TOWARDS THE AUTOMATED INFERENCE OF QUEUEING NETWORK MODELS FROM HIGH-PRECISION LOCATION TRACKING DATA

Tzu-Ching Horng, Nicholas Dingle, Adam Jackson and William Knottenbelt
Department of Computing, Imperial College London,
South Kensington Campus, London SW7 2AZ
Email: {th107,njd200,aj307,wjk}@doc.ic.ac.uk

KEYWORDS

Location Tracking, Spatiotemporal Data Sets, UWB, RFID, Data Mining, Workflow Induction, Queueing Networks, Performance Modelling

ABSTRACT

Traditional methods for deriving performance models of customer flow in real-life systems are manual, time-consuming and prone to human error. This paper proposes an automated four-stage data processing pipeline which takes as input raw high-precision location tracking data and which outputs a queueing network model of customer flow. The pipeline estimates both the structure of the network and the underlying interarrival and service time distributions of its component service centres. We evaluate our method's effectiveness and accuracy in four experimental case studies.

INTRODUCTION

Stochastic models, especially queueing networks, have been widely used to model and analyse the performance and reliability of systems that involve the flow and processing of customers via a network of service centres. Indeed such models provide a powerful "virtual laboratory" for identifying and explaining bottlenecks in existing systems and for exploring the impact on performance of proposed changes to those systems, e.g. changes in customer flows, the number and allocation of resources, system workload, scheduling policies and so on.

Traditionally, constructing a performance model is a four step process: conceptualisation, parameterisation, validation and analysis. The first two steps are essential in building an accurate model, which in turn is crucial in giving confidence in the output of subsequent steps. However, they are typically time-consuming, expensive, intrusive, labour-intensive, and prone to bias and other errors. The limited amount and low quality of manually-collected data often lead to an inaccurate picture of resource flows and poor estimations of model parameters.

In recent years, wireless location tracking technologies based on RFID (Radio Frequency Identification) and UWB (Ultrawide-band) have been increasingly deployed to collect real-time location tracking data. For example, RFID has been applied in patient identification (Chen et al., 2005) 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 (Kim et al., 2006). These technologies offer the advantage of collecting large amounts of high-quality location data in an inexpensive and non-intrusive way. This data is a valuable input in building performance models that can better characterise the operation of real systems.

Raw location tracking data usually contains noise and other extraneous information; thus appropriate data processing methods are necessary for extracting useful information from it. Much research work (such as Gonzalez et al. (2006a), Gonzalez et al. (2006b) and Gonzalez et al. (2006c)) has been done in designing new data models to facilitate RFID data warehousing and inferring high-level information from RFID data, such as probabilistic item flows. However, RFID data has some distinct characteristics when compared to the location tracking data collected by other location tracking technologies, such as sensor networks. In the former case, the location reads of an RFID-tagged item have strict spatial relationships with the RFID readers, as a tagged item's location can only be known when it is scanned by an RFID reader, the location of which is usually fixed. By contrast, in the latter case, the location data is a noisy and frequently incomplete trace of the tagged entities' geographical locations; spatial relationships and interactions between tagged entities must be inferred on the basis of proximity or otherwise.

The aim of our work is to automatically extract a queueing network model of customer flow from observations of a real-life system gathered using a high-precision location tracking system. To achieve this, we present an approach based around a four-stage data processing pipeline, whose initial input is raw location tracking data, i.e. a spatiotemporal data set giving the observed location of tags at various times. The first two stages of the data processing pipeline remove irrelevant location information and use tagged entities' geographical locations to infer high-level spatiotemporal relationships among customer and server entities (including the presence of customers in the service area of servers and customer movements between servers). The third stage of the pipeline extracts occurrence times of arrivals and departures of

Proceedings 23rd European Conference on Modelling and Simulation ©ECMS Javier Otamendi, Andrzej Bargiela, José Luis Montes, Luis Miguel Doncel Pedrera (Editors)
ISBN: 978-0-9553018-8-9 / ISBN: 978-0-9553018-9-6 (CD)

customers at servers. The fourth stage uses these times to estimate the underlying interarrival and service time distributions of the original physical system, and also infers routing probabilities from customer movements. Together, these yield a parameterised queueing network model of the real-life system.

For the purpose of evaluation, we conduct four case studies corresponding to simple queueing systems with known structure and parameters. Goodness of fit tests are used to assess whether the extracted distributions of interarrival time and service time exhibit the same probabilistic characteristics as the real ones.

