

LocTrackJINQS: An Extensible Location-aware Simulation Tool for Multiclass Queueing Networks

Tzu-Ching Horng, Nikolas Anastasiou, Tony Field and William Knottenbelt
Department of Computing,
Imperial College London, South Kensington Campus
London SW7 2AZ
Email: {th107,na405,ajf,wjk}@doc.ic.ac.uk

KEYWORDS

Simulation, Queueing Network, Location Tracking

ABSTRACT

This paper presents `LOCTRACKJINQS`, a flexible and extensible spatio-temporal simulation tool for systems that involve the flow and processing of customers at multiple service centres. Developed based on the multi-class queueing network simulation package `JINQS`, `LOCTRACKJINQS` retains the abstract model specification power of `JINQS` while providing additional low-level models of entity movement. Besides 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.

1 INTRODUCTION

Simulation has long seen its use in both academic and business applications for system performance modelling or evaluation, workflow analysis, process improvement, asset/personnel management, etc. Simulation software tools such as [10], [3], [9], and [4] are offered to help organisations identify their system bottlenecks and gain better insights into the implications of different resource or personnel investments on overall system performance. By using simulation software decision makers can rapidly experiment with different scenarios and compare them at a fraction of the cost of real implementation in the system. However, some simulation tools have limited application domains (e.g. [3], [4]) and many of them also fail to represent the stochastic nature of time delays in the system.

The recent development of real-time location systems (RTLs) enables the automatic collection of large amounts of high-precision location tracking data

in real-time. RTLSs have been widely deployed by enterprises, especially in the fields of supply chain management and healthcare [11, 8], because they provide enhanced insights into asset and personnel flows [2]. These in turn lead to improvements in performance efficiency, asset management and system safety. Several research endeavours have also been made recently to extract workflow patterns and models of system performance from location tracking data. For example, research work conducted in [6] constructs compressed probabilistic flow models for commodities in a supply chain by mining large RFID (Radio Frequency Identification) data sets. [7] and [1] show how stochastic performance models (queueing networks and stochastic Petri nets, respectively) can be automatically derived from high-precision location tracking data.

This paper presents `LOCTRACKJINQS`, originally developed to support location-based research ([7] and [1]), is an open-source simulation library for constructing simulations with location awareness. `LOCTRACKJINQS` is an extension of `JINQS`, a Java simulation library for multiclass queueing networks [5]. `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 [5]. 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 similar to that 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 in data mining location tracking data or for developing location-based applications.

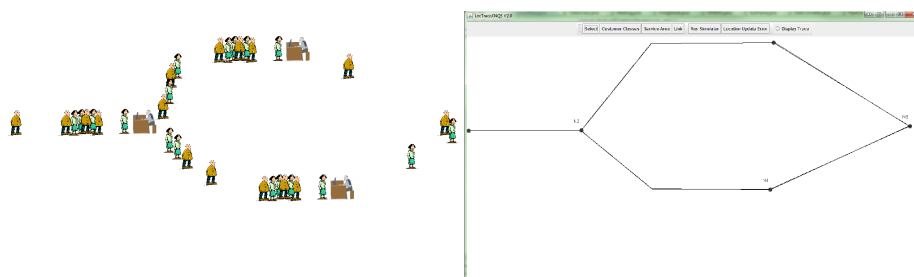
The remainder of the paper first gives an overview of the software architecture of `LOCTRACKJINQS` and explains how the new features are implemented in `LOCTRACKJINQS`. We then present a case study to demonstrate how one can build up a simulator for a real-life system. The paper concludes with a summary of `LOCTRACKJINQS`'s new features and possible future extensions.

