University of London
Imperial College of Science, Technology and Medicine
Department of Computing

Inferring Queueing Network Models from High-precision Location Tracking Data

Tzu-Ching Horng

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of the University of London and
the Diploma of Imperial College, May 2013

Abstract

Stochastic performance models are widely used to analyse the performance and reliability of systems that involve the flow and processing of customers. However, traditional methods of constructing a performance model are typically manual, time-consuming, intrusive and labour-intensive. The limited amount and low quality of manually-collected data often lead to an inaccurate picture of customer flows and poor estimates of model parameters. Driven by advances in wireless sensor technologies, recent real-time location systems (RTLSs) enable the automatic, continuous and unintrusive collection of high-precision location tracking data, in both indoor and outdoor environment. This high-quality data provides an ideal basis for the construction of high-fidelity performance models.

This thesis presents a four-stage data processing pipeline which takes as input high-precision location tracking data and automatically constructs a queueing network performance model approximating the underlying system. The first two stages transform raw location traces into high-level "event logs" recording when and for how long a customer entity requests service from a server entity. The third stage infers the customer flow structure and extracts samples of time delays involved in the system; including service time, customer interarrival time and customer travelling time. The fourth stage parameterises the service process and customer arrival process of the final output queueing network model.

To collect large-enough location traces for the purpose of inference by conducting physical experiments is expensive, labour-intensive and time-consuming. We thus developed LocTrack-JINQS, an open-source simulation library for constructing simulations with location awareness and generating synthetic location tracking data.

Finally we examine the effectiveness of the data processing pipeline through four case studies based on both synthetic and real location tracking data. The results show that the methodology performs with moderate success in inferring multi-class queueing networks composed of single-server queues with FIFO, LIFO and priority-based service disciplines; it is also capable of inferring different routing policies, including simple probabilistic routing, class-based routing and shortest-queue routing.

Acknowledgements

The completion of this research is largely owe to many of those I have worked with during my study at Imperial College London. I especially would like to express my gratitude to my supervisor Dr. William Knottenbelt, for his generosity, open-mindedness, enthusiasm as well as constant support and guidance during my PhD; and to my colleague Nikolas Anastasiou, for all the work we have done together and our personal friendship.

The PhD journey for me has been a long one with countless moments of self-questioning, doubts and confusion. Fortunately, there are many beloved ones always taking my side along the journey. Special thanks to my dearest friends – Pei-Chien Tsai, Tzu-Chun Chen, Cecilia Lopez, Olivier Blesbois – for always being there for me. Last, but most important of all, I owe my parents the deepest gratitude for everything they have given me throughout my life, with their unconditional love.

Dedication

This thesis is dedicated to Tzu-Ting Horng, who understands me the most and always has a place in my heart.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Stochastic models, such as queueing network and Petri nets, have been widely used to model and evaluate the critical performance characteristics – availability, reliability, responsiveness and efficiency – of systems in which a network of service centres process the flow of customers requesting services and resources. Such models are mathematical abstraction of the underlying systems; they not only identify bottlenecks existing in the systems and but also monitor if the QoS (quality of service) targets are being satisfied. They are also a powerful "virtual laboratory" for exploring the impact on performance given changes to those systems in customer flows, the number and allocation of resources, system workload, scheduling policies and so on. Stochastic models have seen a broad range of applications in computer network and telecommunication systems [15, 28, 122], public transportation systems [112, 114, 118, 55, 56, 110, 141], healthcare systems [104, 101, 3, 8, 45, 139, 85, 22, 21] and manufacturing systems [32, 33, 80].

Traditionally, constructing a performance model involves four major steps: conceptualisation, parameterisation, validation and analysis. The first two steps are essential in building an accurate model and thus crucial in giving confidence in the output of subsequent steps. Many of previous studies based their model construction on a prior knowledge of the system structure,

assumptions made from human observations and manually-collected data [45, 104]. Some of them largely simplified the model structure for the purpose of obtaining analytical closed-form results. As a result, the generated models are limited to a very high level. It is difficult to use such high-level models to identify hidden or previously-unknown bottlenecks in the system or to capture the complex interactions among various entities in many real-life systems. Neither can they reflect the changes of arrival and service rates dependent on the time of the day or other conditions. Furthermore, traditional data collection techniques, such as time and motion studies, involve tedious manual tasks such as inspection of video footage, questionnaires, manual collection of timing data, personnel interviews, all of which may disrupt the natural entity flow in the system. This data gathering process is not only time consuming, expensive, but, more importantly, often introduces human bias and errors, which makes the constructed model difficult to validate. For example, previous work on modelling patient arrival patterns and response times in an Accident and Emergency department by S. Au-Yeung [22, 21] exemplified the difficulties of model construction (particularly accurate parameterisation) and validation using manually-collected data. While there was good agreement between *mean* response times emerging from both the model and the data, the *distributions* of response times were not well matched, and there was no straightforward way to identify the causes of discrepancies.

Over the past few decades, the advances in electronics and telecommunications have brought a wide variety of technologies, such as video imaging system, inductive loop detectors and weight-in-motion (WIM), that allow high-quality data to be collected automatically and continuously. In particular, thanks to the rapid development in wireless technologies, as well as the popularisation of sensor-enabled mobile devices, we are approaching an era of the "Internet of things" [18] – where a wide range of heterogeneous physical objects are linked through wireless networks and able to communicate with each other. The physical world itself then becomes a type of information system, churning out large amount of real-time observations for analysis at much lower costs. These physical information networks/systems have become powerful tools that help us not only gain a much enhanced understanding of our environment; but also able to respond to changes, abnormality, insufficiency or inefficiency more promptly and with

better "intelligence". The unprecedented volume and high-quality of information captured by these sensor-enabled networks, combined with theories/techniques in Data Mining, Machine Learning and statistical inference, lay the foundation for many emerging research areas such as "embedded intelligence" [71] or "people-centric sensing" [40], where computers can estimate, understand and then predict human behaviour, as well as interactions among entities in the environment, without minimal human intervention. For example, in response to the growing ageing population, research efforts such as [115, 72, 121, 98] have been devoted to construct sensor-enabled intelligent systems in domestic environments that can assist activities of daily living (ADL); these studies aim to provide solutions ensuring the quality and safety of domestic life for the elderly.

Real-time location tracking and location-aware services is another area that has received great attention in both academia and industry, thanks to the emergence of enabling wireless positioning technologies. Over the past decade, GPS (global positioning system) has been the most popular and widespread positioning system. In addition to traditional navigation, it has enabled numerous commercially successful applications offering location-based (or location-aware) services. Examples include real-time traffic updates [136], local business search services [62], nearest parking lot availability search [63], location-based business advertisements etc. However, due to the requirement of a clear sky view for receiving satellite signals, GPS fails to function properly in many daily environments, such as indoor spaces or urban environment. As people spend most of their time indoors, indoor tracking services, which require high data precision and resolution, is expected to create unprecedented opportunities for business [52]. Accurate indoor positioning is more challenging as there are various obstacles (such as walls, metal objects, mirror, furniture and human bodies) disrupting the propagation of electromagnetic waves and causing multi-path effects. Only recently has indoor positioning reached the sub-metre accuracy (e.g. Ubisense [131], active badge [135], Cricket [50, 117, 116]); we are thus approaching the level of ubiquitous location-awareness. The deployment of real-time location systems (RTLSs) in both outdoor and indoor environments have been growing at an astonishing rate over the past decade. A wide variety of applications in areas across transportation,

manufacturing, supply chain, urban planning and healthcare systems have taken advantage of the rich location information collected by wireless location tracking systems to enhance system efficiency and responsiveness, reduce wastes and improve personnel safety or security [73, 13, 138, 41].

Motivated by the availability of high-precision location data and the much improved visibility it offers to the underlying systems, this research aims to develop a data-processing methodology that takes advantage of high-precision location tracking data obtained with real-time location systems and generates a queueing network model that can accurately describe the underlying system with minimum human intervention. Rather than relying solely on the modeller's perception of how the process "should" be, our approach for model construction is mainly driven by observations from the actual process and the resulting models are thus more subjective. Through automating the processes of conceptualisation, parameterisation and validation of performance models, the time as well as human resources spent in the model construction pipeline is expected to be largely reduced. Automation of model construction processes, continuously fed with large amount of real-time observations, makes it possible to construct performance models based on most recent updates from the monitored systems, instead of long-term historical data. The generated models can thus reflect the most recent status of the system and support fast decision-making in response to sudden changes in the system. As an ultimate goal, we expect most applications of our methodology in managing systems with complex customer flows, high volatility in customer demands and stringent response time requirements.

Raw location tracking data, defined in this research as a spatiotemporal dataset giving the observed locations of tags at various times, usually contains noise and other extraneous information. Appropriate data processing methodologies are thus necessary for extracting useful information from it. Some previous research endeavours (such as [65], [66] and [67]) have been made in designing new data models for RFID data warehousing and processing; such models facilitate inferring high-level information, such as probabilistic item flows, from raw, low-level RFID data. However, RFID is different from other wireless technologies such as sensor network

in that the location of an RFID-tagged item is identified with the RFID reader that detects it. This is because a tagged item's location can only be known when it is scanned by an RFID reader, whose location is usually fixed and known in advance. By contrast, in the latter case, the location data is simply a time-stamped trace of tagged entities' geographical locations. The spatial relationships and interactions between tagged entities must be inferred on the basis of proximity or otherwise. We argue that next few years will see increasing deployments of Internet of things systems for entities tracking in large-scale systems. Recently the London City Airport launched the pilot project taking advantages of multi-modal data collected through the interconnected sensor network and data hub to track, understand and better manage passenger flow and behaviour at the airport. The aim is to enable interactions with passengers at key touch points across the airport through various location-specific services [97].

The developed methodology in this research is based around a four-stage data-processing pipeline, which takes raw high-precision location tracking data as initial inputs and outputs a queueing network model that can best describe the underlying systems (see Figure 1.1). The final output queueing network model can be used as the base model for analytical or simulation-based performance analysis; the application of the final output queueing network model is however beyond the scope of this research. Since this is an extremely broad and challenging problem for customer-processing systems in general, for the present stage we restrict ourselves to systems with multiple customer classes, single-server service and service disciplines including FIFO, LIFO and priority-based. In terms of the customer flow structure, we only consider three routing policies: probabilistic routing, class-based routing and shortest-queue routing.

Although the ultimate goal of this research is to apply the developed methodology to performance analysis of real-life systems, it is necessary, in the development process, to test the accuracy of the data processing pipeline against a system whose underlying processes are fully understood. As one can only has the "perception" of how the dynamics of a real-life system should be, instead of the true reality, we can only evaluate our methodology through either

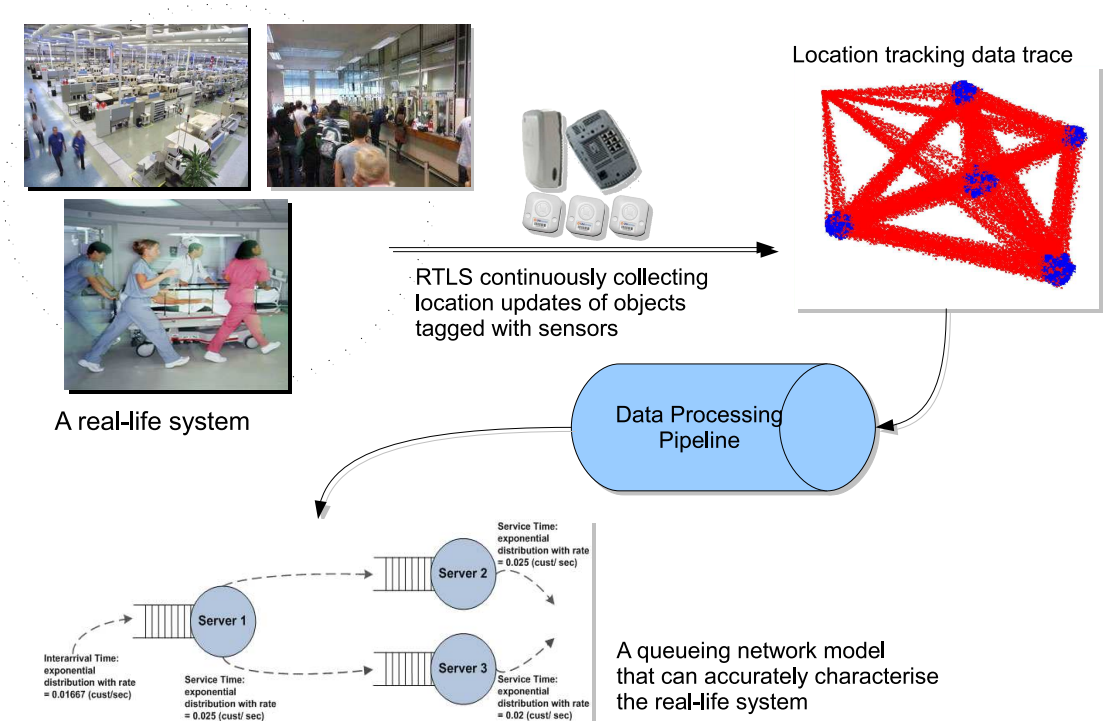


Figure 1.1: The overview of the goal of this research

experiments or simulations. At the initial stage of this research, four experiments have been conducted; in each of them a simple queueing network system with at most two service centres was designed. While the location data collected from the experiments are sufficient for inferring the statistical characteristics in simple queueing systems with only one service centre, it is not the case for scenarios with more service centres existing in the system [76]. The installation and execution of experiments are also time-consuming and labour-intensive. Under tight constraints on hardware equipments and human resources, it is difficult to conduct long-hour and large-scale experiments to collect sufficient data for inferring more complex features. Model construction usually requires large amount of data for parameter estimation, especially for systems involving multiple customer classes and multiple servers employing different service disciplines. While expanding the variety of features the data processing pipeline can infer, we also need location tracking data collected from systems with various settings on service times, customer interarrival times, service disciplines and routing policies. This research is thus motivated to develop LocTrackJINQS, a location-aware simulation tool that can rapidly produce large amounts of location tracking traces from systems with different user-defined scenarios and

settings. In addition to providing an efficient approach for generating location tracking data for evaluation purpose, LocTrackJINQS can be easily extended for different applications or general location-based research.

1.2 Objectives

The primary objects of this research include:

- To develop a methodology that takes high-precision location tracking data (either from simulation or collected by a real-time location tracking system) as inputs and automatically constructing a detail and realistic queueing network model for either a real-life or a simulated customer-processing system. The inferred queueing network model should be able to reflect:
  - The structure of the customer flow in the system.
  - The service time distribution at each service centre.
  - The service discipline adopted at each service centre.
  - The travelling time distribution between each pair of connecting service centres.

  The methodology also identifies the approximate location and size of each service area; it can further discover previously unknown bottleneck in the system.

- To develop a simulator tool that can generate spatiotemporal location data under different user-defined settings regarding the system layout, service process (in terms of service time and service discipline), customers' interarrival time distributions and customer classifications. The tool does not only serve for the purpose of our research; it also aims to facilitate general location-based research.

1.3 Contributions

1.3.1 Inferring Queueing Network Models from High-precision Location Tracking Data

In this research we have developed a methodology based around a four-stage data processing pipeline. The initial input of the pipeline is raw high-precision location tracking data, either collected by a RTLS or generated through simulation. The final output is an inferred queueing network characterising the customer flow, customer arrival pattern (customer interarrival time) and different time delays incurred in the system (service time and customer entities' travelling time between service centres).

The first stage of the data processing pipeline performs basic data cleaning and interpolation. The second stage infers the approximate location of each service centre as well as the size of its associated service area. This stage uses the geographical proximity of tagged customer entities' to the approximated service areas' locations to identify their spatiotemporal relationships and decide if a customer entity is present in a particular service area or it is travelling between service centres. The first part of the third stage extracts samples of service times in each service area, samples of travelling times between each pair of service areas and samples of customer interarrival time. By mining the tagged customer entities' paths in the system, the second part of this stage creates the initial structure of the queueing network model. The final stage of the pipeline fits a hyper-Erlang distribution to each extracted time sample using the G-FIT tool [130] and finalises the structure of the output queueing network accordingly. It also infers routing policies and service disciplines adopted at each service centre. Together, these yield a parameterised queueing network model representing the underlying system.

To evaluate the accuracy of the developed data processing pipeline, this research conducts three case studies based on synthetic data generated through simulations of queueing systems

with known system structure and parameters. In order to evaluate how the data processing pipeline performs in the presence of noisy real-time location tracking data, another case study is conducted based on location traces collected in an experiment environment. The inferred probabilistic models for characterising customer interarrival time and service time are specified as hyper-Erlang distributions. The goodness-of-fit test results presented in the case studies have shown that our data processing pipeline can find the hyper-Erlang distributions that fit well the extracted time delay samples. The first and second moments of the inferred hyper-Erlang distributions are also within small deviations from the true values; which means that the inferred hyper-Erlang distributions exhibit similar probabilistic characteristics as the true underlying distributions. Simulation-based response time analysis is also conducted in the first three case studies based on synthetic data to evaluate how well the inferred queueing network model can approximate the real underlying system in terms of performance-related measurements and response time distributions. The results have show that the inferred queueing network models capture well the behaviour of the true underlying system.

1.3.2 LocTrackJIQNS

This thesis presents LocTrackJINQS, a flexible and extensible spatiotemporal simulation tool for systems that involve the flow and processing of customers at multiple service centres. Developed based on the multiclass queueing network simulation package JINQS, LocTrack-JINQS retains the abstract model specification power of JINQS while providing additional low-level information concerning entity movement such as entity physical location and its speed of movement). Beside traditional performance metrics, LocTrackJINQS produces as output a trace of each entity's location in the system over time. It can thus be used to generate synthetic location tracking data for location-based research or applications (e.g [76], [11], [12]).

1.4 Thesis Outline

The remainder of this thesis is organised as follows:

Chapter 2 presents the theories and other materials relevant to this research. The first three sections introduce fundamental theories in stochastic processes and especially with more details in those closely related to this research, including Poisson process, Markov Chains and Markov process. Some important results in queueing theory are also presented, followed by introduction on several famous queueing network models. Section 2.4 covers elementary theories in distribution fitting and maximum-likelihood parameter estimation. Section 2.5 provides an overview of phase-type distributions and their applications in modelling traffic with heavy-tailness, which has been found common in many domains, especially in computer and network systems. The section then focuses on hyper-Erlang distributions, which is the type of phase-type distribution used in this research for modelling time delays incurred in a system; it also introduces G-FIT, the fitting tool employed for finding the hyper-Erlang distributions that best fit the extracted time delay samples. Section 2.6 provides a brief introduction and comparison of common wireless location tracking technologies. The last part of the chapter presents previous research works most relevant to this research; especially those related to workflow mining, large RFID database mining as well as modelling real-life systems with queueing networks.

Chapter 3 presents the four-stage data processing pipeline developed in this research. It first gives a general description of the initial inputs and final output of the pipeline and the function of each stage. The following sections provide details on how each stage is implemented, including the techniques adopted/adapted and the formats of the input and output data.

Chapter 4 presents LocTrackJINQS, the spatiotemporal simulation tool developed in this research for the purpose of generating location tracking data through simulation. It gives an overview of its software architecture as well as the functionalities LocTrackJINQS supports. It compares LocTrackJINQS with its predecessor JINQS and presents the

major extensions implemented in LocTrackJINQS; before it summarises the required user inputs for constructing simulations with LocTrackJINQS and generated simulation outputs. The chapter is concluded with a simple example demonstrating the user interface of LocTrackJINQS and how LocTrackJINQS can be used to simulate a real-life system.

Chapter 5 The chapter begins with introduction of the methodology adopted in this research for demonstrating the effectiveness and accuracy of the data processing pipeline. It then presents four case studies, three based on synthetic location data and one based on real location tracking data collected from an experiment environment. The chapter is concluded with analysis and discussion on the case study results.

Chapter 6 concludes the thesis by summarising the achievements of this research, discussing potential areas of application and highlighting opportunities for future work.

1.5 Statement of Originality and Publications

I declare that this thesis was composed by myself, and that the work it presents is my own, except where otherwise stated. The following publications arose from work conducted during the course of this PhD:

- **European Conference on Modelling and Simulation** (ECMS '09) [76] presents the early efforts in developing a methodology toward automated inference of queueing network models from high-precision location tracking data. It introduces the very first version of the four-stage data processing pipeline, based on which our methodology is developed. The pipeline estimates the structure of the network, which is specified by routing probabilities and characterises the underlying interarrival and service time distributions of its component service centres. The paper also describes four case studies in which real location tracking data is collected (using Ubisense RTLS [131]) in a controlled experiment environment with previously-known parameter settings (e.g. customer arrival rate and

service rates). The paper later presents evaluation results of our method's effectiveness and accuracy based on data collected from the case studies. Materials from this paper can be found in Chapter 3 and Chapter 5.

- **International ICST Conference on Performance Evaluation Methodologies and Tools** (VALUETOOLS '11) [11] presents an automated technique which takes as input high-precision location tracking data – potentially collected from a real-life system – and constructs a hierarchical Generalised Stochastic Petri Net performance model of the underlying system. [11] extends the four-stage data processing pipeline introduced by [76] but adopts GSPN as its modelling formalism. Different from [76], [11] adapts DBSCAN clustering technique to infer the locations as well as the sizes of the service centres, instead of assuming they are previously-known; it is thus possible to identify the hidden bottlenecks in the system. [11] uses hyper-Erlang distributions for modelling service time at each service centre, time intervals between two contiguous customer arrivals and customer travelling times between a pair of service centres; G-FIT is the tool it mainly uses for distribution fitting. Materials regarding the application of DBSCAN clustering to identify the possible service centre locations and how this research adapts G-FIT for finding the best-fit hyper-Erlang distributions can be found in Chapter 3.

- **International Workshop on Practical Applications of Stochastic Modelling** (PASM '11) [75] presents our work in implementing LocTrackJINQS –a Java-based location-aware simulation tool for multiclass queueing networks. It describes the features LocTrackJINQS inherits from its predecessor JINQS and the major extensions implemented in LocTrackJINQS for the purpose of providing information on low-level entity movement during simulation. The paper gives an overview of LocTrackJINQS's software architecture, followed by discussion on the implementation issues. The paper also illustrates the possible application of LocTrackJINQS by giving a case study of airport custom queues, which exemplifies a multiclass queueing network with customer class-based service time distributions. The paper is concluded with future possible additions to the current version of LocTrackJINQS. Most of the materials of [75] can be found in Chapter 4.

Chapter 2

Background

2.1 Stochastic Processes

The behaviour of a system, or a process, can often be characterised by all the possible states it may occupy and by describing how it moves from one state to another. Thus, a system or a process can be represented by using a *stochastic process*, which is defined as follows:

Definition 2.1 (**Stochastic Process**). *A stochastic process is a family of random variables* $\{X(t), t \in T\}$ *defined on a probability space. The parameter* t *usually represents time and* T *is the parameter space, which can be taken as a set of points in time. The set of values of* $X(t)$ *might take, usually called the state space, represents all the possible states the stochastic process might enter at different time points* ts.

A stochastic process is said to be *stationary* if its state evolution does not depend on the initial time point. Let $\{X(t)\}$ denote the state of a stochastic process at time t and such process is said to be *stationary* if

$$\begin{aligned} &\text{Prob}\{X(t_1) \leq x_1, X(t_2) \leq x_2, \ ..., X(t_n) \leq x_n\} \\ &= \text{Prob}\{X(t_1 + \alpha) \leq x_1, X(t_2 + \alpha) \leq x_2, \ ..., X(t_n + \alpha) \leq x_n\} \end{aligned}$$

for all n and all t_i and x_i, where $i = 1, 2, ..., n$.

When transitions of a stochastic process between states depend on the amount of time the process has spent on the current state, such stochastic process is said to be *non-homogeneous*; otherwise, it is *homogeneous*.

2.1.1 Renewal Processes

A counting process $\{N(t), t > 0\}$, is a stochastic process which counts the number of events up to and including time t. We use the random variable X_n, $n \geq 1$ to denote the time between the occurrence times of event $n-1$ and event n, which allows us to define a renewal process as follows [126, 108]:

Definition 2.2 (**Renewal Process**). *A counting process is a renewal process if the sequence of the nonnegative random variables* $\{X_n, n \geq 1\}$ *that represents the time elapsed between events are independently and identically distributed.*

Let $\{N(t), t > 0\}$ be a renewal process with interrenewal periods $\{X_n, n \geq 1\}$ and let S_n denotes the time at which the nth renewal occurs, i.e.,

$$S_0 = 0,\ S_n = X_1 + X_2 + ... + X_n,\ n \geq 1.$$

We also define here the parameter γ:

$$\gamma \stackrel{\text{def}}{=} \frac{1}{E[X]} \tag{2.1}$$

There are two important results on the limiting behaviour of renewal processes, stated as follows [126, 108]:

Proposition 2.1. *The average number of renewals per unit time satisfies*

$$\frac{N(t)}{t} \to \gamma \text{ as } t \to \infty$$

Thus, γ is also called the *rate* of the renewal process.

We use $M(t)$ to denote the expected number of renewals of a renewal process by time t, $E[N(t)]$. $M(t)$ is also called the *renewal function* of such renewal process.

Proposition 2.2. *The expected rate of renewals,* $\frac{M(t)}{t}$ *satisfies*

$$\frac{M(t)}{t} \to \gamma \text{ as } t \to \infty$$

The derivative of the renewal function $M(t)$, denoted by $m(t)$, is called the *renewal density*. It can also be shown that

$$\lim_{t\to\infty} m(t) = \gamma$$

The results shown above hold for all renewal processes.

2.1.2 Poisson Processes

A Poisson process $\{N(t), t \geq 0\}$ is a special type of continuous-time counting process, defined as follows [126, 108]:

Definition 2.3 (**Poisson Process**). *Let* $\{N(t), t \geq 0\}$ *be the number of the events that occur during the time interval* $(0, t]$. *A counting process* $\{N(t), t \geq 0\}$ *is a Poisson process when it possesses the following properties:*

1. $N(0) = 0$.

2. ***Independent Increments****: the number of events that occur in non-overlapping time intervals are mutually independent.*

3. ***Stationary Increments****: the number of events that occur within any time interval only depends on the length of the interval, regardless of the past history.*

4. For some sufficiently small h and some positive constant $\lambda > 0$,

$$
\begin{aligned}
&Prob\{One\ event\ in\ (t, t+h]\} = \lambda h + o(h), \\
&Prob\{Zero\ event\ in\ (t, t+h]\} = 1 - \lambda h + o(h), \\
&Prob\{More\ than\ one\ event\ in\ (t, t+h]\} = o(h), \\
&where\ \lim_{h \to 0} \frac{o(h)}{h} = 0
\end{aligned}
$$

The definition given by Definition 2.3 is equivalent to saying that:

1. A Poisson process $\{N(t), t \geq 0\}$ with rate $\lambda \geq 0$ and $\{N(0) = 0\}$ is such that the number of events that occur in any time interval of length t has a Poisson distribution as follows:

$$\text{Prob}\{N(t) = n\} = e^{-\lambda t} \frac{(\lambda t)^n}{n!}, \text{n=0, 1, 2, ...}$$

2. If the elapsed times between successive events are independently and identically exponentially distributed with parameter $\lambda \geq 0$, then the stochastic process $\{N(t), t \geq 0\}$ is a Poisson process.

It can be easily shown that for a Poisson process the expected number of renewals by time t, denoted by $M(t)$, is equal to λt and hence its renewal density $m(t) = \lambda$.

2.1.2.1 Poisson Arrivals See Time Averages (PASTA)

There are two different views of observing a queueing system: the view as seen by an arriving customer and that as seen at a random time point, i.e., at equilibrium. Normally these two viewpoints would lead to different statistical outcomes. However, they are statistically equivalent if the arrival process is Poisson and the system reaches equilibrium, which is called PASTA (Poisson Arrivals See Time Averages) property. More formally, if p_n is the probability of the number of customers contained in the system at equilibrium and a_n is the probability of an arriving customer finding that the system having n customers. The PASTA property says

that $a_n = p_n$. The PASTA property arises from the memoryless property of the interarrival time distribution (exponential distribution) of a Poisson process and does not hold for other stochastic processes.

2.1.3 Markov Chain and Markov Processes

A Markov process is a class of stochastic process that satisfies the Markov Property, which is defined as follows [106]:

Definition 2.4 (**Markov Property**). *Let* $\{X(t),\ t \in T$ *and* $X(t) \in S\}$ *be a stochastic process defined on the parameter set* T *and state space* S*. It is said to possess the Markov property if it satisfies*

$$\begin{aligned} &Prob\{X(t_0+t_1) \leq x \mid X(t_0) = x_0,\ X(\tau), -\infty < \tau < t_0\} \\ =&Prob\{X(t_0+t_1) \leq x \mid X(t_0) = x_0\}, \end{aligned} \tag{2.2}$$

for any value of t_0 and for $t_1 > 1$. In other words, the evolution of a Markov process in the future has limited dependency on the previous history of the process and the current state $X(t_0)$ offers sufficient information for the future behaviour of the process.

In our work, we only consider Markov processes with discrete *state space* S, which means that the values of $X(t)$ are nonnegative integers. If the *parameter set* T is discrete then we call the process a *Markov chain* or *Discrete-time Markov Chain*; if T is continuous, then the process is called a *Markov Process* or *Continuous-time Markov Chain*.

2.1.3.1 Discrete-time Markov Chain (DTMC)

Since the parameter set T of a DTMC is discrete, we can represent the set T by the set of nonnegative integers $\{0, 1, 2, ...\}$. Equation 2.2 defining Markov property, when applied to

DTMC, becomes:

$$\begin{aligned} &Prob\{X_n = x_n \mid X_{n-1} = x_{n-1},\ X_{n-2} = x_{n-2},\ ...,\ \} \\ =&Prob\{X_n = x_n \mid X_{n-1} = x_{n-1}\} \end{aligned} \tag{2.3}$$

We denote this *one-step* transition probability $Prob\{X_n = x_n \mid X_{n-1} = x_{n-1}\}$ at time $n-1$ as $p_{ij}(n)$. If the *DTMC*, with its state space S, is said to be *time homogeneous*, it satisfies

$$\begin{aligned} &Prob\{X_n = j \mid X_{n-1} = i\} = Prob\{X_{n+m} = j \mid X_{n+m-1} = i\} \\ &n = 1,\ 2,\ ...,\ m \geq 0,\ i, j \in S \end{aligned}$$

If a DTMC is *time-homogeneous*, we can write

$$p_{ij} = Prob\{X_n = j \mid X_{n-1} = i\},\ n = 1,\ 2,\ ...,\ i, j \in S \tag{2.4}$$

Then the evolution of a *time homogeneous* DTMC can be described using a probability transition matrix P, defined as follows:

$$P^n = \begin{pmatrix} p_{00} & p_{01} & ... & p_{0j} & ... \\ p_{10} & p_{11} & ... & p_{1j} & ... \\ . & . & ... & . & ... \\ . & . & ... & . & ... \\ . & . & ... & . & ... \\ p_{i0} & p_{i1} & ... & p_{ij} & ... \\ . & . & ... & . & ... \\ . & . & ... & . & ... \\ . & . & ... & . & ... \end{pmatrix}.$$

The dimension of the probability transition matrix P is the size of the state space S. Throughout this report, unless specified otherwise, we only consider $time-homogeneous$ DTMCs.

2.1.3.1.1 Chapman-Kolmogorov Equations Let p_{ij}^n be the probability of a DTMC being in state j n steps after being in state i; it is also the $(i,\ j)\,th$ entry of the $n-step$ probability transition matrix P^n. According to *Chapman-Kolmogorov equations*,

$$p_{ij}^n = \sum_{k \in S} p_{ik}^m \, p_{kj}^{n-m} \tag{2.5}$$

We now define the vector

$$\pi^{\mathbf{n}} \stackrel{\text{def}}{=} \left(\begin{array}{cccc} \pi_0^0, & \pi_1^0, & \ldots \end{array} \right) \tag{2.6}$$

where each element π_j^n $(j \in S)$ represents the probability of the the Markov chain being in state i at step n. The dimension of the vector $\pi^{\mathbf{n}}$ is the size of the state space $\mathbf{S}$. $\pi^{\mathbf{n}}$ can then be calculated as follows:

$$\pi^{\mathbf{n}} = \pi^{\mathbf{0}} \, \mathbf{P^n} \tag{2.7}$$

2.1.3.1.2 The Stationary Probability Distribution of a Ergodic DTMC A DTMC is said to be *ergodic* if all the states are positive recurrent, aperiodic and in only one irreducible set (the detailed definitions of positive recurrent, aperiodic and irreducibility can be found in [126]). Let $N_{ij}\,(n)$ be the number of visits to state j n steps after the chain starts at state i. If the chain is ergodic, $N_{ij}\,(n)$ can be seen as a renewal process. We define π_j as the inverse of the mean recurrence time for state j (which is equivalent to the γ defined in Equation 2.1). We can show that given that all the states are aperiodic in a ergodic DTMC,

$$\lim_{n \to \infty} p_{ij}^n = \pi_j \tag{2.8}$$

for any initial state i. That is, the probability of finding an ergodic chain in a specific state i is invariant with time that is, *stationary* with time. The values of π_i, $i \in \mathbf{S}$ are called *stationary probabilities* of the Markov Chain. It can be easily shown that the vector of stationary probabilities, denoted by π, satisfies the following equation:

$$\pi = \pi \mathbf{P} \tag{2.9}$$

An important result on the the stationary probabilities is presented by the following proposition:

Proposition 2.3. *The stationary probabilities of an irreducible Markov chain with one ergodic set and state space* $\mathbf{S}$ *are* ***unique*** *and satisfy*

$$\pi_j = \sum_{k \in \mathbf{S}} \pi_k p_{kj},\ j \in \mathbf{S} \tag{2.10}$$

and

$$\sum_{j \in \mathbf{S}} \pi_j = 1. \tag{2.11}$$

That is, the stationary probabilities can be found by solving the linear equations shown in Equation 2.10 subjecting to the normalisation condition shown in Equation 2.11.

2.1.3.2 Continuous-time Markov Chain (CTMC)

In a DTMC, the state transitions can only happen at discrete time steps, while in a CTMC (or Markov process) the state transition may occur at any point in time. Analogous to the probability transition matrix $\mathbf{P^n}$ of a DTMC at time step n, the state transitions of a CTMC can be characterised by a *generator matrix* $\mathbf{Q}(\mathbf{t})$ at time t. Each entry of the generator matrix $\mathbf{Q}(\mathbf{t})$, denoted as $q_{ij}(t)$, defines the *instantaneous* rate that the chain changes from state i to state j at time t. More formally, given a CTMC with state space $\mathbf{S}$ we define that

$$q_{ij}(t) = \lim_{\tau \to 0} \frac{Prob\{X(t+\tau) = j \mid X(t) = i\}}{\tau},\ i, j \in \mathbf{S} \tag{2.12}$$

The total rate out of state i at time t is

$$q_i(t) = \sum_{j \in \mathbf{S},\, j \neq i} q_{ij}(t),\ \forall i \in \mathbf{S}. \tag{2.13}$$

Therefore, the ith diagonal element of the generator matrix $\mathbf{Q}(\mathbf{t})$, which represents the transition rate corresponding to the chain remaining in the current state, is defined as follows:

$$q_{ii}(t) = -\sum_{j \in \mathbf{S},\, j \neq i} q_{ij}(t),\ \forall i \in \mathbf{S}. \tag{2.14}$$

If a CTMC is time homogeneous, Equation 2.12 and Equation 2.14 become

$$q_{ij} = \lim_{\tau \to 0} \frac{Prob\{X(t+\tau) = j \mid X(t) = i\}}{\tau},\ i, j \in \mathbf{S}$$
$$q_{ii} = -\sum_{j \in \mathbf{S},\, j \neq i} q_{ij},\ \forall i \in \mathbf{S}.$$

Consider a *homogeneous* CTMC and let T_i be the time the chain spends in state i until it moves to another state (which is also called sojourn time or state holding time), we can use the Markov property to prove that

$$Prob\{T_i \leq s + t \mid T_i > s\} = Prob\{T_i \leq t\} \tag{2.15}$$

Equation 2.15 is called the *memoryless property*. Since the only continuous probability distribution that possesses the memory-less property is exponential distribution, it follows that the state holding time (sojourn time) of a CTMC is exponentially distributed. That is, given a state i of a homogeneous CTMC, the probability distribution of the state holding time in state i is

$$F(T_i = x) = 1 - e^{-\mu_i x},\ x > 0$$

where

$$\mu_i = \sum_{j \neq i} q_{ij} = -q_{ii} \tag{2.16}$$

The results given by Equation 2.15 and Equation 2.16 do not hold in the case of *nonhomogeneous* CTMC.

2.1.3.2.1 The Stationary Probability Distribution of a Ergodic CTMC Similar to the discussion on the stationary probability distribution in Section 2.1.3.1, the stationary probabilities of a *ergodic* and *time homogeneous* CTMC with state space $\mathbf{S}$, the stationary probability π_j is given by

$$\pi_j = \lim_{t\to\infty} Prob\{X(t) = j \mid X(0) = i\},\ i, j \in \mathbf{S} \tag{2.17}$$

The stationary probability vector, $\pi = (\pi_0, \pi_1, ...)$, satisfies the following linear equations:

$$\pi\mathbf{Q} = 0, \tag{2.18}$$

$$\sum_{i\in S} \pi_i = 1 \tag{2.19}$$

Analogous to Proposition 2.3,

Proposition 2.4. *For an irreducible and time homogeneous CTMC with only one ergodic set in its state space* $\mathbf{S}$*, the stationary probabilities are unique and can be found by solving the following linear equations:*

$$\sum_{i\in\mathbf{S}} \pi_i q_{ij} = 0,\ \forall j \in \mathbf{S} \tag{2.20}$$

subject to the condition that

$$\sum_{j\in\mathbf{S}} \pi_j = 1. \tag{2.21}$$

2.2 Queueing Theory and Simple Queues

Queueing network is one of the most widely-used methods in stochastic modelling. It models real-life systems as networks of one or more queues interacting with each other. A vast body of related theory, called *queueing theory*, provides convenient tools for quantitative analysis on the efficiency and effectiveness of the system of interest. We call entities that offer service or resources as *Servers*. Those who enter a system requiring service or resources for some period of time are called *Customers*. Some servers contain a buffer with limited or infinite capacity

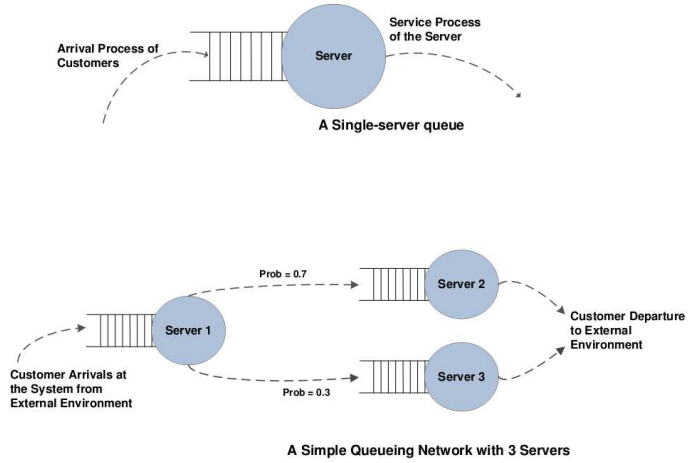


Figure 2.1: Two simple examples of queueing networks

where customers wait for service; this is called *Queue*. However, we can also refer to both server and queue collectively as a *Queue*. Figure 2.1 shows a single-server queue, which is the simplest queueing network and a simple 3-server queueing network.

