1 queue represents the queue length in a system having a single server, where arrivals are determined by a Poisson process and job service times have an exponential distribution. The model name is written in Kendall's notation [4].

- Arrivals occur at rate λ according to a Poisson process and move the process from state i to i + 1.
- Service times have an exponential distribution with rate parameter μ in the M/M/1 queue, where 1/μ is the mean service time.
- A single server serves customers one at a time from the front of the queue, according to a first-come, first-served discipline. When the service is complete the customer leaves the queue and the number of customers in the system reduces by one

The model can be described as a continuous time Markov chain with transition rate matrix

$$\begin{bmatrix} \lambda & 0 & 0 & \cdots & 0 \\ 0 & \mu+\lambda & 0 & \cdots & 0 \\ 0 & 0 & \mu+\lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & 0 & \cdots & \mu \end{bmatrix}$$

C. M/M/c

The M/M/c queue (or Erlang–C model) is a multi-server queuing model, where there are c servers and job service times that are exponentially distributed. M/M/c is a generalization of the M/M/1 queue which considers only a single server. The characteristics of M/M/c are:

- Arrivals occur at rate λ according to a Poisson process and move the process from state i to i+1.
- Service times have an exponential distribution with parameter μ in the M/M/c queue.
- There are c servers, which serve from the front of the queue. If there are less than c jobs, some of the servers will be idle. If there are more than c jobs, the jobs queue in a buffer.
- The buffer is of infinite size, so there is no limit on the number of customers it can contain.

D. Multi-class M/M/c

The simple M/M/c queue, seperti pada menggunakan c servers dengan service rate yang sama. Pada kenyataannya, dalam pelayanan bank misalnya, terdapat beberapa service desk yang melayani jenis service yang berbeda-beda, sehingga masing-masing jenis pelayanan dapat mempunyai service rate yang berbeda-beda. Istilah ini disebut dengan Multi-class services.

Wang dkk, melakukan penelitian untuk menemukan sebuah model dalam menangani antrian pada Two Priority Class [2]. Two class M/M/c service rate at state (q1, q2) is μ1 min(q1, c); independent of q2. Thus, the distribution of Class-1 jobs' sojourn time is [3]:

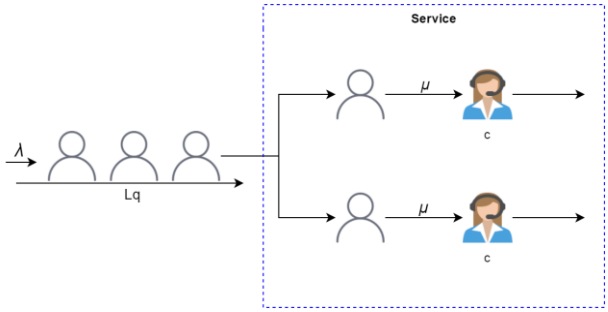

Fig. 3. M/M/c Queuing Model

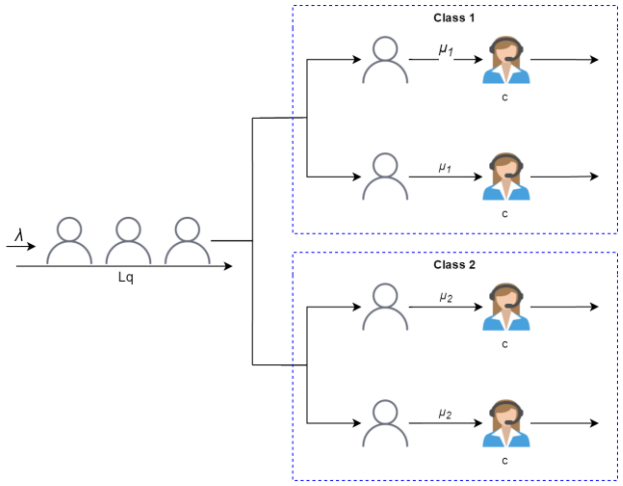

Fig. 1. Multi-class M/M/c Queuing Model with Two Servers

$$P\{S_1 < t\} = 1 - e^{-\mu_1 t} - \frac{(e^{-(c\mu_1-\lambda_1)t} - e^{-\mu_1 t})}{1-(c-\frac{\lambda_1}{\mu_1})} \frac{\lambda_1^c}{\mu_1^c c!}((1-\frac{\lambda_1}{c\mu_1})\sum_{i=0}^{c-1}\frac{\lambda_1^i}{\mu_1^i i!} + \frac{\lambda_1^c}{\mu_1^c c!})^{-1}$$