The remainder of this paper is organised as follows. The next section introduces previous research literature closely related to our work. This is followed by a brief description and comparison of three common location tracking technologies – RFID, WiFi and UWB, of which UWB is our adopted technology. We then describe and illustrate each stage of the developed data processing pipeline, discussing issues encountered during implementation, before presenting four case studies. We conclude with a summary of our results and a discussion of potential future work.

PREVIOUS WORK

Our work shares some high-level similarities with previous research efforts on mining RFID data. These research efforts have been focused on efficiently extracting flow information from massive RFID data sets through innovation of data model design and data compression techniques. For example, Gonzalez et al. (2006c) proposed a new data model for data-warehousing RFID data sets. Their goal was to enable the efficient storage and high-level querying of such data sets, taking into account the fact that several items may be aggregated into single tagged units. Building on this work, Gonzalez et al. (2006a) 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 that maintains the path information of the item flow. Each node in a flowgraph represents a location and edges between nodes correspond to transitions between locations. Gonzalez et al. (2006b) later presented a method to construct compressed probabilistic workflows that describe the general movement paths of items, as well as the significant exceptions. Although our data bears different characteristics from RFID data, our approach is also directed at devising a method to extract flow information from location tracking data.

Workflow induction and process discovery are also closely related research areas. These areas explore techniques that can use event data, which records the activities of an on-going process, to build up a formal model that describes the process's behaviour. Cook and Wolf (1995, 1996) 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 Cook and Wolf, Agrawal et al. (1998) presented an algorithm that uses existing execution logs to model the workflow structure of a given business process as a graph.

Other research work has been conducted in inferring performance models from sample executions. Instead of using pure statistical learning methods, Zhang and Bivens (2007) and Zhang et al. (2007) incorporate domain knowledge into machine learning techniques such as Bayesian networks and neural networks for response time modelling. Sen et al. (2004) present a machine-learning algorithm that can learn the underlying edge-labelled continuous-time Markov chains of a stochastic system based on sample execution trace data.

Our work seeks to develop a data processing method that incorporates the strengths of statistical learning techniques with the aim of extracting service and interarrival time distributions and routing probabilities from location tracking data; together these will specify a queueing network model of customer flow with enhanced accuracy.

LOCATION TRACKING TECHNOLOGIES

Three common technologies that have been employed in location tracking are RFID, UWB and WiFi. The various technologies used in real-time tracking have distinct characteristics which lend them to differing applications. RFID is usually passive, which means that a RFID tag has to be scanned by a reader for its location to be recorded. Tracking with WiFi typically provides an accuracy of the order of 2–7m in 2D while the latest UWB-based systems provide greater accuracy of around 15cm in 3D; the latter allows for detailed detection of customer/server interactions and gives potential to model processes such as contact-based spread of pathogens. WiFi-based systems are, however, often easier to deploy as they use hardware (WiFi access points) that might already be installed.

UWB operates over the frequency range 3–10 GHz in the U.S and 6–8.5 GHz in Europe (The Federal Communications Commission, 2002). Interference with other signals is prevented as UWB emits short-duration high-bandwidth radio pulses at a low power making it particularly suitable for use in radio sensitive environments such as hospitals. The low power does, however, limit the range over which detection is possible but is still able to penetrate walls. The short length of time over which a pulse is emitted also means that a high density of devices can be achieved in a given detection area. 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 (Ubisense) (cf. Figure 1).

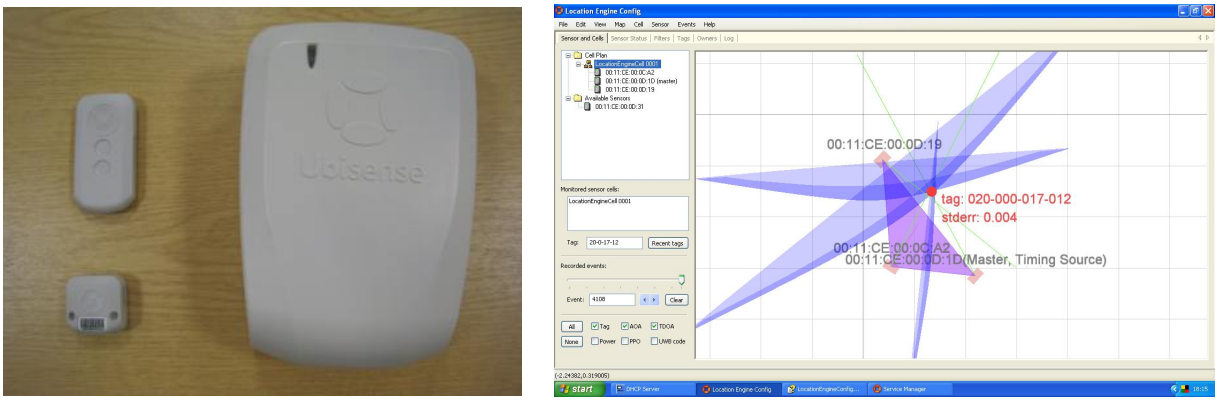


Figure 1: UWB-based sensors and tags (left) use a combination of angle-of-arrival and timed-difference-of-arrival triangulation schemes to enable high-accuracy tag location tracking (right)