2.2.1 Queueing theory

2.2.1.1 Kendall's Notation

A queue can be described by characterising how customers arrive at the queue to request service (customer arrival process) and how the server(s) serve customers (service process and service scheduling discipline). Kendall's notation offers a standardised way to characterise a queueing system by using 6 variables $A/S/k/c/N/Z$ in this notation. They have following interpretations:

- A indicates the customer inter-arrival time distribution; S stands for the service time probability distribution. Some possible distributions include M for Markovian (Poisson arrivals and exponential service times), E for Erlang, D for deterministic, and G for general distribution

- k indicates the number of servers

- c indicates the capacity of the system, which is the maximum number of customers the system can hold. Customers who arrive when the system already contains c customers are rejected from receiving service. It is assumed to be infinite if not specified
- N indicates the size of the customer population
- Z indicates the queue scheduling discipline.

2.2.1.2 Queueing discipline

The scheduling discipline determines how the customers are selected from the queue and taken into service. There are several scheduling disciplines, listed as follows [126]:

- FCFS/FIFO (first come/in, first served/out). This discipline selects the customer who has been waiting for the longest time as the next one to be served.
- LCFS (last come, first served). It takes the customer that last joins the queue for service.
- LCFS-PR (last come, first served, preempt resume). It is LCFS with the preempt resume policy. Preempt resume policies usually classify customers into different priority classes. If a customer of higher priority, say A, arrives while another customer with lower priority, say B, is being served, the service of B will be interrupted. The server then serves A and will resume its service to B later.
- SIRO (service in random order). The server choose the next customer in the queue to be served at random.
- RR (round robin). In this scheduling discipline, when its service begins, each customer is given a fixed amount of service time, which is called *time slice*. The customer's service will get interrupted if it is not completed before the time slice is expired. The interrupted customer will go to the end of the queue waiting for its next turn to be served, until its service gets completed. If the customer's service is finished before the time slice ends, the server will choose the next one at the head of the queue to be served.

- PS (processor sharing). Process sharing discipline can be approximated by RR discipline with the size of time slice approaching zero. All customers in the system are simultaneously in service and receiving the same fraction of the server capacity.

- PRIO (priority scheduling). In this scheduling discipline, each customer is assigned different priorities that determine the order in which they will be served. The server will always select the customer with the highest priority from the queue.

2.2.1.3 Performance Measures

Given a queueing system, there are several measures often used to evaluate the effectiveness of the underlying system; in this research the following performance measures are especially of our interest:

- Mean queue length. This measure describes the average number of customers waiting for service in the system at steady state.

- Response time and queueing time. The time a customer spends inside the system, from the instance when it enters the system to the instance when it leaves the system, is defined as response time or sojourn time. The response time is composed of two elements; the time that the customer spends in the queue waiting to be served, called queueing time; and the service time the customer receive.

- Utilisation. Given a single server queue, the utilisation is defined as the fraction of time that the server is occupied. If customers arrive at the system at rate λ and the single-server processes incoming customers at rate μ, then the utilisation is equal to λ/μ. In the case of multiple-server queueing systems, the utilisation is calculated as $\lambda/c\mu$, where c is the number of the servers.

2.2.1.4 Little's Law

Given the following notations:

- λ denotes the average arrival rate of customers to the system
- $\overline{N}$ denotes the average number of customers in the system
- $\overline{T}$ denotes the average response time (also called sojourn time) in the system

Little's law says that the relationship among λ, $\overline{N}$ and $\overline{T}$ when the system is in steady state can be described using a simple equation as follows:

$$\overline{N} = \lambda \overline{T} \tag{2.22}$$

Little's law may be applied to different individual parts within a queueing system, namely the queue and the server. For example, given the average waiting time spent in the queue $\overline{W}$, and the average queue length $\overline{N_q}$ it follows from Little's Law that

$$\overline{N_q} = \lambda \overline{W}$$

It should be noted that the application of Little's law is independent of the interarrival time distribution, service time distribution, the number of servers in the system and the service discipline. The law only requires that customers are not created or lost inside the system and the system reaches its steady state. Little's law can also be extended to higher moments. For a queueing system in its steady state, let N denote the number of customers the system contains and T denote the response time

$$E\left[N\left(N-1\right) \ldots \left(N-k-1\right)\right] = \lambda^k E\left[T^k\right] \tag{2.23}$$

When $k = 1$, the Equation 2.23 is the Little's law. The expression on the left-hand side of Equation 2.23 is called the k-th factorial moment.

2.2.2 General birth-and-death process and $M/M/1$ queue

Let $X(t)$ be a Markov Process with discrete state space; then $X(t)$ is a birth-and-death process if, for very small Δ_t:

$$\begin{aligned} P\{X(t+h) - X(t) = k | X(t) = n\} & \\ &= \lambda_n \Delta_t + o(\Delta_t), \text{if } k = 1 \\ &= \mu_n \Delta_t + o(\Delta_t), \text{if } k = -1 \\ &= o(\Delta_t), \text{if } |k| > 1 \end{aligned}$$

Please note that the last condition,

$$o(\Delta_t), \text{if } |k| > 1,$$

implies that the probability of more than one events occurring at the same time is approximately zero. From the above definition it is clear that in a birth-and-death process, only the transitions among neighbouring states are allowed.

2.3 Queueing Network

A queueing network is usually presented as a directed graph, where nodes represent service centres in the system; directed links, with associated routing policies, specify the paths customers (also called jobs) follow when they move within the system requesting service/resources. The state of the system is often represented by a tuple of nonnegative integers with each number

specifying the number of customers occupying the corresponding server/queue at a given time point. A queueing network may be open, closed or mixed, depending on whether a fixed population of customers remain within the system or not. An open queueing network has arrivals from external sources as well as departures to external environment, while in a closed queueing network no jobs/customers ever leave the network and the total number of jobs/customers remain the same all the time.

Consider a network with M service centres; at service centre there are n_i customers. The vector $\mathbf{n} = (n_1,...,n_M)$ represents the state of the network and $\pi(\mathbf{n})$ is the stationary probability distribution of the network being in state $\mathbf{n}$. Let π denote the stationary state probability vector of the Markov process and $\mathbf{Q}$ the corresponding transition rate matrix, the stationary state probability π can be solved using the linear system called *global balance equations*:

$$\pi\mathbf{Q} = 0 \tag{2.24}$$

However, only with certain special structures of the matrix $\mathbf{Q}$ can an exact solution of Equation 2.24 be obtained. A large class of queueing networks, either closed or open, has been shown to have a straightforward and computationally efficient solution, often termed *product form solutions* if the queues in the network are *quasireversible*. A queue is said to be quasireversible if its current state, the past departures and the future arrivals are mutually independent. The famous Burke theorem regarding quasireversibility of $M/M/k$ queues is stated as follows:

Theorem 2.5. *(Burke) Consider an $M/M/k$ system (k can be a finite or infinite integer) with arrival rate λ. The system is in steady state. Then the following hold true:*

1. *The departure process is also Poisson with rate λ.*
2. *The number of customers in the system at time t is independent of previous departure times.*

Product form solution of a queueing system of single customer class takes the follow form:

$$\pi(\mathbf{n}) = \frac{1}{G} V(n) \prod_{i=1}^{M} g_i(n_i), \tag{2.25}$$

where G is a normalising factor, n is the total network population, function V is defined in terms of network parameters and g_i is a function of state n_i and depends on the type of service centre i,$1 \leq i \leq M$. For open networks, $G = 1$; while for closed networks $V = 1$.

Similar formula is derived for queueing networks with multiple types of customers. To represent different behaviours (in terms of arrival patterns and service demands) of various customer types, a "chain" is used to gather the customers of the same type. A chain consists of a set of classes that represent different stages of processing in the system for a given type of customer; that is, a customer entity belonging to such type would move from one class to another along the chain. For example, if customer entities of a certain type receives two successive services from two service centres, the chain representing such behaviour connects one class to the other. Let R denote the total number of chains; S denote the state of the network including the number of customer entities at each service centre. The product form solution for the stationary probability of state π in a multiclass queueing network is as follows:

$$\pi(S) = \frac{1}{G} \prod_{r=1}^{R} V_r(K_r) \prod_{i=1}^{M} g_i(n_i), \tag{2.26}$$

where G is still a normalising factor, K_r is the total population from chain r in the network, function V_r is defined in terms of network parameters; and function $g_i(n_i)$ depends on the type of service centre. Similarly $G = 1$ for open networks and $V = 1$ for closed networks.

2.3.1 Jackson Queueing Network

The first important result concerning product-form queueing networks was introduced by Jackson [81] for open queueing networks that satisfy the following conditions:

- Each service centre consists of an infinite FCFS queue; there are c_i exponential servers, each with service rate μ_i.
- External customer arrivals at service centre i are Poisson with rate γ_i.
- After completing service at the service centre i, a customer would proceed to the next service centre j with probability r_{ij} or depart forever from the system with probability $1 - \sum_{j=1}^{M} r_{ij}$.

The routing matrix $\mathbf{R} = [r_{ij}]$ defines the allowed transitions among different service centres in the network. The total customer arrival rate to service centre i, denoted by λ_i, is given by the sum of external arrivals (Poisson) to the service centre i plus arrivals from other internal service centres. The following traffic equation is obtained:

$$\lambda_i = \gamma_i + \sum_{j=1}^{M} \lambda_j r_{ji} \tag{2.27}$$

The Jackson theorem is stated as follows:

Theorem 2.6. *(Jackson) For a Jackson network with effective arrival rate λ_i to service centre i and assuming that $\lambda_i < c_i\mu_i$ for all i (c_i is the number of the servers at service centre i), the following are true in steady state:*

1. *Each service centre behaves as if its customer arrival process were Poisson with rate λ_i.*
2. *The number of customers at any service centre is independent of the number of customers at any other service centre.*

In other words, the joint distribution of numbers of customers in a Jackson queueing network can be calculated as the product of the marginal distributions at individual service centres in the system. That is,

$$P(k) = \prod_{i=1}^{M} P_i(k_i), \tag{2.28}$$

where $P_i(k_i)$ is the probability distribution of having k_i customers at service centre i; it is given as if service centre i functions as an independent $M/M/k$ type of queue. Thus,

$$P(k) = \prod_{i=1}^{M}(1-\rho_i)\rho_i^{k_i}, \text{ where } \rho_i = \frac{\lambda_i}{\mu_i} \tag{2.29}$$

2.3.2 Gordon and Newell networks

Similar product-form solutions are later introduced by Gordon and Newell [68, 69] for closed queueing networks with only exponential, FCFS service centres. Consider a closed single-class queueing network consisting M service centres. Let $N = \sum_{i=1}^{M} n_i$ and the state of the network is again defined as $n = (n_1, n_2, n_3, ..., n_M)$. The total number of states of the network is given by $\begin{pmatrix} N+M-1 \\ M-1 \end{pmatrix}$.
Let matrix $\mathbf{P} = [p_{ij}]$ be the stochastic matrix where its element p_{ij} is the fraction of departures from service centre i that would go next to service centre j. A "visit ratio" for service centre i, denoted by v_i, is introduced to give the mean number of visits to service centre i *relative* to a specific service centre. The visit ratios can also be taken as arrival rates normalised so that the arrival rate at the reference service centre becomes 1. In the following context, this reference service centre is service centre 1. Thus, the system traffic equation is:

$$v_1 = 1 \text{ and } v_i = \sum_{j=1}^{M} v_j p_{ji},\ i = 1, 2, ..., M. \tag{2.30}$$

The equilibrium distribution is given by the following product form:

$$P(n) = P(n_1, n_2, ..., n_M) = \frac{1}{G(N)} \prod_{i=1}^{M} f_i(n_i), \tag{2.31}$$

where $f_i(n_i) = \frac{v_i^{n_i}}{\prod_{k=1}^{n_i} \mu_i(k)}$, if service centre i is load dependent; and $f_i(n_i) = \left(\frac{v_i}{\mu_i}\right)^{n_i}$ if service centre i is load independent.

2.3.3 BCMP queueing network

BCMP theorem, proved by Baskett, Chandy, Muntz and Palacios in their work [27], defines the BCMP queueing networks with product form solutions for open, closed or mixed queueing network models with multiple customer classes, different types service disciplines and service time distributions. BCMP theorem considers four types of multiclass service centres; all of them are quasireversible:

- Type 1: FCFS service discipline and identical exponential service time for each chain.
- Type 2: PS service discipline.
- Type 3: Infinite-server (IS) service centres.
- Type 4: LCFS-PR service discipline.

It is required that for types 2, 3, and 4, the service time distributions should have rational Laplace transforms (e.g. phase-type distributions); the average service rate may depend on the population along each customer chain. At each service centre, the service rate can depend on its population or its population belonging to a particular chain; type 1 service centre can only have its service rate dependent on the overall population. External arrivals in BCMP queueing networks are Poisson. Let $\lambda(n)$ and $\lambda_r(K_r)$ denote the average overall arrival rate to the network dependent on the total number of customers n in the network and on the population K_r along chain r, respectively.

Let the routing matrix for chain r be $P^{(r)} = \left[p^{(r)}_{ic;jd}\right]$. $p^{(r)}_{ic;jd}$ denotes the probability of a customer leaving service centre i in class c and entering service centre j in class d; $p^{(r)}_{ic;0}$ denotes the probability of a customer leaving the network from being in class c at service centre i.

For chain r, the visit ratio at service centre i from customers in class c is denoted as v^r_{ic}. Thus, the visit ratio at service centre i from customers following chain r is $v_{ir} = \sum_{c \in C_{ir}} v^r_{ic}$, where C_{ir}

is the set of classes at service centre i belonging to chain r; this is the traffic equation defined for chain r.

The product form solutions for BCMP networks are given by Equation 2.25 and Equation 2.25, where $V(n) = \prod_{k=0}^{n-1} \lambda(k)$ for open queueing networks; and $V_r(K_r) = \prod_{r=1}^{R} \prod_{k=0}^{K_r-1} \lambda_r(k)$ for closed queueing networks. Let μ_{ir} be the service rate at service centre i for customer belonging to chain r; and $\rho_{ir} = \frac{v_{ir}}{\mu_{ir}}$. For service centres of type 1, 2, and 4, $g_i(n_i) = \prod_{r=1}^{R} \frac{\rho_{ir}^{n_{ir}}}{n_{ir}!}$; while for service centres of type 3, $g_i(n_i) = n_i! \prod_{r=1}^{R} \frac{\rho_{ir}^{n_{ir}}}{n_{ir}!}$.

2.4 Distribution fitting

2.4.1 Maximum Likelihood Method

In this section, we consider a probabilistic model, usually a probability density function $p(\mathbf{x} \mid \Theta)$, governed by a set of parameters Θ. There are also a set of N observations $\mathcal{X}$; each observation is represented by a data vector, i.e. $\mathcal{X} = (\mathbf{x_1}, \ldots, \mathbf{x_N})$. We assume that $(\mathbf{x_1}, \ldots, \mathbf{x_N})$ are independently drawn from the same probability density function $p(\mathbf{x} \mid \Theta)$. The probability of obtaining this set of data, given the chosen probability model $p(\mathbf{x} \mid \Theta)$, where Θ is unknown, is defined as follows:

$$P(\mathcal{X} \mid \mathbf{\Theta}) = \prod_{i=1}^{N} p(x_i \mid \mathbf{\Theta}). \tag{2.32}$$

The function $P(\mathcal{X} \mid \mathbf{\Theta})$ is called the likelihood function, which is a function of the parameter set Θ and where $\mathcal{X}$ is fixed. The idea behind the maximum likelihood method is to find the values of Θ that maximise the likelihood function. That is, the maximum likelihood method is to find $\mathbf{\Theta}^*$ that

$$\mathbf{\Theta}^* = \underset{\mathbf{\Theta}}{\operatorname{argmax}}\ P(\mathcal{X} \mid \mathbf{\Theta}) \tag{2.33}$$

Often it is analytically easier to work with $\mathbf{Log}\left(P\left(\mathcal{X} \mid \mathbf{\Theta}\right)\right)$, which is called the log-likelihood function. $\mathbf{\Theta}^*$ can be found by solving the equations that the first derivatives of the likelihood or log-likelihood function with respect to the unknown parameters are equal to zero.

2.4.2 EM Algorithm

EM algorithm is a popular and efficient tool in finding the Maximum Likelihood (ML) estimates of the parameters of the underlying probability distribution. It is essentially an iterative procedure that gradually maximises the log likelihood function $L\left(\mathbf{\Theta} \mid \mathcal{X}\right)$. Assume that after ith iteration step, the estimates for the parameter set $\mathbf{\Theta}$ is given by $\mathbf{\Theta}^i$ and the log likelihood function calculated based on $\mathbf{\Theta}^i$ is $L\left(\mathbf{\Theta}^i \mid \mathcal{X}\right)$. The EM algorithm finds the new parameter estimates for $i+1$th step that are optimal for increasing the log likelihood function [35]. That is, the algorithm is looking for the update estimate $\mathbf{\Theta}^{i+1}$ that maximises

$$L\left(\mathbf{\Theta}^{i+1} \mid \mathcal{X}\right) - L\left(\mathbf{\Theta}^i \mid \mathcal{X}\right)$$

EM algorithm is especially useful when dealing with cases where missing or hidden variables or data are involved. Sometimes hidden variables are introduced on purpose in order to make the maximum likelihood estimation of $\mathbf{\Theta}$ tractable. Assume that $\mathcal{X}$ is the observed data set, which is called *incomplete data* and a complete data set $\mathcal{Z} = \left(\mathcal{X}, \mathcal{Y}\right)$ (where $\mathcal{Y}$ is the unobserved hidden data set) exists. We then assume the probability density function of $\mathcal{Z}$ given the parameter set $\mathbf{\Theta}$ can be specified by the join probability density function $p\left(\mathbf{x}, \mathbf{y} \mid \mathbf{\Theta}\right)$.

The following two sections describe the two major steps comprising each iteration step in EM algorithm.

2.4.2.1 Expectation step (E step)

This step determines the conditional expected value of the complete-data log-likelihood function given the observed data $\mathcal{X}$ and the current estimate of the parameter set $\mathbf{\Theta}^i$. Here we define

$$\begin{aligned}
&\mathrm{E}_{\mathcal{Z}|\mathcal{X},\mathbf{\Theta}\rangle}\left\{L\left(\mathbf{\Theta} \mid \mathcal{Z}\right)\right\} \\
=&\mathrm{E}_{\mathcal{Z}|\mathcal{X},\mathbf{\Theta}\rangle}\left\{L\left(\mathbf{\Theta} \mid \mathcal{X}, \mathcal{Y}\right)\right\} \\
=&\mathrm{E}_{\mathcal{Z}|\mathcal{X},\mathbf{\Theta}\rangle}\left\{\log P\left(\mathcal{X}, \mathcal{Y} \mid \mathbf{\Theta}\right)\right\} \\
=&\int_{\mathbf{y}\in\mathbf{\Upsilon}} \log p\left(\mathcal{X}, \mathbf{y} \mid \mathbf{\Theta}\right) f\left(\mathbf{y} \mid \mathcal{X}, \mathbf{\Theta^i}\right)\, d\mathbf{y},
\end{aligned}$$

where $\mathbf{\Upsilon}$ is the space of values the vector of unobserved data $\mathbf{y}$ can take and $f\left(\mathbf{y} \mid \chi, \mathbf{\Theta^i}\right)$ is the marginal probability density function of the unobserved data given the observed data and the current parameter estimate

2.4.2.2 Maximisation step (M step)

In this step, the EM algorithm finds an update estimate of that maximise $\mathrm{E}_{\mathcal{Z}|\mathcal{X},\mathbf{\Theta}\rangle}\left\{L\left(\mathbf{\Theta} \mid \mathcal{Z}\right)\right\}$. That is, we are looking for

$$\Theta^{i+1} = \underset{\mathbf{\Theta}}{\operatorname{argmax}}\ \mathrm{E}_{\mathcal{Z}|\mathcal{X},\mathbf{\Theta}\rangle}\left\{L\left(\mathbf{\Theta} \mid \mathcal{Z}\right)\right\}$$

These two steps are repeated as necessary and continually increase the log-likelihood until the algorithm converges to a local maximum of the log-likelihood function. The most widely used application of EM algorithm is to estimate the parameters of mixture probability models, given as follows [31]:

$$p\left(\mathbf{x} \mid \mathbf{\Theta}\right) = \sum_{i=1}^{M} \alpha_i p_i\left(\mathbf{x} \mid \theta_i\right),$$

where $\mathbf{\Theta} = (\alpha_1, ..., \alpha_M\, \theta_1, ..., \theta_M)$ such that $\sum_{i=1}^{M} \alpha_i = 1$ and p_i is a density function with parameter set θ_i.

2.4.3 Statistical goodness-of-fit tests

The goodness-of-fit test is commonly used to evaluate the compatibility of a set of random sample data with a theoretical probability distribution function. In other words, these tests are used to estimate how well the assumed distribution(s) fit(s) to the given sample data. Three most popular goodness of fit tests are Kolmogorov-Smirnov test, Anderson Darling test and Chi-squared test, all of which test the following two hypotheses:

- H_0: The given sample data are governed by the assumed probability distribution.
- H_1: The given sample data are not drawn from the assumed probability distribution.

Different goodness-of-fit tests have different ways of calculating measures estimating the discrepancy between collected data and the theoretical probability distributions being tested. Such measures, called test statistics, are used in testing the two hypotheses mentioned above. If the test statistics exceed critical values, which are calculated based on specified significance levels (α), the null hypothesis H_0 is rejected; otherwise, it is accepted that the given sample is drawn from the assumed probability distribution. Section 2.4.4, Section 2.4.5 and Section 2.4.6 give an overview of Kolmogorov-Smirnov test, Anderson Darling test and Chi-squared test, respectively.

2.4.4 Kolmogorov-Smirnov test

Kolmogorov-Smirnov test is based on the empirical cumulative distribution function ($ECDF$) of the random sample. Assume that we have N *ordered i.i.d* observations $\{X_1, X_2, X_3, ..., X_N\}$; the empirical CDF of which is defined as:

$$E_N(i) = \frac{n(i)}{N}, \tag{2.34}$$

where $n(i)$ is the number of observations less than X_i.

The Kolmogorov-Smirnov test statistic is referred to as D value; it calculates the largest absolute vertical difference between the theoretical CDF and the empirical CDF calculated from the sample:

$$D = \max_{1 \leq i \leq N} \left\{ F\left(X_i\right) - \frac{i-1}{N},\ \frac{i}{N} - F\left(X_i\right) \right\}, \tag{2.35}$$

where F is the theoretical cumulative distribution of the distribution being tested.

The Kolmogorov-Smirnov test is a non-parametric technique and the critical values are not calculated based on any any distributional assumption (for example, the normal distribution assumption is very common in statistical tests). However, its most serious limitation is that the distribution being tested must be fully specified. Also, Kolmogorov-Smirnov cannot be used for testing discrete distributions [109].

2.4.5 Anderson-Darling test

Anderson-Darling test is a modification of Kolmogorov-Smirnov test. Given a set of n *ordered* observations $\{X_1, X_2, X_3, ..., X_N\}$, the Anderson-Darling test statistic is calculated as follows:

$$A^2 = -N - S, \tag{2.36}$$

where

$$S = \sum_{i=1}^{N} \frac{(2i-1)}{N} \left[ln\left(F\left(X_i\right)\right) + ln\left(1 - F\left(X_{N+1-i}\right)\right) \right], \tag{2.37}$$

where F is the cumulative distribution function of the distribution being tested.

The differences between Anderson-Darling test and Kolmogorov-Smirnov test are [109]:

1. Anderson-Darling test gives more weight to the tail than does Kolmogorov-Smirnov test (Kolmogorov-Smirnov is more sensitive near the centre of the distribution).

2. Anderson-Darling test calculates critical values depending on specific distributions. This means that critical values must be calculated for each different distribution being tested.

3. The types of distributions that can be tested using Anderson-Darling test are limited to normal, lognormal, exponential, Weibull, extreme value type I, and logistic distributions.

2.4.6 Chi-square test

The Chi-square test procedure starts with grouping the observed data into different bins (classes) in a frequency table. Given a data set of size N and the data are grouped into k different bins, the Chi-square test statistic is calculated as follows:

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}, \tag{2.38}$$

where O_i is the observed frequency of data falling in class i; that is, the number of observed data grouped in class i. E_i is the expected frequency of group i calculated from the cumulative distribution being tested as follows:

$$E_i = N\left(F\left(X_u^i\right) - F\left(X_l^i\right)\right),$$

where X_u^i is the upper bound of class i and X_l^i is the lower bound of class i. F is the cumulative distribution function being tested and N is the size of the data set. The Chi-square test statistic follows approximately a Chi-square distribution with $k - p - 1$ degrees of freedom where k is the number of non-empty classes used to bin data and p is the number of estimated parameters for the distribution being tested.

An advantageous feature of the Chi-square test over Kolmogorov-Smirnov test and Anderson-Darling test is that it can be applied to both discrete and continuous distributions, while the other two tests can only be used in the cases of continuous distributions. One of its major disadvantages is that the testing results are sensitive to the way in which the sample data are

binned and there is no optimal way of grouping the data. Another disadvantage is that the Chi-square test requires a sufficient sample size in order for the Chi-square approximation to be valid [109].

2.4.7 Model Selection Using AIC

Information theorists believe that, instead of a "single" working hypothesis (the best model) inferred from the dichotomic results (significant or non-significant) of traditional null-hypothesis tests, there are several "multiple working hypotheses" or "models", that are well-supported by the empirical data observed from the reality [39, 38]. Thus, a set of one or more candidate models, with good empirical support, should be taken for considerations for making statistical inferences from data. These candidate models can also be ranked by how "close" they are from the true reality.

S. Kullback and R. A. Leibler's seminal work [90] – based on the theoretical foundation of Boltzmann's entropy and the second law of thermodynamics – proposed a quantity to measure information lost when approximating model is used to approximate the full reality; such quantity is called Kullback-Leibler information or K-L information. Assume a model g, over a parameter space θ, is used to approximate f, which represents the full reality, the K-L information $\mathbf{I}(f, g)$ is defined as:

$$\begin{aligned}\mathbf{I}(f,g) &= \int f(x)\, log\left(\frac{f(x)}{g(x|\theta)}\right) \mathrm{d}x \\ &= \int f(x)\, log\left(f(x)\right)\, dx - \int f(x)\, log\left(g(x|\theta)\right)\, dx \\ &= \mathbf{E}_f\left[log\left(f(x)\right)\right] - \mathbf{E}_f\left[log\left(g(x|\theta)\right)\right]\end{aligned}$$

As the quantity $\mathbf{E}_f\left[log\left(f(x)\right)\right]$ is a constant; hence, to compare models in the candidate model set, only $\mathbf{E}_f\left[log\left(g(x|\theta)\right)\right]$ needs to be estimated. The model that is closest to the reality is the

one with the least information loss (thus, the smallest $\mathbf{I}(f, g)$), compared with the others in the candidate model set.

Akaike [7] later introduced a model selection criterion based on K-L information, which is essentially an estimate of $\mathbf{E}_f[log(f(x))]$ calculated based on the the maximum likelihood estimator of θ. The criterion is called Akaike's information criterion (AIC) and defined as:

$$\mathbf{AIC} = -2log\left(L\left(\widehat{\theta} \mid data\right)\right) + 2\mathbf{K}, \tag{2.39}$$

where L is the likelihood function; $\widehat{\theta}$ is the maximum likelihood estimator; and $\mathbf{K}$ is the number of estimated parameters in the model. The derivations and related theoretical details can be found in literature including

Another version of AIC, called *adjusted* AIC and denoted as $\mathrm{AIC_c}$, is proposed by [127, 79] for cases where sample sizes are small. Given the sample size n, $\mathrm{AIC_c}$ is defined as:

$$\mathbf{AIC} = -2log\left(L\left(\widehat{\theta} \mid data\right)\right) + 2\mathbf{K} + \frac{2\mathbf{K}(K+1)}{n - \mathbf{K} - 1}. \tag{2.40}$$

It is recommended that $\mathrm{AIC_c}$, instead of AIC, should be used unless $\mathbf{K}/n$ is larger than about 40; when n is large, $\mathrm{AIC_c}$ will approximate AIC. Thus, in practice $\mathrm{AIC_c}$ is generally suitable except for the cases where the underlying probability distribution is strongly skewed.

2.5 Phase-type Distributions and their applications in traffic modelling

The central idea of traffic modelling is to construct analytical models that capture the most important statistical properties of traffic measurement data [130]. Traffic models not only provide an approximation of system demands and capabilities based on previous history; it also enables prediction of system performance under different traffic scenarios for the purpose of system

design and management. Hence, accurate traffic modelling is vital to accurate performance evaluation of the underlying systems.

One of the most widely used and oldest traffic model is the memoryless Poisson model; which assumes the underlying distribution is exponential. However, such assumption may not be applicable in the vast majority of real life systems. Over the last two decades, evidence has been accumulated that traffic flows in many computer and wired/wireless network systems (e.g Ethernet [93], World Wide Web traffic [51, 17]) exhibit properties including self-similarity, fractality, and long-range dependency. Previous research has shown that self-similarity of traffic in these systems can be attributed by the "heavy tails" observed in the distributions of data objects transferred or stored in the system (such as file sizes stored on the Web servers and data files transferred through the Internet [51]) or time delays involved in the system (such as user "think" time [51] and users' cell residence time [88] in wireless mobile networks). A distribution of a random variable X is said to be heavy-tailed if

$$\overline{F(x)} \sim cx^{-\alpha} \text{ and } 0 < \alpha < 2, \tag{2.41}$$

where $\overline{F(x)} = 1 - F(x)$ and $F(x)$ is the cumulative distribution of X. In addition, heavy-tailed distributions have many applications in analyses of economic, financial and physical real-life systems; for example, it has been found that distributions of healthcare utilisation such as length of staying and inpatient costs generally are right-skewed with fat tails [64].

Previous studies have shown that distributions with such properties behave very differently from the exponential distribution in that the probability of very large observations is non-negligible [51, 120, 111]. Approximating such distributions by using the conventional Poisson model would underestimate some important performance measures such as response time and queue length [88, 120]. Thus, choosing appropriate statistical models to approximate observation data with heavy-tailness has gained great attention in the research area of analytical

performance modelling.

Markovian Arrival Process (MAP) or Batch Markovian Arrival Process (BMAP) are stochastic models often used to approximate distributions with high-variability, long-range dependence or both [77]. Phase-type distributions, specified by a number of exponential phases and the interactions among them, is one of the most widely-used MAP models in approximating distributions exhibiting high variabilities. Phase-type distributions are "dense"; which means that they can approximate arbitrarily closely to any positive real-valued distribution. Apart from denseness, phase-type distribution are mathematically tractable. Exact solutions are often possible to obtain with phase-type distributions, either algorithmically or numerically, especially when they are applied to problems where there are explicit solutions with exponential distributions. One can thus turn to the well-established performance analysis in the domain of CTMC using phase-type distributions [19, 120].

2.5.1 Overview of phase-type distributions

Consider a Markov process MP on a finite state space, defined by $\{0, 1, 2, 3, ..., p\}$; state 0 is absorbing while the others are transient. The infinitesimal generator $\mathbf{Q}$ can be blocked partitioned as follows:

$$\mathbf{Q} = \begin{pmatrix} 0 & 0, \cdots, 0 \\ \mathbf{t} & \mathbf{T} \end{pmatrix},$$

where $\mathbf{t}$ is called the "exit vector"; its i-th component $\mathbf{t}_i$ is the conditional intensity of the absorption state 0 if the underlying MP is in state i. Matrix $\mathbf{T}$ is a $p \times p$ non-singular (thus, invertible) matrix; p is the number of transient states. Given an initial probability π of the MP with infinitesimal generator $\mathbf{Q}$, the time to absorption of the MP is said to be phase-type distributed. The basic statistical characteristics of a phase-type distribution with parameters $(\pi, \mathbf{T})$ are:

- cumulative distribution function: $F(x) = 1 - \pi e^{\mathbf{T}x}\mathbf{e}$
- density function: $f(x) = \pi e^{\mathbf{T}x}(-\mathbf{T} \cdot \mathbf{e})$
- the n-th moment: $m_n = (-1)^n \cdot n!\pi\mathbf{T}^{-n}\mathbf{e}$,

where $\mathbf{e}$ is a column vector of ones whose size is the total number of phases.

Based on the interaction among states in the Markov Chain whose absorption time defines the phase-type distribution, several common classes of phase-type distributions are defined.

Acyclic Phase-type If the Markov chain whose absorption time defines a phase-type distribution is acyclic (that is, no state visited more than once in the Markov chain), such phase-type distribution is called an "acyclic" phase-type distribution.

Coxian A phase-type distribution is Coxian if it is acyclic; it has one unique initial non-absorbing state; and for each state the next non-absorbing state is unique.

Hyperexponential A phase-type distribution is hyperexponential if it is acyclic; and for any state the next state is the absorbing state. That is, a mixture of exponential distributions.

Erlang A phase-type distribution is Erlang if it is acyclic; there is one unique, non-absorbing initial state; the sojourn time at each state is identically distributed and each state only has one unique next state. An Erlang distribution with n phases is essentially the sum of n i.i.d. exponential random variables.

Hyper-Erlang A hyper-Erlang distribution is a mixture of several Erlang distributions.

2.5.2 Fitting traffic measurements with phase-type distributions

Finding the parameters of the phase-type distribution that best fits the empirical measurement data is essentially an optimisation problem. Phase-type fitting techniques can be classified based on their optimisation criteria. Some of them are based on optimising only certain properties of the distribution; for example, matching various moments of measurement data (e.g., [37],

[34], [86]) is the most widely-used method in this category. The others minimise the cross entropy measures based on the maximum likelihood estimation (MLE) methods (e.g. [19], [120], [88], [129]). Moment-matching techniques are computationally efficient; however, the techniques are more suitable for phase-type distributions with only a limited number of exponential phases [120, 86]. Despite MLE methods can be applied to a much wider variety of phase-type distributions, MLE methods have short-comings that they fail to approximate accurately the tails of distributions as the fitting procedures search for global maximum [19, 77]. However, some MLE-based methods have been proposed to fit different partitions (e.g. body and tail) of a given continuous distribution into several phase-type distributions in order to capture the tail behaviour [120].

Among fitting methods proposed in either of these two categories, many are applied to only a subset of phase-type distributions, instead of general phase-type distributions. Fitting general phase-type distributions with the number of phases above two or three is too complex and computationally costly [130]. Targeting at only a subset of distributions narrows down the search space and thus makes it possible to develop algorithms with much better computational efficiency. Acyclic phase-type models are proposed by [78, 34]; such class of distributions has a canonical representation and the number of parameters for fitting is reduced to $2N$, instead of $N^2 + N$ in the general case. Other works such as use mixtures of phase-type distributions, such as hyper-exponential or hyper-Erlang distributions, for approximating general distributions [86, 60, 88, 130].

Of particular relevance to our research is the use of EM algorithm, one of the most widely-used MLE-based algorithms, to approximate long-tailed measured data sets or continuous distributions by mixtures of phase-type distributions. Based on the algorithm proposed by [60], [88] developed an EM-based algorithm to fit strictly monotone non-negative distributions into hyper-exponential distributions. The developed algorithm is efficient and gives good fitting results; however, hyperexponential distributions can only be used to approximate distributions whose squared coefficient of variations are at least one. Thümmler et *al.* later extended the

fitting procedure of [88] and developed G-FIT [130, 129], a distribution fitting tool that uses mixtures of Erlang distributions (that is, hyper-Erlang distributions) to approximate general non-zero distributions. Thümmler et *al.* showed that mixtures of Erlang distributions in theory are as powerful as acyclic or general phase-type distributions once the numbers of phases reach infinity. Thus, hyper-Erlang distributions can approximate a much broader set of non-negative distributions compared to hyper-exponential distributions; while their reduced form allows for an efficient fitting algorithm. There are several case studies in [130, 129] based on both synthetic data and real traffic traces to demonstrate the effectiveness of G-FIT in approximating empirical data, compared to the other methods such as PhFit [78].

2.5.2.1 Hyper-Erlang Distributions and G-FIT

This research adopts mixtures of Erlang distributions for approximating the general distributions of time delays involved in a system; G-FIT is the fitting tool employed to find the hyper-Erlang distribution that best fits a given data trace. The properties of mixtures of Erlang distributions are presented as follows.

The probability density function of an hyper-Erlang distribution with M Erlang branches, denoted by $f\,(x;M,R,A,\Lambda)$, is specified by three M-directional vectors; $R = \{r_i, i = 1, 2, ..., M\}$ contains the number of phases for each Erlang branch; $A = \{\alpha_i, i = 1, 2, ..., M\}$ defines the weight (or the initial probability) for each Erlang branch; and $\Lambda = \{\lambda_i, i = 1, 2, ..., M\}$ is a vector with the rate of each Erlang branch in the mixture. The vector R is subject to the constraint $\sum_{m=1}^{M} \alpha_m = 1$.

The probability density function is defined as:

$$f_X\,(x) = \sum_{m=1}^{M} \alpha_m \frac{(\lambda_m x)^{r_m - 1}}{(r_m - 1)!} \lambda_m e^{-\lambda_m x}\ , \tag{2.42}$$

and the cumulative distribution function is defined as:

$$F_X(x) = 1 - \sum_{m=1}^{M} \alpha_m \sum_{i=0}^{r_m-1} \frac{(\lambda_m x)^i}{i!} e^{-\lambda_m x} \ . \tag{2.43}$$

The i-th moment $E[X^i]$ is given by

$$E\left[X^i\right] = \sum_{m=1}^{M} \alpha_m \frac{(r_m x + i - 1)!}{(r_m - 1)!} \frac{1}{\lambda_m^i} \ . \tag{2.44}$$

The total number of states of a hyper-Erlang distribution is therefore:

$$N = \sum_{m=1}^{M} r_m \tag{2.45}$$

Given a fixed state number N as input, G-FIT searches through all the possible combinations of the number of branches m and the number of phases in each Erlang branch to find the hyper-Erlang that generates the highest log-likelihood value and thus is considered as best-fit to the empirical data trace. The total number of all possible settings of N-state hyper-Erlang distributions is given by $\varphi_N(N, 0)$ in [130], where:

$$\varphi_m(n, j) = \sum_{i=j}^{\lfloor n/m \rfloor} \varphi_{m-1}(n - i, i) \text{ and } \varphi_1(n, j) = \begin{cases} 0 & \text{, if } j > n, \\ 1 & \text{, if } j \leq n. \end{cases}$$

The final output from G-FIT includes the total branch number M and three M-directional vectors R, A and Λ for specifying the best-fit hyper-Erlang distribution.

2.6 Wireless Real-time Location Systems

A positioning system determines the location of an object or individual in a particular space, such as an enterprise facility, business premise or public building. Ideally the positioning is conducted in real time, with the ability to track the location of the object as it move around a space. The access to real-time location information has enabled numerous location-aware commercial applications, as well as innovative research areas such as ubiquitous computing, "Internet of things" [18], "embedded intelligence" [71] and "people-centric sensing" [40]. This section offers a brief overview of wireless location techniques and currently popular location tracking systems. Section 2.6.1 introduces the basic measuring principles and corresponding localisation methods employed by most of the location systems nowadays for position determination; Section 2.6.2 gives an overview of some most popular wireless location solutions available in the industry.

2.6.1 Localisation Techniques

Localisation is the process of determining positions of tracked objects in the environment. A location system usually applies one or multiple localisation techniques to locate objects; the location information it offer can be absolute, relative or proximity. This section gives an overview of basic principles for 2D/3D localisation techniques based on different measurements (more details regarding localisation techniques for wireless positioning systems can be found in [99, 113, 96]); three major types of localisation methods are introduced as follows: triangulation, scene analysis (fingerprinting) and proximity.