2 SOFTWARE ARCHITECTURE AND MAJOR EXTENSIONS

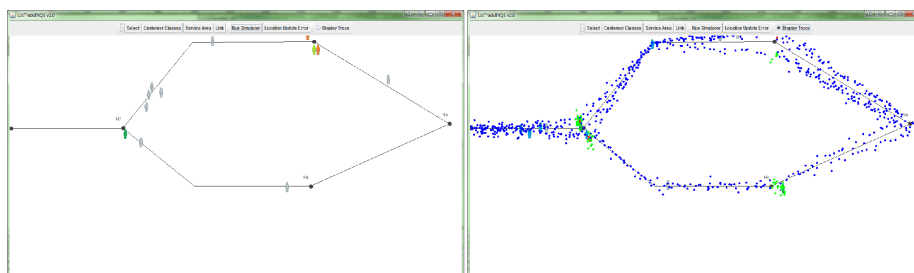
2.1 Overview

`LOCTRACKJINQS` is a simulation library that offers functionalities for simulating a real-life customer-processing system as a queueing network with low-level location information (see Figure 1). The user can not only specify the high-level features of the network such as the customer flow structure and time delay distributions (e.g. service time and inter-arrival time); but also low-level ones including entities' geographic locations and their moving speeds and paths.

`LOCTRACKJINQS` inherits many features from `JINQS`. In both `JINQS` and



(a)



(b)

Figure 1: An example of simulating a real-life system using LOCTRACKJINQS: Figure 1(a) demonstrates how a customer processing system is represented as a high-level queuing network with low-level location information; Figure 1(b) shows a screen shot of the simulation in progress and the generated location traces.

LOCTRACKJINQS, there are two main Java packages: **network** and **tools** [5]. 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 [5].

JINQS supports simulations of queueing networks with features including multiple servers, multiple customer classes and various queueing disciplines (e.g. FIFO, LIFO and priority-based); these are also supported by LOCTRACKJINQS. Inherited from JINQS, LOCTRACKJINQS also 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 population) [5]; the summary of all the measures can be displayed 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. As mentioned earlier, JINQS can only be used for constructing high-level abstract 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.

The following sections give more details on the important additions and modifications made to JINQS for implementing the new features.

2.2 Queueing Network Structure

In JINQS, a queueing network comprises three main types of entities, defined by **Node**, **Link** and **Customer** classes (see Figure 2). A network is structured as a collection of **Nodes** connected together by **Links**. The network is populated by **Customers**, which move among different **Nodes** along the connecting **Links** and request service or resources from the system. A subclass of **Node**, called **Source**, represents entry points where **Customers** are injected into the network according to a user-specified interarrival time distribution; **Sink** nodes are where **Customers** exit the network.

LOCTRACKJINQS defines queueing networks in a similar way, but introduces the entities' geographical locations (specified as 2D Cartesian coordinates)

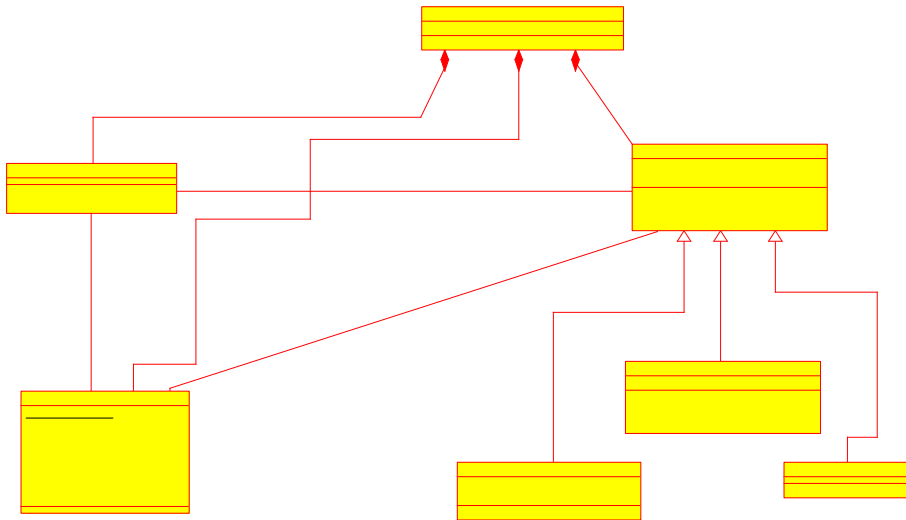


Figure 2: UML diagram of important classes in JINQS

and allows meaning to be assigned to their spatial relationships (e.g. defining how close a customer must be to a server in order to be served). `LOCTRACK-JINQS` 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 3 gives an overview of `INode` and the important types of `INode`. `LOCTRACKJINQS` uses three classes (all implementing the `INode` interface) – that is `Server` and subclasses `InfiniteServer` and `QueueingServer` – to define three basic types of service points in the system (see Figure 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.