E. Service Oriented Architecture

Service Oriented Architecture (SOA) is a way of designing software as interoperable software services. A service is a software component that is available over a network and can act either as a client or a server or be composed by combining other services [4]. Services in a SOA are lightweight, loosely coupled and communicate using messages over well-defined interfaces. Today, most SOAs are built using Web Services.

Because SOAs are built on light-weight and interoperable services, they are inherently flexible and scalable. The flexibility comes from the fact that services can be replaced according to changing needs and that new services can be created by combining other services. SOAs scale well thanks to their parallel nature as services can easily be duplicated or migrated as needed [5].

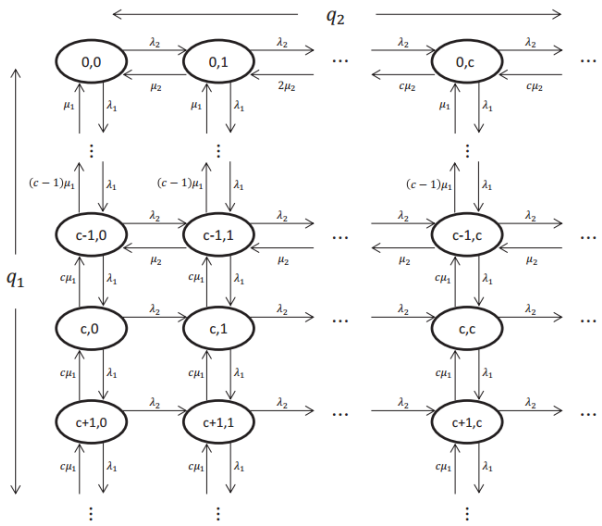

Fig. 2. MC of the M/M/c queue with two priority classes

F. *SOAP and REST*

A SOA can be rendered through the use of any service-based technology, such as SOAP or REST. Compared to each other, SOAP is more mature and standardized, widely supported and transport protocol independent while REST gives developers larger freedom of choice and is easy to get started with thanks to the agility of the design and the lightweight approach. REST also works well for applications with a Web Interface since the basic REST operations are supported by any Web Browser. However, REST lacks standards for security, quality of service (QoS), etc and support for business automation languages whereas SOAP is highly standardized and supported. A more in-depth discussion of these technologies is given by Pautasso et al [6].

Performance wise, REST has better cache support, lightweight requests and responses as well as easier response parsing. REST also supports a large number of parallel clients and servers and reduces network traffic due to less overhead [7].

III. S-QUE SYSTEM DESIGN

S-Que merupakan sebuah sistem yang digunakan untuk memanage antrian pelayanan. S-Que merupakan sebuah jawaban dari sistem antrian konvensional yang masih umum digunakan pada pusat-pusat pelayanan. Sistem antrian konvensional hanya memberikan nomor antrian kepada konsumen yang membutuhkan pelayanan, tanpa memberikan perkiraan kapan pelayanan akan diberikan. Sistem seperti ini tentu saja sangat tidak menyenangkan, terutama bagi konsumen dengan waktu yang terbatas, karena konsumen harus menunggu giliran pelayanan dengan terus menerus berada pada pusat pelayanan.

Di sisi lain, S-Que memberikan kepastian kepada konsumen melalui dua channel : estimasi dan real-time. Estimasi pelayanan diberikan kepada konsumen bersamaan dengan tiket antrian yang diambil konsumen. Selain itu, konsumen dapat juga mengecek kondisi antrian terkini setiap saat melalui web. S-Que mengadopsi teori antrian Multi-class Erlang-C (M/M/c) model yang telah dikembangkan oleh Wang et al [2] serta mengadopsi pendekatan SOA yang diimplementasikan dalam RESTful Web Service.