2.6.1.1 Triangulation methods

Triangulation methods estimate a target's location taking advantage of triangles' geometric properties. There are two types of triangulation methods: lateration and angulation. Lateration derives an object's position according to its distances from multiple reference points in the space;

angulation locates an object by measuring angles relative to the reference points. Measurements commonly used for for lateration localisation include:

Time of arrival (TOA) measures the absolute travel time of a signal from a transmitter to a receiver. The distance of the target object is the signal travelling time multiplied by the wave speed. TOA relies on precise time synchronisation among transmitters and receivers in the system. Also, a receiver needs to know the absolute time when a signal pulse was sent; thus it is necessary to attach a timestamp to the transmitted signal in order for the signal travel distance to be estimated. TOA is particularly challenging to apply in indoor environments where multipath conditions are common.

Time difference of arrival (TDOA). Instead of the absolute value of arrival time (TOA), the time difference at which the signal arrives at different measuring units in the system (TDOA) is measured to determine the relative position of the mobile transmitter. Using time differences of TOA measurements offers the advantage that it is not relevant if clocks at receivers and transmitters are not completely synchronised; neither does the receiver need to know the absolute timestamp when the signal was transmitted.

Received Signal Strength (RSS). One of the major drawback of using TOA or TDOA for position estimation is the requirement for line-of-sight between the transmitter and the receiver; the measurement accuracy would be largely compromised in indoor environments where radio propagation suffers much from multipath effect. An alternative approach is to estimate the distance of the target object from multiple measuring units by calculating the signal path loss due to propagation. The attenuation of transmitted signal strength can be modelled using theoretical or empirical methods. However, the features of the signal strength attenuation are site-specific and therefore the path loss model has to be parameterised for different environments.

Round-trip time-of-flight (RTOF) measures the time-of-flight of the signal making a roundtrip between a receiver and a transmitter. This method avoids the need for time synchronisation between transmitters and receivers. The drawback of this method is the longer measurement times required, which causes large location update latencies.

Angulation localisation is based on the measurements of angle of arrival (AOA). AOA is defined as the angle between the propagation direction of a radio-frequency wave incident on some fixed reference direction, referred to as orientation. AOAs are measured against the orientation, represented in degrees in a clockwise direction from the North [113]. There are two types of angle-of-arrival measurement techniques; one is basically to make use of the anisotropy in the reception pattern of an antenna; the second one estimates the AOA measurements from the phase differences in the arrival of a wave front [99]. AOA-based position determination requires as few as two measuring units in 2D and three measuring units for 3D positioning. No time synchronisation between transmitters and receivers is required either. The accuracy of AOA measurements is limited by the directivity of the antenna, shadowing and multipath reflections; when the target object is far away, the accuracy of AOA-based methods would be also compromised [70]. In addition, line-of-sight from the transmitter to the receiver is required for AOA measurements [99].

2.6.1.2 Scene analysis methods, Fingerprinting

The quantity used by scene analysis methods is usually radio frequency RSSI; but it can also be performed with audio or images [103]. Scene analysis methods usually comprise two stages: offline stage and online stage. In the offline stage, a site survey is conducted to collect data regarding the location coordinates/labels and construct mapping between various locations and the features (that is, fingerprints) of received signals from the nearby base stations or measuring units. Maps of fingerprints can be set up by either empirical measurements or analytical modelling [103]. The online stage is to estimate a target's location by comparing the observed signal features with the closest location fingerprints predefined at the offline stage.
Fingerprinting performance can reach accuracy at metre level, depending on the numbers per squared meters of base stations and calibration points where the fingerprints are taken. The major drawbacks of fingerprinting methods is that radio fingerprinting is a costly process; also, changes of settings in the environment might necessitate recalculation of predefined signal fingerprints.

2.6.1.3 Proximity methods, Cell of origin

Proximity, also called cell of origin (COO), is a simple positioning technique that determines an object's position by assuming it to collocate with the anchor point where the strongest signal is received. Such method only provides symbolic relative location information and its accuracy depends on the density of anchor points deployed in the space and signal range. Proximity method is suitable for applications with low accuracy requirements, such as physical contact detection, automatic ID recognition (e.g. RFID) and mobile network positioning systems [96].

2.6.2 Wireless Positioning Technologies and Systems

It is beyond the scope of this thesis to provide a complete overview of location positioning systems available till now. Here we only give briefly introduction to some most common types of location positioning systems, classified based on the underlying wireless technologies used for location tracking: GPS-based systems; infra-red system; ultrasonic system; radio frequency (RF) systems, including radio frequency identification (RFID), wireless local area networks (WLANs) and ultra-wide band (UWB).

2.6.2.1 GPS-based

The global positioning system (GPS) is the most popular and widespread positioning technology in the world. The GPS system contains 24 to 32 satellites, also called space vehicles (SVs), travelling along medium Earth orbit. GPS satellites continuously broadcast signals with information including their current positions and the timestamp when the message is transmitted. A GPS receiver employs a triangulation process to calculate its position using signals sent from multiple GPS satellites. Data from at least four satellites are required to compute three dimensional position (x, y, z) along with the timestamp. Resolution of position data can reach 1 to 5 meters with the GPS system.

This technology is currently implemented in many mobile devices, thanks to the availability of chip-size GPS receiver. It is foreseeable that in the near future the majority of the mobile wireless devices will be equipped with real-time location information. However, GPS cannot function well in the indoor environments. Attenuation and multipath reflections of the line-of-sight (LOS) signal (or direct path) by the walls, floors, and surface of buildings are the main factors preventing typical GPS receivers from functioning in the indoor environments or urban spaces.

To address the failure of GPS for reliable indoor positioning, Locata Corporation, an Australia-based company, developed a new positioning technology, named Locata [26, 23], for high-precision positioning in both indoor and outdoor environments. Locata technology is based on a type of ground-based time-synchronised pseudolite transceiver called LocataLite. Multiple LocataLites can form a network called LocataNet. Such pseudolite transceivers emit GPS-like signals that can be received by suitable mobile devices (a Locata) [24]; and allow single-point positioning using carrier-phase measurements. It has been shown that Locata technology can reach precision at centimeter level [25].

2.6.2.2 Infra-red

Infrared (IR) wavelengths are longer than that of visible light, but shorter than that of terahertz radiation. There are three methods of using infrared signals for positioning [103]:

- use of active beacons
- infrared imaging using natural (i.e. thermal) radiation
- artificial light sources

The active beacon approach involves placement of fixed infrared receivers at known locations throughout an indoor space, in order to detect the infrared beacons emitted by mobile devices whose positions are unknown. One of the early and widely recognized IR indoor positioning

systems is the Active Badge System [135] developed by AT&T Cambridge for positioning people at room level. A small infrared beacon is carried by personnel being tracked and emits short IR pulses with unique identifier codes every 10-15 seconds. One major disadvantage of Active Badge is that it is not suitable for real-time applications as the location update rate is more than 10 seconds [54].

Positioning using natural infrared radiation are known as passive infrared localisation. Sensors, operating in the long wavelength infrared spectrum (8 μm to 15 μm, also known as the thermography region), are able to detect objects in the surrounding from their natural thermal emissions; thus, there is no need for active IR beacons. One example of passive IR location system is IR.Loc, developed by Ambiplex [9]. Sensors in IR.Loc determine the angle of incidence to a heat source; given multiple sensors installed at known locations, the location of a heat source can be determined at a measurement rate of 50 Hz, with reported location accuracy up to 20-30 centimeter at an operating range of 10 metre [103].

Optical IR indoor positioning systems are based on active IR light sources and IR sensitive CCD (charge coupled device) cameras. Such approach has seen most applications in motion detection [103]. The motion sensing device known as Kinect, developed by Microsoft for the video game console Xbox, uses continuously projected infrared light to obtain 3D scene information with a webcam style of infrared camera. People can be tracked simultaneously up to a distance of 3.5 metre at a frame rate of 30 Hz with reported accuracy of 1 cm at 2 m distance.

The IR-based systems can reach high accuracy of positioning estimation. One of the major advantages of using IR for positioning is that many existing devices (e.g. mobile phones, TV sets, PDA, etc) are equipped with IR sources [70]. The system architecture is also relatively simple, which does not require time-consuming and expensive installation and maintenance. However, IR-based positioning systems do not work well in complex indoor environments due to its requirement of line-of-sight and its inability to penetrate opaque obstacles.

2.6.2.3 Ultra-sound

Ultra-sound is another media that can be used for position measurement. The relative distance or range between two devices can be estimated from Time of Arrival (TOA) measurements of Ultra Sound (US) pulses which travel from an US emitter to an US receiver. The Active Bats system developed by AT&T Cambridge [20] and The Cricket system from MIT [116, 117] are two most well-known ultrasound-based positioning systems. In both systems the relative distance or range between two devices can be estimated from TOA measurements of US pulses, along with triangulation methods.

The Active Bats system uses badges or tags that periodically emit ultrasonic pulses, which send information to a matrix of receivers embedded on the ceiling at known locations. With sufficient number of receivers installed, it is reported that the system can reach accuracy up to 3 centimeter for 95 percent of the readings in a 3D location estimation scenario using 100 measurements within a $10m^3$ volume [54]. The major drawback of the Active Bats system is the cost and complexity involved in installing receivers on the ceiling, which largely reduces the scalability of the system.

The MIT Cricket system uses a combination of RF and ultrasound technologies to provide location information to the tagged host devices [116]. Location beacons are mounted on the ceiling or inside the wall; they not only publish information on an RF channel but also transmit concurrent ultrasonic pulses. Once listeners, which are attached to the tracked objects, receive RF signals, they listen for the corresponding ultrasonic pulses. When the ultrasonic pulse arrives, the listener obtains a distance estimate relative to the sender beacon by taking advantage of the difference in propagation speeds between RF and ultrasound [50]. As the position is calculated locally at the listener mounted on the tracked object, Cricket also provides the advantage of protecting user privacy. The Cricket system can provide a position estimation accuracy of 10 centimetre and an orientation accuracy of 3 degree.

2.6.2.4 Radio Frequency Identification (RFID)

Radio-frequency identification (RFID) is a non-contact identification technology using electromagnetic transmission to an RF compatible integrated circuit, to store and retrieve a tracked object's data such as its identification. An RFID system has several basic components, including RFID readers (with antennas attached to it), RFID tags (or RFID transponders), and the communication between them. When a RFID reader energises an antenna, the tags in its coverage area would get activated and transmit their identification codes, possibly along with other data, back to the reader. RFID can be taken as the next generation of barcodes; however, unlike barcodes, it does not require line-of-sight reading and the identification can be conducted meters away [134, 70].

RFID tags contain a unique identification number called an Electronic Product Code (EPC), and potentially additional information of interest. There are two types of RFID tags: passive and active. Passive tags operate without having their own power supply. When the RF signal from a reader is received at a tag, electric current is induced at the tag's antenna and powers its IC to transmit response with information back to the reader. Passive RFID tags are smaller, lighter and inexpensive. However the coverage range of passive RFID tags is limited, around 1-2 meters. Active RFID tags are small transceivers with internal batteries; they actively broadcast signals to readers in proximity and can transmit over much longer distance as far as 100 meters comparing to passive tags. Active tags are usually heavier and larger in volume; they are also more costly and thus are suitable for tracking high value goods like vehicles, aircraft parts and large containers of goods [96, 107].

Three primary frequency bands are being used for RFID [107, 134]:

- Low frequency (125/134KHz). Most commonly used for access control, animal tracking, and asset tracking.
- High frequency (13.56 MHz). Used in cases where medium data rate is acceptable and

read ranges under 1.5 metres.

- Ultra-high frequency (850–950 MHz). Offers the longest read ranges of up to about 3 meters and high reading speeds.

The most common positioning method used by RFID systems is the proximity method; that is, the system collocates the presence of a person wearing an RFID tag with the RFID reader that detects it. The accuracy of an RFID system is highly dependent on the density of reader deployment. RFID technology has the advantage of requiring no direct line-of-sight. It can detect tags at high speeds and RFID tags can be read in any environment. In addition, RFID tags are cheap and thus offer a cost effective solution for positioning.

2.6.3 Ultra-wide band (UWB)

One of the common problems encountered in using RF-based positioning systems is that accuracy is often compromised due to the the multipath distortion of radio signals reflected by walls in indoor environments. The ultra-wideband (UWB) technology operates with signal bandwidth either exceeding 20 percent of the centre frequency or at least 500 MHz. Due to the large bandwidth, UWB signal is characterised by ultra-short duration pulses (less than 1 ns), which are easy to distinguish from signals generated from multipath. The low frequency parts of the UWB signal spectrum are able to penetrate building materials such as concrete, glass and wood [102]. Greater precision of location readings up to 15 cm in 3D space can thus be achieved, allowing for detailed detection of entities' interactions in space and offering potential to model processes such as contact-based spread of infectious agents. UWB requires no direct line-of-sight; UWB sensors (tags) are also cheap and consume less power than conventional RF-based sensors, making UWB-based positioning a cost-effective solution.

The Ubisense Company [131], which is founded by engineers from AT&T Cambridge, provides a real-time tracking system based on UWB technology. Ubisense System comprises a network of sensors which are deployed in the existing network infrastructure and grouped into several

cells; each cell requires at least four sensors or readers and can cover an area of up to 400m^2 [70]. Each sensor is assigned an IP address by a DHCP Server. One of the sensors in a single cell takes the role as the master server, offering services to the other sensors in the same cell. Each sensor determines the angle of arrival (AOA) and the time of arrival (TOA) of the incoming RF pulses sent from a specific tag. Non-master sensors would take turn to send out messages containing the AOA and TOA measurements of the tag to the master server. Calculation of the location of a tag is performed using time difference of arrival (TDOA) and angle of arrival (AOA) of the radio frequency pulses at different sensors [131]. The time delay of the position estimations is short; the location update rate can be up to 20 times per second. The accuracy offered by Ubisense can reach tens of centimeters. Ubisense system is also highly scalable; throughout buildings or collections of buildings, an unlimited number of cells can be networked together in a manner similar to cellular phone networks.

2.6.4 Wireless Local Area Network (WLAN)

The mid-range wireless local area network (WLAN) standard (IEEE 802.11 standard), also called Wi-Fi, operates in the 2.4-GHz Industrial, Scientific and Medical (ISM) band. WiFi has been proliferating as the primary standard for wireless LANs in enterprise facilities and households worldwide [96]; most mobile devices, such as PDAs, laptops and mobile phones, nowadays are Wi-Fi-enabled.

One of the issues shared by many positioning technologies is the proprietary nature of the readers and tags; as well as the need for an infrastructure separate from the existing data network in the facility. This makes deployment of positioning systems costly and difficult to scale. It is thus attractive to use an existing WLAN infrastructure for indoor positioning by simply adding a location server.

The received signal strength indication (RSSI) values of the transmitted RF signals recorded at different WLAN access points (APs) are used to estimation the targets. RSSI values depend

to a large extent on the propagation environment and it is difficult to correctly model the relationship between RSSI values and any given position in the space. As a result fingerprinting methods, based around simple comparison of empirical measurements without developing a theoretical model have become more favorable than analytical modelling [103].

WLAN-based positioning typically covers ranges from 50 metres to 100 metres; line-of-sight is not required. The accuracy of location estimations based on the WLAN signal strength is affected by various factors in indoor environments including movement and orientation of human body, the overlapping of APs, walls, doors, etc [96]. The accuracy of typical WLAN positioning systems using RSSI is approximately 3 to 30 metres, with an update rate in the range of few seconds.

Ekahau is a completely software-based WiFi-based positioning system that utilises existing Wi-Fi access points installed in a facility and radio cards already present in the user devices [57]. The Ekahau system consists two major parts: site survey and positioning engine. Site survey is a program that conducts site calibration before the real-time position estimations; the positioning engine conducts real-time location tracking. The accuracy of Ekahau positioning system can achieve 1 metre, if there are three or more overlapping APs.

2.6.5 Location-aware applications

Wireless location tracking technologies have been rapidly deployed across the business world to enhance productivity and reduce associated costs and inefficiencies. There has been particularly great interest in areas such as supply chain management, healthcare systems and transportation. Many leading-edge companies such as Wal-Mart, Proctor & Gamble, and Unilever have either started adopting or experimented with RFID-enabled processes at different locations (e.g. factories, transportation facilities, retailer warehouses and store shelves) along their manufacturing and supply chain systems. These projects have indicated the potential of RFID in identifying sources of inefficiencies, errors or disruptions within the supply chains and thus

offering opportunities for improvement [87, 92, 13]. RFID has also been applied in patient identification [44, 133] to enhance the efficiency and effectiveness of management in hospital Emergency Departments. Combined technologies, such as sensor networks and RFID, have also been deployed, for example in a hospital blood bag management system that continuously monitors the temperature of stored blood bags as well as their locations [89]. [105] also identifies various key wireless sensor technologies adopted in Intelligent Transportation Systems (ITS) for the purposes of reducing congesting, achieving cost savings to stakeholders, enhancing safety, monitoring environmental impacts etc. These technologies offer the advantage of collecting large amounts of high-quality location data in an inexpensive and non-intrusive way. This data provides an unprecedented level of transparency to complex systems or processes which cannot be achieved visually or through manually-collected data. It is thus a valuable input in building performance models that can better characterise the operation of real systems and provide an enhanced understanding of complicated or highly-volatile processes.

2.7 Other Related Work

2.7.1 Workflow Mining

A large portion of our work, especially the part of mining customer flow structures, is closely related to the research area of workflow mining and process discovery. A workflow model, which specifies the order in which different tasks are conducted in a given system, is the basis of a variety of business process management systems such as ERP (Enterprise resource planning), CRM (Customer relationship management), supply chain management, and B2B [132]. To eliminate potential model bias introduced by the lack of full knowledge of the underlying process, research works in workflow mining and process discovery utilise collected runtime event data (or transactional data), which records the activities of an on-going process in time order, to build up a formal model that describes the process's behaviour or to identify the problems in the process. For the purpose of analysing software processes, [137] proposed an event-based

model, which uses different types of events to mark important time points during different activities in a process, such as the BEGIN and END events of an activity. [47, 48] later proposed three methods, ranging from the purely algorithmic to the purely statistical, to discover and produce formal models corresponding to actual process executions. In contrast to the finite state machine approach of [47, 48], [4] presented an algorithm that uses existing execution logs to model the workflow structure of a given business process as a graph.

Gonzalez et *al.* further extended works in workflow induction from event logs to mining RFID data in order to extract commodity flow structures in manufacturing and supply chain systems. These studies have been focused on innovating data model structures and data compression techniques in order to efficiently extract from massive RFID datasets flow information including commodity flow structures, transition probabilities and duration distributions at various locations along a supply chain. [67] proposed a new data model for warehousing RFID data sets. Targeted at applications in supply chain management, their goal was to enable the efficient storage and to facilitate high-level analysis of such data sets. They achieve large data compression by taking into account the fact that in a typical supply chain several items may be aggregated into single tagged units and certain path segments of little interest can be ignored or merged. Built on this work, [65] further presented a method to build up a warehouse of item flows, which is referred to as flowcubes. The flowcube is different from traditional data cubes in that its measure is not a scalar aggregate but a flowgraph, which is a tree-shaped graph with its nodes representing locations and edges between nodes corresponding to transitions between locations. The flowgraph maintains not only the path structure of item flows but also path-related information, including duration (i.e. the time an item spend at a certain location) and transition probabilities. [66] later presented a method to construct compressed probabilistic workflow models which not only capture the general movement paths of items and time they spent at different locations but also the significant deviations. This workflow model takes into account the observations that the probability of an item spending certain amount of time at its current location and the probability of the item moving to another particular location can be both affected by the time it has spent on the previous locations in the path.

This is different from traditional Hidden Markov Models (HMM), where time spent on previous states is independent from the current state's duration and transition probabilities.

This research adopts similar data-processing methodologies with previous research works in mining workflow patterns from observation data, especially [137, 65, 67, 66]. Raw location tracking data, in the form of (x, y, z) Cartesian coordinates, are gradually reduced and refined into high-level descriptions similar to event logs, which record in time order entities' activities in the system and their spatial relationships with each other. This helps extract time durations of customer entities interacting (i.e. requesting services) with server entities. A tree-like structure, similar to the flowgraph proposed in [65], is employed during customer flow mining to keep track of paths taken by customer entities; tree nodes, representing service centres in the system, maintain customer entities' sojourn time traces and their next destinations after departure.

However, general location tracking data can be different from RFID data in that many location systems output entities' location updates as absolute locations; while RFID systems collocate a tracked entity with the reader, whose location is known in advance, that detects it. That is, one can directly identify from RFID data if two entities are in proximity to each other according to their location updates around the same time. Many location systems, however, output absolute location data, which is often in the form of coordinates relative to certain reference points. In this case, the starting and ending timings of interactions among entities (e.g. a doctor treating a patient) need to be inferred based on their proximity to each other.

2.7.2 Constructing Queueing Models for Performance Analysis of Real-life Systems

Queueing models have been frequently adopted to help organisations such as transportation, call centres and telecommunication, determine capacity levels required to respond to demands in a timely fashion. Many of these studies focus on constructing analytical models with closed-

form solutions for important performance measures, such as response time, queueing time and utilisation of resources [8, 32, 45, 139, 3, 59, 104, 49]. In order to obtain tractable solutions, the structures of the queueing network models in these studies are usually high-level abstraction of the underlying processes; which takes the form of a number of inter-connected nodes, representing "stages" a customer entity goes through during its stay in the system. For example, most of the previous research works modelled processes in healthcare systems by assuming that patients would progress through a series of conceptual phases – patients arrive, waiting, receive treatment before they are discharged (e.g. [49, 104]). Poisson models are commonly adopted in many of these studies for approximating the service and customer arrival processes. This assumes the time gaps between two contiguous events are exponentially distributed [8, 94] with time-stationary rates. Such analysis offers an efficient way to predict the performance measures of the system as a whole and gives key insights to the utilisation of various resources in the system; which is valuable for high-level strategic planning.

Nonetheless, queueing network models do not provide complete and in-depth understanding of complex processes often encountered in real-life systems. Approximating the system with only few stages makes the resulting models fail to reflect detailed interactions among different entities involved in the systems [46, 53]. What constitutes a "stage" in such high-level models might not always have a clear correspondence in real-life systems and can change with time [104]. The assumptions made in basic queueing theory – stationary Poisson arrival and exponential service time – are hardly realistic [119, 80, 8]. There have been evidences showing that in many real-life systems, service times might not follow exponential distributions; they might also be varied with time and thus non-stationary. For example, patients arrival rates in many healthcare systems have exhibited seasonality and might have sudden surges within a short time window [8, 94, 74, 21]. In addition, analytical performance models give steady-state results results; it is not clear and difficult to verify if a given real-life system would ever reach steady-state [8, 80]. Although some previous studies on modelling real-life systems resort to discrete-event simulation (DES) when addressing complex problems with many interacting elements and time-varying features [45, 46, 140, 36], building a realistic simulation model typically

requires considerable input data for calibration and parameterisation. It also takes a significant amount of time for conducting on-site observational studies for custom logic development and data collection [94].

Another challenge frequently encountered in modelling real-life systems, either analytical or simulation-based, is the difficulty to obtain sufficient, high-quality and detailed data for model conceptualisation and parameterisation. Long-hour, labour-intensive observational studies are often required in order to obtain high-level understanding of customer flows in the system [104, 46, 49]; review of historical data from existing data management system, either electronic or paper-based, involves tremendous manual efforts and often introduces human bias and errors. As consequence, many existing studies base their models on little data; for example, [3, 104, 140] only obtain two days, seven days and one month, respectively, of observations. It has been shown that lack of detailed data makes it difficult to develop models that provide good agreement with empirical performance measurements [45, 46]. The work of [22, 21] on modelling patient arrival patterns and response times in an Accident and Emergency department exemplified the difficulties in accurate parameterisation and validation using data collected through existing technologies. The developed model only reaches good agreement in mean response times; the distributions of response times were not well matched and there was no straightforward way to identify the causes of discrepancies.

Moreover, many existing studies are case-study based; the developed models are tailored to the specific customer flow structures, as well as service and customer arrival patterns in the case-study organisations [49, 36, 3]. The processes involved in model development, including field studies, data collection, model conceptualisation, parameterisation and validation, are largely manual and rely on modeller's domain knowledge of the underlying systems. Models constructed using such methodology are difficult to apply to other organisations without arduous human efforts for model restructuring and re-parameterisation.

There are other related research works dedicated to inferring important queue statistics from sample executions collected. [91] applied order statistics and developed an $O(N^5)$ (where N is the number of times when service completion immediately followed by service commencement during the busy period) algorithm to deduce queue statistics (e.g., transient expected queue length, mean waiting time in the queue and the probability distribution of the number of customers in queue seen by a random customer's arrival) solely from transactional data (i.e., times of service commencement and service completion during the busy period). Larson's algorithm assumes Poisson arrival process and that during the busy period service of the next customer begins immediately after service completion of the previous customer. [30] later developed an improved $O(N^3)$ algorithm to deduce transient queue lengths and the waiting times of customers during the busy period from the same type of transaction data as in [91]. The algorithm is further generalised to the cases of Poisson arrival processes with time-varying arrival rates as well as the cases of arrival processes with arbitrary distributions. An $O(N)$ on-line algorithm is also developed to update the estimate of queue length after each customer departure. These studies do not attempt to infer service processes – service time distributions and service disciplines. The applicability of these studies is limited to simple queueing systems with only one server queue and where the arrival process is assumed to be Poisson.

Chapter 3

Inferring Queueing Networks from High-precision Location Tracking Data

3.1 Introduction

This research presents a methodology that takes raw location tracking data – collected either from a real-life customer-processing system or through simulation – as initial input; and automatically infers a queueing network model that can accurately characterise the physical customer flow and the service and arrival processes of the underlying system. This research has restricted the types of systems that can be inferred to those with multiple customer classes, single-server service and service disciplines including FIFO, LIFO and priority-based. For the customer flow structure, only probabilistic, class-based and shortest-queue routing policies are considered in this research.

Our approach is based on the four-stage data processing pipeline, as demonstrated in Figure 3.1. The input and output data of each stage are stored as database tables, which are designed to facilitate data queries and data processing for the following stage. After each stage is completed, the inputs from the previous stage can be discarded. Stage 1 performs basic data preprocessing and smoothing. The second stage has two parts. The first part infers the locations of service

centres in the system and their approximate sizes. The second part estimates when a tagged customer entity enters or leaves a particular service centre based on their proximity to each other; it also infers the evolution of population inside each service centre. Stage 3 mines all the extracted customer paths that have been observed inside the system and infers the basic structure of the customer flow and the routing policies associated with each branching point. Samples of all the time delays incurred in the system, including service times at each service centre, time intervals between customer arrivals at the system and customer travelling times between each pair of connecting service centres, are also extracted in Stage 3. At the final stage we fit a hyper-Erlang distribution to each extracted time-delay sample using the G-FIT tool [130, 129]; we infer the service discipline employed at each service centre by comparing the orders of customer arrivals and those of customer departures. With information extracted above, a fully-parameterised the queueing network model that best describes the underlying system is constructed as the final output of the data processing pipeline.

3.2 Stage 1

In this research we assume a typical location update reading from an RTLS is of the form (`tagName, type, time, x, y`), as exemplified in Table 3.1. `tagName` is a unique identifier for each entity in the system and `type` indicates the category a tag belongs to. This category is application specific and it can be used as an indicator for further information regarding the particular tag. For example, if we have multiple customer classes, `type` can be used to identify the customer class of the particular tag. `time` is the timestamp of each location update and `x`, `y` are the location of the tag in a 2D Cartesian coordinate system.

Location tracking data in reality does not appear as a smooth trace describing the object's path. Many RTLS systems only offer the "best effort" to meet the location update frequency requirements, usually specified by the system manager as the the time gap between two contiguous location updates. As what we observed through experiments with real-time location tracking system (Ubisense RTLS), the actual location update frequency for a particular tag

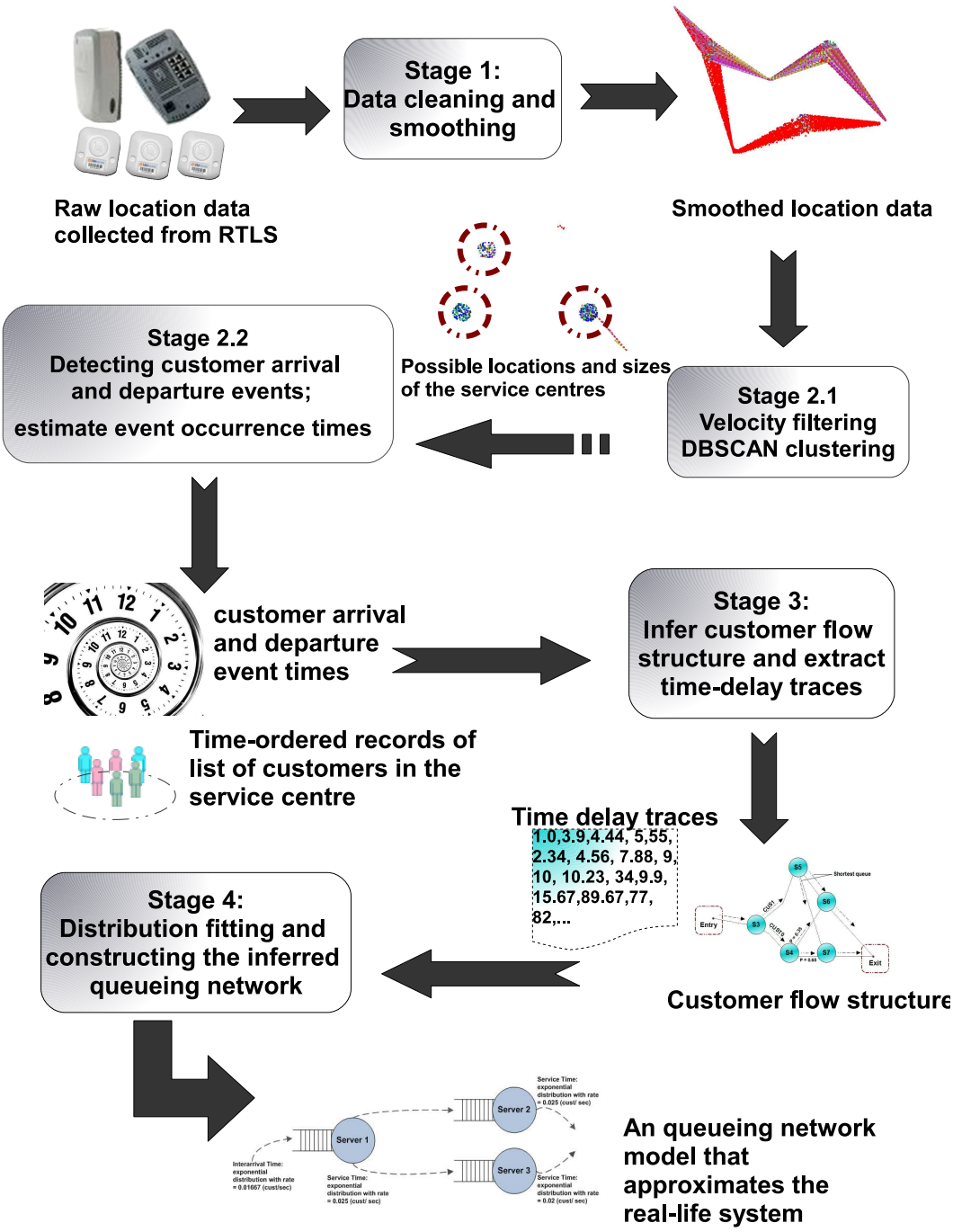


Figure 3.1: Overview of the four-stage data processing pipeline

Table 3.1: Example of raw location trace table after Stage 1

tagID	type	time	x	y
SERVER1	SERVER	150.311	11.496	1.409
CUST011	CUSTOMER01	150.405	3.277	3.513
CUST010	CUSTOMER02	150.654	11.437	1.602
CUST011	CUSTOMER01	150.935	3.277	3.515
CUST010	CUSTOMER02	150.982	11.449	1.595
SERVER1	SERVER	151.231	11.501	1.422
CUST011	CUSTOMER01	151.278	3.273	3.509

fluctuates from time to time, depending on whether the tag is static or moving, and more often, on whether there are many other tags competing for update time slots. Also, due to noise and other artefacts, location updates collected from RTLS systems contain many erratic reads with sudden deviation from the tag's actual movement path. At this stage, we first filter out location reads that are beyond the monitored area. We then conduct simple linear interpolation of location updates and calculate for each tagged entity its velocity curve as the average velocity for each time interval between consecutive readings. That is, if the distance between two location readings at times t_i and t_{i+1} is d_{i+1}, the corresponding point on the raw velocity curve is $((t_i + t_{i+1})/2, d_{i+1}/(t_{i+1} - t_i))$.

3.2.1 Output

The output of the first stage is smoothed time-ordered location traces with additional velocity information; the data is of the form (`tagName, type, time, x, y, vx, vy`), where `vx` and `vy` refers to the tagged entity's velocity along `x` and `y` axis, respectively. Table 3.2 gives an example of the output smoothed location traces from Stage 1.

Table 3.2: Example of location trace table after Stage 1

tagID	type	time	x	y	vx	vy
SERVER1	SERVER	150.311	11.496	1.409	-0.239	-0.104
CUST011	CUSTOMER01	150.405	3.277	3.513	0.004	0.000
CUST010	CUSTOMER02	150.654	11.437	1.602	0.031	-0.016
CUST011	CUSTOMER01	150.935	3.277	3.515	-0.007	-0.009
CUST010	CUSTOMER02	150.982	11.449	1.595	0.031	-0.017
SERVER1	SERVER	151.231	11.501	1.422	0.002	0.002
CUST011	CUSTOMER01	151.278	3.273	3.509	-0.007	-0.008

3.3 Stage 2

Given the smoothed location trace outputted from Stage 1, Stage 2 transforms the primitive low-level location data (in Cartesian coordinates) into high-level descriptions of the tagged entities' movements in the system (e.g. a tagged customer entity arriving at or departing from a particular service centre). Stage 2 comprises two sub-stages. The first part applies DBSCAN clustering algorithm [58] on low-speed location data points in order to infer the approximate area of each service centre and other previously-unknown bottlenecks in the system. Based on the proximity between a customer entity and a particular service centre, the second part estimates when the customer entity entered the service centre requesting service (either being served or waiting for service) and when it left. The outputs from Stage 2 include two database tables: **customerevents** and **serverpopulation**; **customerevents** records the estimated timestamps when customer entities enter or leave a service area; **serverpopulation** keeps track of changes in each service centre's population (or queue) with time.

3.3.1 Inferring locations of service centres

In this research, we assume that server entities are mostly static in the system; therefore, when customer entities are interacting with the server entities, either receiving or waiting for service, they stop or move at very slow speeds relative to the static server entities. Figure 3.2 displays the raw location data readings collected by a RTLS over a period of time; the readings are presented as dots which are colour-coded based on the tracked entities' speeds. The colour ranges from blue, representing speeds close to or equal to zero; to red, representing speeds of or above a certain threshold. We can observe from Figure 3.2 that the regions or "clusters", where location readings gather closely together with very low speeds, suggest possible areas for service areas and other unknown bottlenecks. This means that by discovering these low-speed clusters of location readings, we are able to pinpoint the locations and approximate sizes of possible service areas and other previously-unknown bottlenecks.

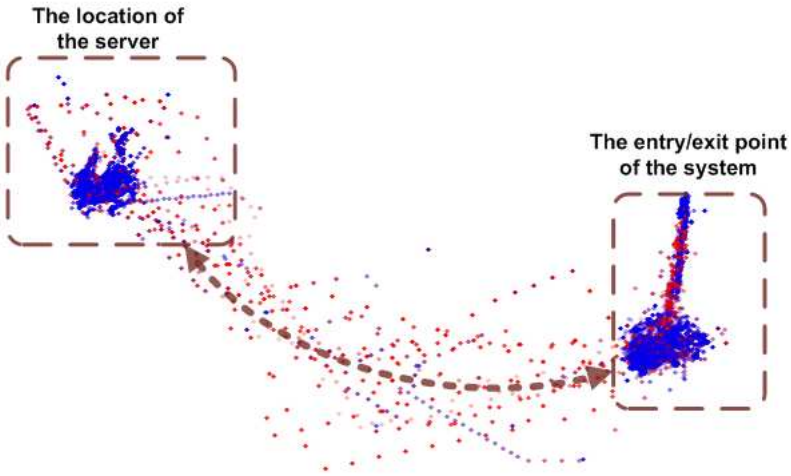


Figure 3.2: Visualisation of the raw location tracking data collected by a RTLS system to show location updates in red dots when the tagged entities are moving and in blue dots when they are static.

This is achieved through a two-step processing technique (see Figure 3.3). The first step is to filter out those customer entities' location readings with relatively high speeds compared to static server entities. The second step applies DBSCAN clustering algorithm to the remaining dataset to identify the areas with high data concentration as possible locations of service areas and other bottlenecks.

3.3.1.1 Velocity Filtering

At this step, we first use the output from Stage 1 (see Table 3.2) to calculate each tagged entity's speeds at different timestamps over its sojourn in the system. We assume that except for the time when a tagged customer entity is inside a service area, it is travelling with a constant speed from one service area to another. Thus, the time-ordered speed curve of a tagged customer entity has several high plateaus, which corresponds to its relatively high-speed movements among service areas; and the low-speed plateaus lying between capture its low-speed movements when the customer entity is requesting service inside one or more service areas in the system. Most of the data points with below-than-average speeds would be concentrated around zero with a

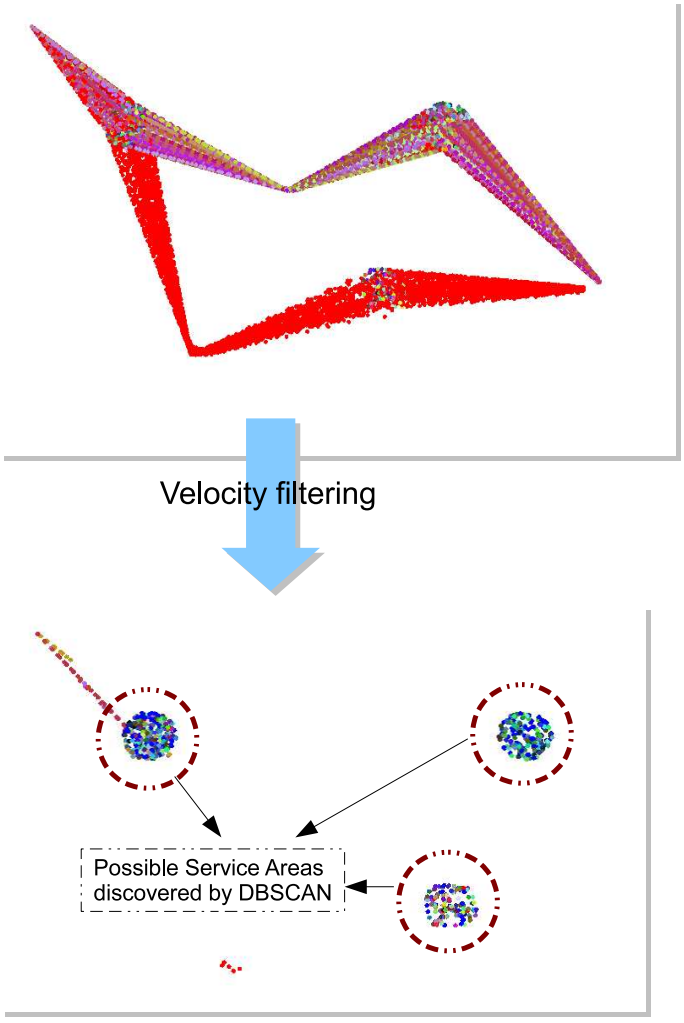


Figure 3.3: An overview of velocity filtering and DBSCAN clustering processes

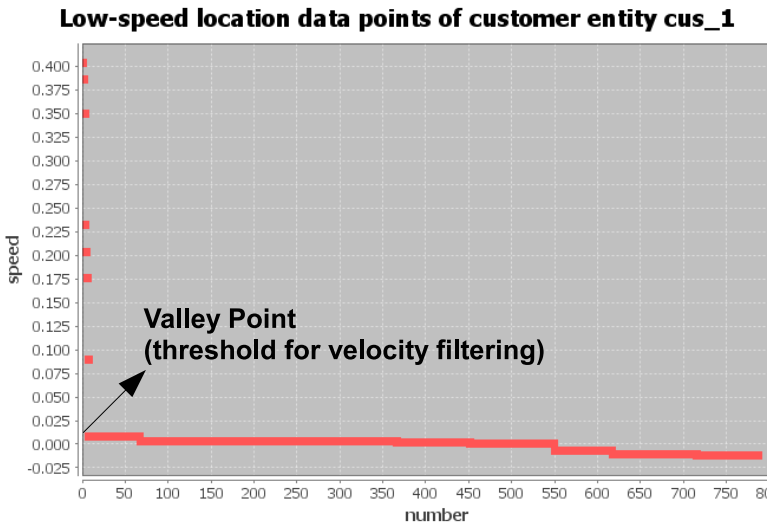


Figure 3.4: An example of a customer entity's low-speed data point distribution and the valley point that is taken as the threshold for velocity filtering

small number of outliners, as shown by Figure 3.4, where low-speed data points are plotted in descending order by value. From Figure 3.4 we observe that the data points grouped close to zero have much smaller variation, contributing to a flatter line; while the outliners data points are more deviated from each other and make a much steeper line. These outliners mostly contain noisy data or the data points collected when the customer entity is accelerating to leave a service centre or decelerating while entering a service centre. We use an indicator called *speed change rate*, defined as the difference of two values in an ordered list divided by the number of data points between them, to measure changes of slopes in different segments of the speed curve.