INFERRING PERFORMANCE MODELS FROM HIGH-PRECISION LOCATION TRACKING DATA

Data Processing Pipeline

This paper presents an automated four-stage data processing pipeline. The input to our pipeline is the raw location data from a real-life physical system (gathered via a high-precision UWB-based sensor network) and the output is a queueing network describing customer flow in that system. This queueing network is specified by routing probabilities and interarrival and service time distributions estimated from the location data.

Figure 2 gives an overview of the data processing pipeline. The input and output data of each stage are stored as database tables, which are designed to facilitate data querying and further manipulation. After each stage is completed, the inputs from the previous stage can be discarded.

Stage 1

Raw location tracking data is stored in the **Traces** table as a stream of tuples of the form `(tagName, type, time, x, y, z, stderr)`. `tagName` is a unique identifier for each tag and `type` refers to the category a tag belongs to. In this research there are two types of tagged entities of interest: `Server` and `Customer`. `Server` entities are those offering services or resources in a system; `Customer` entities are those requesting services or resources. `time` is the timestamp when the location update took place; ideally this is a taken using a high-resolution timer with millisecond resolution or better. `x`, `y`, `z` are the locations of the tags expressed in a Cartesian coordinate system. `stderr` expresses the location tracking system's estimate of how much the measured location might deviate from the actual location.

Using the `type` field in the table, the **Traces** table can be split into the **Server Traces** table and the **Customer Traces** table, which store the location tracking data for `Server` and `Customer` entities, respectively. The data stored in both tables is of the form `(tagName, time, x, y, z, stderr)` and is sorted by time. Furthermore, in Stage 1, linear data interpolation is conducted to fill out gaps in trace data when two contiguous location updates for a tag take more than four times longer than would be expected (in our case tags should update their position four times a second).

Stage 2

After the first stage of the data processing, the data is still very low-level. The main purpose of the second stage is to extract relevant higher-level data. This includes at what time a customer arrives at a server, when the server finishes serving the customer and the number of customers being served or queueing for service at a server at a specific timestamp. We assume each server in the system has a user-defined service area, which is a circular area with given radius. If a tag is detected within the service area of a particular server at a given timestamp, we assume that the customer entity that owns the tag is either being served by the server or is waiting to be served in the queue of the server. If the tag is found to be located within user-specified entry or exit areas, the customer entity corresponding to the tag is considered as entering or leaving the system. In other cases, the customer entity is assumed to be moving between servers and/or entry/exit areas.

The outputs of the second stage of data processing are **Service Load Traces** tables (one table for each server) and one **Customer Location Traces** table. The contents of these two types of table are as follows:

Service Load Traces tables The data stored in these tables is of the form `(time, count, ⟨customer list⟩)` and is ordered by time. The `count` field is the total number of the customer entities that are located inside the server's service area at the given timestamp. The field `⟨customer list⟩` maintains the tag IDs of the customer entities. Table 1 gives an example of the **Service Load Traces** table.

The main challenge in this stage of data processing is that location tracking data in reality does not appear as a smooth trace describing the object's path;

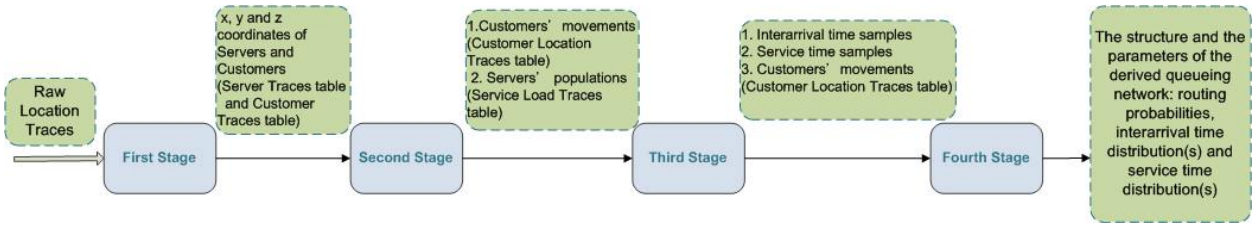


Figure 2: Four-stage data processing pipeline

time	count	customerlist
13.3163	1	Customer0
13.4882	1	Customer0
13.6633	0	
13.8360	1	Customer1
13.9894	2	Customer1, Customer0
14.1501	2	Customer1, Customer0
14.3084	2	Customer1, Customer0
14.4703	2	Customer1, Customer0
14.6341	2	Customer1, Customer0