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 service area are busy.

Like `JINQS`, `LOCTRACKJINQS` supports multiple customer classes, allowing for simulations of scenarios where `Customer` entities have class-dependent interarrival time distributions, service time distributions, routing probabilities

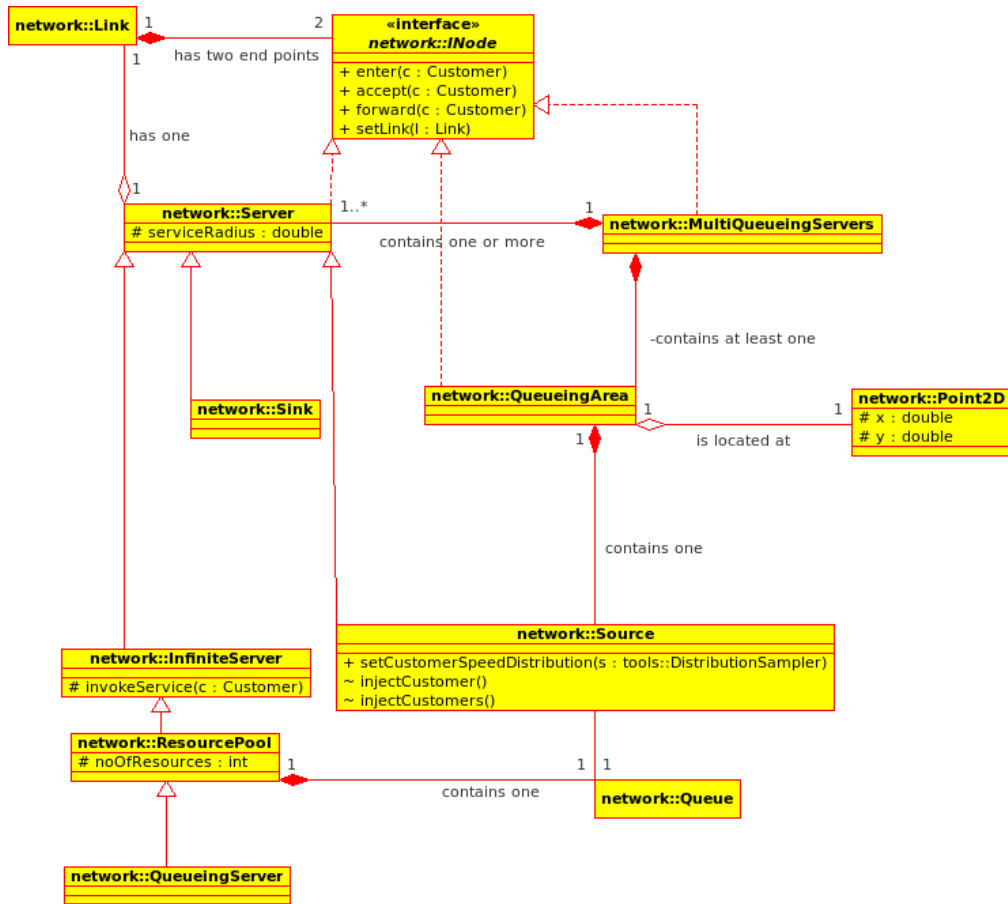


Figure 3: UML diagram of the important classes implementing `INode` interface in `LOCTRACKJINQS`

and service priority. 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). 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 `Customers` between two `Node` entities takes place in no time, a call to the `moveCustomers()` function of a `TransportLink` updates the locations of the `Customers` following the link based