An iterative approach is developed to find out the first transition point, referred to as "valley point", where the speed curve first experiences abrupt change of slope (i.e. speed change rate). Such "valley point" is used as a threshold that differentiates the part with speed data points homogeneously close to zero and the outliners. Please note that this approach can be applied to datasets where the majority of data points are similar to each other (i.e. with small variations) and a few outliners with values largely deviated from the average.

The filtered low-speed data points are first sorted in descending order by their values. The average speed change rate, which is the average slope of the sorted speed curve as shown in Figure 3.4, for the entire sorted low-speed dataset is then calculated, denoted as $SpeedCh_{all}$. The iterative method searches for the range of data points where the "valley point" most likely takes place. The search starts with the largest data points within a certain window size; the data point of the smallest value in the subset is denoted as p_{min}. The speed change rate between p_{min} and any other data point within this window is calculated. If all the speed change rates calculated are above $SpeedCh_{all}$, it means this subset of data points only include the outliners and might not contain the valley point; then the search moves on to the next window of data points. The same process continues until we find the the "smallest" data value at which the speed change rate relative to the p_{min} in the subset changes from above the $SpeedCh_{all}$ to below the $SpeedCh_{all}$; such point pinpoints the potential value of the valley point. The window size of data to examine at each iteration is at the user's discretion. In this research we choose the window size to be ten percent of the dataset size. Such choice is largely heuristic. It is based on the observation that outliners are usually of the small percentage of the entire sorted low-speed dataset and using such window size the valley point can be found within a few iterations in most cases.

3.3.1.2 DBSCAN clustering location traces

After the locations readings with higher speeds have been filtered out, the density-based clustering algorithm DBSCAN is applied to group the remaining location readings. Those data points are likely to be captured when the tagged entities are in the the approximate locations and sizes of service centres, as well as other hidden bottlenecks in the system. In the following subsections we follow the notations introduced by [58]: *Eps-neighbourhood* of a point p in a dataset D, denoted by $N_{Eps}(p)$, is defined as

$$N_{Eps}(p) = \{q \in D \mid dist(p, q) \leq Eps\};$$

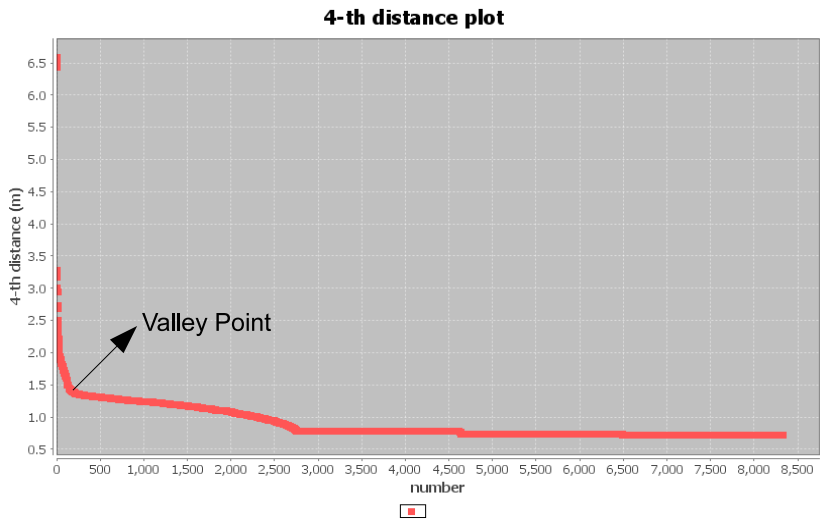


Figure 3.5: 4-th distance plot with its first valley point identified

that is, the set of points in D that lie in a circular area of radius Eps around the point p. DBSCAN algorithm requires two user-input parameters: $MinPts$ and Eps; these two inputs define the "sparest" cluster in the dataset. $MinPts$ is to decide the minimum number of data points inside $N_{Eps}(p)$ to make p fulfill the *core point* condition; a *core point* is a data point that resides inside of a cluster, as opposed to a *border point* which is at the border of the cluster (please refer to [58] and [123] for further explanations). The radius Eps of the $N_{Eps}(p)$ with only $MinPts$ number of data points inside is the distance between p and its MinPt $-$ th nearest neighbour.

In this research we follow the recommendation by [58] to use four as the value of $MinPts$; larger values do not produce any significant difference in results and are also computationally prohibitive [58]. For each of the remaining data points in the filtered dataset, we compute its distance from the fourth nearest neighbours (called $4\text{th} - \text{dist}$); then we sort the calculated $4\text{th} - \text{dist}$ values in descending order, as shown in Figure 3.5. According to [58] and [123], the first "valley point" of the $4\text{th} - \text{dist}$ graph, as shown in Figure 3.5, is chosen to be the Eps input parameter to the DBSCAN clustering algorithm.

Instead of manually selecting the first "valley point" from the $4\text{th} - \text{dist}$ graph as in [58], this

research applies the same technique used to find the filtering threshold in the low-speed curve to automatically find the valley point in the $4\text{th} - \text{dist}$ curve. Also note that if the data points are stored in a linear data structure, the brutal force of finding a single point's kth-nearest neighbour within a dataset of size n would take n comparisons and the total computation complexity would reach $O(n^2)$. In our implementation, we use KD-tree data structure [29] and to store the filtered location readings; which allows the time complexity of a kth-nearest neighbour search to go down to only $O(log(n))$ on average.

3.3.2 Generation of event table

At the end of the clustering step, each low-speed location data point is either assigned a cluster label or classified as noise. To identify when a customer entity enters or leaves the whole system being monitored, "artificial" clusters, representing the entry areas and exit areas in the system, are created and the data points falling in these areas are retrieved and assigned the corresponding cluster labels.

Here we assume if a customer entity and a server entity occur in the same clusters around the same time, the customer entity is interacting with the server entity; that is, either receiving service or waiting to be served. Thus, the overlapping time period when the location data points from a customer entity are clustered together with those from a particular service entity can be used to approximate the duration when a customer entity stays inside this server entity's service area. Our current implementation assumes that each cluster, except those added later to represent exit and entrance areas, either represents a service centre containing a server entity (called server cluster) or a developing system bottleneck. Please note here that we assume no multiple server entities existing in the same service centre.

The location data points from all the tracked entities are first traversed in time order. When a customer entity's data point is assigned a cluster label different from the previous one, it is considered that the customer entity might have left a service centre and entered another one.

However, a change in cluster labellng does not necessarily represent a real customer departure or arrival event. Noise and other artefacts might cause location updates to have sudden deviation from the true locations. More importantly, although velocity filtering in the previous stage has screened out most of the location data points captured when customer entities are travelling, it does not guarantee that the remaining data points are "only" those captured when customer entities are requesting service in service centres, despite their relatively high proximity to server entities locations. This is because the speed threshold used to filter out a customer entity's data points with higher-than-average speeds is decided solely based on its own speed trace, independent of the other entities. If a customer entity was passing through a service centre at a speed below its average, it is possible that its location updates around that time are clustered together with those from the server entity in the service centre; while in reality it did not request any service inside the service centre. The velocity filtering algorithm does not work well either when a customer entity's speeds throughout the data collection period does not exhibit much variation (that is, the speed curve is very flat). In this case, most of the customer entity's location data points would not be filtered out and very likely be grouped into a closest cluster.

As mentioned above, a customer entity normally has zero or very small relative speed to a certain server entity if it is actually requesting its service. This is considered as a "common feature" shared by the majority of the data points clustered together with those from server entities. As we have assumed that the server entities are mostly static, we can apply the velocity filtering technique described previously to find out the speed threshold for each cluster. We use it to rule out data points that are not "similar" to those captured during the time when customer entities are requesting service from the server entity. A customer entity is only considered to be "inside" a service centre demanding service when its data points have speeds below this "similarity" threshold.

To judge whether a real departure event occurred, a look-ahead action is taken; this is essentially to examine the cluster labels of the next few (in time order) data points from the same

customer entity within a certain time window. The size of the time-window is user-defined and should be based on the estimated minimum time for a customer entity to make a round trip from the current service centre to the entry/exit areas or any other service centre and then back to the current service centre again. A voting mechanism is then applied to count how many data points within this time-window have the same cluster labels as the previous one; that is, to support the hypothesis that the customer entity still remains in the previous cluster. Only if the majority of the data points within the time-window do not share the same cluster label as the previous one can the change in the cluster label represent a real departure event.

The next step after confirming a customer entity's arrival or departure event is to estimate the most probable time when the event occurred. The departure event occurrence time is approximated by averaging two values, both of which are obtained from querying the location trace table. The first value is called the last appearance time, which is the last timestamp when the customer entity is still considered requesting service in the service centre (i.e. the last timestamp of the customer entity' low-speed data point being clustered together with the service entities'). The second value is called first disappearance time; which is the first timestamp, after the last appearance time, when the customer entity is either moving at a speed larger than the velocity threshold of the cluster or it is physically outside the service centre's range. A similar approach is used to estimate arrival times using the last time the customer was observed outside a service centre and its first appearance time within the service centre.

3.3.3 Output

After this stage the processed data can be stored into two database tables: `customerevents` table and `serverpopulation` table

- **customerevents table**. As exemplified by Table 3.3, the `customerevents` table records each tagged customer entity's path of movement as well as the timing and location of its activities during its sojourn time within the system. Since only the customer arrival and

departure events are relevant in deriving performance metrics, we use these two types of events to represent a customer entity's activities inside a system. Thus, instead of the Cartesian coordinates in the raw location traces, in the `customerevents` table a customer entity's locations are represented by the tag IDs of the service centres it has visited. Beside the tag IDs of service centres, we use "ENTRY" and "EXIT" to identify the entrance and exit areas in the monitored system, respectively. A customer entity's path started when the customer entity left the entrance area and ended when it arrived at the exit area.

- **serverpopulation table**. The table `serverpopulation` contains data fields `(serverID, time, count, customers)` (see Table 3.4) and the data is ordered by time. The `count` field is the total number of the customer entities that are located inside the service centre, which belongs to the server identified as `serverID`. The field `time`, records the time-stamp at which the population inside a certain service centre is updated. The `customers` field maintains a list of the customer entities' tag IDs in the service centre. If (`S1`, $t1$, $c1$, $\langle list1 \rangle$) and (`S1`, $t2$, $c2$, $\langle list2 \rangle$) are two contiguous updates of population in the service centre of server `S1`, there are $c1$ customer entities, whose tag IDs are listed as in $\langle list1 \rangle$, inside `SERVER1`'s service centre from $t1$ to $t2$. Table 3.4 gives an example of the **serverpopulation** table.

Table 3.3: Example of **customerevents** table

customerID	type	location	time	event type
CUST010	CUSTOMER02	SERVER1	140.550	ENTRY
CUST011	CUSTOMER01	ENTRY	170.671	DEPARTURE
CUST010	CUSTOMER02	SERVER1	160.230	DEPARTURE
CUST011	CUSTOMER01	SERVER1	200.888	DEPARTURE

Table 3.4: Example of records stored in the **serverpopulation** table

serverID	time	count	customerlist
SERVER1	138.003	1	CUST004
SERVER1	140.550	2	CUST004, CUST010
SERVER1	153.950	1	CUST010
SERVER1	160.230	0	
SERVER1	200.888	1	CUST011

3.4 Stage 3

Stage 3 is implemented as two parallel processes, each taking data stored in one of the output tables from Stage 2 as inputs. The outputs from this stage include an initial structure of the queueing network representing the system, as well as time delay samples of three types – customer interarrival time, customer travelling time between a pair of connecting service centres and service time at each service centre.

The first part processes the table **customerevents** to discover customer behaviour, which is described by the customer flow structure, customer arrival process to the entire system and customer travelling time among service centres. The inferred customer flow structure outputted from this part is specified by the routing policy associated with each branching point; it is the "blueprint" for constructing the final output queueing network. Measurements of customer interarrival time at the system and customer travelling time among service centres are also extracted; they will be used in the later stage for parameterising the probabilistic models representing the customer arrival process and customer travelling time delays. The second part uses the information from **serverpopulation** to extract service time samples at each service centre.

3.4.1 Customer flow structure mining and extraction of travelling time and interarrival time samples

The data stored in the **customerevents** table can be seen as a time-ordered stream of event logs recording customer entities' activities, especially their interactions with service entities, in the system. Stage 3 adapts a simplified version of the method proposed by [4], which is basically a counting process for workflow mining from event logs, to discover the structure of customer flow in the system. One point to note is that tags can be "recycled" after the tracked customer entities who carry them leave the system, so a single tag ID may represent different

customer entities at different timings. We use the events where a customer entity left the entry area and entered the exit area to mark the beginning and the end of a customer entity's path in the system, respectively.

Before the customer flow mining process, records from each customer entity in the **customerevents** table are processed one by one. The path each customer has travelled in the system is constructed and represented as a chronologically-ordered string of server entities' tag IDs whom the customer entity has requested services from. For example, after the customer entity `CUSTOMER0` entered the system it first requested services from server entity `S1` and then from server entity `S2` before it left the system; the string that represents its travelling path is " `ENTRY S1 S2 EXIT`". Once all the customer paths are constructed, we conduct customer flow mining by using a tree-like data structure, called customer flow graph, to keep track of all the possible routes customer entities have taken in the system. The customer flow graph is essentially a directed graph with nodes representing service centres in the system connected together by directed edges showing how customer entities move around the system. Each node in the customer flow graph maintains the number of visiting customers from each customer class, as well as their arrival and departure times.

To extract customer arrival times at the whole system, we identify the timings when customer entities leave the system's entry area. As we assume that the arrival processes of different customer classes are independent from each other, for each customer class customer interarrival time is the time gap between two contiguous customer arrivals to the system from the same customer class. For each pair of connecting service centres, we also measure how long customer entities have spent in travelling between them. It is defined as the time elapsed after the customer entity has left one service centre before it entered the next one.

3.4.1.1 Routing policies mining

Three types of routing policies can be inferred by the data processing pipeline – probabilistic routing, class-based routing and shortest-queue routing. At the departure of a customer entity, if its next destination is decided based on its assigned customer class, it is called class-based routing; if the customer entity goes for the service centre with the shortest queue length, compared to the other possible destinations, it is called shortest-queue routing; if the routing follows certain probabilistic distribution, it is called probabilistic routing.

After the customer flow mining process, for each branching point in the customer flow graph a frequency count table is generated, as exemplified in Table 3.5. The frequency count table can be seen as two two-way contingency tables; which summarise respectively the numbers of customer entities who go for the next destination categorised by two variables – customer class or whether the destination service centre has the shortest queue. Table 3.5 gives an example of such frequency count table for a branching point from server entity `S1` to any of the three service entities `S2`, `S3`, `S4`. There are, for example, 30 customer entities that have taken the path from `S1` to `S2` with all of them classified as `CUSTOMER0`; 10 out of these 30 customer entities chose `S2` as their next destination because `S2` had the shortest waiting queue compared to `S3` and `S4`.

Table 3.5: Example of a frequency table for different possible routing types along a single link with multiple branches

Link	CUSTOMER0	CUSTOMER1	CUSTOMER2	Shortest routing count	Total count
S1->S2	30	0	0	10	30
S1->S3	20	0	10	5	30
S1->S4	1	30	3	4	34

We then conduct Chi-square contingency table analysis (see [6] and [5] for theories regarding one-way and two-way categorical contingency table analysis) to test if either of these two variables show strong association with the choice of next destinations. If the Chi-square test results show that one of the variables has strong statistical association (at significance level 0.05) with

which path the customer entities were taking, the routing policy corresponding to the variable would be taken as the most-likely routing policy. If none of the Chi-square test results reject the null hypothesis, we assume the routing of customers only follows certain probability distribution and is independent of the customers' classifications or the conditions of the possible destinations.

In the case where Chi-square test results indicate that both variables have significant association with the choice of the next destination, we calculate the percentages of customer entities that take the "right" paths if one of the candidate routing policies is the true one. For example, as shown in Table 3.5 there are 19 out of 94 customers who left `S1` would choose their next destination because it has the shortest queue; if the shortest-queue routing is the true routing policy, around 20% of the customers are following the "right" path. This percentage is called the "success rate" of a candidate routing policy; high success rate of a particular candidate policy means most of time the customers indeed are routed based on such policy. To calculate the success rate for class-based routing policies is less straightforward. Given a set of customer classes there will be multiple combinations of possible class-based routing. We only choose the one best supported by the observed frequency counts; in the case shown in Table 3.5, the most possible class-based routing would be `CUSTOMER0` going to `S2`, `CUSTOMER1` going to `S4` and `CUSTOMER2` going to `S3` (according to which destination that have been chosen the most frequently within a customer class). The routing policy with the higher success rate (that is, better supported by the observed data) is chosen to approximate the underlying routing policy.

3.4.2 Extracting service time traces and inferring service disciplines

3.4.2.1 Extracting service time trace at each service centre

For each service centre, we process observations of its population at different timestamps (as recorded in table **serverpopulation**) and retrieve those timestamps of customer departure or arrival events by detecting differences in the ⟨`customer list`⟩ fields between contiguous

records. For example, if at timestamp $t1$, the ⟨`customer list`⟩ field is ⟨`CUST000, CUST001, CUST002`⟩ and at the next timestamp $t2$, the ⟨`customer list`⟩ field is ⟨`CUST001, CUST002`⟩, then at $t2$ there is one departure event of `CUST000` leaving the service centre; vice versa for detecting occurrence of a customer arrival event.

One of the assumptions we have made about service process is that service preemption is not allowed at a service centre; but there is no other assumption regarding the service disciplines. Another assumption made is that if the service centre is empty at the arrival of a customer entity, the customer entity will receive service immediately and its service time would be its departure time minus the arrival time. Otherwise, it will be served at the departure of the previous customer entity and the service time is estimated as the difference between the previous departure time and the customer entity's own departure time. Table 3.6 shows a snapshot of service centre `SERVER1`'s records between timestamp 12.423 and 24.308. From the table, we observe that the population at the service centre `SERVER1` is first empty; then a customer entity `CUST000` arrived at timestamp 13.316 and left the service centre at timestamp 15.488. Thus, the service time `Customer0` has received from `SERVER1` is estimated as $15.488 - 13.316$, which is 2.172. From the following records shown in Table 3.6, we observe that the customer entity `CUST002` left the service centre at timestamp 20.150, which can be assumed to be the service start time for the customer entity `CUST004`; with the departure time of `CUST004` as 24.308, the service time for `CUST004` is then estimated as $24.308 - 20.150 = 4.158$.

Table 3.6: Example of records in **serverpopulation** table from a server entity with tag ID `SERVER1`

serverID	time	count	customerlist
SERVER1	12.423	0	
SERVER1	13.316	1	CUST000
SERVER1	15.488	0	
SERVER1	16.836	1	CUST002
SERVER1	18.989	2	CUST004, CUST002
SERVER1	20.150	1	CUST004
SERVER1	24.308	0	

3.4.2.2 Inferring service disciplines

Our data processing pipeline can infer if the server entity adopts one of the following service disciplines – FIFO, LIFO and priority-based. For inferring priority-based service discipline, it is assumed that the priority is decided based on the customer class and each customer class is assigned a different priority order; if two customer entities are of the same customer class (thus, of the same priority), they are served based on FIFO principle.

By observing the population evolution with time inside a service centre (as recorded in the **serverpopulation** table), we are able to extract the orders in which customer entities arrive at and depart from a service centre. Given the order of customer entities arriving at a service centre, its service discipline decides the order of customer entities' leaving the service centre. To decide which service discipline is the most likely one adopted in a service centre, we use a "scoring" scheme to rank the candidate service disciplines by how accurate they are in generating customer departure patterns compared with the real observed ones.

One thing to note is that the service discipline is only applied when there are more than one customer entities queueing and competing for service. In cases where the service centre is empty (that is, no customer queueing) or only one customer entity is waiting to be served, the first incoming customer entity or the only customer entity in queue will be the next one to get served, respectively. In addition, as we assume no service preemption, the timing when the next customer entity is selected to be served (i.e. when the service starts) is approximated by the timing when the previously-served customer entity left the service centre. That is, only when a customer departure event takes place and at the same time the population inside the service centre is larger than one does the server entity employs the associated service discipline to select the next customer to serve.

While mining the records in **serverpopulation** table, a list of customer entities' tag IDs, arranged in the order of their arrivals, is maintained for each service centre to keep track of its population evolutions as time proceeds; this is referred to as the *customer incoming order list* in the following context. At the occurrence of a customer departure event with population in the service centre larger than one, we apply different candidate service disciplines to"predict" the next customer entity in the queue to be served. As there is no service preemption, the next-to-be-served customer entity should also be the next one that leaves the service centre. We then decide which service policy selects the "right" customer entity to serve when the next customer departure event is observed at the service centre. The one that has made the right prediction would get one score and the one that scores the highest over the entire data collection period is the most likely service discipline applied by the service centre.

Table 3.7 shows an example of **serverpopulation** with the lists of customer entities inside the service centre `SERVER1` between timestamp 12.423 and 24.308. When the customer entity `CUST002` arrived at `SERVER1` at 13.316, the service centre is empty and thus `CUST002` was offered service immediately. While `CUST002` was receiving service, customer entity `CUST004` arrived at 15.488, followed by the arrivals of `CUST005` and `CUST006`. As `CUST002` left the service centre at 20.150, there were more than one customer entities remaining in the queue and the customer incoming order list is (`CUST004`, `CUST005`, `CUST006`). If the true service discipline is FIFO the next customer entity to be served would be the first one on the customer incoming order list, which is `CUST004`; and the last one on the list, which is `CUST006`, if the service discipline is LIFO. Assume there are two customer classes `CLASS0` and `CLASS1`; and `CUST004` belongs to `CLASS1` while the rest are of `CLASS0` class. If the service discipline is priority-based depending on the customer class, there are two possibilities: `CLASS0` has higher priority than `CLASS1` (denoted as (`CLASS0`, `CLASS1`)) or the other way around (denoted as (`CLASS1`, `CLASS0`)). If the priority-based discipline (`CLASS0`, `CLASS1`) is applied, `CUST005` will be the next one that gets served; and `CUST004` in the case of (`CLASS0`, `CLASS1`). At timestamp 24.308, another departure event took place and it was `CUST005` that left the service centre. In this case, the service discipline candidate (`CLASS0`, `CLASS1`) gets one score.

Table 3.7: Example of records in **serverpopulation** table from a server entity with tag ID SERVER1

serverID	time	count	customerlist
SERVER1	12.423	0	
SERVER1	13.316	1	CUST002
SERVER1	15.488	2	CUST002, CUST004
SERVER1	16.836	3	CUST002, CUST004, CUST005
SERVER1	18.989	4	CUST002, CUST004, CUST005, CUST006
SERVER1	20.150	3	CUST004, CUST005, CUST006
SERVER1	24.308	2	CUST004, CUST006

3.4.3 Output

At the end of the Stage 3, an initial structure of the customer flow is extracted, which is specified as a network of server nodes, representing service centres in the system, connected with each other by directed links. At each branching point there is an inferred routing policy associated with it. This serves as the blueprint for constructing the final output queueing network model in the following stage. In addition, this stage extracts samples of time delays involved in the system, including service times and customer travelling times among service centres, as well as customer interarrival time.

3.5 Stage 4

3.5.1 Performance traces distribution fitting

After time traces (service time, travelling time and customer interarrival time) are extracted in Stage 4, the final stage of the data processing pipeline uses G-FIT [130, 129] to identify the best-fit hyper-Erlang to each of the traces. In the following context, we use *HErD* to denote a hyper-Erlang distribution; the number of branches in *HErD* is denoted as m; for the i-th branch its weight is denoted as α_i, the number of phases as r_i and the rate as λ_i. A *HErD* is thus specified by three m-dimensional vectors: $R = \{r_i, i = 1, 2, ..., m\}$; $A = \{\alpha_i, i = 1, 2, ..., m\}$;

$\Lambda = \{\lambda_i, i = 1, 2, ..., m\}$. The total number of states in a *HErD*, denoted as N, is the sum of the numbers of phases in all branches.

For a given trace, we run G-FIT with a range of possible state numbers, normally from 5 to 15 (the hyper-Erlang with state number above 10 has displayed good fitting results according to [130, 129]). Given a fixed state number N, G-FIT searches through all the possible combinations of m and $\mathbf{r}$ to find the corresponding Λ that generates the highest log-likelihood and thus is considered as best-fit to the empirical data trace.

In order to make the search process more efficient, the squared coefficient of variation c_v^2 is used as guidance to narrow down the search space for the optimal setting of m and R based on the recommendations by [130, 129]. If the extracted samples have a c_v^2 less than one, we use *HErD* with no more than three Erlang branches. For traces with c_v^2 larger than one we limit our search to hyper-Erlang distributions with at least $N - 2$ branches. Note that when the branch number reaches N, the *HErD* becomes a hyper-exponential distribution, which display good fitting performance for heavy-tailed distributions with large squared coefficient of variations.

The other modification we made is the use of adjusted Akaike Information Criterion (AICc) criterion as the measure for selecting the best-fit *HErD* model to avoid data over-fitting (please refer to Chapter 2 for an introduction on model selection based on AIC). As the number of parameters being estimated in a *HErD* mainly depends on the number of branches m, by using AICc, instead of log-likelihood, as the selection criterion, the best-fit *HErD* would be the one with the smallest branch number among those with highest log-likelihood values.

To further cut down the computation time, before the distribution-fitting process we first identify customer classes that might have received the same or very similar service qualities (in terms of service time distributions) at each server centre. This is decided based on the testing results of Mann-Whitney U test, which tests if two sets of data points are likely sampled

from the same or similar distribution. Mann-Whitney U test is chosen over the traditional Student t-test for the same purpose because it can be applied to cases where sample sizes are small or samples do not follow a normal distribution; these are often the cases in most real-life applications. We then combine service time observations sampled for different customer classes if the Mann-Whitney U test fails to reject the null hypothesis that they come from the same distribution. By doing so, we are able to decrease the number of G-FIT runs and thus the total computation time of the distribution-fitting process. Another advantage of such approach is that in general larger sample sizes lead to better precision when estimating parameters using statistical methods.

3.5.2 Output

The final step of Stage 4 is to construct a multiclass queueing network best representing underlying system. The basic structure of the output queueing network model is similar to the customer flow structure inferred at Stage 3. Single-server queues are created corresponding to service centres in the system and they are connected to each other by links directing the customer flow based on the inferred routing policies. Service process at each service centre is parameterised according to the inferred service discipline and the hyper-Erlang distribution best-fit to the extracted service time samples. Similarly, the customer arrival process to the whole system is specified by the best-fit hyper-Erlang distribution characterising the time gaps between two contiguous customer arrivals at the system.

To model customer travelling times incurred between a pair of connecting service centres, for each link we insert an "internal" infinite-server node, which is a server node that has infinite resources and can offer service to arriving customers immediately (that is, no queue needed); its service time distribution is specified as the hyper-Erlang distribution best characterising the extracted travelling time samples along the corresponding link. However, the "internal" infinite-server node modelling the customer travelling time along a certain link is not an entirely independent server node. It acts more like a "delegate" node to the server node at the end

of the link. The introduction of such internal infinite server node is due to the fact that its "service", which models the travelling time, is only triggered if the corresponding link is chosen based on the associated routing policy. While probabilistic routing and class-based routing are respectively based solely on some probabilistic distribution and the customer entity's classification, some of the routing policies, such as shortest-queue routing, depend on the condition of the destination server node. Thus, the internal infinite server node along a link basically can be seen as the delegate of the server node at the end of the link; it receives the incoming customers and forwards them to the server node it represents. The other nodes can make inquiries at it about the condition (such as the number of customer entities waiting) at the end server node before they forward customer entities. Figure 3.6 illustrates a queueing network model derived from location tracking data collected in an example system.

The constructed queuing network model that represents the underlying system can be used for further performance analysis through either analytical or simulation approach. Although the applications of the final output queueing network model are beyond this research, this research has developed a simulation tool – LocTrackJINQS, which can take in a queueing network specification, create simulation, and generate performance-related statistics including first and second moments of response times at each service centre and the whole system, mean queue length and utilisation of each service centre. The following chapter will give a detailed introduction of LocTrackJINQS, which is later used in the case studies discussed in Chapter 5 for evaluating the data-processing pipeline.

3.6 Summary

This chapter introduced the four-stage data process pipeline developed in this research for inferring queueing network models based on high-precision location tracking data. The first stage of the data processing pipeline performs basic data cleaning and interpolation. The

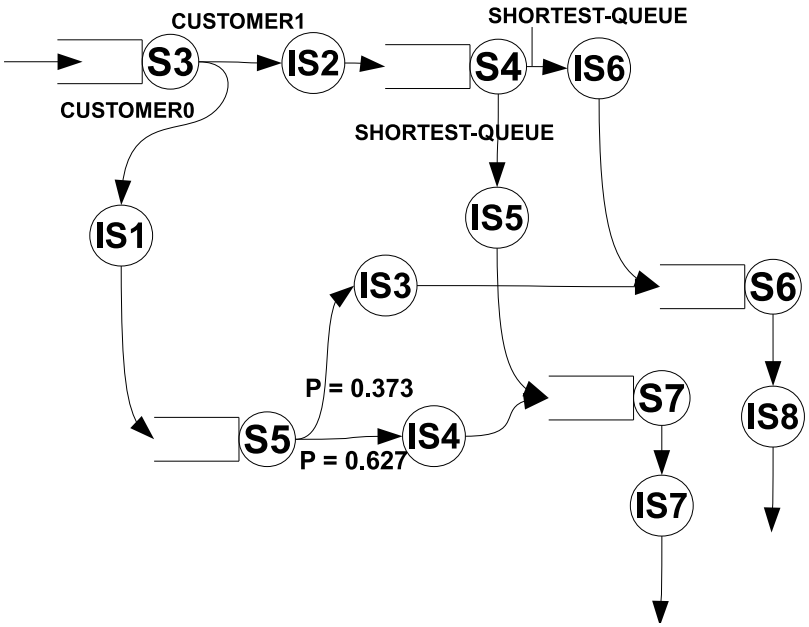


Figure 3.6: Example of a queueing network model as final output of the data processing pipeline

second stage transforms the low-level location data into high-level, time-ordered descriptions of customer entities' activities in the system. It first infers the approximate locations and ranges of each service centre as well as the size of its associated service area. This stage uses the proximity of tagged customer entities' geographical locations to the approximated service areas' locations to identify their high-level spatiotemporal relationships; for example if a customer entity is present in a particular service area or it is travelling between service centres. The first part of the third stage extracts samples of service times in each service area, samples of customer travelling times between each pair of service areas and samples of customer interarrival time. By mining the tagged customer entities' paths in the system, the second part of this stage creates the initial structure of the queueing network model. The final stage of the pipeline fits a hyper-Erlang distribution to each extracted time sample using the G-FIT tool [130] and finalises the structure of the output queueing network model accordingly. It also infers routing policies and service disciplines adopted at each service centres. Together, these yield a parameterised queueing network model of the real-life or simulated system.

Chapter 4

LocTrackJINQS: A Location-aware Simulation Tool for Multiclass QN Models

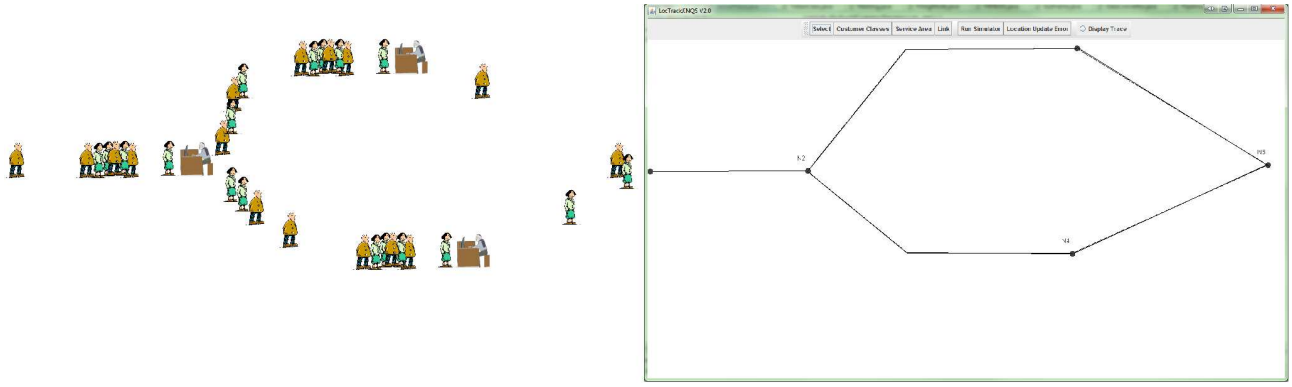
4.1 Introduction

Discrete-event simulation (DES) models are computer models built to mimic the dynamic behaviour of a real-life process as it evolves with time. They provide visualisation and quantitative performance analysis of the system under different scenarios. DES models systems as a network of queues where state changes occur at discrete points of time; they are generally stochastic with randomness introduced by using statistical distributions for sampling event occurrence times or other system behaviour. Specific attributes are assigned to each entity, which determine what happens to them throughout the simulation. DES models have long seen its use in both academic and business applications for system performance modelling and evaluation, workflow analysis, process improvement, asset/personnel management etc. By using DES models decision makers can rapidly experiment with different "what-if" scenarios and compare them at a fraction of the cost of real implementation in the system. Compared to analytical approach, DES models are flexible and able to deal with variability and uncertainty; many of the existing

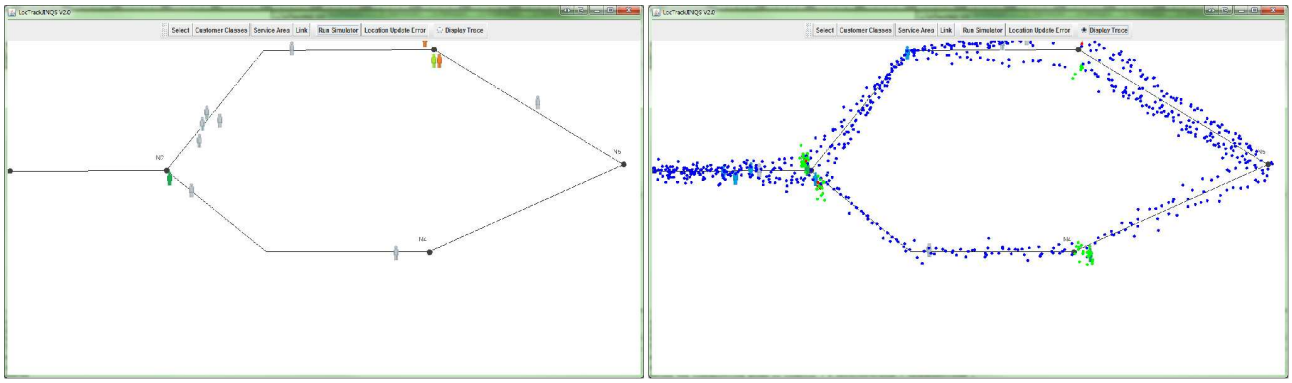
DES software packages offer graphical interfaces to facilitate visual demonstration of system bottlenecks and abnormalities.

Several simulation software tools, such as SIMUL8 [125], WITNESS [100], Enterprise Dynamics [1], MANUPLAN [128], JMT [124], are offered to help managers of different types of service provision systems identify system bottlenecks and gain better insights into the implications of different resource or personnel investments on overall system performance. However, very few existing simulation tools are designed to support location-based research. Most of them do not simulate individual entities' physical movements in a real-life system; nor do they maintain entities' low-level location data during simulation. For conducting location-based research or developing location-aware applications, the availability of large amount of location data generated in various scenarios is necessary. It is also important that the data are generated from a well-understood system so that the developed methodologies or applications can be evaluated or tested against the "reality". Although real-time location data can be collected through laboratory experiments, the process can be time-consuming, labour-intensive; the installation of real-time location systems may be prohibitively costly as well. Thus, this research is motivated to develop an in-house location-aware simulation tool that can rapidly produce large amounts of location tracking traces from different user-defined scenarios and settings.

LocTrackJINQS, developed in this research to support our location-based research ([76] and [11]), is an open-source simulation library that offers functionalities for simulating a real-life customer-processing system as a queueing network while providing low-level location information for each tracked entity (see Figure 4.1). The user can not only specify the high-level features of the network such as the customer flow structure and distributions characterising different time delays (e.g. service time and customer interarrival time) incurred in the system; but also low-level information such as service entities' geographic locations and customer entities' physical moving speeds and paths.



(a) A real-life system



(b) A real-life system

Figure 4.1: An example of simulating a real-life system using LocTrackJINQS: Figure 4.1a demonstrates how a customer processing system is represented as a high-level queueing network with low-level location information; Figure 4.1b shows a screen shot of the simulation in progress and the generated location traces.

LocTrackJINQS is an extension of JINQS, a Java simulation library for multiclass queueing networks developed by [61]. JINQS provides a suite of primitives that allow developers to rapidly build simulations for a wide range of stochastic queueing network models. It offers not only simplicity for simulation construction but also flexibility for application-specific functionalities through the use of inheritance [61]. However, JINQS only allows the creation of high-level simulations of abstract queueing networks and does not support realistic low-level features found in the physical world. This limitation makes it difficult to support simulations that can approximate entities' physical movements in a real-life system, where entities' travelling time might significantly influence the overall system response time. LocTrackJINQS retains the abstract high-level model specification power of JINQS while providing additional low-level models of entity movement. LocTrackJINQS also provides primitives for generating synthetic location tracking data that potentially contain location reading errors and missing data, which are commonly encountered in location observations collected from actual real-time location tracking systems. This eliminates the need for the heavy upfront investment and long-running observation periods that an RTLS installation requires, which benefits research involving mining large location tracking datasets or for developing location-based applications.