Table 1: Example **Service Load Traces** table

tagName	time	location
Customer1	13.2983	moving
Customer2	13.3847	moving
Customer0	13.4027	Server3
Customer1	13.4194	moving
Customer2	13.4405	moving
Customer0	13.4631	Server3
Customer1	13.5592	moving
Customer2	13.5816	moving
Customer0	13.6023	Server3
Customer1	13.6240	moving

Table 2: Example **Customer Location Traces** table

noise and other artefacts such as radio reflections might result in location updates with sudden deviation from the actual locations. The erratic location updates might lead to the conclusion that a customer entity has entered the service area when in fact it has not. A window-based voting mechanism is therefore used to avoid this situation. In each time window, defined as the time between two `Server` location updates, there is one counter for each entity that has visited the service area of a certain `Server`. Assume that there are more than one location reads of a customer tag, say `Customer1`, between two timestamps $t1$ and $t2$. When checking whether `Customer1` was inside the service area of a `Server`, say `Server1`, its counter value is incremented by one if `Customer1` is detected inside `Server1`'s service area and decreased by one if not. If at time $t2$ the value of the counter is positive, it means that there are more readings between $t1$ and $t2$ supporting the notion that `Customer1` is within `Server1`'s service area. If the counter's value is negative, then `Customer1` is not included in ⟨`customer list`⟩.

Customer Location Traces table The data stored in this table is of the form ⟨`tagName, time, location`⟩ ordered by time. Instead of exact Cartesian coordinates, the location information is reduced into a higher-level description. For example, as shown in Table 2, we can see that `Customer1` is moving at time 13.2983 and `Customer0` is within a service area of `Server3` at time 13.4027.

Stage 3

The major work in this stage is to detect arrival and departure events from the **Service Load Traces** tables and calculate customer interarrival times and service times. We process each record in each **Service Load Traces** table and detect any differences in the ⟨`customer list`⟩ fields between two contiguous records. For example, if at timestamp $t1$, the ⟨`customer list`⟩ field is ⟨`Customer0, Customer1, Customer2`⟩ and at the next timestamp $t2$, the ⟨`customer list`⟩ field is ⟨`Customer1, Customer3`⟩, then between $t1$ and $t2$ there might be one arrival event of `Customer3` entering the service area and two departure events of `Customer0` and `Customer2` leaving the service area.

However, a missing tag ID in the ⟨`customer list`⟩ field does not necessarily represent "real" departure events. Infrequent location updates or erratic location reads of the tags between two contiguous timestamps might contribute to this phenomenon. In this case, the missing tag IDs should still remain in the ⟨`customer list`⟩ field. In order to judge whether a real departure event occurred, a look-ahead action is taken; this is essentially a query to the **Customer Location Traces** table to check whether the missing Customer entity shows up again in the server's service area within a certain amount of time from the current timestamp. The amount of time is based on the minimum time for a customer entity to make a round trip from the current server to the entry/exit areas and back to the current server, and is another user-supplied input.

For arriving customers, it is assumed that database entries represent true arrival events. This is because of the window-based voting mechanism applied in the second stage of data processing.

The next step after detecting a customer arrival or de-

parture event is to estimate the most probable time when the event occurred. The estimation of departure event occurrence time is the average of two values, both of which are obtained from querying the **Customer Location Traces** table. The first value is called first disappearance time. It refers to the first timestamp at which the customer's location is not inside the server's service area after it begins service. The second value is called the last appearance time, which is the last timestamp at which the departing customer remained in the service area between its service start time and the first disappearance time. A similar approach is used to estimate arrival times using the last time the customer was observed outside a service area and its first appearance time within the service area. We note the differences between the two times were typically not as significant for arrivals as for departures, since the tags we use are designed to update their positions more frequently whilst moving.

The outputs of the third stage are the extracted interarrival time and service time samples for each `Server`.