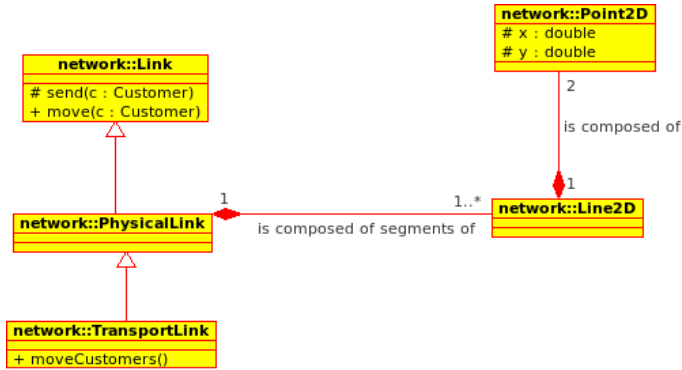


Figure 4: UML diagram of Link, PhysicalLink and TransportLink

on each `Customer` entity’s speed and direction of movement.

2.3 Generating Synthetic Location Tracking Data

Usually the entities we are interested in monitoring in a physical customer processing system are either those providing service/resources or those who are requesting service/resources, which correspond to `Server` (and its subclasses) and `Customer`, respectively. LOCTRACKJINQS introduces a new class `NetworkElement` as the superclass of both `Server` and `Customer` classes (see Figure 5); it corresponds to an entity in a system that is “tagged” and thus “monitored” by the RTLS. Another new class, `NetworkMonitor`, is implemented to simulate the behaviour of a RTLS tracking and to output tagged entities’ latest geographical locations. Similar to some of the existing RTLSs, each tag is assigned a QoS (Quality of Service) value, which is the tag’s location update rate requirement and defined as an attribute in `NetworkElement`. At each location update time slot, `NetworkMonitor` selects a tag and outputs its latest location with the best effort to satisfy each tag’s QoS, while avoiding starvation (i.e. a tag’s location is not updated for a long period of time). One thing to note is that different wireless technologies adopted by RTLS (e.g. Ultra Wide Band, Wi-Fi, and RFID), along with other factors (e.g. the density of sensors in range and objects that might interrupt sensors’ signal) contribute in practice to different error/noise patterns/distributions on location readings. It is thus impractical to assume any noise distributions in LOCTRACKJINQS. Accordingly, LOCTRACKJINQS allows the user to specify appropriate error distributions. A typical generated location update is of the form $(\text{tagName}, \text{type}, \text{time}, x, y, \text{stderr})$. `tagName` is a unique identifier for each entity in the system. `type` indicates the entity’s class, which can be used to differentiate a customer entity from a service point or to specify which customer class a customer entity belongs to. `time` is the timestamp when the location update is recorded and `x` and `y` specify the location of the tag. `stderr` is the expected deviation between the generated location update and the actual location of the entity.

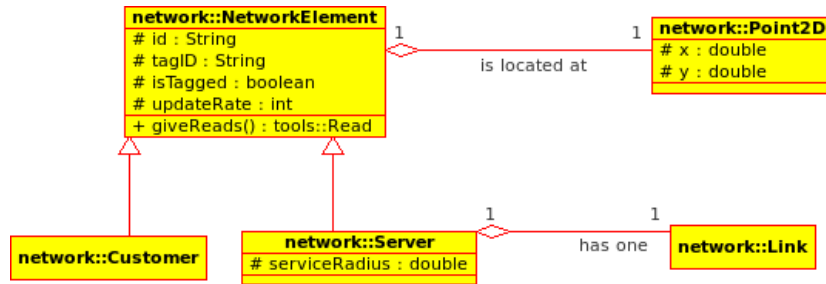


Figure 5: UML diagram of `NetworkElement` and its two subclasses `Server` and `Customer`

2.4 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 `moveCustomers()` method 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.

2.5 Graphical 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, to specify the parameters of a customer processing system and to animate it without having to write code (as is always required in JINQS). The facilities provided by the GUI are illustrated in the next section.

3 SIMULATION CASE STUDY

For the purpose of demonstrating the new capabilities implemented in `LOCTRACKJINQS` we present a case study of a small airport with two passport control and two security check points. Each of the check points has its own queue to hold waiting customers, who may be of three different classes:

- Pilots and flight attendants. This customer class has high priority.
- EU passengers. This customer class has low priority.
- Non-EU passengers. This customer class also has low priority.