The remainder of the chapter first gives an overview of the new functionalities supported by LocTrackJINQS, compared to its predecessor JINQS. Section 4.3 presents the software architecture of LocTrackJINQS and important additions as well as modifications made to JINQS for implementing the new features. Section 4.4 outlines user inputs required for constructing simulations using LocTrackJINQS; and outputs generated from simulation. At the end of the chapter a case study is presented to demonstrate how one can use LocTrackJINQS to build up a model simulating a real-life system.

4.2 JINQS and Major Extensions in LocTrackJINQS

LocTrackJINQS inherits many features from JINQS. In both JINQS and LocTrackJINQS, there are two main Java packages: **network** and **tools** [61]. Classes in **network** are used to define the structure of queueing networks. Package **tools** provides utility classes for setting up simulations, defining common families of probability distributions and calculating performance metrics. The two packages have been designed to be easily extensible; developers can add on application-specific features only by subclassing those existing classes and overriding the inherited methods [61].

JINQS supports simulations of queueing networks with features including multiple servers, multiple customer classes and various queueing disciplines (e.g. FIFO, LIFO, priority-based and service preemption); these are also supported by LocTrackJINQS. Inherited from JINQS, LocTrackJINQS provides primitives for maintaining performance measurements at each service point as well as for the whole network. Two types of measurement variables are maintained – customer-oriented (e.g. mean response time) and system-oriented (e.g. mean queue length and mean overall population in the system) [61]; the summary of all the measures would be outputted at the end of a simulation.

Extended from the features mentioned above, there are three main distinguishing features implemented in LocTrackJINQS:

- Support for location-aware simulations. JINQS only supports construction of high-level QN simulations, where each entity has no physical geographical location in the system and entities travel from one server to another instantaneously. LocTrackJINQS introduces location-related features to support more realistic simulations of real-life customer processing systems. In particular, each entity in the queueing network is assigned a geographical location represented in a 2D Cartesian coordinate system; entity movements occur along user-defined paths at speeds sampled from a user-specified distribution.

- Generate location updates. One of the applications of LocTrackJINQS is to support research on mining agent flow patterns. It thus offers the ability to generate synthetic, yet reasonably realistic, location tracking data traces from its simulations.

- Graphic user interface. LocTrackJINQS offers a graphical interface (defined in package **gui**) for setting up the simulation environment, specifying related parameters and monitoring the simulation process visually.

Other than the new features mentioned above, LocTrackJINQS supports two additional routing policies other than probabilistic routing. One is class-based routing, where customer entities follow certain branching paths depending on their belonging classes; the other one is shortest-queue routing, where customer entities would choose the servers with shortest queues as their next destinations.

4.3 Implementation

4.3.1 Queueing Network Structure

In JINQS, a queueing network comprises three main types of entities, defined by `Node`, `Link` and `Customer` classes (see Figure 4.2). A network is structured as a collection of `Node`s connected together by directed `Link`s. The network is populated by `Customer`s, which move among different `Node`s along the connecting `Link`s and request service or resources at one or more `Node`s. A subclass of `Node`, called `Source`, represents entry points where `Customer`s are injected into the network according to a user-specified interarrival time distribution; `Sink` nodes are where `Customer`s exit the network.

LocTrackJINQS defines queueing networks in a similar way, but introduces the entities' geographical locations (specified as 2D Cartesian coordinates) and allows meaning to be assigned to entities' relative distance (e.g. defining how close a customer must be to a server in order to

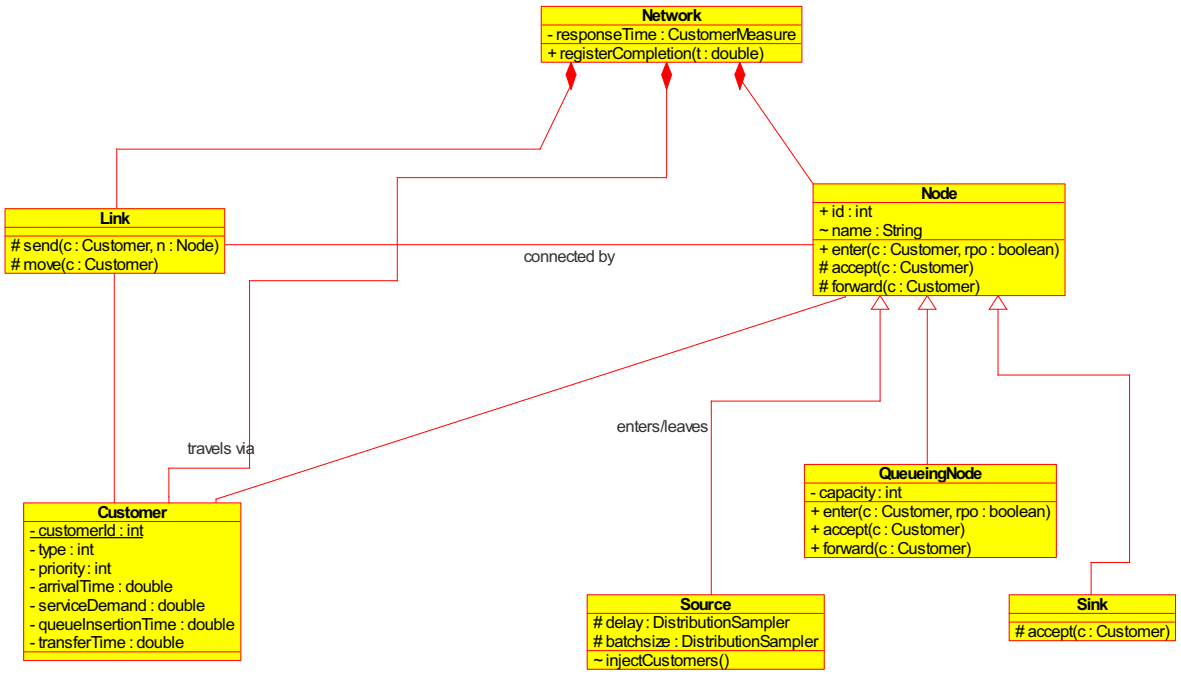


Figure 4.2: UML diagram of important classes in JINQS

be served). LocTrackJINQS further extends the notion of `Node` to include any space that a `Customer` entity might enter and stay for a period of time before departing for its next destination. This is implemented as an interface called `INode`, which defines the basic functions for accepting and forwarding `Customer` entities. Classes implementing this interface include the entry and exit points of the system, defined by `Source` and `Sink` classes (as in JINQS), respectively; a `Server` (comprising one or more service points and a corresponding service area within which `Customer` entities can request and receive service) or a cluster of servers (known as a `MultiQueueingServer`) which share a common queue. Figure 4.3 gives an overview of `INode` and the important classes implementing `INode`.

LocTrackJINQS uses three classes (all implementing the `INode` interface) – `Server` and its subclasses `InfiniteServer` and `QueueingServer` – to define three basic types of service points in a system (see Figure 4.3). Service points are currently assumed to have fixed locations. Their service areas are assumed to be circular (the radii are user-specified) due to the fact that a `Customer` entity's proximity to a service point is used as the criterion to decide whether the `Customer` entity has entered the service point's service area, either being served or queueing for service.

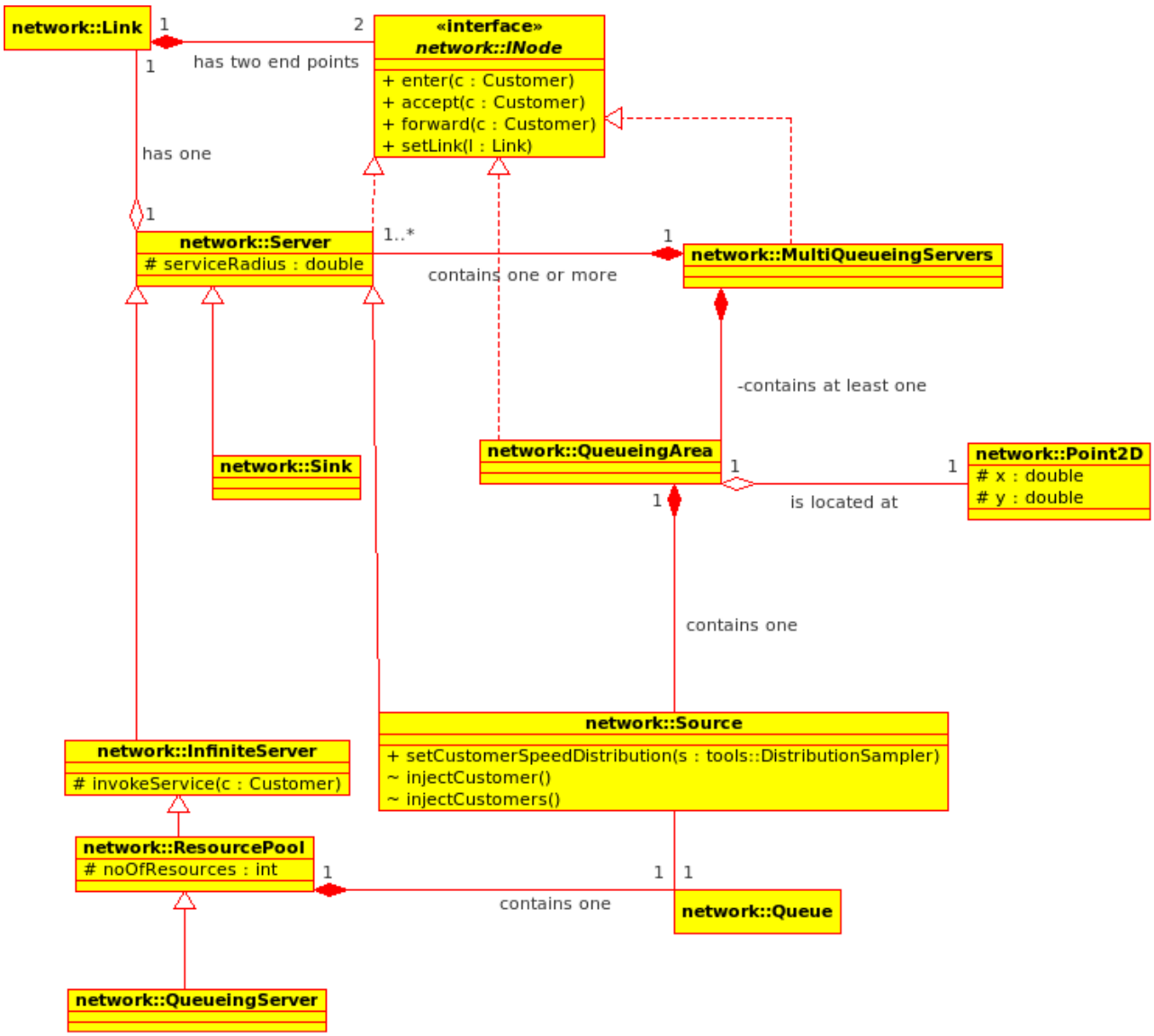


Figure 4.3: UML diagram of the important classes implementing `INode` interface in LocTrackJINQS

An instance of the `Server` class provides no "service" to the customers; after it accepts a `Customer` entity to its service area, it immediately forwards `Customer` entities to their next destinations (selected probabilistically from outgoing links). An `InfiniteServer` entity provides immediate service (i.e. no waiting time) to incoming `Customer` entities; the service time for each served `Customer` entity follows a user-specified distribution. A `QueueingServer` provides a limited pool of service points. Thus incoming `Customer` entities must queue for service if all the service points within the same service area are busy.

Like JINQS, LocTrackJINQS supports multiple customer classes, allowing for simulations of scenarios where interarrival time distributions, service time distributions, routing probabilities and service priorities are class-dependent. Queueing discipline is priority-based and may be FIFO, LIFO or random within customer classes of equivalent priority.

The class `MultiQueueingServers` supports simulations of some commonly-seen scenarios in daily life, where several geographically-separated service points share one or more common queueing areas (an instance of the `QueueingArea` class); examples of these types of systems include post-office counters or hospital treatment rooms with patients waiting in a common waiting area.

In order to simulate `Customer` entities travelling from one location to another, LocTrackJINQS introduces classes `PhysicalLink` and `TransportLink` as subclasses of `Link` (as shown in Figure 4.4). Instead of the abstract connection that a `Link` provides, a `PhysicalLink` represents the physical path that `Customer` entities follow when moving between two `INode` entities. A path is composed of several line segments connected to each other by break points. Unlike JINQS, where the forwarding of `Customer`s between two `Node` entities takes place in no time, a call to the `moveCustomers()` function of a `TransportLink` updates the locations of the `Customer`s following the link based on each `Customer` entity's speed and direction of movement.

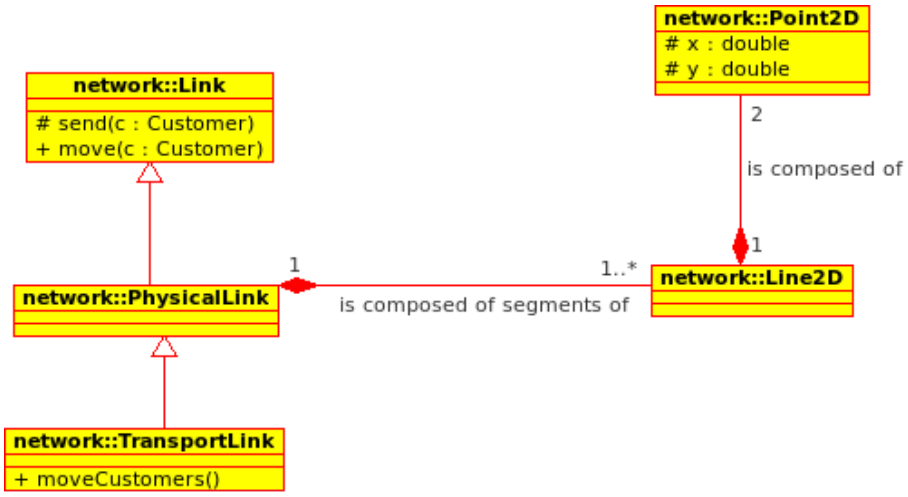


Figure 4.4: UML diagram of `Link`, `PhyscialLink` and `TransportLink`

4.3.2 Generating Synthetic Location Tracking Data

4.3.3 New `Event` classes

Like JINQS, LocTrackJINQS follows the discrete-event simulation model. Under this model, a time-ordered diary of simulation events is maintained and time "hops" to the next event of interest. Processing (or invoking) an event may result in other events being added to the event diary. In both JINQS and LocTrackJINQS the `Event` class and its subclasses define possible events, such as a customer arrival event or a service completion event. We introduction two new subclasses of the `Event` class to support the location-based features implemented in LocTrackJINQS:

- `TransportCustomersEvent` class. The triggering of such an event invokes the method `moveCustomers()` of each `TransportLink` entity. By scheduling such an event to occur on a regular basis (e.g. every few milliseconds), we are able to simulate customer movement at a high resolution.

- `TagReadEvent` classes. When such an event is triggered, it invokes the `updateTagReads()`

method defined in `NetworkMonitor`. According to their update rates, the "read" location of the tags (i.e. their true location adjusted according to user-defined error/noise distributions) are output to the trace file.

4.3.4 Graphic User Interface

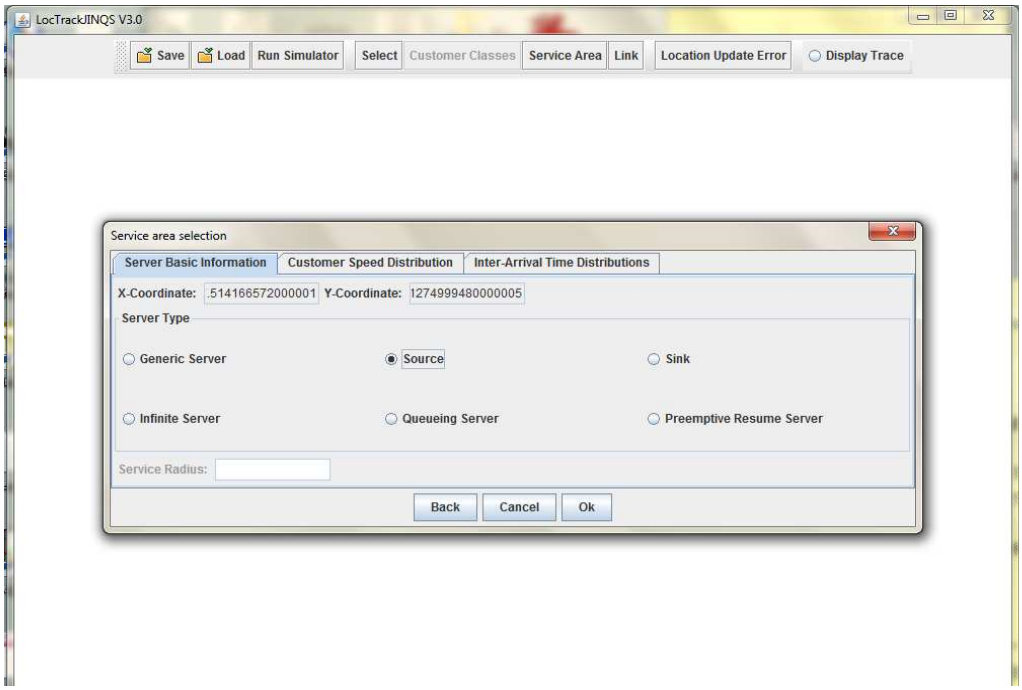
To facilitate the creation and visualisation of location-enhanced simulation models, LocTrackJINQS includes a graphical user interface. This allows users to easily lay out the topology of the queueing network model and specify the parameters of a customer processing system and without having to write codes, as required by JINQS. Figure 4.5 and Figure 4.6 show several screenshots of the GUI, demonstrating how to construct a queueing network simulation using LocTrackJINQS GUI.

4.4 User Inputs and Simulation Outputs

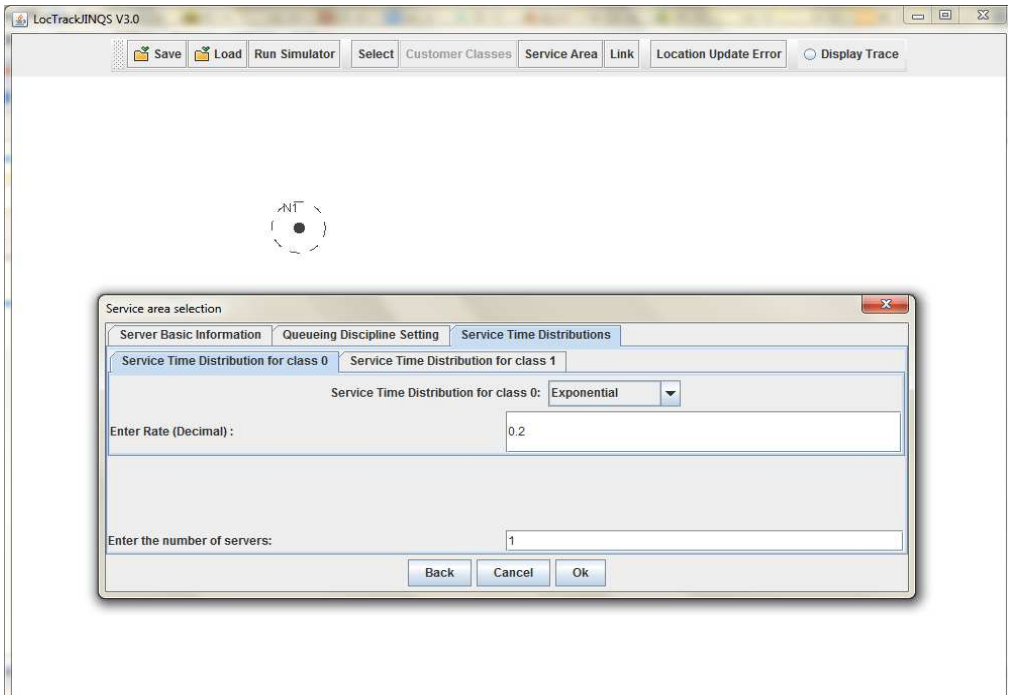
4.4.1 User Inputs

To set up a location-aware simulation using LocTrackJINQS, users need to provide following inputs:

- Locations of entries and exits, where customer entities enter and leave the system, respectively.
- Locations of service centres in the system, as well as the radii of their service areas (which are assumed to be circular).
- Number of customer classes.
- For each service centre:
    1. The number of active server entities in the service centre.

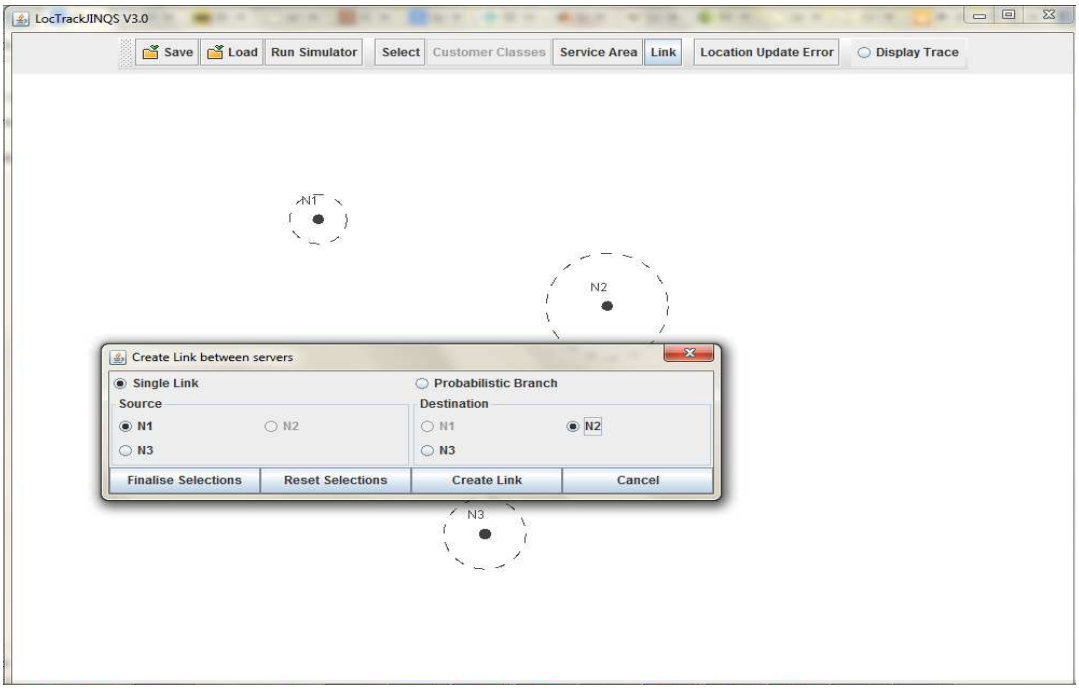


(a) Selecting types of nodes to create

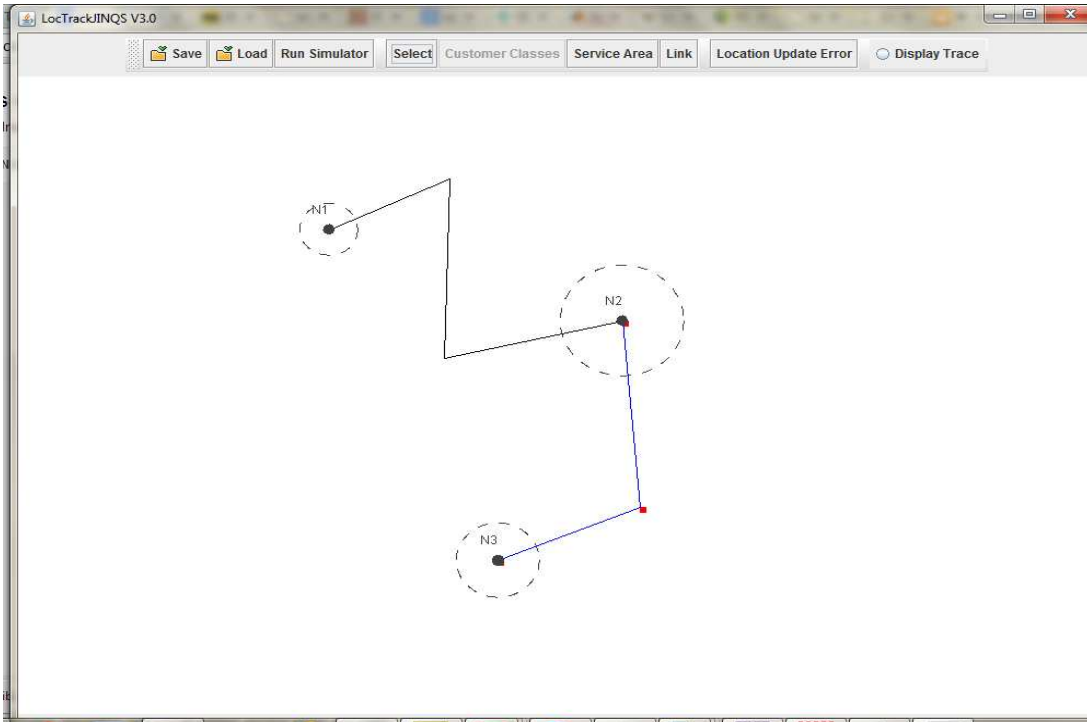


(b) Specifying the settings for a server node

Figure 4.5: Screenshots of LocTrackJINQS GUI : Figure 4.5a demonstrates how a user creates different types of nodes in the queueing network; Figure 4.5b shows how to specify settings (e.g. service time distribution, service discipline and number of server entities) for a server node



(a) Adding a directed link between a pair of server nodes



(b) Creating a physical customer path with several turning points

Figure 4.6: Screenshots of LocTrackJINQS GUI : Figure 4.6a demonstrates how a user creates directed links connecting different of nodes in the queueing network; Figure 4.6b shows that a user can introduce break points on a created directed link, representing turning points of a physical customer path.

2. The service discipline employed in the service centre. Supported service disciplines include FIFO, LIFO, priority-based and service preemption.

3. Service time distributions. Several built-in probability distributions are provided in LocTrackJINQS, including: uniform, deterministic, Gaussian, exponential, Cauchy, Erlang, Pareto, Weibull, Gamma, Geometric, and mixtures of Erlang distributions (including mixtures of exponential distributions).

- Customer interarrival time distribution for each customer class.

- Topology of the simulated queueing system:

  1. Directed links connecting pairs of service centres.

  2. In the presence of branching points, where there are several outgoing links from a service centre, associated routing policies should be specified. LocTrackJINQS supports probabilistic, customer-class based and shortest-queue routing policies.

- Customer entities' velocities. The assigned velocity can be customer-class dependent.

- Parameters regarding outputting location updates:

  1. Location update rates for different types of entities being tracked in the system (e.g. server entities or customer entities).

  2. Distribution of deviations of location readings from tracked entities' real positions; this is to simulate data errors.

- Simulation parameters, including simulation duration and the length of warm-up period.

4.4.2 Simulation Outputs

As JINQS, outputs from LocTrackJINQS include statistics of important performance measures obtained from the simulation, including:

- First three moments of customer queueing time and system response time experienced by customer entities at each service or during their staying at the system; the same metrics are also calculated within each customer class.
- Mean queue length at each service centre.

The validation of performance measures outputted by LocTrackJINQS has been conducted by [10]; which compares the performance metrics from simulation of a single $M/M/1$ queue system with those calculated from analytical formula.

During the simulation, LocTrackJINQS will output the location trace in the form of (`tagName, type, time, x, y`). `tagName` is the unique identifier of the tag attached to a tracked entity in the system; the `type` field is used to categorise tracked entities. The categorisation can be tailored to specific applications; for example, if we have multiple customer classes, `type` can be used to denote the class a particular tagged customer entity belongs to. `time` is the timestamp of the outputted location update and `x`, `y` are the location of the tag in a 2D Cartesian coordinate system.

4.5 Software Demonstration

To demonstrate the new capabilities implemented in LocTrackJINQS we present a case study simulating a real-life scenario, described as follows:
The city council recently opened up a new job centre in a busy commercial area. The job centre is currently staffed with five employees; one receptionist, two advisors and two assistants. The high-level topology of the job centre is depicted in Figure 4.7. Once visitors arrive at the reception of the job centre (represented as `N3`), they are directed to different advisors (`N4` and `N5`) based on their visiting purposes, which can be one of the followings:

- National insurance number application. Referred to as `NIN`.

- Job consultancy. Referred to as `JOB`.
- Benefit claims. Referred to as `BEN`.

After meeting with advisor `N4`, `JOB` and `BEN` visitors will continue to submit documents with either one of the two assistants (represented as `N6` and `N7`), depending on who has the shorter queue in the front. While a small percentage of the `NIN` visitors would be sent to advisor `N4`, the rest will select randomly either one of the assistants `N6` and `N7` for form submission.

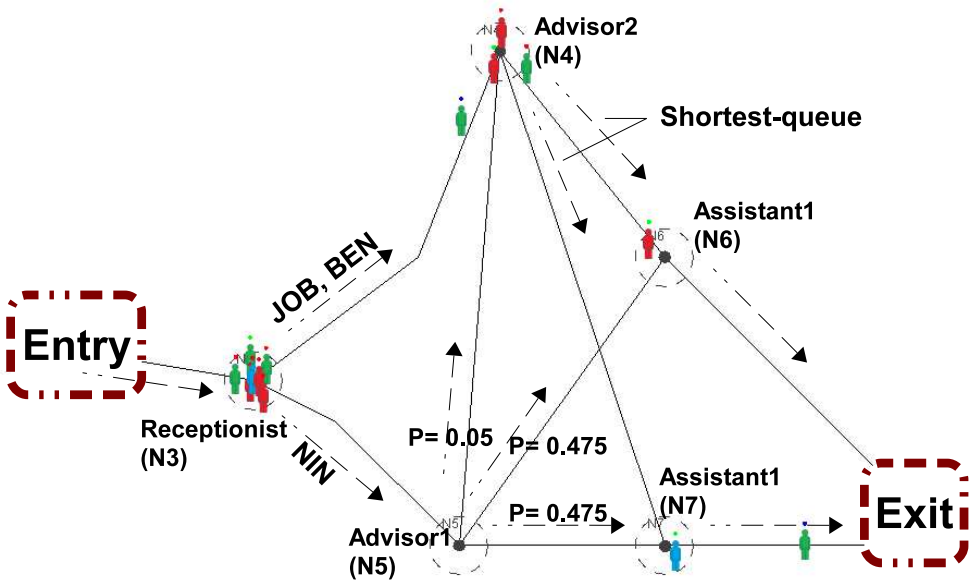


Figure 4.7: Job centre customer flow structure

The speed of all customers in this simulation environment is assumed to follow a deterministic distribution with a mean 0.9. The remaining parameters of the simulation – i.e. interarrival time and service time distribution for each customer class – are listed in Table 4.1.

Table 4.2 shows the calculated mean response time, its variance, and the mean queue length experienced by different customer classes at each service centre and for the whole system.

Service Centre Settings					
Service Centre	N3	N4	N5	N6	N7
Service discipline	FIFO	(NIN,JOB/BEN)	FIFO	FIFO	FIFO
Service time (NIN)	D (15)	Er $(8, 0.65)$	$HErD$ $\boldsymbol{\lambda} = (0.5, 0.2)$	HEr $\boldsymbol{\lambda} = (0.75, 0.1)$ $\mathbf{r} = (7, 3)$ $\boldsymbol{\omega} = (0.55, 0.45)$	HEr $\boldsymbol{\lambda} = (0.75, 0.1)$ $\mathbf{r} = (7, 3)$ $\boldsymbol{\omega} = (0.55, 0.45)$
Service time (JOB)	D (10)	Er $(8, 0.65)$	N/A	Er $(5, 0.35)$	$(5, 0.35)$
Service time (BEN)	D (10)	Er $(8, 0.65)$	N/A	D (7.0)	D (7.0)
Interarrival time	NIN: $Exp\,(0.02)$				
	JOB: $Exp\,(0.03)$				
	BEN: $Exp\,(0.03)$				

Table 4.1: Service and arrival processes specification

Table 4.2 shows the performance measures outputted by LocTrackJINQS, including: the mean and variance of response times experienced by all customers and by different customer classes at individual service centre and during their staying in the entire system; mean queue length at each service centre; utilisation at each service centre. As shown by Table 4.2, visitors classified as NIN experience longest response time on average, followed by JOB and then BEN. In general, visitors across different customer classes spend most time in the system at the reception area, represented as N3 in Figure 4.7; the utilisation of the receptionist also reached 0.9, twenty percent above the second highest one, 0.75 at N4 (representing one of the advisor in the job centre). This indicates the reception might be the major bottleneck of the system; and it is worth investigating the possibility of adding another staff at the reception to expedite the visitor processing time.

4.6 Summary

To address the lack of open-source simulation tools that can support location-based research, this research developed its in-house location-aware simulation tool –LocTrackJINQS – that simulates a system as a network of queues with user-specified service processes and customer arrival processes. As its predecessor JINQS, LocTrackJINQS supports high-level queueing network simulations; however it also simulates entities' physical movements during their sojourn

		System	N3	N4	N5
Aggregate	μ_{rt}	165.394	90.991	25.567	34.414
	σ^2_{rt}	7328.935	6314.373	307.249	507.999
NIN	μ_{rt}	175.735	90.453	17.213	34.414
	σ^2_{rt}	7306.065	6245.327	50.204	507.999
JOB	μ_{rt}	171.438	95.094	26.291	N/A
	σ^2_{rt}	7193.812	6331.521	312.782	N/A
BEN	μ_{rt}	152.508	87.245	25.132	N/A
	σ^2_{rt}	7206.800	6312.711	307.510	N/A
	$\overline{n_q}$	N/A	6.37	0.81	0.19
	ρ	N/A	0.90	0.75	0.49
		N6	N7		
Aggregate	μ_{rt}	15.059	15.722		
	σ^2_{rt}	151.210	186.378		
NIN	μ_{rt}	21.588	22.731		
	σ^2_{rt}	323.920	342.764		
JOB	μ_{rt}	16.227	16.248		
	σ^2_{rt}	92.250	95.209		
BEN	μ_{rt}	10.087	9.675		
	σ^2_{rt}	66.464	66.305		
	$\overline{n_q}$	0.14	0.10		
	ρ	0.56	0.421		

Table 4.2: Mean (μ_{rt}, in seconds) and the variance (σ^2_{rt}, in seconds ^{2}) of the response time for different customer classes at each service centre. $\overline{n_q}$ and ρ represent the mean queue length and utilisation at each service centre, respectively

in the system and maintains individual entities' low-level location information. Along with important performance measures (including moments of customer queuing and sojourn times; mean queue lengths; utilisation of each server entity), LocTrackJINQS also outputs location traces over the simulation period.

Chapter 5

Case Studies

This research takes a simulation-based approach for evaluating the developed data processing pipeline. The ultimate goal for our methodology is to apply with location tracking data collected from real-life systems. However, for the purpose of evaluating the accuracy of the data processing pipeline, it is necessary to have full knowledge of the underlying system in order to estimate how well the inferred queueing network approximates the true system behaviour. As it is extremely unlikely to obtain a complete picture of dynamics in real-life systems, controlled laboratory experiments and simulation are more feasible options. In the preliminary stage of this research, we conducted several experiments where Ubisense RTLS was employed to collect location tracking data in a controlled environment with predefined system settings (including service time distributions and customer interarrival time distributions) [76]. The work in [76] has demonstrated our early-stage data processing pipeline could capture the statistic properties (such as first and second moments) of the underlying service and customer arrival processes. Though it was difficult to stage larger-scale and long-hour experiments with more complex system settings, due to the limits on hardware and personnel resources. Such constraints also make the experiment approach less flexible as it is time-costly to conduct experiments for multiple times. Thus, using synthetic data generated from simulation is a much more appealing option for the purpose of evaluation at the current stage of this research.

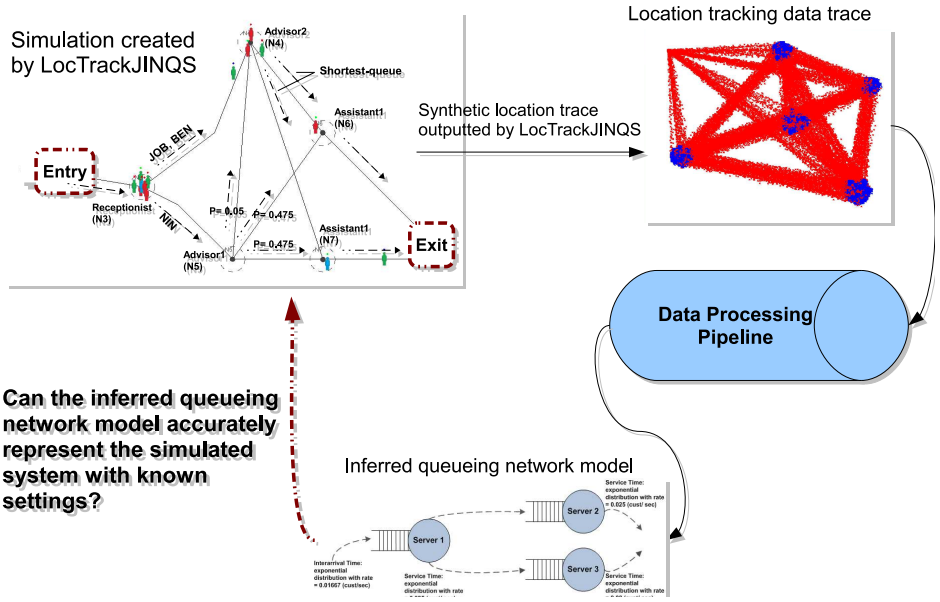


Figure 5.1: Illustration of the simulation-based evaluation methodology in this research

This chapter first presents three case studies with synthetic location tracking data generated by the in-house simulation tool LocTrackJINQS. As demonstrated in Figure 5.1, we first use LocTrackJINQS to simulate a user-specified queueing network system and generate synthetic location tracking data from it. The generated location traces will serve as inputs to the data processing pipeline. The inferred queueing network model outputted from the pipeline is then compared against the true system settings. The first case study is a simple queueing network system with four single-server queues with service times following different distributions and customer flows are routed based on certain probability distribution. The second case study presents a more complicated scenario where multiple customer classes are considered and different service disciplines, including FIFO, LIFO and priority-based, are applied at different server centres. Case study 3 is a simplified version of the real-life scenario presented in Chapter 4.5, with higher level of complexity of customer flow routing policies and offering different service qualities, in terms of serving priorities and service time distributions, for different customer classes. Each case study simulates a medium-size system for 3000 time ticks, equivalent to about 10 hours in reality, with around 400 customer visits. This setting is common in many real-life systems such as a post-office, city hall, shopping mall, sport centre or a small hospital. This is also to evaluate how well the data processing pipeline would perform with datasets of

moderate sizes. The location update error has been set to be normally distributed with a mean of 0.05 m and a standard deviation of 0.01 m.

At the end of the chapter, the fourth case study is also presented with real location tracking data collected in one of the experiments conducted in the preliminary stage of this research. This is to demonstrate the performance of the data processing pipeline with real location tracking data containing data noise or errors with unknown distributions.

For each of the case studies, we first present the system settings, including the system customer flow structure specified in terms of the routing policies; the distribution settings for service time at each service centre and the customer interarrival time to the whole system. We then conduct Kolmogorov-Smirnov test estimating the fitness of the best-fit hyper-Erlang distribution to the extracted sample; we compare the first and second moments of the best-fit hyper-Erlang distribution and the corresponding theoretical distribution. To demonstrate the inferred queueing network models indeed capture the underlying systems' performance features, for the first three case studies we use LocTrackJINQS to construct simulations based on the inferred queueing network settings, compute the key performance measures (including first and second moments of response time, mean queue length and utilisation at each service centre) and compare them with the ones generated from the simulation based on the true underlying model. Each simulation lasts 1 000 000 time ticks with warm up time 50 000 time ticks. The chapter is then concluded by analysis and discussion on the case study results.