Stage 4

The data stored in the **Customer Location Traces** table can be seen as a stream of event logs that trace all the activities of the `Customer` entities in time order. Stage 4 uses this table to mine the structure of customer flow in the system by a method that is a simplified adaptation of the workflow mining methods proposed by Agrawal et al. (1998). One feature worth noticing is that tags can be "recycled" after the tracked customers leave the system, so one tag ID can actually represent different customers. It is therefore necessary to identify the entry and exit points of customer flow in the system. The mining process is basically a counting process. As each record in the **Customer Location Traces** table is processed, a tracking table is used for recording all the possible paths or sub-paths the customers have passed through so far, and the number of customers that followed a specific path. Table 3 gives an example of such a table. From it we can conclude that in this specific system, 27 out of 28 customers first visited `Server1` while one customer first visited `Server2`. Among the 27 customers, 19 customers went on to `Server2` next, while 8 proceeded to `Server3`. It is straightforward to estimate routing probabilities from such a table.

path	count	nextFlow
`Server1`	27	`Server1` `Server2`, `Server1` `Server3`
`Server2`	1	
`Server1` `Server2`	19	
`Server1` `Server3`	8	

Table 3: Example **Customer Flow** table

Having now identified the structure of a high-level queueing network through routing probabilities, the next step is to characterise the arrival process and service process associated with each (single-server) queueing node. We do this by fitting candidate distributions to the interarrival and service time samples generated in the previous stage of the pipeline. Specifically, we use the samples' coefficient of variation, c_v, as a simple indicator of the underlying distribution as follows:

- If $c_v \simeq 0$, the underlying distribution is likely to be deterministic.
- If $0 < c_v < 1$, the underlying distribution is likely to be hypo-exponential.
- If $c_v \simeq 1$, the underlying distribution is likely to be exponential.
- If $c_v >> 1$, the underlying distribution is likely to be hyper-exponential.

Having thus selected a candidate distribution, we use maximum-likelihood estimation to obtain distribution parameters. We then conduct three different goodness of fit tests, including Kolmogorov-Smirnov, Anderson-Darling, and Chi-squared tests, to measure the compatibility of the sample with the experiment's true probability distribution function.

CASE STUDIES

In this section, we perform evaluation and analysis of our data processing pipeline using four case studies. For each case study, we designed an environment whose customer flow resembles the one in a simple queueing system with known parameters. We then staged experiments where volunteers transported tags between single-server service areas at times sampled from known distributions. Positions of tags were continuously recorded using a Ubisense Location Tracking System. Figure 3 illustrates the structure and parameter settings of the approximated queueing systems.

We use the first three case studies to show that extracted interarrival and service time distributions are similar in statistical behaviour to the actual underlying distributions. We use the fourth case study to show how the proposed data processing method copes when inferring customer routing probabilities.

Results and Discussion

Figure 4 shows the historical paths of all tags during the first of our experiments. The tags' locations are colour-coded; the red positions indicate when the tag was moving, while the blue locations are when it remained (almost) static. From this, we can infer the locations of the service and entry/exit areas and the paths between them.

Service Time Distributions

Table 4 lists the coefficients of variation calculated from the extracted service time samples in the first three case studies, which are compared with the coefficients of variations of the actual underlying distributions.