The speed of all customers in this simulation environment is assumed to follow a Normal distribution with a mean and standard deviation of 0.3 and 0.1 m/s respectively. The remaining parameters of the simulation – i.e. interarrival time and service time distribution for each customer class – are depicted in Table 1. In all cases service priority-based with a FIFO queueing discipline within each priority class. The location update error has been set to be normally distributed with a mean of 0.15m and a standard deviation of 0.2m.

	Pilots & Flight Attendants	EU Passengers	Non-EU Passengers
Source (Interarrival time)	Exp(0.02)	Exp(0.33)	Exp(0.2)
Passport Control (Service time)	Normal(2, 0.5)	Exp(0.25)	Erlang(2, 0.33)
Security Check (Service time)	Exp(0.2)	Erlang(4, 0.1)	Erlang(4, 0.1)

Table 1: Interarrival and service parameters for each customer class.

The high-level topology of the airport environment is depicted in Figure 6. It can be easily constructed using the GUI features of LOCTRACKJINQS as shown in Figure 7. Here the Source node (N1) is connected to the two passport control nodes (N2 and N3) through via two branches having equal probability. These two nodes are then connected with to server (N4) which is an infinite server with zero service time and which routes customers to one of two security check points (N5 and N6) with equal probability. Customers exit the system by the Sink node (N7).

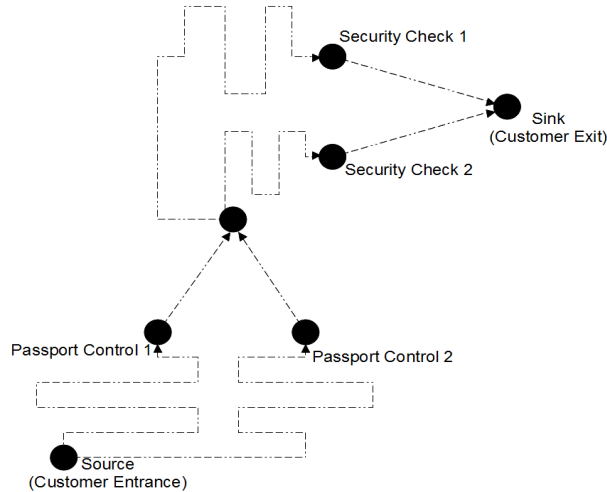


Figure 6: High-level description of the airport layout. Dashed arrow lines indicate the customer flow in the system.

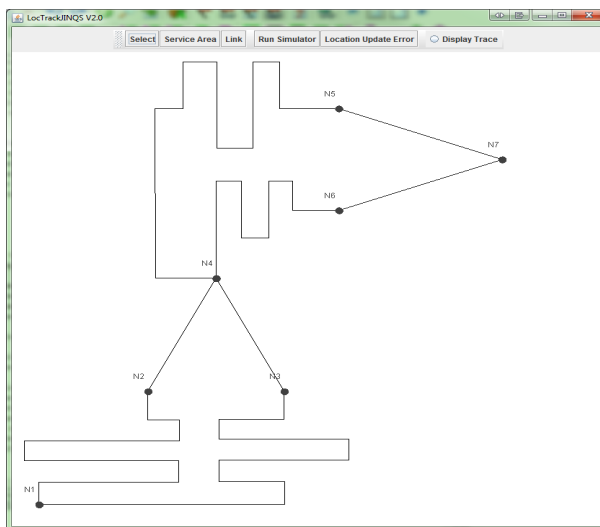


Figure 7: The airport simulation model as created using the GUI of LOCTRACK-JINQS

Figures 8 and 9 show the on-going simulation process at different time instants. Light blue, grey and dark blue coloured graphical figures are used to represent moving pilots and flight attendants, EU, and non-EU passengers respectively. In the same order as above, red, orange and pink colours are used to indicate that the entities are queueing. Three different shades of green are used to represent the customers from each class receiving service.

Table 2 and Table 3 shows the calculated mean response time, its standard deviation, and the mean queue length experienced by different customer classes at each service point and for the whole system. As shown in Table 2 and Table 3, the customer class with the highest priority (pilots and flight attendants) receive much better service quality than the other customer classes with lower priorities. Their mean response times at passport control and security check points are close to the assigned service time distributions and have low standard deviations, which means that they do not need to queue for long period of time and the service they enjoy is consistently of good quality. The other two customer classes experience much lower service qualities, with long response times with high standard deviations. From Table 2, we can also see that customers of European/UK and Non-EU passengers encounter long queues at passport control and security check points.