A. *Use Case*

There are three types of users of the system, customer, service officer, and administrator. Customers are people who take queue and get the ticket, and also check the current queue status. Service officers are people who call customers for service. While administrator is user who manage the system, including add, edit, and delete service channels and service officers. The Use case for system is shown in Fig. 4.

Pertama-tama customer perlu untuk mengambil antrian. Sebelum mengirimkan respons yang akan dicetak pada tiket antrian, sistem akan mengassign sebuah nomor antrian, dan mengkalkukasi perkiraan pelayanan berdasarkan jumlah antrian dan jumlah service counter yang tersedia, sehingga pada tiket antrian akan tercetak nomor antrian beserta perkiraan waktu pelayanan. Proses ini dilakukan oleh server antrian yang berkomunikasi dengan client yang berupa tombol antrian melalui teknologi Web service.

Fig. 4. S-Que Use Case

Customer juga dapat mengecek kondisi antrian secara real-time dengan menggunakan web browser pada computer maupun smartphone.

Sementara itu, Service officer akan memanggil customer dengan menggunakan client yang dilengkapi tombol start dan stop. Perekaman mulai dan selesainya pelayanan berfungsi untuk menghitung rata-rata lama pelayanan pada setiap kelas pelayanan.

B. *SOA Design*

Dari use case pada Fig. 4, kemudian di dekomposisi menjadi Services dan Components yang akan digunakan pada sistem. Pada Fig. 5 digambarkan tentang hubungan antara services, components, dan functional components. Services terdiri dari enam buah service yang terbagi dalam tiga modul : modul service, modul counter, dan modul ticket. Sedangkan service component menyediakan functionalitas untuk setiap modul

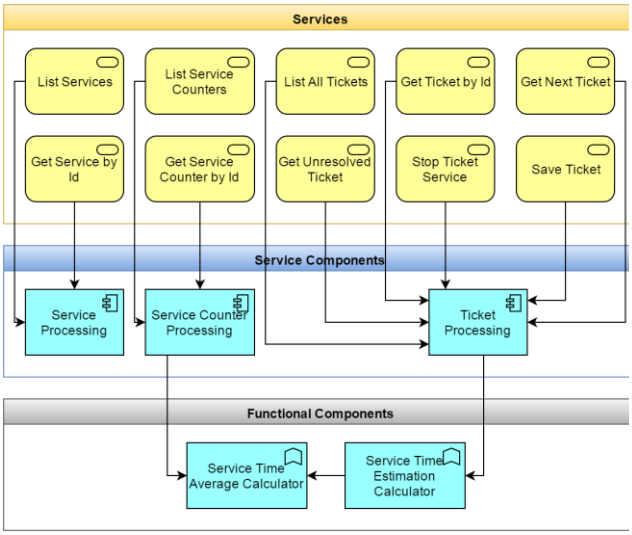

Fig. 5. SOA Architecture Layer

service, seperti : koneksi ke database, meng-consume service lain, maupun kalkulasi logic.

Sementara itu, functional components terdiri dari dua fungsi, yaitu : service time average calculator dan service time estimation calculator. Service time average berfungsi untuk melalukan penghitungan rata-rata lama service untuk setiap jenis pelayanan. Penghitungan ini didasarkan pada historical data yang ada. Sementara service time estimation calculator berfungsi untuk melakukan estimasi kapan pelayanan akan diberikan, berdasarkan metode multi-class M/M/c.

C. *User Interface*

While it is possible for users to use the REST service interface to work with the S-Que System, we think most people would appreciate not having to interact directly with Web Services. We therefore provide a Web Application on top of the REST interface which clients and reviewers use to interact with the system.

IV. Implementation and Deployment

A. *Web Service*

RESTful Web Service pada S-Que diimplementasikan dengan menggunakan bahasa pemrograman Python dan Flask framework. Flask framework telah menyediakan method-method dasar untuk berkomunikasi melalui HTTP, seperti POST, GET, PUT, PATCH, and DELETE.