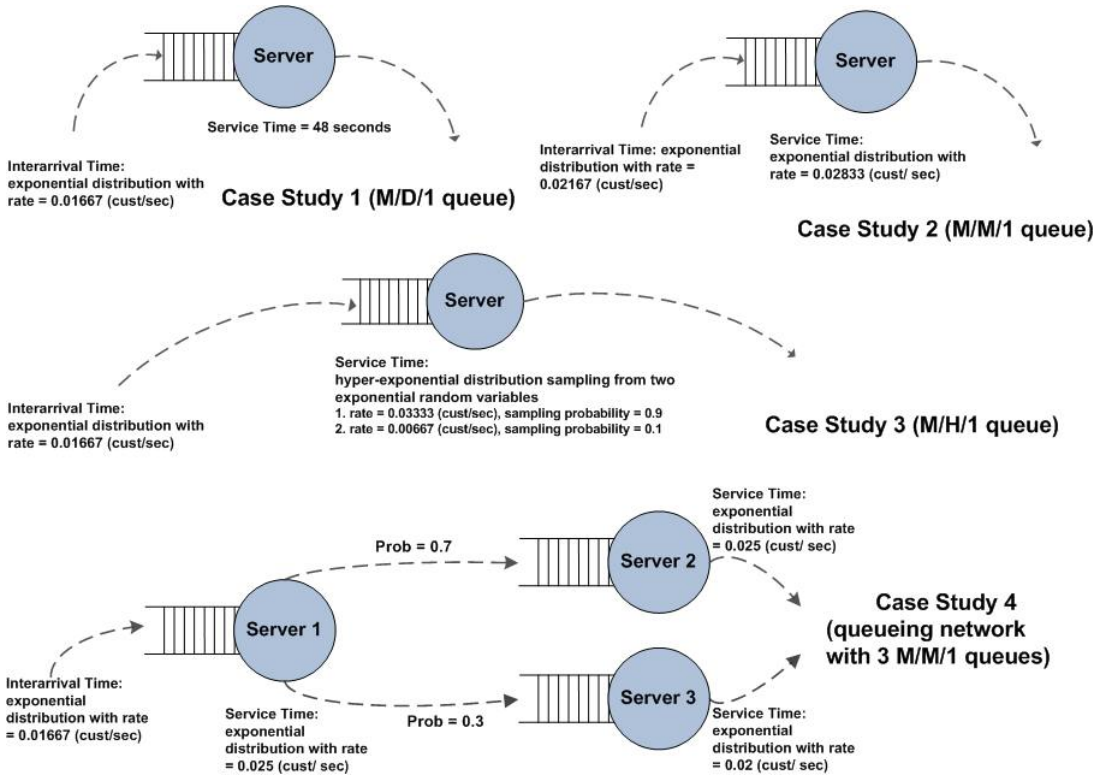


Figure 3: The settings of the four case studies

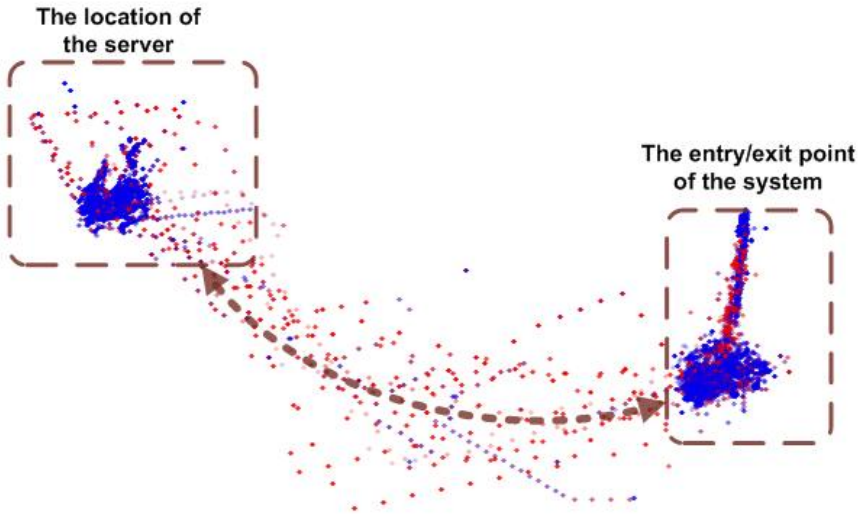
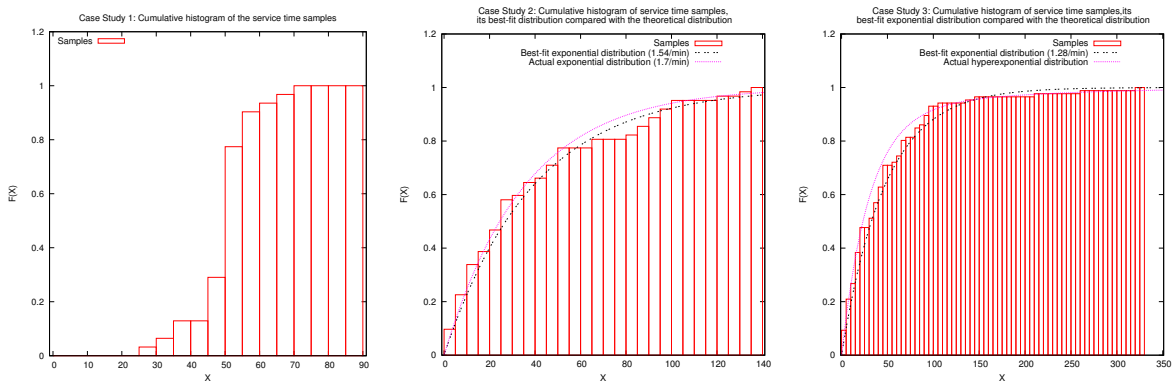


Figure 4: Visualisation of the raw location tracking data from Case Study 1 to show moving tag positions (red dots) and static tag positions (blue dots).



(a) Case Study 1 (b) Case Study 2 (c) Case Study 3

Figure 5: Cumulative histograms of the extracted service time samples in the three case studies.