For the convenience of presentation, we use the following notations to specify the distributions characterising time delays involved in a system:

- $D\,(t)$: deterministic distribution with parameter t.
- $Exp\,(\lambda)$: exponential distribution with rate λ.
- $Er\,(\lambda, r)$: Erlang distribution with rate λ and phase length r.

- *HExp* $(\boldsymbol{\lambda}, \boldsymbol{\omega})$: hyper-exponential distribution with two parameter vectors $(\boldsymbol{\lambda})$ and $(\boldsymbol{\omega})$. The dimension of $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$ vectors is the number of exponential branches; the values of each dimension in $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$ specify the rate and the weight associated with each branch.

- *HErD* $(\boldsymbol{\lambda}, \boldsymbol{\omega}, \mathbf{r})$: hyper-Erlang distribution with three parameter vectors $\boldsymbol{\lambda}$, $\boldsymbol{\omega}$ and $\mathbf{r}$. The definitions of parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$ are similar to those defined for *HExp* with one additional parameter vector $\mathbf{r}$, which specifies the number of phases at each branch.

5.1 Case Study 1: Simple Queueing Network

5.1.1 System settings

Figure 5.2 depicts the customer flow structure of the system designed for the first case study: a simple queueing network with four single-server queues offering service based on FIFO discipline and service time distributions specified in Table 5.1.

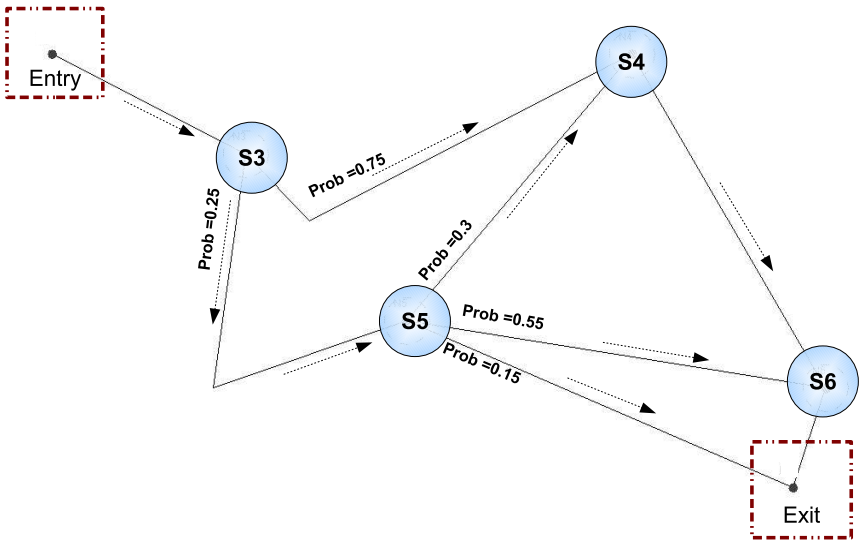


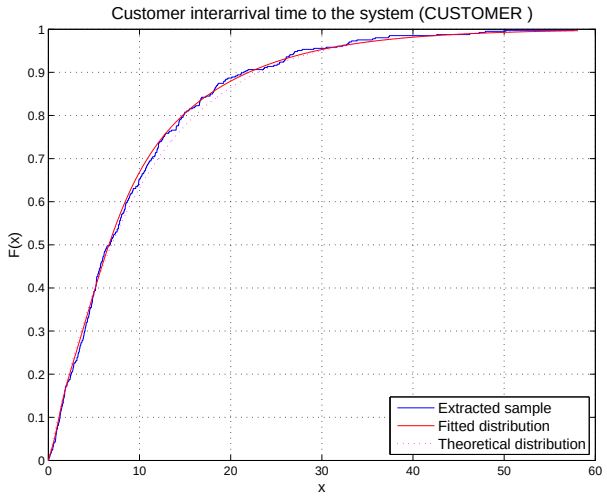
Figure 5.2: Customer flow structure of system for case study 1

Service Centre Settings				
Service Centre	S3	S4	S5	S6
Service Centre	FIFO	FIFO	FIFO	FIFO
Service time	Er $(3, 0.6)$	Er $(5, 1.0)$	$HErD$ $\boldsymbol{\lambda} = (0.7, 0.3, 0.2)$ $\mathbf{r} = (7, 2, 3)$ $\boldsymbol{\omega} = (0.4, 0.25, 0.35)$	D (5.0)
Interarrival time Setting	$Exp\,(0.1)$			

Table 5.1: Service and arrival processes settings for case study 1

5.1.2 Distribution fitting results

Figure 5.3, Figure 5.4, Figure 5.5, Figure 5.6 and Figure 5.7 display the distribution fitting results comparing the best-fit hyper-Erlang distributions found by G-FIT to the corresponding extracted time delay samples and the theoretical distributions. In all cases except service time at S6 (see Figure 5.7), K-S test results suggest that the extracted time delay samples follow the best-fit hyper-Erlang distributions. The first moment and second moment (expressed as squared coefficient of variation, denoted as c_v^2) of the best-fit hyper-Erlang distributions are compared against the theoretical values (specified between parentheses); the percentage of errors for both moments are also calculated. The differences in first moments between the fitting hyper-Erlang distributions and the theoretical ones in all cases are as small as less than two percent; and mostly below one percent difference in second moments.

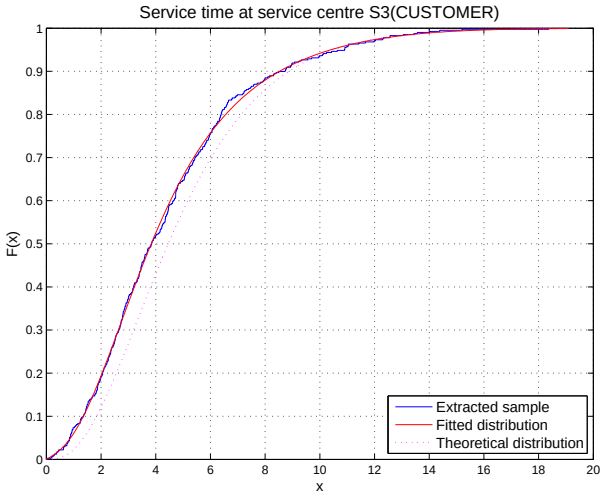


(a)

Fitting results on customer interarrival time	
Best-fit $HErD$	$\boldsymbol{\lambda} = (0.088, 1.422)$ $\mathbf{r} = (1, 4)$ $\boldsymbol{\omega} = (0.969, 0.031)$
K-S test	$p = 0.919$ K-S statistic $= 0.0323$ H_0 not rejected
First moment (error%)	$11.021\,(11.021)$ $-3.21 \times 10^{-5}\%$
c_v^2 (error%)	$1.030\,(1.034)$ -0.625%

(b)

Figure 5.3: Comparing the best-fit hyper-Erlang distribution with the extracted sample and the theoretical distribution for customer interarrival time to the system in case study 1

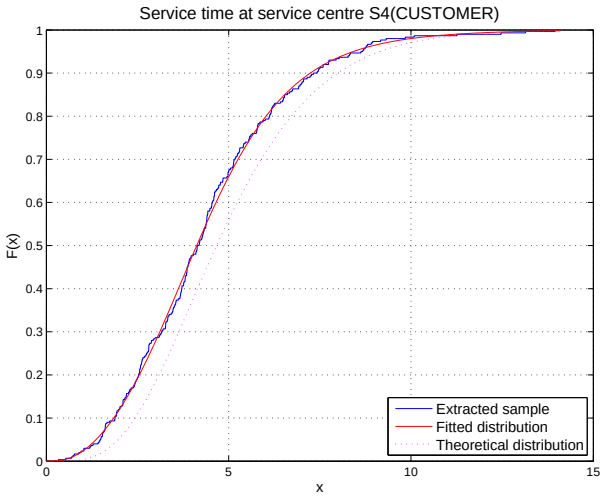


(a)

Fitting results on service time at S3	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.955, 1.110)$ $\mathbf{r} = (4, 10)$ $\boldsymbol{\omega} = (0.799, 0.201)$
K-S test	$p = 0.997$ K-S statistic: 0.023 H_0 not rejected
First moment (error%)	5.154 (5.154) -9.18×10^{-5}%
c_v^2 (error%)	0.332 (0.334) -0.536%

(b)

Figure 5.4: Comparing the best-fit hyper-Erlang distribution with extracted sample and theoretical distribution for service time at service centre S3 in case study 1



(a)

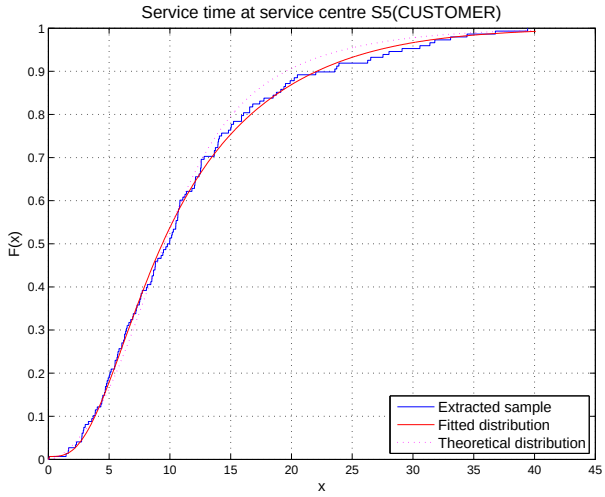
Fitting results on service time at S4	
Best-fit *HErD*	$\boldsymbol{\lambda} = (1.035)$ $\mathbf{r} = (5)$ $\boldsymbol{\omega} = (1.0)$
K-S test	$p = 0.820$ K-S statistic $= 0.043$ H_0 not rejected
First moment (error%)	4.830 (4.830) -1.24×10^{-4}%
c_v^2 (error%)	0.197 (0.2) -1.308%

(b)

Figure 5.5: Comparing the best-fit hyper-Erlang distribution with extracted sample and theoretical distribution for service time at service centre S4 in case study 1

5.1.3 Inferred queueing network model and response time analysis

The structure of the inferred queueing network is presented in Figure 5.8. As mentioned in Chapter 3, the customer travelling time between a pair of connecting service centres is modelled by an infinite server node, whose service time distribution follows the hyper-Erlang distribution best fit to the extracted customer travelling time sample. These additional infinite server nodes are labelled with IDs starting with **IS**. In Figure 5.8 the inferred routing probabilies are also specified at each branching point, which are similar to the designed customer flow structure

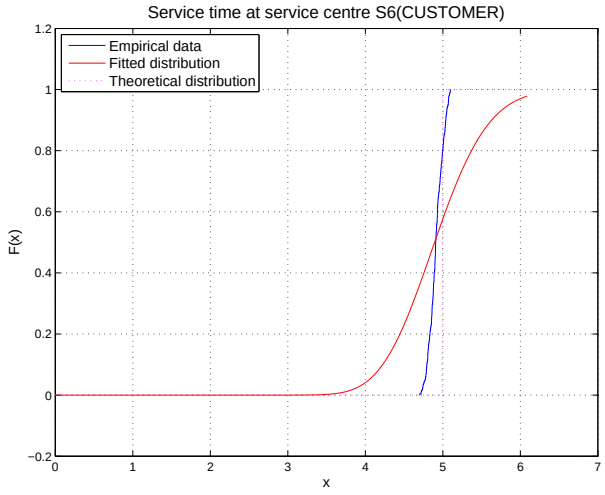


(a)

Fitting results on service time at S5	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.177, 1.226)$ $\boldsymbol{r} = (2, 9)$ $\boldsymbol{\omega} = (0.558, 0.441)$
K-S test	$p = 0.807$ K-S statistic $= 0.064$ H_0 not rejected
First moment (error%)	9.545 (9.545) $-1.38e-004\%$
c_v^2 (error%)	0.429 (0.462) -7.067%

(b)

Figure 5.6: Comparing the best-fit hyper-Erlang distribution with extracted sample and theoretical distribution for service time at service centre S5 in case study 1



(a)

Fitting results on service time at S6	
Best-fit *HErD*	$\boldsymbol{\lambda} = (16.279)$ $\boldsymbol{r} = (80)$ $\boldsymbol{\omega} = (1.0)$
K-S test	$p = 1.55 \times 10^{-32}$ K-S statistic $= 0.370$ H_0 not rejected
First moment (error%)	4.914 (5.000) -1.7%
c_v^2 (error)	0.013 (0.000) 0.013

(b)

Figure 5.7: Comparing the best-fit hyper-Erlang distribution with extracted samples and theoretical distribution for service time at service centre S6 in case study 1

shown in Figure 5.2. Figure 5.9b lists the first and second moments of the response times, as well as mean queue length and utilisation at each service centre generated through simulation of the inferred queueing network; the simulation results based on the true underlying system settings are specified next in parentheses.

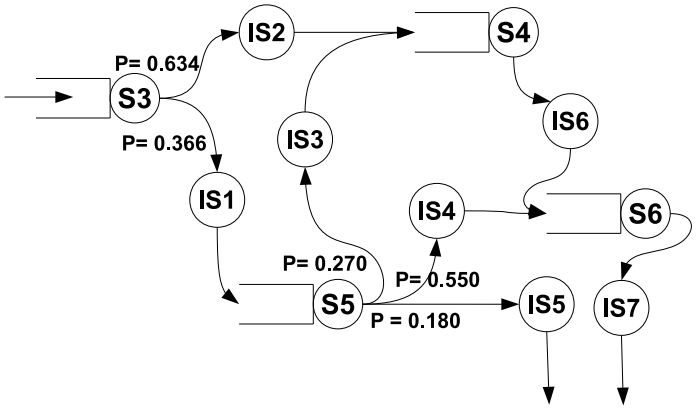
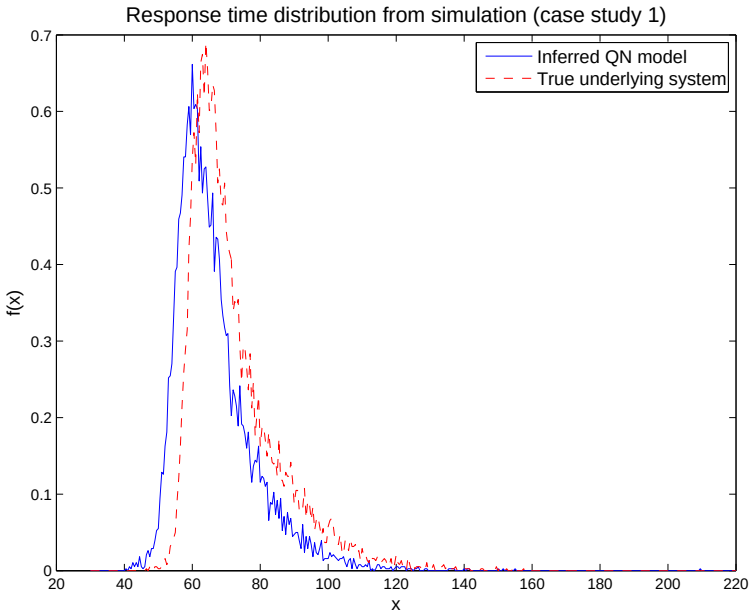


Figure 5.8: Inferred queueing network model for case study 1



(a) Response time distributions generated through simulation

		System	S3	S4	S5	S6
Aggregate	μ_{rt}	66.721 (72.716)	7.945 (8.336)	5.837 (6.369)	12.894 (14.735)	5.997 (6.341)
	cv^2_{rt}	0.032 (0.033)	0.494 (0.551)	0.274 (0.315)	0.567 (0.483)	0.123 (0.132)
	$\overline{n_q}$	N/A	0.25 (0.33)	0.06 (0.1)	0.09 (0.14)	0.19 (0.12)
	ρ	N/A	0.45 (0.50)	0.31 (0.38)	0.29 (0.38)	0.4 (0.47)

(b) Mean (μ_{rt}, in seconds) and the squared coefficient of variation (cv^2_{rt}) of customer response time at each service centre in case study 1. $\overline{n_q}$ and ρ represent the mean queue length and utilisation at each service centre, respectively

Figure 5.9: Response time analysis for case study 1

5.2 Case Study 2: Multiclass Queueing Network with Different Service Disciplines

5.2.1 System settings

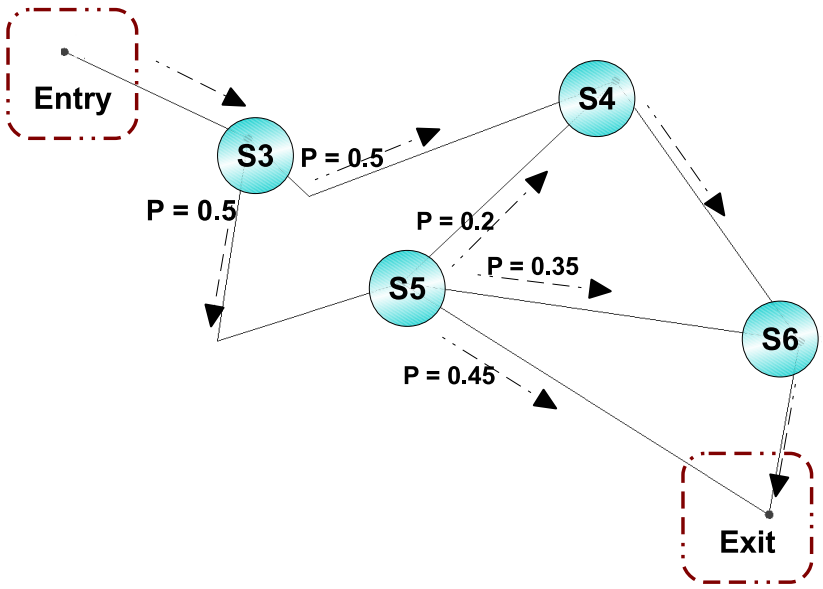


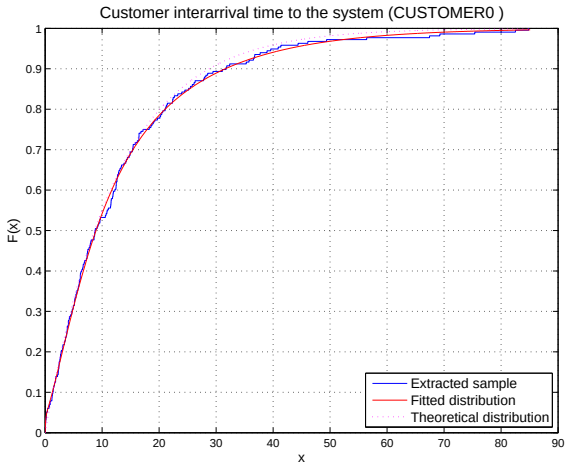
Figure 5.10: Customer flow structure of system for case study 2

Service Centre Settings				
Service Centre	S3	S4	S5	S6
Service discipline	(CUSTOMER1,CUSTOMER0)	(CUSTOMER1,CUSTOMER0)	LIFO	FIFO
Service time settings (CUSTOMER0)	*Er* $(3, 0.6)$	*HExp* $\boldsymbol{\lambda} = (0.1, 0.35)$ $\boldsymbol{\omega} = (0.75, 0.25)$	*ErD* $(5, 1.0)$	*D* (8.0)
Service time settings (CUSTOMER1)	as above	*HErD* $\boldsymbol{\lambda} = (0.85, 0.3)$ $\mathbf{r} = (4, 2)$ $\boldsymbol{\omega} = (0.7, 0.3)$	*HErD* $\boldsymbol{\lambda} = (0.7, 0.3)$ $\mathbf{r} = (6, 3)$ $\boldsymbol{\omega} = (0.65, 0.35)$	*Er* $(5, 0.9)$
Interarrival time	CUSTOMER0: $Exp\,(0.1)$			
	CUSTOMER1: $Exp\,(0.05)$			

Table 5.2: Service and arrival processes specification for case study 2

As depicted in Figure 5.10 the customer flow structure of the system in the second case study is similar to that of case study 1. In addition, case study 2 introduces two customer classes – `CUSTOMER0` and `CUSTOMER1` – with `CUSTOMER1` enjoying higher priority over `CUSTOMER0`. The four service centres in the system employ different service disciplines as listed in Table 5.2.

5.2.2 Distribution fitting results

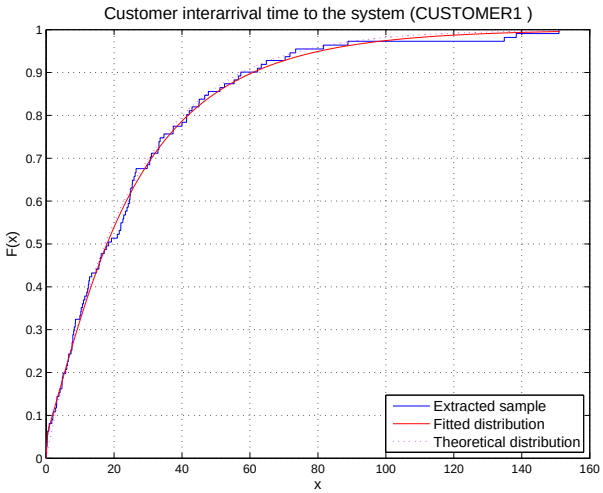


(a)

Fitting results on customer interarrival time for `CUSTOMER0`	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.061, 0.108, 10.942, 0.237)$ $\mathbf{r} = (1, 1, 1, 2)$ $\boldsymbol{\omega} = (0.676, 0.022, 0.037, 0.265)$
K-S test	$p = 0.911$ K-S statistic $= 0.037$ H_0 not rejected
First moment (error%)	13.510 (12.5) 8.077%
c_v^2 (error%)	1.160 (1.000) 16.007%

(b)

Figure 5.11: Comparing the best-fit hyper-Erlang distribution with extracted sample and the theoretical distribution for interarrival time of `CUSTOMER0` in case study 2

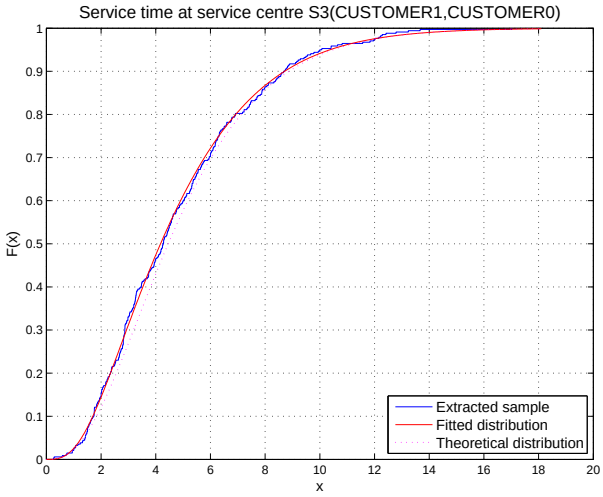


(a)

Fitting results on customer interarrival time for `CUSTOMER1`	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.034, 0.102, 8.060)$ $\mathbf{r} = (1, 2, 2)$ $\boldsymbol{\omega} = (0.785, 0.166, 0.049)$
K-S test	$p = 0.980$ K-S statistic $= 0.043$ H_0 not rejected
First moment (error%)	26.154 (25) 4.616%
c_v^2 (error%)	1.091 (1.000) 9.064%

(b)

Figure 5.12: Comparing the best-fit hyper-Erlang distribution with extracted sample and the theoretical distribution for interarrival time of `CUSTOMER1` in case study 2



(a)

Fitting results on service times at S3 for both CUSTOMER0 and CUSTOMER1	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.592, 1.990)$ $\mathbf{r} = (3, 5)$ $\boldsymbol{\omega} = (0.890, 0.110)$
K-S test	$p = 0.958$ K-S statistic $= 0.027$ H_0 not rejected
First moment (error%)	4.789 (5) −4.223%
c_v^2 (error%)	0.367 (0.333) 9.975%

(b)

Figure 5.13: Comparing the best-fit hyper-Erlang distribution with extracted sample and theoretical distribution for service time at service centre S3 in case study 2

Figure 5.11 and Figure 5.12 demonstrate the distribution fitting results for customer interarrival times from two different customer classes – CUSTOMER0 and CUSTOMER1. Figure 5.13 presents the fitting results for service time at service centre S3 for both CUSTOMER0 and CUSTOMER0 classes. Note here that the data processing pipeline extracts service time samples separately for each individual customer class. The Mann-Whitney U test result comparing the service time samples from these two customer classes has failed to reject the null hypothesis that they are from the same distribution; thus they are combined into one single sample before the distribution fitting step. This agrees with the service setting for S3 in case study 2.

Please note here that Figure 5.14 only displays the comparison between the extracted samples and the best-fit hyper-Erlang distribution. While the theoretical service time settings are different for CUSTOMER0 and CUSTOMER1, the result of Mann-Whitney U test comparing the extracted samples for service times of CUSTOMER0 and CUSTOMER1 suggests that there is no significant difference (at significance level 0.05) between the underlying distributions of these two samples. We argue that the data extraction error should not be the main cause for the data processing pipeline failing to discover the distinctions of service demands from these two customer classes. Figure 5.17a and Figure 5.17b compare the histograms of service time samples extracted by the data processing pipeline and the actual samples generated from the simulation for CUSTOMER0

and CUSTOMER1; both of them display high resemblance to each other and the K-S test results also failed to reject the null hypothesis that the extracted service time sample is significantly different (at significance level 0.05) from the simulation-generated one in both cases. As the extracted service time sample for CUSTOMER0 only contains 132 data points and the extracted sample is even smaller, 62 for CUSTOMER1, the extracted service time samples might not be large enough to exhibit the distinction between service time distributions for these two customer classes.

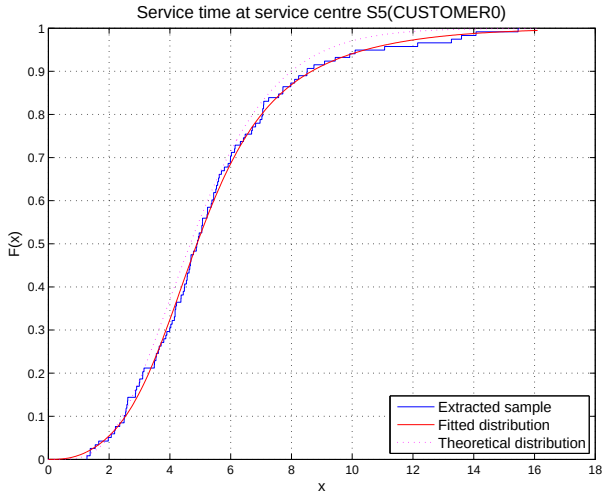
(a)

Fitting results on service times at S4 for both CUSTOMER0 and CUSTOMER1	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.124, 0.408, 6.642)$ $\mathbf{r} = (2, 2, 2)$ $\boldsymbol{\omega} = (0.264, 0.674, 0.062)$
K-S test	$p = 0.906$ K-S statistic: 0.0398 H_0 not rejected
First moment (error%)	7.596(N/A) N/A
c_v^2 (error%)	1.216(N/A) N/A

(b)

Figure 5.14: Comparing the best-fit hyper-Erlang distribution with extracted sample and theoretical distribution for service time at service centre S4 in case study 2

Figure 5.15, Figure 5.16, Figure 5.18, Figure 5.19 show the distribution fitting analysis for service times at service centres S5 and S6. Except service time for CUSTOMER0 at S6 (which is a deterministic case), the K-S test results have shown in general good fitness to the extracted service time samples. The first moments of the best-fit hyper-Erlang distributions are within less than 10% of difference from the theoretical values in all cases; larger error rates are observed for second moments, but most of time they are still below 15%.

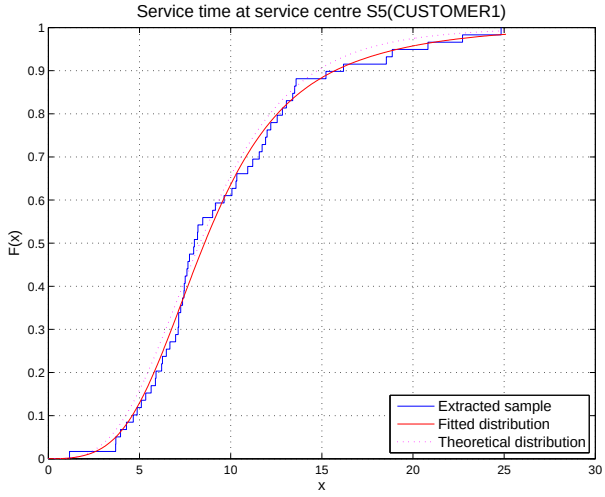


(a)

Fitting results on service time for CUSTOMER0 at S5	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.530, 2.228)$ $\mathbf{r} = (3, 11)$ $\boldsymbol{\omega} = (0.568, 0.432)$
K-S test	$p = 0.996$ K-S statistic: 0.0362 H_0 not rejected
First moment (error%)	5.350 (5) 7.004%
c_v^2 (error%)	0.250 (0.2) 25.142%

(b)

Figure 5.15: Comparing the best-fit hyper-Erlang distribution with extracted sample and theoretical distribution for service time for CUSTOMER0 at service centre S5 in case study 2



(a)

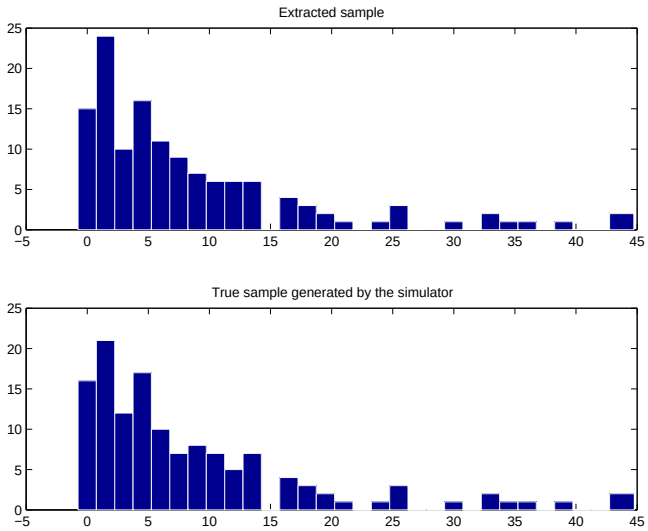
Fitting results on service time for CUSTOMER1 at S5	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.269, 0.956)$ $\mathbf{r} = (3, 8)$ $\boldsymbol{\omega} = (0.435, 0.565)$
K-S test	$p = 0.806$ K-S statistic: 0.081 H_0 not rejected
First moment (error%)	9.580 (9.071) 5.607%
c_v^2 (error%)	0.271 (0.244) 11.101%

(b)

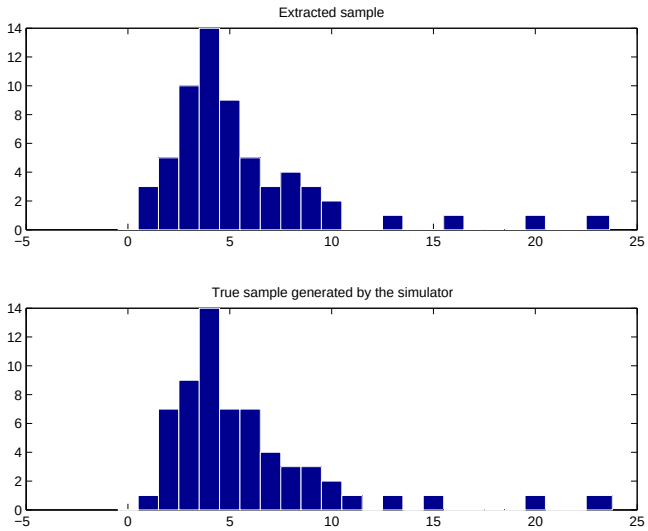
Figure 5.16: Comparing the best-fit hyper-Erlang distribution with extracted sample and theoretical distribution for service time for CUSTOMER1 at service centre S5 in case study 2

5.2.3 Inferred queueing network model and response time analysis

Figure 5.20a shows the customer flow structure of the output queueing network model, which in general agrees with the true probabilistic settings of the underlying system. Figure 5.20b lists the total scores each candidate service discipline obtains in predicting correctly the next customer to be served at each service centre when there are more than one customers in queue. The most likely service discipline is the one with the highest score (in bold face in Figure 5.20b); it shows that the data processing pipeline has selected the correct service disciplines. From



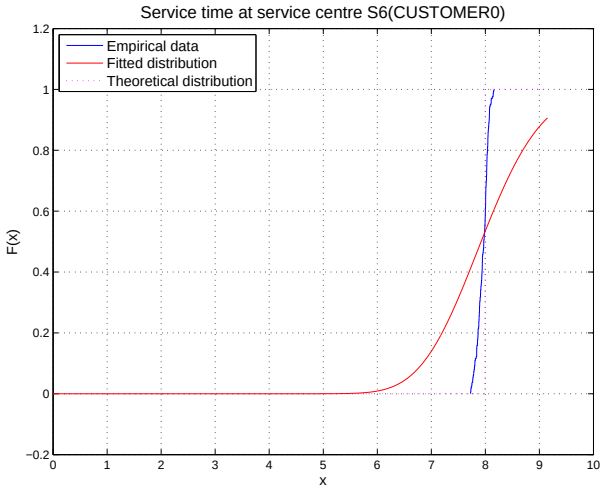
(a) Histograms of the extracted service time sample and the true sample generated by the simulator for `CUSTOMER0` at `S4`



(b) Histograms of the extracted service time sample and the true sample generated by the simulator for `CUSTOMER1` at `S4`

Figure 5.17: Figure 5.17a and Figure 5.17b show that the extracted service time samples for `CUSTOMER0` and `CUSTOMER1` have similar distributions with the synthetic ones generated by the simulation; two-sample K-S test results also fail to reject the null hypothesis that the extracted service time samples for `CUSTOMER0` (p-value= 0.998, K-S statistic= 0.046) and `CUSTOMER1` (p-value= 0.984, K-S statistic = 0.081) are not significantly different at significance level 0.05 from the synthetic samples.

Figure 5.21c, we observe that the performance measures generated by simulation based on the inferred queueing network in general share similar scale of quantities comparing to those generated by simulation based on the true underlying system settings. Figure 5.21a and Figure 5.21b also show that the response time distributions of the inferred queueing network indeed follow the trend of those of the theoretical system for both customer classes.

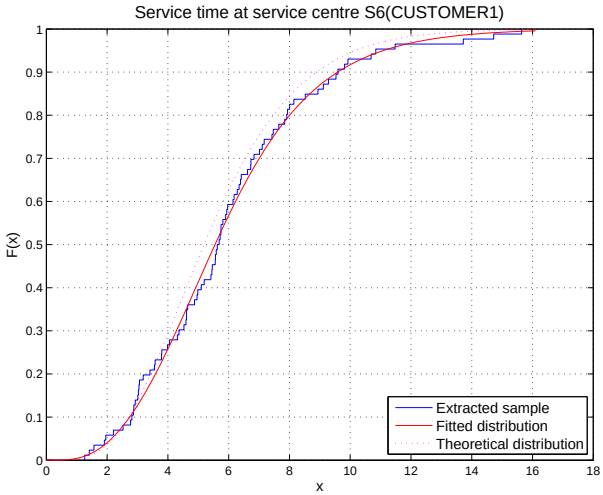


(a)

Fitting results on service time for `CUSTOMER0` at `S6`	
Best-fit *HErD*	$\boldsymbol{\lambda} = (10.057)$ $\mathbf{r} = (80)$ $\boldsymbol{\omega} = (1.0)$
K-S test	$p = 4.136e - 25$ K-S statistic: 0.412 H_0 rejected
First moment (error%)	7.955 (8) −0.567%
c_v^2	0.013 (0.000)

(b)

Figure 5.18: Comparing the best-fit hyper-Erlang distribution with extracted sample and theoretical distribution for service time for `CUSTOMER0` at service centre `S6` in case study 2

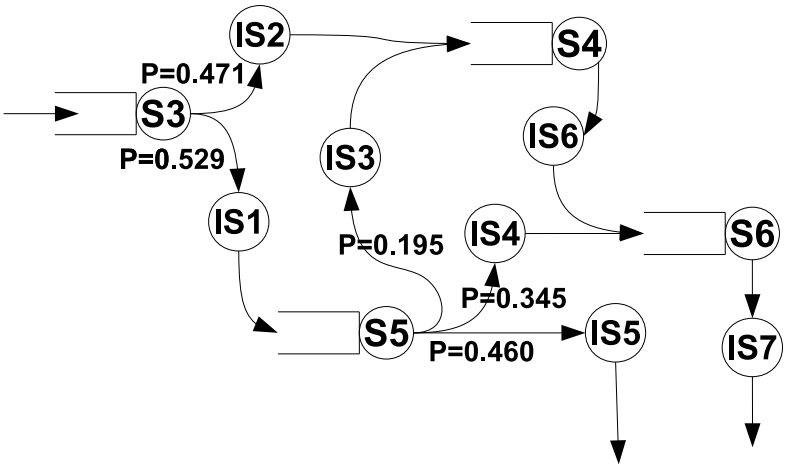


(a)

Fitting results on service time for `CUSTOMER1` at `S6`	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.679, 1.786)$ $\mathbf{r} = (4, 11)$ $\boldsymbol{\omega} = (0.825, 0.175)$
K-S test	$p = 0.888$ K-S statistic: 0.061 H_0 not rejected
First moment (error%)	5.938 (5.556) 6.893%
c_v^2 (error%)	0.220 (0.2) 10.234%

(b)

Figure 5.19: Comparing the best-fit hyper-Erlang distribution with extracted sample and theoretical distribution for service time for `CUSTOMER1` at service centre `S6` in case study 2

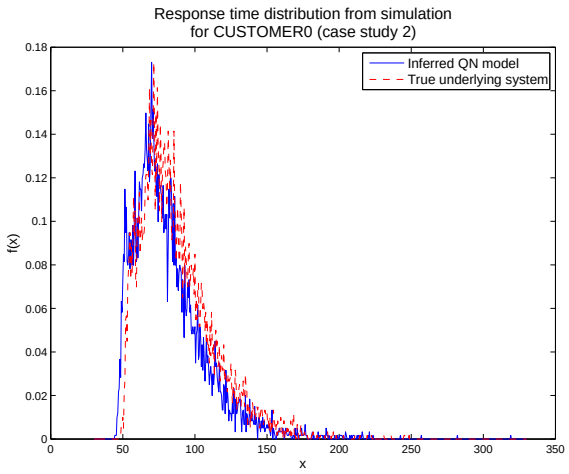


(a) Inferred queueing network structure

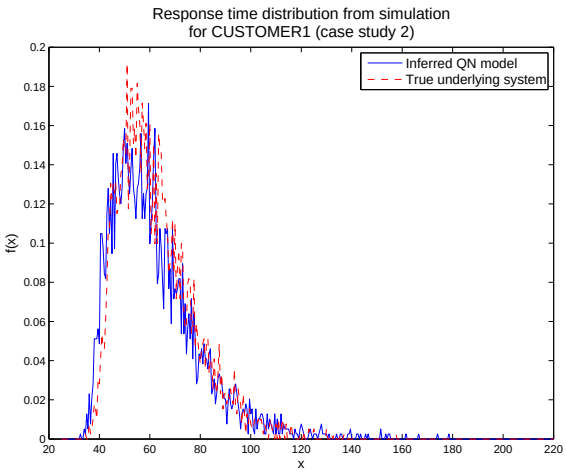
Server ID	S3	S4	S5	S6
FIFO	59	40	0	**90**
LIFO	11	7	**12**	0
(CUSTOMER1,CUSTOMER0)	**81**	**55**	5	67
(CUSTOMER0,CUSTOMER1)	44	67	1	71

(b) Compare the scores of candidate service disciplines in predicting correctly the following customer to be served; the one with the highest score (in bold text) is selected.