4 CONCLUSION AND FUTURE WORK

This paper 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 location information of entities in the system and allows meaning to be assigned to their spatial relationships. To facilitate studies on location tracking data mining and analy-

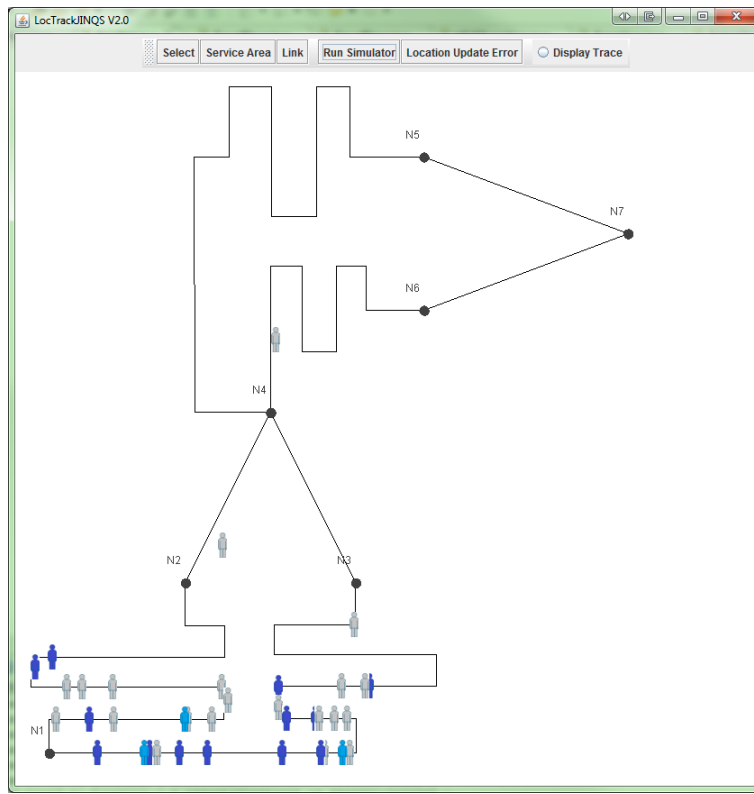


Figure 8: The airport simulation in an early stage.

sis, LOCTRACKJINQS provides functionalities for generating synthetic location tracking data from the simulation environment. LOCTRACKJINQS also provides a graphical user interface to support visual construction of queueing network models and parameter setting for the simulation. This was demonstrated in the context of an airport case study. Possible future work includes support for more sophisticated customer routing policies (e.g. join shortest queue), 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, the service areas can have different shapes from being circular; instead of having fixed locations, server points can move around providing service to customers (e.g. nurses visiting hospitalised patients).

References

- [1] N. Anastasiou, T.-C. Horng, and W. Knottenbelt. Deriving Generalised Stochastic Petri Net performance models from high-precision location tracking data. In *Proc. 5th International ICST Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2011)*, Paris, France, May 2011. To appear.