B. *All S-Que User Interface*

All of user interface are implemented as a Web-based interface. Web-based interface is aimed to be simpler, platform independent, and responsive. User interface is built in top of twitter bootstrap library.

C. *S-Que Ticket*

S-Que Ticket masih menggunakan sistem manual, artinya customer perlu hadir untuk menekan tombol jenis service di lokasi service, dan belum dapat digunakan secara remote. Secara fisik, tiket yang diprint berukuran sama dengan tiket antrian pada umumnya (ukuran POS). Akan tetapi informasi yang terdapat pada ticket antrian lebih lengkap, mencakup : nomor antrian customer, nomor antrian yang sedang berjalan, jenis service yang diinginkan, perkiraan waktu pelayanan, dan QR code untuk pengecekan secara real-time.

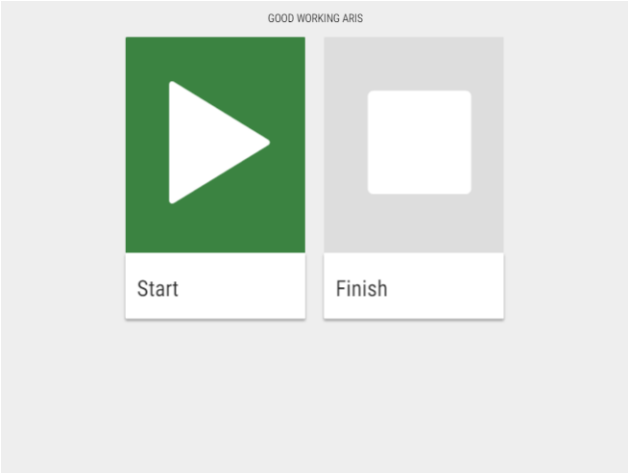

Fig. 7. Screenshot of S-Que Service Counter on Tablet Screen

```
@self.route('/api/list/', methods=['GET'])
def api_list():
   tickets = Ticket.query.all()
   return Response(json.dumps([i.serialize for i
        in tickets]),
        mimetype='application/json')

@self.route('/api/id/<id>/', methods=['GET'])
def api_by_id(id):
   ticket =
        Ticket.query.filter_by(ticket_id=id)\
      .first()
   return Response(json.dumps(ticket.serialize),
        mimetype='application/json')

@self.route('/api/next/<counter>',
        methods=['GET'])
def api_next_ticket(counter):
   next_ticket = Ticket.query.filter_by(\
        service_date=datetime.datetime.now()\
        .date(), service_start_time=None)\
        .order_by(Ticket.ticket_order.asc())\
        .first()

   if next_ticket:
      next_ticket.service_counter = int(counter)
      next_ticket.service_start_time = datetime\
        .datetime.now()

      db.session.commit()

      return Response(json.dumps(next_ticket\
        .serialize if next_ticket else None),\
        mimetype='application/json')

@self.route('/api/unresolved/<counter>',\
   methods=['GET'])
def api_unresolved_ticket(counter):
   unresolved_ticket =
        Ticket.query.filter(and_(\
      Ticket.service_start_time!=None,\
      Ticket.service_finish_time==None,\
      Ticket.service_date==datetime.datetime\
        .now().date(),\
      Ticket.service_counter==int(counter)\
   )).first()

   return Response(json.dumps(unresolved_ticket\
      .serialize if unresolved_ticket else None)\
      , mimetype='application/json')

@self.route('/api/stop/<id>', methods=['GET'])
def api_stop_ticket(id):
   finished_ticket = Ticket.query.filter_by(\
      ticket_id=id).first()

   if finished_ticket:
      finished_ticket.service_finish_time =\
        datetime.datetime.now()
      db.session.commit()

   return Response(json.dumps({'success' :\
      True}), mimetype='application/json')
```