	Case Study 2		**Case Study 3**	
	Kolmogorov-Smirnov		Kolmogorov-Smirnov	
test statistic	0.0855		0.0851	
α	0.1	0.05	0.1	0.05
critical values	0.1526	0.1696	0.13	0.1444
rejected?	No	No	No	No
	Anderson-Darling		Anderson-Darling	
test statistic	0.4121		0.4868	
α	0.1	0.05	0.1	0.05
critical values	1.929	2.502	1.929	2.502
rejected?	No	No	No	No
	Chi-squared		Chi-squared	
test statistic	4.671		5.325	
α	0.1	0.05	0.1	0.05
critical values	9.236	11.07	10.64	12.59
rejected?	No	No	No	No
estimated λ	0.0256 cust/sec		0.0213 cust/sec	
actual λ	0.02833 cust/sec		hyper-exponential with λ_1= 0.03333 (prob_1=0.9) λ_2= 0.00667 (prob_2=0.1)	

Table 5: The results of three goodness of fit tests at significance levels 0.1 and 0.05 applied to the extracted service time samples from Case Study 2 and Case Study 3. The null hypothesis is that the sample is extracted from an exponential distribution. The estimated λ is the rate of the best-fit exponential distribution (in cust/sec).

Case Study	c_v (estimated)	c_v (actual)
Case Study 1	0.17	0.00
Case Study 2	0.93	1.00
Case Study 3	1.15	1.57

Table 4: The coefficients of variation c_v of the extracted service time samples compared to their actual values

The designed probability distribution of the service time for Case Study 1 is deterministic. The calculated c_v is very low (0.17), suggesting that the service time distribution is likely to be a deterministic distribution. Indeed, the cumulative histogram obtained from the service time samples is similar to a step function with an inflexion point between 45 and 55 seconds (see Figure 5(a)), with a sample mean of 51 seconds. This differs slightly from the true value of 48 seconds, a deviation that is expected, since the case study was executed under human control with a slight delay when a customer finished being served and began to leave the server's service area.

The c_v of service time sample in Case Study 2 is very close to the actual value 1, which is the c_v of an exponential distribution. The results of all three goodness of fit tests (see Table 5) also accept that the null hypothesis that the service time sample is extracted from an exponential distribution. The estimated parameter of the best-fit exponential distribution is 0.02561 (cust/sec), which is close to the theoretical value 0.02833 (cust/sec), as shown in Figure 5(b).

The c_v of the service time samples in Case Study 3 (see Table 4) and the goodness of fit test results (see Table 5) all indicate that the service time is very likely to be sampled from an exponential distribution. However, the true service time distribution should be a hyper-exponential distribution with c_v around 1.5. From Figure 5(c), we can see that the shapes of the actual service time distri-

bution and the best-fit exponential distribution are very similar to each other. Given the limited size of the extracted samples, it is reasonable to make the statistical inference that the underlying distribution of the extracted service time sample is likely to be an exponential distribution with $\lambda = 0.0213$ cust/sec.

Case Study	c_v (estimated)	c_v (actual)
Case Study 1	0.89	1.00
Case Study 2	0.93	1.00
Case Study 3	0.87	1.00

Table 6: The coefficient of variation c_v of the extracted interarrival time samples compared to their actual values

Interarrival time

Table 6 shows the coefficients of variation calculated from the extracted interarrival time samples in the first three case studies. All of them are close to 1, which means that we infer the interarrival time samples are likely to be extracted from exponential distributions (which in fact they are).

Table 7 presents the results of applying three goodness of fit tests to the extracted interarrival time samples for the three case studies. The results show that the hypothesis that the underlying probability distribution of the sample is exponential cannot be rejected. It also compares the estimated λ of the best-fit exponential distribution with the actual values.

Routing Probabilities

path	count	nextFlow
`Server1`	27	`Server1` `Server2`, `Server1` `Server3`
`Server2`	1	
`Server1` `Server2`	19	`Server1` `Server2` `Server3`
`Server1` `Server3`	4	
`Server1` `Server2` `Server3`	4	

Table 8: Customer flow analysis for Case Study 4

Table 8 shows the result of flow analysis (conducted in the fourth stage of the data processing pipeline) for Case Study 4. From the table, we can see that 19 out of 27 customers who completed service at `Server1` went to `Server2` for service and only 4 went to `Server3` for service. In the actual setting, 70 percent of the customers go to `Server2` and 30 percent of the customers go to `Server3` after `Server1`. The experimental results differ from the actual probabilities because, as shown in Table 8, the flow analysis discovers some nonexistent paths, such as `Server1` `Server2` `Server3` and `Server2`. This is due to the fact that some of the tags' locations were not updated regularly on account of the tags entering their "sleep" mode after being stationary for a period of time (Ubisense).

CONCLUSION AND SUMMARY

This paper has presented a four-stage data processing pipeline for extracting queueing models (specified in terms of routing probabilities, interarrival time distributions and service time distributions) from high-precision location tracking data. The pipeline gradually refines noisy low-level data (e.g. entities' Cartesian coordinates) into high-level historical traces of customer and server activities. From derived samples of customer interarrival times and service completion times, a simple workflow mining algorithm infers customer flow routing probabilities and fits arrival and service time distributions. These comprise a simple queueing network model of the monitored system made up of single-server nodes.