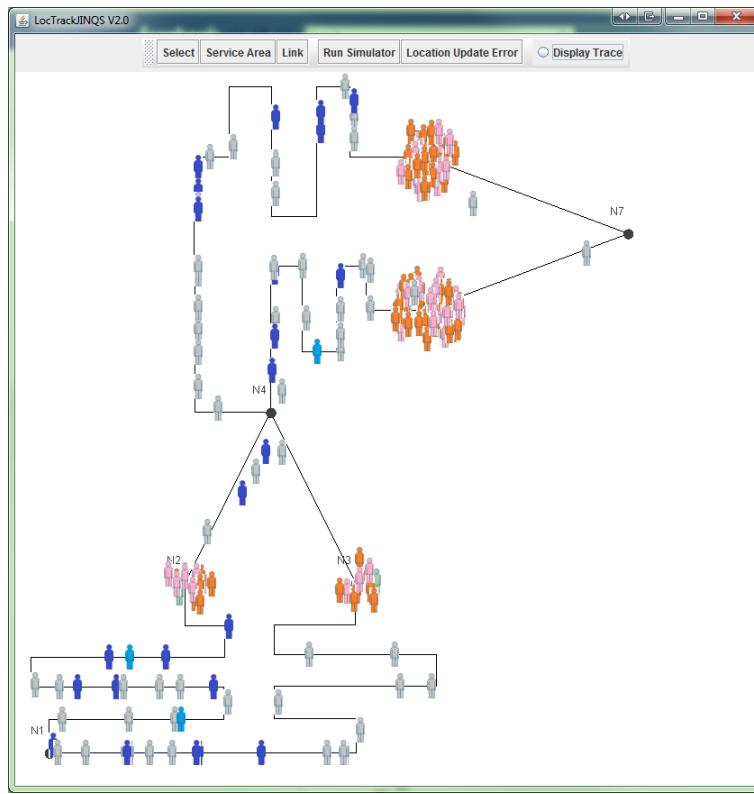


Figure 9: The airport simulation as time progresses.

- [2] M. Baudin and A. Rao. RFID applications in manufacturing. Technical report, Manufacturing Management and Technology Institute, 2005.
- [3] S. Boschi, M. Di Ianni, P. Crescenzi, G. Rossi, and P. Vocca. MOMOSE: a mobility model simulation environment for mobile wireless ad-hoc networks. In *Proc. 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems Workshops*, SIMU-TOOLS '08, pages 38:1–38:10, 2008.
- [4] Cambridge Systematics Inc., PB Consult, and System Metric Group Inc. Analytical tools for asset management. Technical Report 545, National Cooperative Highway Research Program, 2005.
- [5] T. Field. JINQS: An Extensible Library for Simulating Multiclass Queueing Networks V1.0. <http://www.doc.ic.ac.uk/~ajf/Research/manual.pdf>, 2006.
- [6] 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.
- [7] T.-C. Horng, N. Dingle, A. Jackson, and W. Knottenbelt. Towards the automated inference of queueing network models from high-precision loca-

		Passport Control 1 (N2)	Passport Control 2 (N3)	Security Check 1 (N5)	Security Check 2 (N6)
Class 0	μ_{rt}	8.754	4.830	24.979	39.0346
	σ_{rt}	4.793	2.918	17.221	22.602
Class 1	μ_{rt}	164.708	153.794	297.914	354.599
	σ_{rt}	97.861	84.034	177.663	216.591
Class 2	μ_{rt}	155.025	152.709	272.432	322.936
	σ_{rt}	88.892	96.924	200.560	178.817
	μ_q	49.747	37.124	54.837	50.263

Table 2: Mean (μ_{rt}) and standard deviation (σ_{rt}) – in seconds – of the response time for each customer class and service point of the airport simulation. Customer classes 0, 1 and 2 correspond to Pilots & Flight Attendants, EU passengers and non-EU passengers respectively. μ_q represents the mean queue length for each service point.

	Class 0	Class 1	Class 2	Aggregate
μ_{rt}	246.160	520.750	479.937	420.563
σ_{rt}	82.376	229.931	223.166	225.557

Table 3: Mean (μ_{rt}) and standard deviation (σ_{rt}) – in seconds – of the response time for each customer class, as well as the aggregate for the system. Customer classes 0, 1 and 2 correspond to Pilots & Flight Attendants, EU passengers and non-EU passengers respectively.

tion tracking data. In *Proc. 23rd European Conference on Modelling and Simulation (ECMS 2009)*, pages 664–674, May 2009.

- [8] 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.
- [9] P. L. Markt and M. H. Mayer. WITNESS simulation software: a flexible suite of simulation tools. In *Proc. 29th Winter Simulation Conference, WSC '97*, pages 711–717, Washington, DC, USA, 1997.
- [10] Simul8. <http://www.simul8.com/>.
- [11] M. Vankipuram, K. Kahol, T. Cohen, and V. L. Patel. Toward automated workflow analysis and visualization in clinical environments. *Journal of Biomedical Informatics*, 2010.