Fig. 6. Code Snippet for Python Flask Web Service

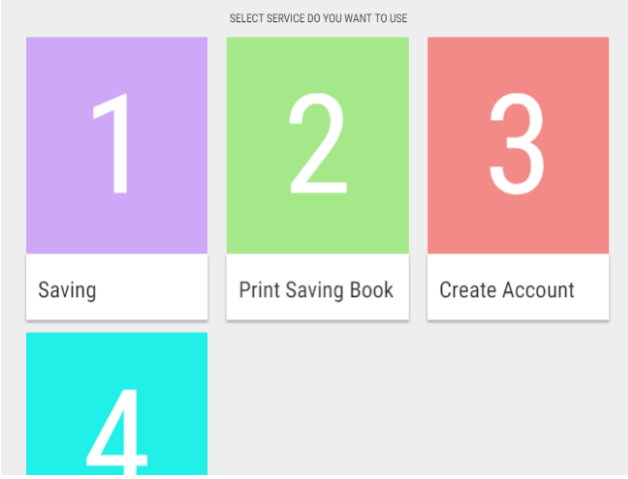

Fig. 8. Screenshot of S-Que Ticketing Service on Tablet Screen

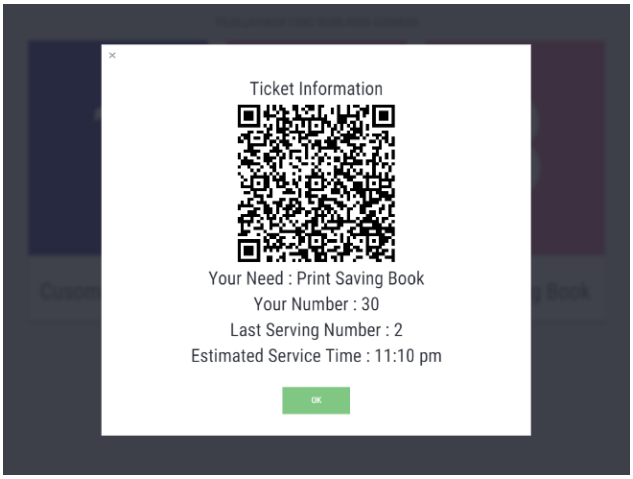

Fig. 10. Screenshot of S-Que Ticket Information after Selecting Service on Tablet Screen

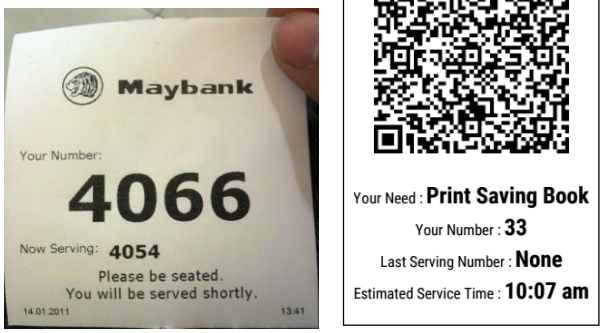

Fig. 9. Conventional vs S-Que Queuing Ticket

REFERENCES

[1] [1] J. F. C. Kingman, "The first Erlang century—and the next," *Queueing Syst.*, vol. 63, no. 1–4, pp. 3–12, Dec. 2009.

[2] [2] J. Wang, O. Baron, and A. Scheller-Wolf, "M/M/c Queue with Two Priority Classes," *Oper. Res.*, vol. 63, no. 3, pp. 733–749, Jun. 2015.

[3] [3] J. A. Buzacott and J. G. Shanthikumar, *Stochastic models of manufacturing systems*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[4] [4] P. F. Brown and R. M. Hamilton, "Reference model for service oriented architecture 1.0. Tech. rep.," 12-Oct-2006. [Online]. Available: http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf. [Accessed: 01-May-2016].

[5] [5] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

[6] [6] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful Web Services vs. 'Big'' Web Services: Making the Right Architectural Decision," in *Proceedings of the 17th International Conference on World Wide Web*, New York, NY, USA, 2008, pp. 805–814.

[7] [7] S. Kumari and S. K. Rath, "Performance comparison of SOAP and REST based Web Services for Enterprise Application Integration," in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2015, pp. 1656–1660.