Through four case studies conducted in this project, the developed data processing method has shown moderate success in extracting performance models that display the true probabilistic features of the underlying system. The success is especially obvious with systems of lower levels of complexity and randomness.

In the future we intend to extend our methodology to support advanced features such as multiserver nodes, multiple classes of customer, customer priority schemes, and sophisticated arrival processes (e.g. correlated input traffic). We also wish to experiment with the application of such techniques in the context of real-life systems (e.g. healthcare environments).

REFERENCES

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.

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.

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.

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.

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, 2006a.

	Case Study 1		Case Study 2		Case Study 3	
	Kolmogorov-Smirnov		Kolmogorov-Smirnov		Kolmogorov-Smirnov	
test statistic	0.1514		0.068		0.0804	
α	0.1	0.05	0.1	0.05	0.1	0.05
critical values	0.2109	0.2342	0.1538	0.1709	0.1257	0.1396
rejected?	No	No	No	No	No	No
	Anderson-Darling		Anderson-Darling		Anderson-Darling	
test statistic	0.8881		0.3704		0.7681	
α	0.1	0.05	0.1	0.05	0.1	0.05
critical values	1.929	2.502	1.929	2.502	1.929	2.502
rejected?	No	No	No	No	No	No
	Chi-squared		Chi-squared		Chi-squared	
test statistic	9.76		2.739		9.76	
α	0.1	0.05	0.1	0.05	0.1	0.05
critical values	10.64	12.59	9.236	11.07	10.64	12.59
rejected?	No	No	No	No	No	No
estimated λ (cust/sec)	0.0195		0.01845		0.01715	
actual λ (cust/sec)	0.01667		0.02167		0.01667	

Table 7: The results of three goodness of fit tests applied to the extracted interarrival time samples from three case studies. The null hypothesis is that the sample is from an exponential distribution. The estimated λ is the rate of the exponential distribution with the best fit to the sample data.

H. Gonzalez, J. Han, and X. Li. Mining compressed commodity workflows from massive RFID data sets. In *CIKM '06: Proc. 15th ACM International Conference on Information and Knowledge Management*, pages 162–171, New York, NY, USA, 2006b. ACM.

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, 2006c. IEEE Computer Society.

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 *PWC '06: Proc. 11th International Conference on Personal Wireless Communications*, pages 506–517, 2006.

K. Sen, M. Viswanathan, and G. Agha. Learning continuous time Markov chains from sample executions. *QEST '04: Proc. First International Conference on the Quantitative Evaluation of Systems*, pages 146–155, Sept. 2004.

The Federal Communications Commission. *Part 15 Regulations, Subpart F –Ultra-Wideband Operation*, February 2002.

Ubisense. Ubisense location system. `http://www.ubisense.net`.

R. Zhang and A.J. Bivens. Comparing the use of bayesian networks and neural networks in response time modeling for service-oriented systems. In *SOCP '07: Proc. 2007 Workshop on Service-oriented Computing Performance: Aspects, Issues, and Approaches*, pages 67–74, New York, NY, USA, 2007. ACM.

R. Zhang, A.J. Bivens, and I. Rezek. Efficient statistical performance modeling for autonomic, service-oriented systems. In *Proc. 21st IEEE International Parallel and Distributed Processing Symposium*, page 61, 2007.

Tzu-Ching Horng completed her MSc degree in Computing Science from Imperial College London in 2008. She also holds a MSc degree in Transportation System (Civil and Environmental Engineering) from Massachusetts Institute of Technology and a BSc in Civil Engineering from National Taiwan University. She is currently working toward her PhD degree in Department of Computing at Imperial College London.

Nicholas Dingle obtained an MSc degree in Computing Science from Imperial College London in 2001, and a PhD in Computing Science from the same institution in October 2004. He is currently employed as a Research Associate in the Department of Computing investigating the performance of virtualised data storage systems.

Adam Jackson was awarded an MSc in Computing Science from Imperial College London in 2008. Adam also holds a PhD in Biochemistry from the University of Cambridge and a BSc in Biochemistry from the University of Bristol.

William Knottenbelt completed his BSc (Hons) and MSc degrees in Computer Science at the University of Cape Town in South Africa. He obtained his PhD in Computing from Imperial College London in February 2000, and was subsequently appointed as a Lecturer in the Department of Computing in October 2000. Now a Senior Lecturer, his research interests include parallel computing and stochastic performance modelling.