Figure 5.20: The inferred queueing network and the inferred service time discipline at each service centre in case study 3



(a)



(b)

		System	S3	S4
Aggregate	μ_{rt}	75.480 (79.338)	9.020 (9.679)	15.555 (14.979)
	cv^2_{rt}	0.106 (0.102)	0.801 (0.723)	1.359 (1.424)
CUSTOMER0	μ_{rt}	82.194 (87.823)	9.903 (10.831)	17.059 (17.496)
	cv^2_{rt}	0.092 (0.082)	0.888 (0.749)	1.413 (1.350)
CUSTOMER1	μ_{rt}	62.249 (62.718)	7.425 (7.383)	12.813 (9.847)
	cv^2_{rt}	0.076 (0.062)	0.390 (0.361)	1.017 (0.922)
	$\overline{n_q}$	N/A	0.46 (0.54)	0.51 (0.53)
	ρ	N/A	0.54 (0.59)	0.48 (0.51)
		S5	S6	
Aggregate	μ_{rt}	10.240 (8.410)	13.909 (13.949)	
	cv^2_{rt}	1.040 (0.683)	0.451 (0.386)	
CUSTOMER0	μ_{rt}	8.379 (6.930)	14.856 (14.833)	
	cv^2_{rt}	1.164 (0.735)	0.399 (0.346)	
CUSTOMER1	μ_{rt}	13.566 (11.282)	12.050 (12.136)	
	cv^2_{rt}	0.764 (0.481)	0.554 (0.462)	
	$\overline{n_q}$	0.46 (0.54)	0.57 (0.72)	
	ρ	0.40 (0.38)	0.63 (0.66)	

(c) Mean (μ_{rt}, in seconds) and the squared coefficient of variation (cv^2_{rt}) of the response time for each customer class at each service centre in case study 2. $\overline{n_q}$ and ρ represent the mean queue length and utilisation at each service centre, respectively

Figure 5.21: Response time analysis for case study 2

5.3 Case Study 3: Queueing Network with Different Routing Policies

5.3.1 System settings

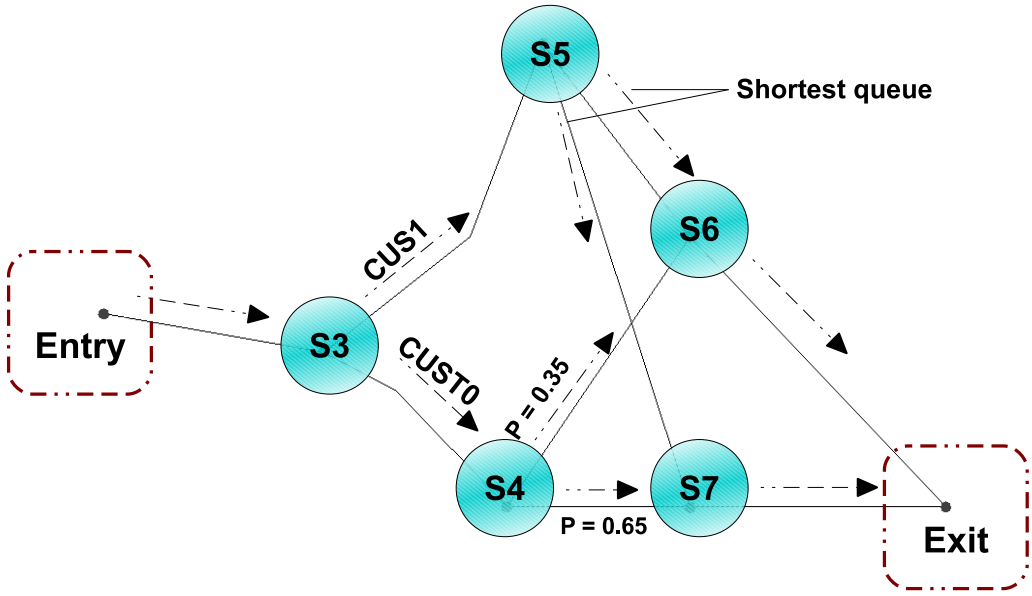


Figure 5.22: Customer flow structure of system for case study 3

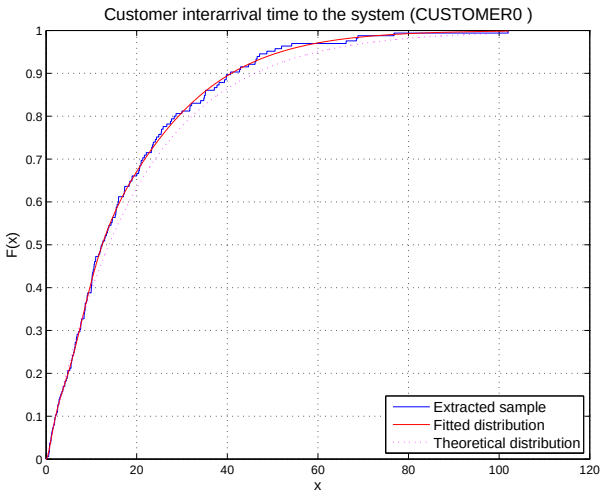
Service Centre Settings					
Service Centre	S3	S4	S5	S6	S7
Service discipline	(CUSTOMER0,CUSTOMER1)	FIFO	FIFO	FIFO	FIFO
Service time (CUSTOMER0)	D (4.5)	N/A	$HErD$ $\boldsymbol{\lambda} = (0.6, 1.5, 0.05)$ $\mathbf{r} = (4, 7, 1)$ $\boldsymbol{\omega} = (0.30, 0.35, 0.35)$	Er $(6, 0.75)$	Er $(3, 0.65)$
Service time (CUSTOMER1)	ErD $(3, 0.6)$	$HErD$ $\boldsymbol{\lambda} = (1.0, 0.3)$ $\mathbf{r} = (4, 2)$ $\boldsymbol{\omega} = (0.7, 0.3)$	N/A	as above	$HErD$ $\boldsymbol{\lambda} = (0.75, 0.1)$ $\mathbf{r} = (4, 1)$ $\boldsymbol{\omega} = (0.65, 0.35)$
Interarrival time	CUSTOMER0: $Exp\,(0.05)$				
	CUSTOMER1: $Exp\,(0.1)$				

Table 5.3: Service and arrival processes specification for case study 3

In this case study, the customer flow structure is a slightly simplified version of the one presented in Section 4.5. This is to demonstrate the capability of our data processing pipeline in inferring a simple real-life scenario with service centres supporting multiple customer classes, class-based service qualities and different routing policies.

As shown in Figure 5.22 after the customers pass through the service centre `S3`, they are directed to different destinations according to the their classifications – `CUSTOMER0` and `CUSTOMER1` in this case. Later the `CUSTOMER0` visitors would choose either of the two service centres `S5` and `S6` based on certain probability distribution as their last stops before leaving the system; while `CUSTOMER1` visitors tend to choose the service centre with the shortest queue length. Table 5.3 shows the specifications of the time delays involved in the system.

5.3.2 Distribution fitting results



Fitting results on customer interarrival time (`CUSTOMER0`)	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.081, 0.955, 0.764)$ $\mathbf{r} = (2, 2, 7)$ $\boldsymbol{\omega} = (0.630, 0.153, 0.217)$
K-S test	$p = 1.000$ K-S statistic $= 0.023$ H_0 not rejected
First moment (error%)	17.918 (20) −10.41%
c_v^2 (error%)	0.875 (1.000) −12.48%

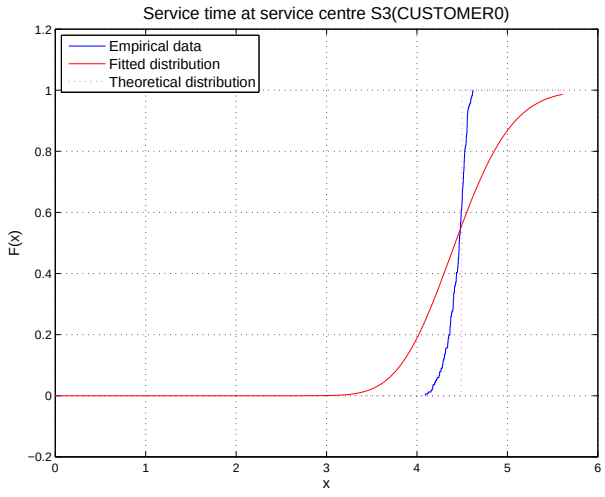
Figure 5.23: Comparing the best-fit hyper-Erlang distribution with extracted sample and the theoretical distribution for for the interarrival time of `CUSTOMER0` in case study 3

As presented by the figures listed below (see Figure 5.23, Figure 5.24, Figure 5.25, Figure 5.26 Figure 5.27, Figure 5.28 Figure 5.29, Figure 5.30 , Figure 5.31), K-S test results again show that the best-fit hyper-Erlang distributions in general have good fitness to the extracted time-delay

Fitting results on customer interarrival time (CUSTOMER0)	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.104)$ $\mathbf{r} = (1)$ $\boldsymbol{\omega} = (1.0)$
K-S test	$p = 0.996$ K-S statistic $= 0.022$ H_0 not rejected
First moment (error%)	9.609 (10) -3.915%
c_v^2 (error%)	1.000 (1.000) 0.00%

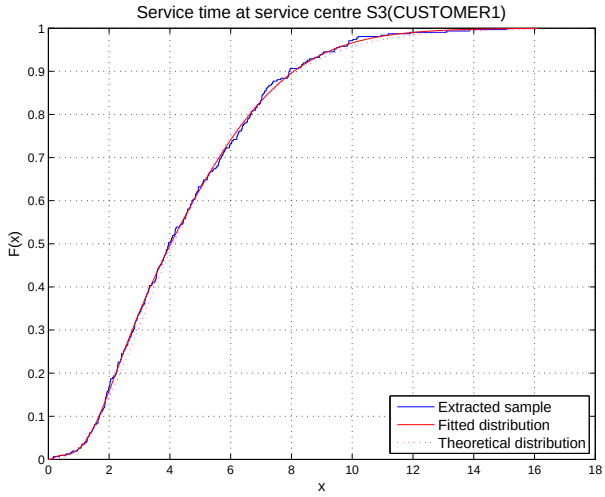
Figure 5.24: Comparing the best-fit hyper-Erlang distribution with extracted sample and the theoretical distribution for interarrival time of CUSTOMER1 in case study 3

samples; the fitting hyper-Erlang distributions also match well in first and second moments with those calculated from the true underlying distributions. We are also able to distinguish the case (see Figure 5.29) where the service demands from visitors belonging to CUSTOMER0 and CUSTOMER1 types are following the same distribution.



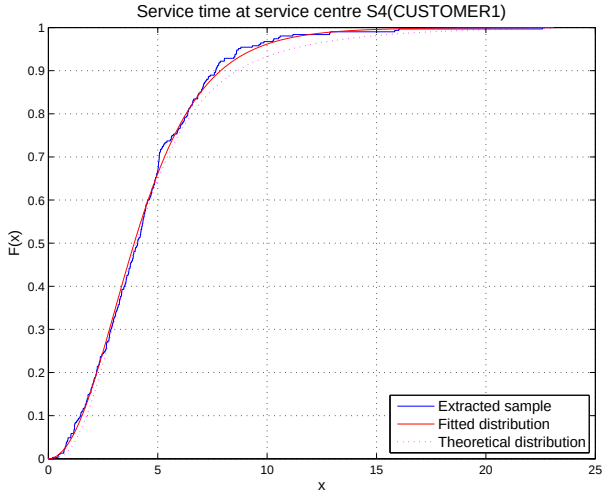
Fitting results on service time for CUSTOMER0 at S3	
Best-fit *HErD*	$\boldsymbol{\lambda} = (18.012)$ $\mathbf{r} = (80)$ $\boldsymbol{\omega} = (1.0)$
K-S test	$p = 3.313e-18$ K-S statistic $= 0.348$ H_0 rejected
First moment (error%)	4.44138482378806 (4.5) -1.30%
c_v^2 (error%)	0.0125 (0.00) N/A

Figure 5.25: Comparing the best-fit hyper-Erlang distribution with extracted samples and theoretical distribution for service time for CUSTOMER0 at service centre S3 in case study 3



Fitting results on service time for CUSTOMER1 at S3	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.347, 0.700, 1.141)$ $\mathbf{r} = (1, 3, 12)$ $\boldsymbol{\omega} = (0.110, 0.833, 0.057)$
K-S test	$p = 0.999$ K-S statistic= 0.020 H_0 not rejected
First moment (error%)	4.513 (4.615) −2.224%
c_v^2 (error%)	0.326 (0.333) −2.235%

Figure 5.26: Comparing the best-fit hyper-Erlang distribution with extracted samples and theoretical distribution for service time for CUSTOMER1 at service centre S3 in case study 3

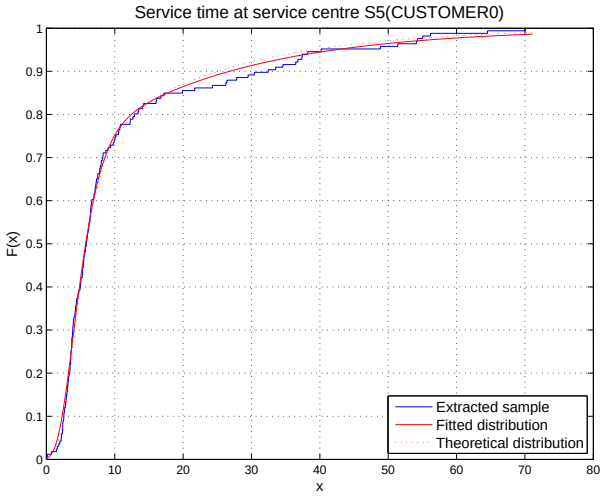


Fitting results on service time for CUSTOMER1 at S4	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.455, 0.902)$ $\mathbf{r} = (2, 4)$ $\boldsymbol{\omega} = (0.462, 0.538)$
K-S test	$p = 0.688$ K-S statistic: 0.040 H_0 not rejected
First moment (error%)	4.416 (4.800) −8.000%
c_v^2 (error%)	0.364 (0.476) −23.406%

Figure 5.27: Comparing the best-fit hyper-Erlang distribution with extracted samples and theoretical distribution for service time for CUSTOMER1 at service centre S4 in case study 3

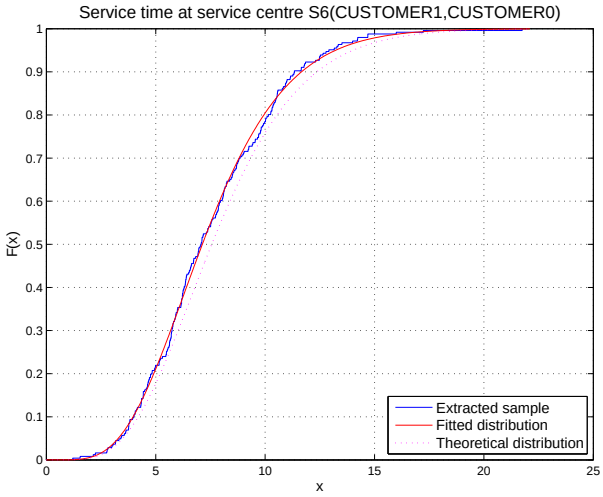
5.3.3 Inferred queueing network model and response time analysis

Figure 5.32a presents the structure of the queueing network model outputted by the data processing pipeline and the most likely service discipline employed at each service centre. Please note here that the routing of customer entities from the server node S4 depends on the queue lengths of nodes S6 and S7, rather than the connecting infinite server nodes IS5 and IS6 (as infinite server nodes do not have queue). As shown in Figure 5.32b, the service discipline that scores the highest at S3 is priority-based discipline with CUSTOMER0 enjoying higher priority



Fitting results on service time for CUSTOMER0 at S5	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.044, 0.761)$ $\mathbf{r} = (1, 4)$ $\boldsymbol{\omega} = (0.331, 0.669)$
K-S test	$p = 0.888$ K-S statistic: 0.044 H_0 not rejected
First moment (error%)	10.923 (10.633) 2.725%
c_v^2 (error%)	1.971 (1.701) 15.888%

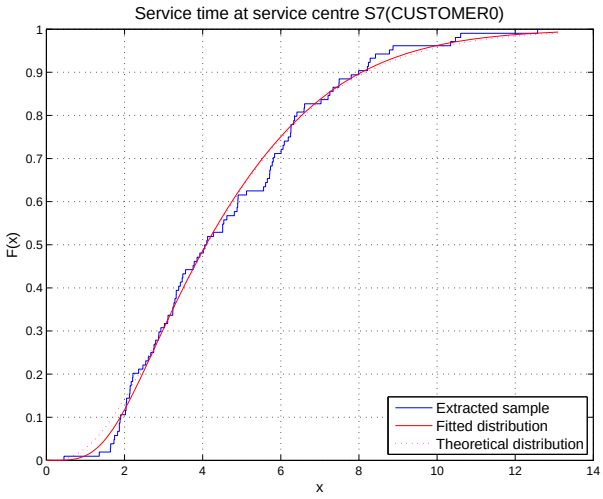
Figure 5.28: Comparing the best-fit hyper-Erlang distribution with extracted samples and theoretical distribution for service time for CUSTOMER0 at service centre S5 in case study 3



Fitting results on service time for CUSTOMER0 and CUSTOMER1 at S6	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.795)$ $\mathbf{r} = (6)$ $\boldsymbol{\omega} = (1)$
K-S test	$p =, 0.937$ K-S statistic: 0.033 H_0 not rejected
First moment (error%)	7.544 (8.000) −5.701%
c_v^2 (error%)	0.167 (0.167) 0.0%

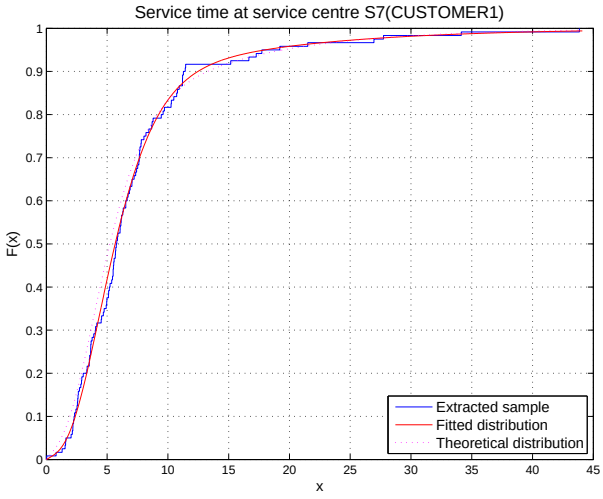
Figure 5.29: Comparing the best-fit hyper-Erlang distribution with extracted samples and theoretical distribution for service time for both CUSTOMER0 and CUSTOMER1 at service centre S6 in case study 3

than CUSTOMER1; FIFO is applied in the rest of the service centres. Note that service centres S4 and S5 only receive one type of customers (due to the class-based routing at S3) and thus the two priority-based service disciplines share the same scores as FIFO (because we have assumed that customers of the same class would be served based on FIFO discipline). Figure 5.33c displays the response time statistics and performance-related measurements from simulation based on the outputted queueing network model; most of these values again do not have large discrepancies from those generated by simulation based on true system settings.



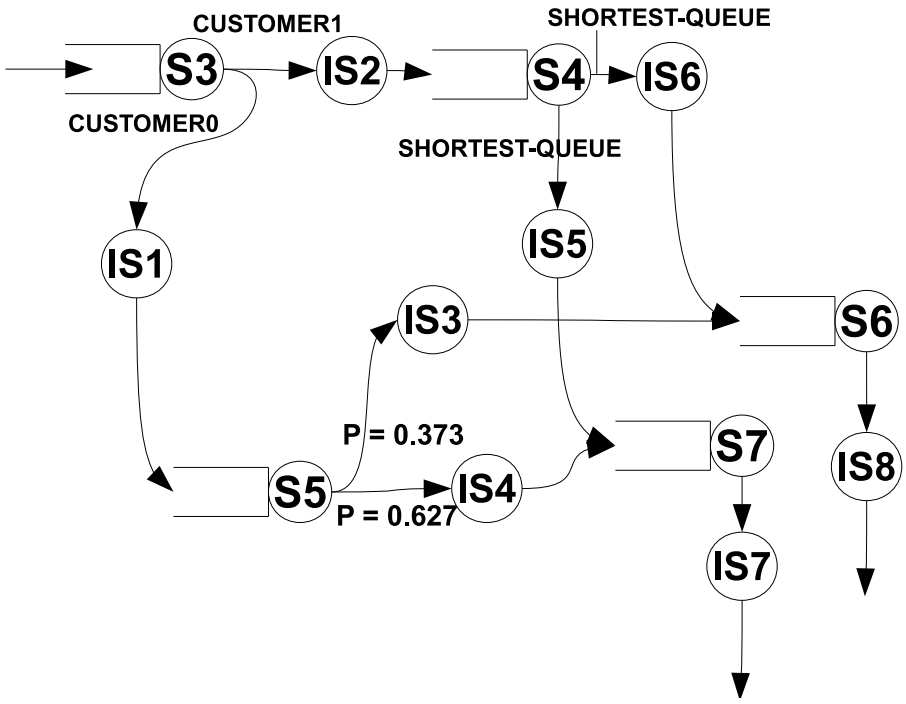
Fitting results on service time for CUSTOMER0 at S7	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.785, 2.692)$ $\mathbf{r} = (4, 7)$ $\boldsymbol{\omega} = (0.807, 0.193)$
K-S test	$p = 0.521$ K-S statistic: 0.078 H_0 not rejected
First moment (error%)	4.615 (4.615) −0.018%
c_v^2 (error%)	0.300 (0.333) −9.906%

Figure 5.30: Comparing the best-fit hyper-Erlang distribution with extracted samples and theoretical distribution for service time for CUSTOMER0 at service centre S7 in case study 3



Fitting results on service time for CUSTOMER1 at S7	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.080, 0.679)$ $\mathbf{r} = (1, 4)$ $\boldsymbol{\omega} = (0.206, 0.794)$
K-S test	$p = 0.787$ K-S statistic: 0.058 H_0 not rejected
First moment (error%)	7.238 (6.967) 3.891%
c_v^2 (error%)	0.872 (0.918) −5.038%

Figure 5.31: Comparing the best-fit hyper-Erlang distribution with extracted samples and theoretical distribution for service time for CUSTOMER1 at service centre S7 in case study 3

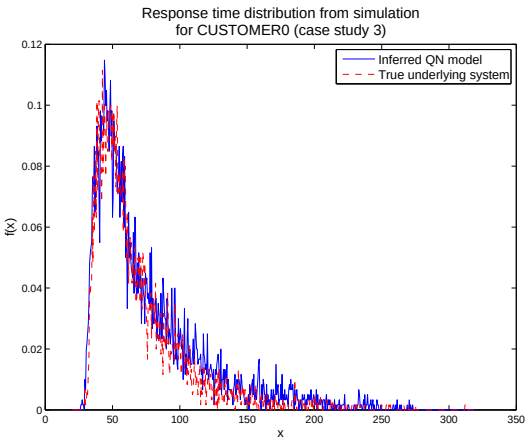


(a) Inferred queueing network structure

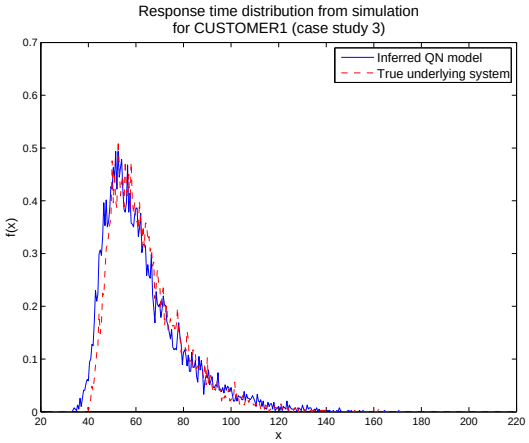
Server ID	S3	S4	S5	S6	S7
FIFO	114	**26**	**61**	**34**	**26**
LIFO	27	0	12	0	0
(CUSTOMER1,CUSTOMER0)	104	26	61	23	20
(CUSTOMER0,CUSTOMER1)	**167**	26	61	31	18

(b) Compare the scores of candidate service disciplines in predicting correctly the following customer to be served; the one with the highest score (in bold text) is selected.

Figure 5.32: The inferred queueing network and the inferred service time discipline at each service centre in case study 3



(a)



(b)

		System	S3	S4	S5
Aggregate	μ_{rt}	67.597 (65.879)	11.535 (10.518)	6.016 (7.165)	34.101 (25.926)
	cv^2_{rt}	0.182 (0.123)	0.964 (0.793)	0.496 (0.644)	1.352 (1.472)
CUSTOMER0	μ_{rt}	75.788 (68.869)	6.979 (6.934)	N/A	34.101 (25.926)
	cv^2_{rt}	0.300 (0.0.239)	0.165 (0.173)	N/A	1.352 (1.472)
CUSTOMER1	μ_{rt}	63.103 (64.404)	14.034 (12.286)	6.016 (7.165)	N/A
	cv^2_{rt}	0.072 (0.056)	1.569 (0.778)	0.496 (0.644)	N/A
	$\overline{n_q}$	N/A	1.13 (0.88)	0.17 (0.23)	1.32 (0.74)
	ρ	N/A	0.72 (0.68)	0.45 (0.48)	0.62 (0.53)

		S6	S7		
Aggregate	μ_{rt}	11.755 (12.009)	11.535 (10.518)		
	cv^2_{rt}	0.346 (0.314)	0.964 (0.793)		
CUSTOMER0	μ_{rt}	13.148 (13.277)	6.979 (6.934)		
	cv^2_{rt}	0.357 (0.315)	0.165 (0.173)		
CUSTOMER1	μ_{rt}	11.270 (11.635)	14.034 (12.286)		
	cv^2_{rt}	0.332 (0.307)	1.569 (0.778)		
	$\overline{n_q}$	0.34 (0.303)	1.13 (0.88)		
	ρ	0.63 (0.61)	0.72 (0.68)		

(c) Mean (μ_{rt}, in seconds) and the squared coefficient of variation (cv^2_{rt}) of the response time for each customer class at each service centre in case study 3. $\overline{n_q}$ and ρ represent the mean queue length and utilisation at each service centre, respectively

Figure 5.33: Response time analysis for case study 3

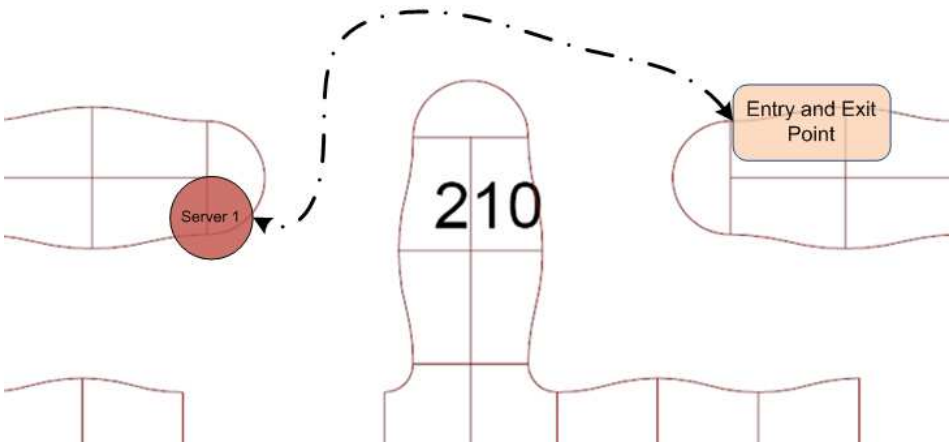


Figure 5.34: The deployment of the experiments for Case Study 1, 2 and 3

5.4 Case Study 4: Experiment Data

This section presents a case study where our proposed data processing methodology is evaluated using real location tracking data collected from a real-time location system. The location observations were captured during one of the four experiments conducted at preliminary stage of this research. For each experiment, we designed an environment whose customer flow resembles the one in a simple queueing system with known service time and customer interarrival time distributions. During the experiment volunteers transported tags between single-server service areas at times sampled from known distributions; the positions of tags were continuously recorded using a Ubisense Location Tracking System with four readers. Section 5.4.1 gives a more detailed description of how experiments were conducted and the settings of the experiment chosen for case study 4; followed by the analysis of the results are discussed in Section 5.4.2. For this case study, as we did not collect exact response times during the experiment, here we only compare the inferred customer interarrival time and service time distributions to the real experiment settings.

5.4.1 Experiment design and settings

Figure 5.34 illustrates the deployment of the experiments. The dash lines in the figure depict the designed customer flow of the experiment system. There is a single imaginary entry/exit

point, which is the location at which the volunteers (that is, customers) are considered arriving at or departing from the system. Once the volunteers "leave" the system, they return the tags they are holding at the entry/exit point. The main challenge in conducting the experiments is to direct volunteers for how long they should wait in front of a server, when they should leave a server's service area and which server they should move to, based on the designed customer flow structure and customer interarrival time and service time distributions. A simple alarm program is developed for this purpose. This program will pop out an alarm message when a certain amount of time is elapsed after the user presses the "start" button. This amount of time gap is generated according to the probability distribution specified by the user. At each server's location and the entry/exit point, there is a computer with this alarm program installed to instruct volunteers to leave their current location when the alarm message pops out. Figure 5.35 shows the historical paths of all tags during one of the experiments. The tags' locations are colour-coded; the red dots indicate when the tag was moving relatively fast, while the blue dots indicate when it remained (almost) static. From this, we can observe that tagged entities are clustered at low speeds around the locations of server entities and the entry/exit areas; as well as customer entities' moving paths between them.

Here we use data from one of the experiments to demonstrate how the data processing pipeline performs with real location tracking data. The system settings of the experiment is shown in Figure 5.36; it is essentially a simple single-queue server with exponentially-distributed service time and customer interarrival time. The experiment lasted around one hour (3357 seconds) and about 65000 rows of location data are collected; about 60 service time and customer interarrival time samples are extracted respectively.

5.4.2 Results

As demonstrated by Figure 5.37 and Figure 5.38, K-S test results show again that the best-fit hyper-Erlang distributions have good fitness to the extracted time delay samples. The fitting hyper-Erlang distribution for service time at `Server1` exhibits good agreement to the theoretical

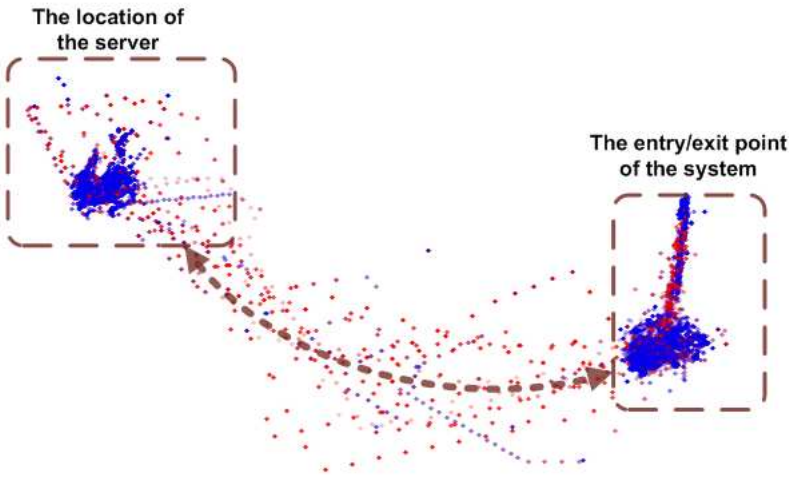


Figure 5.35: Visualisation of the raw location tracking data from one experiment showing moving tag positions (red dots) and static tag positions (blue dots).

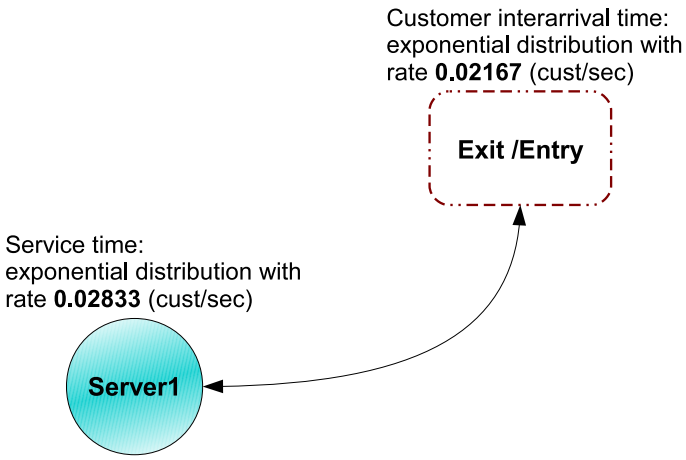
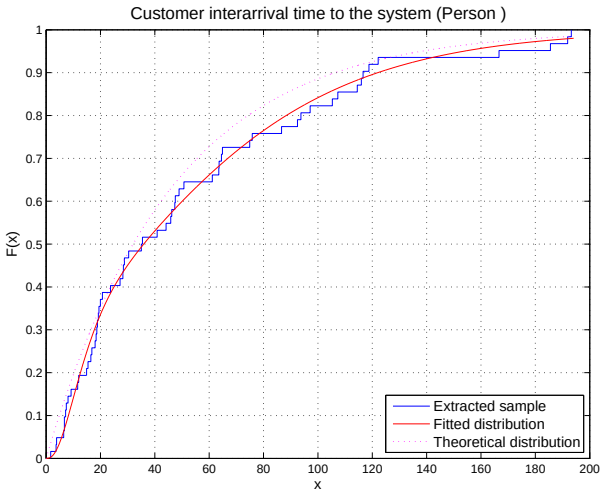


Figure 5.36: Experiment system settings

values of the first and second moments, with both less than 5.00% difference. In the case of fitting the customer interarrival time distribution, larger deviations from the theoretical values are observed; around 15.00% of difference in the first moment and 10.00% of difference in the second moment. Identifying a customer entity's arrival time to the system is more difficult under the experiment depolyment where the customer entities leave the tags at the exit/entry area for recycling after their departure from the system. Update rates in the Ubisense system can be varied in response to tag activity; when stationary, tags would get into a "sleep mode", not transitting signals, in order to conserve power [131]. As the recycled tags were kept idle at the entry/exit area, we have observed tags' locations were not updated frequently in the area and therefore might miss the location information when tags were attached to another arriving customer and started moving again.



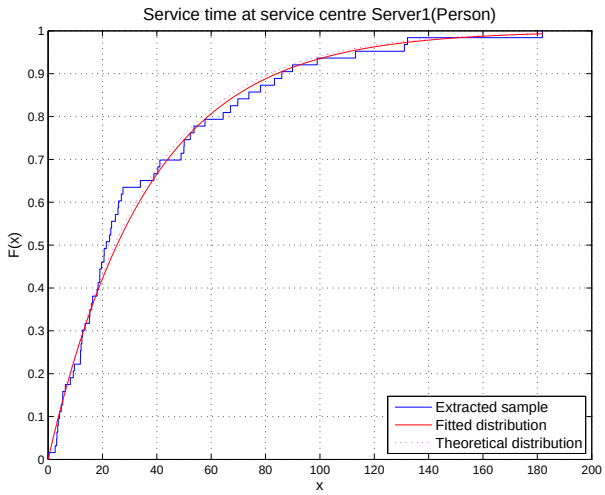
Fitting results on customer interarrival time	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.0277, 0.214)$ $\mathbf{r} = (2, 3)$ $\boldsymbol{\omega} = (0.670, 0.330)$
K-S test	$p = 0.654$ K-S statistic $= 0.090$ H_0 not rejected
First moment (error%)	53.018 (46.147) 14.89%
c_v^2 (error%)	0.896 (1.00) −10.36%

Figure 5.37: Comparing the best-fit hyper-Erlang distribution with extracted samples and theoretical distribution for customer interarrival time

5.5 Discussion

5.5.1 Time delay sample extraction and distribution fitting

The distribution fitting results from the case studies show that the adapted version of G-FIT is able to effectively find the hyper-Erlang distributions with good fitness to the extracted sam-



Fitting results on service time at `Server1`	
Best-fit *HErD*	$\boldsymbol{\lambda} = (0.0273754)$ $\mathbf{r} = (1)$ $\boldsymbol{\omega} = (1.0)$
K-S test	$p = 0.454$ K-S statistic $= 0.105$ H_0 not rejected
First moment (error%)	36.53 (35.298) 3.48%
c_v^2 (error%)	1.00 (1.00) 0.00%

Figure 5.38: Comparing the best-fit hyper-Erlang distribution with extracted samples and theoretical distribution for service time at service centre `Server1`

ples. In most of the cases, the K-S test results suggest that the extracted samples follow the best-fit hyper-Erlang distributions; the cumulative distributions of the best-fit hyper-Erlang curves also follow closely to the cumulative histogram from the extracted samples. There are few exceptions (for example, Figure 5.18), where the K-S test results reject the null hypothesis. However, in these cases the underlying distributions are deterministic and to approximate a deterministic distribution the total phase length of an Erlang distribution can reach up to infinity. In our adapted version of G-FIT, we set the upper limit of the state space to be 80 for samples showing small variability (i.e. $c_v^2 < 0.01$) and the best-fit hyper-Erlang distributions in these cases all reach 80. Thus, the limit set on the G-FIT's searching space might be the main cause for failing to find the hyper-Erlang distributions with good fitness to the extracted samples in the case of deterministic distributions. As the extracted samples in these cases have c_v^2 close to zero and their means lie within a small error range around the theoretical ones; instead of raising the state space limit of G-FIT to an even larger number, when outputting the final inferred queueing network model a deterministic distribution with its mean equal to the sample mean is used to replace the best-fit hyper-Erlang distribution.

Although in the majority of cases G-FIT did not find the best-fit hyper-Erlang distributions exactly the same as their corresponding theoretical distributions, when comparing their first

moments and c_v^2 we also have observed rather small discrepancies, with less than fifteen percentage of difference in most of cases. The CDF curves of the best-fit hyper-Erlang distributions in general follow well those of their corresponding theoretical distributions. One thing worth noting is that in the presence of unknown sources of data noise and errors, the data processing pipeline can still capture with moderate accuracy the statistic properties of the service and customer arrival processes. This is demonstrated by case study 4 with location data collected through experiment.

However we reckon several causes, listed as below, contributing to the best-fit hyper-Erlang distributions deviating from their corresponding theoretical distributions:

- Small sample size. Statistical inference methods are subject to bias when the sample size is small. In many cases where the best-fit hyper-Erlang distributions show larger discrepancies with the theoretical ones (for example, see Figure 5.11, Figure 5.15, Figure 5.16 Figure 5.19 and Figure 5.23), the extracted sample sizes tend to smaller between 100 to 200; comparing to 250 to 400 in those cases where the best-fit hyper-Erlang distributions match well with the corresponding theoretical distributions.

- Sample extraction bias. We have found it challenging to extract the service time samples due to the difficulties of estimating when the service to a customer entity starts and ends. At the moment we assume that the service time for a customer entity starts from the departure of the previous customer entity and ends at its own departure. However, in reality there is a time gap between the end of service and the departure of the customer when the customer entity is moving to leave the service centre; furthermore this time gap varies among different customer entities, depending on their locations inside the service centres when receiving service, as well as on their individual speeds. This uncertainty is difficult to avoid completely and hence contributes to the bias in extracted service time samples.

By applying Mann-Whitney U test on the extracted samples for different customer classes. we are able to differentiate cases where different customer classes have different service demands or different arrival behaviour (for example, see Figure 5.11, Figure 5.12, Figure 5.15, Figure 5.16); and those cases where different customer classes only enjoy different service priorities but not in terms of service times (see Figure 5.13 and Figure 5.29 for example). Although for the service times at server `S4` Mann-Whitney U test fails to tell that underlying distributions of the service demands from `CUSTOMER0` and `CUSTOMER1` are different, it is due to the fact that the extracted samples are not large enough to display the difference. The results shown above also have demonstrated the effectiveness and accuracy of G-FIT in finding the best-fit hyper-Erlang distribution given samples of sizes less than 400 (some of the extracted samples for an individual customer class are even as small as around 100). Using the squared coefficient of variation as guidance to narrow down the search space of G-FIT does not compromise the performance of G-FIT. The best-fit hyper-Erlang distributions in general matches well the first and second moments of the theoretical distributions; this mean that they serve as decent approximation to the theoretical distributions.

5.5.2 Customer flow routing

The results from the first three case studies has demonstrated that the data processing pipeline is able to learn the customer flow structure, specified by the routing scheme at each branching point, of the underlying system. However, as shown in Figure 5.8, Figure 5.20a and Figure 5.32a, the routing probabilities do not match exactly those designed for the case studies. Besides the probabilistic behaviour of the underlying system, another explanation is that branches that are traversed by only less than one percent of the customer entities are not included in the final customer flow structure. Most of time these "rarely-visited" customer paths result from infrequent location updates and small service times. We have found that the data processing pipeline has difficulty detecting whether a customer entity has received service inside a service centre if its service time is very small (e.g less than 0.1 seconds), especially when the customer entity's locations are not updated frequently enough. In this case, the customer entity would

be considered never having visited the service area and going directly to its next destination; thus an "incorrect" customer path would be discovered. This problem happens even more often if there are many tagged entities simultaneously existing in the system competing for update time slots of the location tracking systems. Hence, to avoid constructing a customer path which is not well-supported by the data collected, these less-traversed customer paths are removed when constructing the final queueing network.

5.5.3 Response time analysis

As demonstrated by the response time analysis in each case study with synthetic data, the key performance-based measurements, including first and second moments of customer response times, mean queue lengths and utilisation at each service centre, generated by simulation based on the inferred queueing network models mostly share similar scales in mean values and variations with those generated from simulation based on true system settings. Thus, the queueing network models outputted from our data processing pipeline are able to characterise the important performance-related features of the underlying system and thus serve as decent approximation to the underlying system, especially if the inference is supported by sufficient data samples.

5.5.4 Computation Time

In this section, we take the case study 4 (with experiment data) as example to discuss the computation complexity required by different stages of the data processing pipeline. Please note here that all the case studies presented above were ran on a personal computer with Intel®Core i5-2435M CPU 2.4GHz running 32-bit operating system. There are around 65000 data points in the data set.

We have observed that the second stage, especially the part involving DBSCAN clustering, requires substantial amount of computation time, comparing to other stages along the data

processing pipeline. To process the location tracking data collected from the experiment, DBSCAN clustering and inferring service centre locations takes 389689 milliseconds; while mining the customer flow structure and extracting time delay samples at Stage 3 takes around 95215 milliseconds and the service time distribution fitting and constructing the final inferred queueing network model at Stage 4 takes only 4248 milliseconds. That is to say, DBSCAN clustering contributes almost 80% of the computation time along the data processing pipeline.

The longer computation time at Stage 2 can be attributed to multiple neighbourhood queries required for each data point in the DBSCAN clustering algorithm. The computation time of DBSCAN is thus $O(N * r(k))$, where $r(k)$ denotes the computation complexity of k-th nearest neighbor retrieval. Theoretically, using tree-shaped spatial index structures such as KD tree or R* tree can enhance the efficiency of data access in a spatial database, cutting down $r(k)$ to $O(\log N)$ (compared to $O(N^2)$ with a linear index structure). However, such complexity improvement cannot be achieved in cases where datasets are simply too large or data distributions are particularly dense in certain regions [16]. The latter case is expected in our data sets, as dense regions (that is, clusters) correspond to approximate locations of service centres and potential bottlenecks. There have been research works existing in modifying or parallelising the original DBSCAN algorithm to improve the computation complexity [16, 95]. However, we argue that for the data processing pipeline to be applied for rapid decision support in real-life systems it is necessary to adopt on-line clustering approach in the next phase of this research; more detailed discussion on this respect will be presented in Chapter 6.

5.6 Summary

This chapter presented four case studies to evaluate the performance and accuracy of the developed data processing pipeline; the first three case studies are based on synthetic location data generated using LocTrackJINQS and the last one is based on real location data collected from one of the experiments conducted in the preliminary stage of the research. Results

from all four case studies demonstrated that the developed data processing pipeline is able to derive queueing network performance models that reflect the true probabilistic features of service and arrival processes for a range of multiclass customer-processing systems. The K-S test results have shown that the adapted version of G-FIT can effectively identify the hyper-Erlang distributions with good fitness to the extracted time delay samples; their first and second moments in most of cases agree with those of theoretical distributions. Through simulation, we also show that the inferred queueing network approximates well the true underlying system in terms of response time distributions and common performance measurements such as mean queue length and utilisation of each service centre. Most notably, when the data processing pipeline is evaluated against real location tracking data in case study 4, the data processing pipeline still could capture, with decent accuracy the behaviour of the service and customer arrival processes, despite data issues such as noise and infrequent updates.

Chapter 6

Conclusion

6.1 Summary of Thesis Achievements

6.1.1 Develop a methodology for inferring a queueing network model from high-precision location tracking data

This thesis has presented a four-stage data processing pipeline for deriving queueing network models from high-precision location tracking data. The inferred queueing network model is specified in terms of routing policies, interarrival time distributions, service time distributions and service disciplines. The first stage takes in the raw location tracking data and performs basic data cleaning and interpolation. The second stage uses DBSCAN clustering algorithm to approximate the locations and sizes of the service centres and other potential bottlenecks existing in the system. It gradually refines the low-level location tracking data into high-level "event logs" describing the interactions among customer and server entities based on their proximity to each other. The third stage mines the tagged customers' movement paths and constructs the basic structure of the customer flow as well as the routing policy associated with each branching point. This stage also extracts samples of service times at each service centre, customer interarrival times at the system and travelling times between each pair of connecting service centres. At the fourth stage we use an adapted version of G-FIT tool [130] to find the

hyper-Erlang distributions that can best characterise the extracted time delay samples. By observing the customer entities' incoming orders and departure orders we infer the most likely service discipline applied at each service centre. The final output of the pipeline is a queueing network model constructed in accordance to the inferred customer flow structure, routing policies, service discipline at each service centre, as well as distributions that characterise different time delays involved in the system (including service times, customer interarrival times and customer travelling times).

6.1.2 Implementation of LocTrackJINQS

This research also presented LocTrackJINQS, a location-aware simulation tool developed based on the discrete-event simulation library for queueing networks, JINQS. It inherits many characteristics from JINQS, such as extensibility and simplicity for simulation construction, but incorporates low-level location information, including physical locations and movement speeds of tracked entities in the system. To facilitate general location-based research and applications, LocTrackJINQS provides functionalities for generating synthetic location tracking data from a user-defined queueing system. LocTrackJINQS also provides a graphical user interface to support visual construction of queueing network simulation.

6.1.3 Evaluate the accuracy of the developed data processing pipeline

This research carried out four case studies to evaluate the performance and accuracy of the developed data processing pipeline in deriving queueing network models for a range of multi-class customer-processing systems. The developed data processing pipeline has performed with moderate success in extracting queueing network performance models that reflect the true probabilistic features of service and arrival processes in the underlying system. This is true even when the data processing pipeline is applied to real location tracking data containing many noise and errors. The K-S test results in the case studies have shown that the adapted version of G-FIT can effectively identify the hyper-Erlang distributions with good fitness to the

extracted time delay samples; their first and second moments in most of cases match well with those of theoretical distributions. Furthermore, through simulation-based evaluation, we show that the inferred queueing network approximates well the true underlying system in terms of response time distributions and key performance measurements such as mean queue length and utilisation at each service centre.

6.2 Applications

The most direct applications of our work should be in performance analysis of small-scale specialised systems where the entities involved can be tagged with sensor tags. Systems whose underlying processes are complex, highly volatile and difficult to understand with only visual observations or manually-collected data will see most benefits of adopting our methodology for system performance evaluation and planning. Such systems include healthcare systems, sport centres, shopping malls, museums, restaurants, or special event venues. The methodology can also be applied to various aspects in enterprise environment, such as supply chain, manufacturing systems or warehouse, to identify the wastes, inefficiencies, insufficiencies or abnormalities in the system.

With the prevalence of sensor-enabled vehicles and personal mobile devices, the "Internet of things" has expanded beyond specific domains to large-scale, community-level environments, such as urban space. Massive, multi-modal and real-time observational data collected through these devices provide opportunities to study large-scale human movements and interactions among human, objects and the surroundings. Our data processing pipeline should tap such data sources to address many previously challenging issues in terms of large-system management and planning. For example, we can combine location data collected from multiple sources such as GPS data from private car users, sensor records collected through sensor-enabled public bicycles or smart card readers in the public transport system and infer a performance model that characterises the utilisation of different transportation facilities by urban dwellers or identify

the "hot spots" where congestion takes place in real time.

6.3 Future work

Our work has demonstrated the possibilities of inferring simple queueing systems with well-defined structures. Here we propose several issues we should address in the following phases: extend the current data processing pipeline to incorporate systems with more complex features; extend the functionalities of LocTrackJINQS to support more real-life features; work toward an online methodology where the parameterisations of the inferred queueing network models would "evolve" with latest incoming data streams under memory constraints. Ultimately, we hope to experiment with the application of such techniques in the context of real-life systems.

6.3.1 Extend the variety of features that can be inferred

At the present stage, our data processing pipeline is only capable of inferring queuing models for a limited variety of customer processing systems; we assume single-server service with no queue capacity limits and with service disciplines including only FIFO, LIFO and priority-based and with no service preemption. The routing of the customer flow at each branching point is also restricted to follow only one of the three routing policies: probabilistic routing, class-based routing and shortest-queue routing.

In the near future, we plan to further extend the data processing pipeline to incorporate more features, including:

- Multiple-server queues.
- Multiple servers sharing an external queueing area. This feature is seen in many real-life systems in our daily life; for example, the common patient waiting room outside different treatment rooms in a hospital; or the waiting area at the city council for people with different requests.

- Queues with limited capacity. The incoming customers will be turned away when the queue has reached its maximum capacity.

- More types of service disciplines. Include service preemption and longer/short service time first.

- Hybrid routing policies. Routing of customer entities may depend on more than one routing policies; for example, at a branching point with three branches, thirty percent of the customers would take the first branch, while the rest seventy percent would choose either of the other two branches based on shortest-queue policy (in this case the routing scheme is a combination of probabilistic routing and shortest-queue routing).

6.3.2 Extensions on LocTrackJINQS

Possible future work in extending the functionalities of LocTrackJINQS includes multi-dimensional customer classifications (e.g. a customer can belong to more than one classes based on multiple class definitions), as well as customer speed settings (e.g. customers would slow down when approaching a service area). We also plan to relax some of the current restrictions. For example, instead of having fixed locations, server entities can move around providing service to customers (e.g. nurses visiting hospitalised patients); the queueing area can be of arbitrary shapes (e.g. linear, rectangle) other than being circular.

6.3.3 Online data processing pipeline

Currently our data processing pipeline is based on an offline approach. This means that the inferred queuing network model can only describe the underlying system according to the historical data set and not able to reflect time-varying characteristics unless the pipeline is run again on the latest data set. It would be a major drawback as the inferred queueing network will be slow to react to sudden changes in the monitored environment, while the major values of location tracking data lie in its ability to reflect on real-time basis the latest status of any

tracked entity.

The following step would be to adopt an online and more flexible clustering algorithm for discovering service centres' locations. The DBSCAN clustering algorithm, adopted as part of the current version of data processing pipeline, is a static offline one. It assumes the definition of a "dense" area, in terms of the parameters Eps and $MinPts$, does not change with time; to redefine it one must apply the algorithm again on the new dataset. In reality, the "dense" areas, usually where service centres and possible system bottlenecks are located, might grow, fade or displace themselves with time. The other drawback of DBSCAN clustering algorithm is that it assumes the cluster structures across a system are uniform and can be characterised globally using the same parameters Eps and $MinPts$; while it is often found that in a single system some dense areas might appear "thinner" than the others [14]. Furthermore, an offline clustering algorithm is usually memory-heavy, requiring loading all the data at once; this is especially not feasible with real-time location tracking data, which comes in large amount in short period of time. Existing online density-based clustering algorithms, such as CluStream [2] and DenStream [42], offer memory-light solutions that work with streaming data. Both methods keep a set of "micro-clusters" with a simple summary of their statistics (normally including the data density and first/second moments of distances in each dimension to the origin); these micro-clusters are taken as potential "seeds" forming parts of bigger clusters. The locations and sizes of these "micro-clusters" can evolve with new incoming data; the clustering can be applied directly on these micro-clusters periodically to discover the latest cluster structure in the system.

Another issue in this aspect is that the inference process for characterising those time delays involved in the system are also static. Currently we assume a "best-fit" model characterising the time delay involved in the system; however, the chosen model might have the best fitness to the data collected thus far, while other candidate models might better approximate the "true" underlying distribution in reality. Instead of conducting distribution fitting frequently, we can maintain a set of possible models that are well-supported by the data already collected. We

weight these candidate models based on their relative "distance" from the reality according to their respective calculated AIC numbers; the output model is the weighted average of the outputs from the members in the candidate model set, rather than from only the single model chosen as the best. With new data coming in, we periodically conduct goodness-of-fit tests to check the fitness of the candidate model set to the most recent dataset. Only when above certain percentage of the members in the candidate set fail the goodness-of-fit test do we re-conduct the distribution fitting process to find a new set of candidate model. This approach allows us to avoid the bias created by selecting a single model that might fit well a subset of the data but not the whole reality; it also allow us to gradually update our model outputs responding to the latest data.

6.3.4 Investigate different filtering and smoothing techniques for location tracking data

Currently we use simple methods such as location interpolation, window-based voting method and look-ahead action to handle errors and missing found in the data. However, these methods might not be adequate when coping with tracking data collected from physical sensoring devices, which are notoriously noisy, containing much higher percentages of missing and unreliable location readings. Thus, we should develop a more systematic approach to filter and smooth out raw location tracking data. For example, we can filter out the noise based on the velocity trace of the tracked entities. If a tag's location update has large deviations from the other updates within a small time-window, thus creating a velocity jump, it's very likely that this data point is an erroneous reading and can be ignored. Previous research efforts in preprocessing noisy sensor data (especially RFID and GPS data) should be further investigated. For example, [83, 82] proposed a five-stage pipeline framework for sensor data cleaning, based on the assumption that sensor data are largely homogeneous (i.e. without location readings contradicting against each other) if it is collected within a small time frame and from the sensors located in proximity; [84] later developed an adaptive model for cleaning RFID data, which can automatically and continuously adjust smoothing window sizes based on the observed location

readings; [43] presented a data-driven framework for smoothing GPS data, which is generalization of the classical weighted moving average technique, taking into account both the object's trajectory and its speed trend.

6.3.5 Support more complex customer flow structures

The current workflow mining algorithm simply summarises the paths followed by the customer entities in the system. The generated workflow cannot express many stochastic features that are common in real-life systems. For example, the probability of a customer entity taking a certain path might be based on its previous trajectory in the system. It would also be an interesting topic to identify the frequently-followed paths in the system, the exceptions to the frequently-followed paths and the conditional probabilities (e.g. on time spent on previous locations in the path) of these exceptions.

6.3.6 Applications to real-life systems

We hope in the future to carry out experiments with real location tracking data collected from a real-life system where the underlying processes are unknown. We are particularly interested in an A&E unit of a hospital, where the performance requirements are high and the manually-collected data are not sufficient for constructing a performance model with good approximation to the reality. This will help us identify the practical difficulties or problems we overlook previously, especially in the aspects of raw location data cleaning, time-varying characteristics and complex customer routing structures, when applying our developed methodology to a real-life system and make further improvements or adjustments accordingly.

Bibliography

[1] Enterprise dynamics. `http://www.incontrolsim.com/`.

[2] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *VLDB '03: Proc. the 29th international conference on Very large data bases*, pages 81–92. VLDB Endowment, 2003.

[3] G. Agrawal. A queuing model for health centre. In *ICEIE '10: Proc. International Conference On Electronics and Information Engineering*, volume 1, pages 542–547, 2010.

[4] R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In *EDBT '98: Proc. 6th International Conference on Extending Database Technology*, pages 469–483, London, UK, 1998. Springer-Verlag.

[5] A. Agresti. *An Introduction to Categorical Data Analysis*. Wiley-Interscience, 1st edition, Feb. 1996.

[6] A. Agresti and B. Finlay. *Statistical Methods for the Social Sciences*. Prentice Hall, 4 edition, Jan. 2008.

[7] H. Akaike. Information theory and an extension of the maximum likelihood principle. In P. B. N. and C. F., editors, *Proc. of the 2nd International Symposium on Information Theory*, pages 267–281, 1973.

[8] S. L. Albin, J. Barrett, D. Ito, and J. E. Mueller. A queueing network analysis of a health center. *Queueing Systems*, 7:51–61, 1990.

[9] Ambiplex. Passive infrared localization technologies, 2011. `http://www.ambiplex.com/`.

[10] N. Anastasiou. *Automated Construction of Petri Net Performance Models from High-Precision Location Tracking Data*. PhD thesis, Imperial College London, 2013.

[11] N. Anastasiou, T.-C. Horng, and W. Knottenbelt. Deriving generalised stochastic Petri net performance models from high-precision location tracking data. In *VALUETOOLS '11: Proc. the 5th International ICST Conference on Performance Evaluation Methodologies and Tools*, pages 91–100, 2011.

[12] N. Anastasiou, W. Knottenbelt, and A. Marin. Automatic synchronisation detection in Petri net performance models derived from location tracking data. In *Computer Performance Engineering*, volume 6977 of *Lecture Notes in Computer Science*, pages 29–41. Springer Berlin/Heidelberg, 2011.

[13] R. Angeles. RFID technologies: Supply-chain applications and implementation issues. *Information Systems Management*, 22:51–65, 2005.

[14] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: ordering points to identify the clustering structure. In *SIGMOD '99: Proc. the 1999 ACM SIGMOD international conference on Management of Data*, pages 49–60, New York, NY, USA, 1999. ACM.

[15] D. Ardagna, M. Tanelli, M. Lovera, and L. Zhang. Black-box performance models for virtualized web service applications. In *WOSP/SIPEW '10: Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, pages 153–164, New York, NY, USA, 2010. ACM.

[16] D. Arlia and M. Coppola. Experiments in parallel clustering with DBSCAN. In *Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing*, pages 326–331, London, UK, UK, 2001. Springer-Verlag.

[17] M. Arlitt and C. Williamson. Internet Web servers: workload characterization and performance implications. *IEEE/ACM Transactions on Networking*, 5(5):631–645, 1997.

[18] K. Ashton. That "Internet of Things" thing, June 2009. `http://www.rfidjournal.com/article/view/4986`.

[19] S. Asmussen, O. Nerman, and M. Olsson. Fitting phase-type distributions via the EM algorithm. *Scandinavian Journal of Statistics*, 23(4):419–441, Dec. 1996.

[20] AT&T. The bat system, 2008. `http://www.cl.cam.ac.uk/research/dtg/attarchive/bat/`.

[21] S. Au-Yeung. *Response times in healthcare systems*. PhD thesis, Imperial College London, Jan. 2008.

[22] S. Au-Yeung, P. Harrison, and W. Knottenbelt. A queueing network model of patient flow in an accident and emergency department. In *20th Annual European and Simulation Modelling Conference*, pages 60–67, Sept. 2006.

[23] J. Barnes, C. Rizos, M. Kanli, and A. Pahwa. A positioning technology for classically difficult GNSS environments from Locata. In *IEEE/ION PLANS*, pages 715–721, 2006.

[24] J. Barnes, C. Rizos, J. Wang, D. Small, and G. Voigt. High precision indoor and outdoor positioning using LocataNet. *Proc. International Symposium on GPS/GNSS*, 2003.

[25] J. Barnes, C. Rizos, J. Wang, D. Small, G. Voigt, and N. Gambale. Locata: A new positioning technology for high precision indoor and outdoor positioning, 2003.

[26] J. Barnes, C. Rizos, J. Wang, D. Small, G. Voigt, and N. Gambale. Locata: The positioning technology of the future, 2003.

[27] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *Journal of the ACM*, 22(2):248–260, Apr. 1975.

[28] F. Benevenuto, C. Fernandes, M. Santos, V. Almeida, J. Almeida, G. J. Janakiraman, and J. R. Santos. Performance models for virtualized applications. In *ISPA'06: Proc. the 2006 International Conference on Frontiers of High Performance Computing and Networking*, pages 427–439, Berlin, Heidelberg, 2006. Springer-Verlag.

[29] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, Sept. 1975.

[30] D. J. Bertsimas and L. D. Servi. Deducing queueing from transactional data: The queue inference engine, revisited. *Operations Research*, 40(Supplement-2):S217–S228, 1992.

[31] J. Bilmes. A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical report, University of California, Berkeley, 1997.

[32] G. R. Bitran and S. Dasu. A review of open queueing network models of manufacturing systems. *Queueing Systems*, 12:95–134, 1992.

[33] G. R. Bitran and R. Morabito. Open queueing networks: Optimization and performance evaluation models for discrete manufacturing systems. In *Production and Operations Management*, pages 163–193, 1996.

[34] A. Bobbio, A. Horváth, and M. Telek. Matching three moments with minimal acyclic phase type distributions. *Stochastic Models*, 21(2-3):303–326, 2005.

[35] S. Borman. The expectation maximization algorithm: A short tutorial. Technical report, 2004.

[36] S. Brenner, Z. Zeng, Y. Liu, J. Wang, J. Li, and P. K. Howard. Modeling and analysis of the emergency department at University of Kentucky Chandler hospital using simulations. *Journal of Emergency Nursing*, 36(4):303 – 310, 2010.

[37] P. Buchholz and J. Kriege. A heuristic approach for fitting maps to moments and joint moments. In *QEST '09: International Conference on the Quantitative Evaluation of Systems*, pages 53–62, 2009.

[38] K. P. Burnham and D. R. Anderson. *Model selection and multimodel inference: A practical information-theoretic approach*. Springer, jul 2002.

[39] K. P. Burnham and D. R. Anderson. Multimodel inference: understanding AIC and BIC in model selection. *Sociological Methods & Research*, 33(2):261–304, nov 2004.

[40] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn. The rise of people-centric sensing. *IEEE Internet Computing*, 12(4):12–21, July 2008.

[41] E. Campo and C. M. Detecting abnormal behaviour by real-time monitoring of patients. In *Proc. AAAI-02 Workshop on Automation as Caregiver*, pages 24–30, 2002.

[42] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *SIAM Conference on Data Mining*, pages 328–339, 2006.

[43] F. Chazal, D. Chen, L. Guibas, X. Jiang, and C. Sommer. Data-driven trajectory smoothing. In *GIS '11: Proc. of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 251–260, New York, NY, USA, 2011. ACM.

[44] C. I. Chen, C. Y. Liu, Y. C. Lia, C. C. Chao, C. T. Liu, C. F. Chen, and C. F. Kuand. Pervasive observation medicine: the application of RFID to improve patient safety in observation unit of hospital emergency department. *Studies in Health Technology and Informatics*, 116:311–315, 2005.

[45] T. J. Coats and S. Michalis. Mathematical modelling of patient flow through an accident and emergency department. *Emergency Medicine Journal*, 18(3):190–192, 2001.

[46] L. Connelly and A. Bair. Discrete event simulation of emergency department activity: a platform for system-level operations research. *Academic Emergency Medicine*, 11(11):1177–85, 2004.

[47] J. E. Cook and A. Wolf. Automating process discovery through event-data analysis. In *ICSE '95: Proc. 17th International Conference on Software Engineering*, pages 73–82, Seattle, Washington, U.S.A, 1995.

[48] J. E. Cook and A. Wolf. Discovering models of software processes from event-based data. Research Report Technical Report CU-CS-819-96, Computer Science Dept., Univ. of Colorado, 1996.

[49] M. J. CôTé and W. E. Stein. A stochastic model for a visit to the doctor's office. *Mathematical and Computer Modelling*, 45(3-4):309–323, Feb. 2007.

[50] Cricket. The cricket indoor location system. `http://cricket.csail.mit.edu/`.

[51] M. Crovella and A. Bestavros. Self-similarity in world wide web traffic: evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, Dec 1997.

[52] Z. Da, F. Xia, Z. Yang, L. Yao, and W. Zhao. Localization technologies for indoor human tracking. *The 5th International Conference on Future Information Technology*, abs/1003.1833, 2010.

[53] S. de Treville and A. van Ackere. Equipping students to reduce lead times: The role of queuing-theory-based modeling. *Interfaces*, 36(2):165–173, Mar. 2006.

[54] G. Deak, K. Curran, and J. Condell. Review: A survey of active and passive indoor localisation systems. *Computer Communications*, 35(16):1939–1954, Sept. 2012.

[55] A. Di Febbraro, D. Giglio, and N. Sacco. Urban traffic control structure based on hybrid Petri nets. *IEEE Transactions on Intelligent Transportation Systems*, 5(4):224 – 237, dec. 2004.

[56] A. Di Febbraro and N. Sacco. On modelling urban transportation networks via hybrid Petri nets. *Control Engineering Practice*, 12(10):1225 – 1239, 2004.

[57] Ekahau. `http://www.ekahau.com/`.

[58] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[59] M. Fackrell. Modelling healthcare systems with phase-type distributions. *Health Care Management Science*, 12(1):11–26, March 2009.

[60] A. Feldmann and W. Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. *Performance Evaluation*, 31(3V4):245–279, 1998.

[61] T. Field. JINQS: An Extensible Library for Simulating Multiclass Queueing Networks V1.0. http://www.doc.ic.ac.uk/ ajf/Research/manual.pdf, 2006.

[62] Foursquare. https://foursquare.com/.

[63] K. Freeman. 7 city parking apps to save you time, money and gas, Mar. 2012. http://www.cl.cam.ac.uk/research/dtg/attarchive/bat/.

[64] J. C. Gardiner. Modeling heavy-tailed distributions in healthcare utilization by parametric and bayesian methods. Technical report, Department of Epidemiology and Biostatistics, Michigan State University, 2012.

[65] H. Gonzalez, J. Han, and X. Li. Flowcube: constructing RFID flowcubes for multi-dimensional analysis of commodity flows. In *VLDB '06: Proc. 32nd International Conference on Very Large Databases*, pages 834–845. VLDB Endowment, 2006.

[66] H. Gonzalez, J. Han, and X. Li. Mining compressed commodity workflows from massive RFID data sets. In *Proc. 15th ACM International Conference on Information and Knowledge Management*, pages 162–171, New York, NY, USA, 2006. ACM.

[67] H. Gonzalez, J. Han, X. Li, and D. Klabjan. Warehousing and analyzing massive RFID data sets. In *ICDE '06: Proc. 22nd International Conference on Data Engineering*, page 83, Washington, DC, USA, 2006. IEEE Computer Society.

[68] W. J. Gordon and G. F. Newell. Closed queueing systems with exponential servers. *Operation Research*, 15(2):254–265, 1967.

[69] W. J. Gordon and G. F. Newell. Closed queueing systems with restricted length queues. *Operation Research*, 15(2):266–277, 1967.

[70] Y. Gu, A. Lo, and I. Niemegeers. A survey of indoor positioning systems for wireless personal networks. *Communications Surveys Tutorials, IEEE*, 11(1):13–32, 2009.

[71] B. Guo, D. Zhang, and Z. W. Living with Internet of things: The emergence of embedded intelligence. In *ITHINGSCPSCOM '11: Proc. 2011 International Conference on Internet*

of Things and 4th International Conference on Cyber, Physical and Social Computing, pages 297–304, Washington, DC, USA, 2011. IEEE Computer Society.

[72] V. Guralnik and K. Z. Haigh. Learning models of human behaviour with sequential patterns. In *Proc. AAAI-02 Workshop on Automation as Caregiver*, pages 24–30, 2002.

[73] T. Hansen, J. Bardram, and M. Soegaard. Moving out of the lab: Deploying pervasive technologies in a hospital. *IEEE Pervasive Computing*, 5(3):24 –31, 2006.

[74] G. W. Harrison, A. Shafer, and M. Mackay. Modelling variability in hospital bed occupancy. *Health Care Management Science*, 8(4):325 – 334, 2005.

[75] T.-C. Horng, N. Anastasiou, T. Field, and W. Knottenbelt. LocTrackJINQS: An extensible location-aware simulation tool for multiclass queueing networks. *Electron. Notes Theor. Comput. Sci.*, 275:93–104, Sept. 2011.

[76] T.-C. Horng, N. Dingle, A. Jackson, and W. Knottenbelt. Towards the automated inference of queueing network models from high-precision location tracking data. In *ECMS '09: Proc. 23rd European Conference on Modelling and Simulation*, pages 664–674, May 2009.

[77] A. Horváth and M. Telek. Approximating heavy tailed behaviour with phase type distributions. *3rd International Conference on Matrix-Analitic Methods in Stochastic models*, pages 191–214, 2000.

[78] A. Horváth and M. Telek. Phfit: A general phase-type fitting tool. In *TOOLS '02: Proc. 12th International Conference on Computer Performance Evaluation, Modelling Techniques and Tools*, pages 82–91. Springer-Verlag, 2002.

[79] C. M. Hurvich and C.-L. Tsai. Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307, June 1989.

[80] J. Jackman and E. Johnson. The role of queueing network models in performance evaluation of manufacturing systems. *The Journal of the Operational Research Society*, 44(8):797–807, 1993.

[81] J. R. Jackson. Jobshop-like queueing systems. *Management Science*, 10(1):131–142, October 1963.

[82] S. Jeffery, G. Alonso, M. Franklin, W. Hong, and J. Widom. Declarative support for sensor data cleaning. *Lecture Notes in Computer Science*, 2006.

[83] S. Jeffery, G. Alonso, M. Franklin, W. Hong, and J. Widom. A pipelined framework for online cleaning of sensor data streams. In *ICDE '06: Proc. 22nd International Conference on Data Engineering*, pages 140–140, April 2006.

[84] S. Jeffery, M. Garofalakis, and M. Franklin. Adaptive cleaning for RFID data streams. In *VLDB '06: Proc. 32nd International Conference on Very large Data Bases*, pages 163–174. VLDB Endowment, 2006.

[85] L. Jingshan and P. Howard. Modeling and analysis of hospital emergency department: An analytical framework and problem formulation. In *CASE '10: Proc. IEEE Conference on Automation Science and Engineering*, pages 897 –902, 2010.

[86] M. A. Johnson and M. R. Taaffe. Matching moments to phase distributions: Mixtures of Erlang distributions of common order. *Stochastic Models*, 5:711–743, 1989.

[87] M. Kärkkäinen. Increasing efficiency in the supply chain for short shelf life goods using RFID tagging. *International Journal of Retail and Distribution Management*, 31(10):529 – 536, 2003.

[88] R. E. A. Khayari, R. Sadre, and B. R. Haverkort. Fitting World-Wide Web request traces with the EM-algorithm. *Performance Evaluation*, 52(2-3):175–191, 2003.

[89] S. J. Kim, S. K. Yoo, H. O. Kim, H. S. Bae, J. J. Park, K. J. Seo, and B. C. Chang. Smart blood bag management system in a hospital environment. In *Proc. 11th International Conference on Personal Wireless Communications*, pages 506–517, 2006.

[90] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:49–86, 1951.

[91] R. C. Larson. The queue inference engine: deducing queue statistics from transactional data. *Management Science*, 36(5):586–601, 1990.

[92] Y. Lee, F. Cheng, and Y. Leung. Exploring the impact of RFID on supply chain dynamics. In *Proc. the 2004 Winter Simulation Conference*, volume 2, pages 1145–1152, Dec. 2004.

[93] W. E. Leland, W. Willinger, M. S. Taqqu, and D. V. Wilson. On the self-similar nature of Ethernet traffic. *ACM SIGCOMM Computer Communication Review*, 25(1):202–213, Jan. 1995.

[94] J. Li. Modeling and analysis of hospital emergency department: An analytical framework and problem formulation. In *2010 IEEE Conference on Automation Science and Engineering*, pages 897 – 902, 2010.

[95] B. Liu. A fast density-based clustering algorithm for large databases. In *2006 International Conference on Machine Learning and Cybernetics*, pages 996–1000, 2006.

[96] H. Liu, H. Darabi, P. Banerjee, and J. Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(6):1067–1080, Nov. 2007.

[97] London City Airport. The "Internet of things" will be trialled at London City Airport. `http://www.londoncityairport.com/News/ReadArticle/93`.

[98] S. M. Mahmoud, A. Lotfi, and C. Langensiepen. Abnormal behaviours identification for an elder's life activities using dissimilarity measurements. In *PETRA '11: Proc. of the 4th International Conference on Pervasive Technologies Related to Assistive Environments*, pages 25:1–25:5, New York, NY, USA, 2011. ACM.

[99] G. Mao, B. Fidan, and B. D. O. Anderson. Wireless sensor network localization techniques. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 51(10):2529–2553, July 2007.

[100] P. L. Markt and M. H. Mayer. WITNESS simulation software: a flexible suite of simulation tools. In *WSC '97: Proc. 29th Winter Simulation Conference*, pages 711–717, Washington, DC, USA, 1997.

[101] A. Marshall and L. McCrink. Discrete conditional phase-type model for patient waiting time with a logistic regression component to predict patient admission to hospital. In *CBMS 09': 22nd IEEE International Symposium on Computer-Based Medical Systems*, pages 1–6, 2009.

[102] R. Mautz. Overview of current indoor positioning systems. *Geodesy And Cartography*, 35(1):18–22, 2009.

[103] R. Mautz. *Indoor Positioning Technologies*. PhD thesis, ETH Zurich, Feb. 2012.

[104] L. Mayhew and D. Smith. Using queuing theory to analyse the government's 4-h completion time target in accident and emergency departments. *Health Care Management Science*, 11(1):11–21, March 2008.

[105] L. E. Y. Mimbela and L. A. Klein. Summary of vehicle detection and surveillance technologies used in intelligent transportation systems. Technical report, May 1999.

[106] I. Mitrani. *Probabilistic Modelling*. Cambridge University Press, 1998.

[107] MSDN. RFID: An introduction. `http://msdn.microsoft.com/en-us/library/aa479355.aspx`.

[108] R. Nelson. *Probability, Stochastic Processes, and Queuing Theory: The Mathematics of Computer Performance Modelling*. Springer-Verlag, May 1995.

[109] NIST/SEMATECH. e-Handbook of Statistical Methods, May 2006. `http://www.itl.nist.gov/div898/handbook/`.

[110] L. Pasini and S. Feliziani. Networks of queues for the simulation of urban flow systems: An intermodal traffic system. *Task Quarterly*, 8(1):121–130, 2004.

[111] V. Paxson and S. Floyd. Wide area traffic: the failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226 –244, jun 1995.

[112] J. Peng, Y. Siyuan, and S. Dinghua. Open queueing network model of shanghai public transportation problem. *Journal of Shanghai University (English Edition)*, 2:96–99, 1998.

[113] R. Peng and M. Sichitiu. Angle of arrival localization for wireless sensor networks. In *SECON '06: IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, volume 1, pages 374–382, 2006.

[114] W. P. Peterson and L. M. Wein. Heavy traffic analysis of a transportation network model. *Journal of Applied Probability*, 33(3):pp. 870–885, 1996.

[115] M. Philipose, K. P. Fishkin, M. Perkowitz, D. J. Patterson, D. Fox, H. Kautz, and D. Hahnel. Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 3(4):50–57, Oct. 2004.

[116] N. B. Priyantha. The Cricket indoor location system. Technical report, Massachusetts Institute of Technology, 2005.

[117] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket location-support system. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 32–43, New York, NY, USA, 2000. ACM.

[118] Y. Qu, L. Li, Y. Liu, Y. Chen, and Y. Dai. Travel routes estimation in transportation systems modeled by Petri Nets. In *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 73 –77, july 2010.

[119] B. Rabta, A. Alp, and G. Reiner. Queueing networks modeling software for manufacturing. In *Rapid Modelling for Increasing Competitiveness*, pages 15–23. Springer London, 2009.

[120] A. Riska, V. Diev, and E. Smirni. An EM-based technique for approximating long-tailed data sets with PH distributions. *Performance Evaluation*, 55:147–164, 2004.

[121] F. Rivera-Illingworth, V. Callaghan, and H. Hagras. A neural network agent based approach to activity detection in AmI environments. In *Proceedings of the IEEE International Workshop on Intelligent Environments*, pages 92–99.

[122] D. A. Rolls, G. Michailidis, and F. Hernández-Campos. Queueing analysis of network traffic: methodology and visualization tools. *Computer Networks*, 48(3):447–473, June 2005.

[123] J. Sander, M. Ester, H. P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, june 1998.

[124] G. Serazzi. Java modelling tools. `http://jmt.sourceforge.net/`.

[125] Simul8. `http://www.simul8.com/`.

[126] W. Stewart. *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling*. Princeton University Press, 2009.

[127] N. Sugiura. Further analysts of the data by Akaike' s information criterion and the finite corrections. *Communications in Statistics - Theory and Methods*, 7(1):13–26, 1978.

[128] R. Suri and G. W. Diehl. MANUPLAN: a precursor to simulation for complex manufacturing systems. In *Proceedings of the 17th Conference on Winter Simulation*, pages 411–420, New York, NY, USA, 1985. ACM.

[129] A. Thümmler. A novel approach for fitting probability distributions to real trace data with the EM algorithm. In *DSN '05: Proc. of the 2005 International Conference on Dependable Systems and Networks*, pages 712–721, Washington, DC, USA, 2005.

[130] A. Thümmler, P. Buchholz, and M. Telek. A novel approach for phase-type fitting with the EM algorithm. *IEEE Transactions on Dependable and Secure Computing*, 3:245–258, 2005.

[131] Ubisense. Ubisense location system. `http://www.ubisense.net`.

[132] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: a survey of issues and approaches. *Data & Knowledge Engineering*, 47(2):237–267, Nov. 2003.

[133] M. Vankipuram, K. Kahol, T. Cohen, and V. L. Patel. Toward automated workflow analysis and visualization in clinical environments. *Journal of Biomedical Informatics*, 44(3):432–440, june 2010.

[134] R. Want. An introduction to RFID technology. *IEEE Pervasive Computing*, 5(1):25–33, Jan.-March.

[135] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, Jan. 1992.

[136] Waze. `http://www.waze.com/`.

[137] A. Wolf and D. Rosenblum. A study in software process data capture and analysis. In *Proceedings of the 2nd International Conference on the Software Process*, pages 115–124, Feb. 1993.

[138] F. Wu, F. Kuo, and L. Liu. The application of RFID on drug safety of inpatient nursing healthcare. In *ICEC '05: Proc. 7th International Conference on Electronic Commerce*, pages 85–92, August 2005.

[139] H. Xie, T. Chaussalet, and M. Rees. A semi-open queueing network approach to the analysis of patient flow in healthcare systems. In *CBMS '07: Proc. 12th IEEE International Symposium on Computer-Based Medical Systems*, pages 719–724, Washington, DC, USA, 2007. IEEE Computer Society.

[140] Z. Zhen, M. Xiaoji, H. Yao, L. Jingshan, and B. Deborah. A simulation study to improve quality of care in the emergency department of a community hospital. *Journal of Emergency Nursing*, 38(4):322 – 328, 2012.

[141] F. Zhu, Z. Wang, F.-Y. Wang, and S. Tang. Modeling interactions in artificial transportation systems using Petri net. In *ITSC '06: Proc. IEEE Intelligent Transportation Systems Conference*, pages 1131–1136, sept. 2006.