Performance modelling in the pub — speculative computation and dynamic pricing

Ashok Argent-Katwala[*]

Abstract

Using a simplified pub as a setting, we build a useful example for explaining assorted mechanisms in queueing theory. We consider 'ordinary' customers, negative customers, triggers and negative queue lengths.

The example is extended to provide an approach to dynamic pricing using a prioritised queue where different customers could pursue different strategies to fulfil their needs.

Supply of the key resource is time-consuming to create, can be pregenerated but decays after time and is wasted if a customer is not matched to it.

1 Introduction

We present a light-hearted approach to explain some concepts from queueing theory by setting them in a comfortable and hopefully familiar environment.

Our simplified pub is built up in section 2, starting from a simple queue then adding negative customers, negative queue lengths and triggers to help explain each queueing concept in turn.

Dynamic pricing is discussed in section 3, looking to extending the model to support mini-auctions for the customers. Section 4 considers some possible extensions to the model and we conclude in section 5.

2 The Model

2.1 Simple queue

Figure 1 shows our simplest model, a first-come first-served (FCFS) queue, where arrivals and departures are Poisson processes with rates λ and μ respectively. Each customer orders a single pint of stout, which is not poured until they reach the front of the queue, and the next customer is not dealt with until the pint has settled and been given to the customer. It is reasonable to treat service as a Poisson process since, despite Guinness advertising, there is still considerable variation in how quickly each pint is poured, depending on a variety of factors.

[*]Department of Computing, Imperial College of Science, Technology and Medicine, 180 Queen's Gate, London, SW7 2BZ. email abkk97@doc.ic.ac.uk

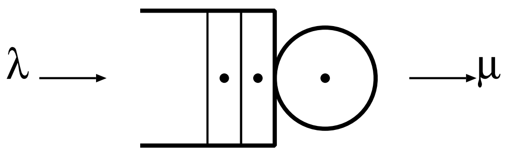

Figure 1: Simple queue

2.2 Multiple servers

We could take up to the first k people in the queue as being in service, using a service centre which processes the first k at the head of a single bar queue, each with some processing rate, μ_i, as shown in figure 2. This represents having k bar staff on duty, or smarter bar staff who deal with several orders at once, as is typical (the limiting factor often being the flow of the taps).

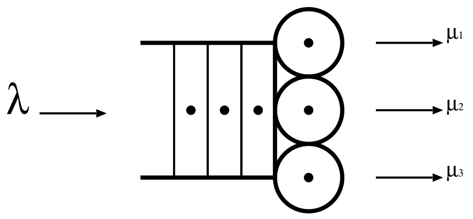

Figure 2: Multiple Servers

2.3 Negative customers

Now consider a customer who, while queueing, is thrown out of the bar (the details are left to your imagination). This corresponds in the model to the arrival of negative customers, which arrive as an independent Poisson process with rate κ_{exit}, as shown in figure 3. On arriving at an empty queue traditional negative customers—as introduced by Gelenbe et al in [Ge91]—disappear and have no effect.

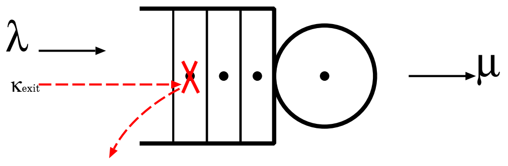

Figure 3: Negative customers

When a negative customer arrives to a queue where everyone is in service we may or may not allow preemption and lose the customer in service. We may also want to preserve the work done on their behalf. We handle something like this in section 2.6 by adding an extra queue.

There is also merit in considering negative customers which remove particular people from the queue—for instance in a shadier pub than this the police may drag someone away for a particular crime, rather than wanting to pull anyone out of the queue. In terms of the model this would be best represented by a negative customers acting only on certain classes of customer. Gelenbe and others have used this approach in his G-Networks and still retained a product form steady-state solution [Fo95].

We can also use the notion of negative customers to represent service completion, rather than expulsion, as follows.

2.4 Negative queue length

We could treat a service completion from the bar queue as a kind of negative customer arriving. By allowing the queue length to go below zero we can cleanly represent a buffer of pints poured, standing on the bar waiting for customers to buy them. A new arrival to such a system is 'serviced' immediately, and brings the queue length up by one (towards zero).

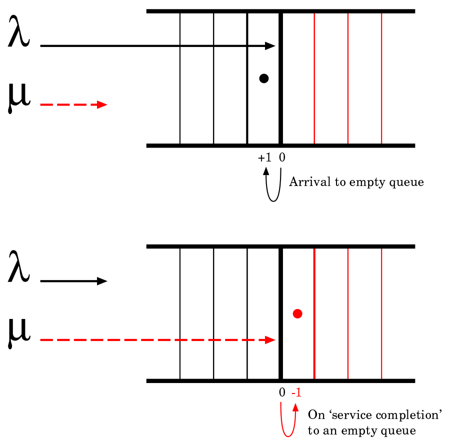

Figure 4: Negative queue length

Plainly this is only particularly useful where the work done for one customer is much the same as that done for any other, otherwise a new arrival would still need more work done to service them. We could allow for some portion of the work to be useful to another client, and some of it wasted. This could be represented explicitly by carrying a weight on the customers below zero (which may age, as discussed in section 3 using ideas from [Ha02]) or by splitting out the non-transferable work into an extra processing node.

In some sense, which is the 'positive' arrival is arbitrary, and the two streams of arrivals are just pushing the state along a one-dimensional line.

2.5 Triggers

Triggers are events that can fire when a customer—negative or positive—arrives at a particular queue. They cause an arrival at another queue, and may themselves also cause triggers to fire.

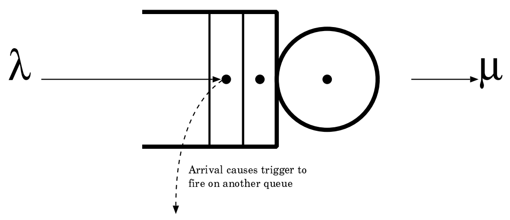

Figure 5: Triggers

In the context of a pub this is akin to a regular arriving. As well as queueing at the bar, the manager will also likely say hello or have a chat. If the manager is already chatting to a regular, then he may mentally note to chat to the new arrival later. So, the new arrival is essentially queueing for his attention, in parallel to queueing at the bar. Moreover, a service completion after talking to the manager may cause the customer to be promoted to the head of the queue at the bar out of turn.

We can also make use of them to monitor particular facets of the system, and maintain the progression we intend, as shall be seen in the next example.

2.6 Speculative work and wastage

We now add an explicit pair of queues to represent a pint being poured, and it lingering before becoming no use—going flat, say—and being thrown away as wastage. The bar queue is now constrained to having only a positive queue length. Arrivals to the pouring queue are controlled by people in the queue at the bar. By using a positive trigger for every arrival to the pouring queue, everyone at the bar's service begins swiftly (compared with waiting for each pint to settle in turn before moving on).

When a pint leaves the standing queue, it causes an ordinary negative trigger on the bar queue. If the bar queue is empty this is then wasted, since it will leave the queue length at zero.

Note that a customer will get the pint when it leaves the pouring queue, which triggers a new negative arrival at the standing queue, ensuring the same pint isn't drunk twice.

Thus we have two sorts of negative arrival, one representing service completion—removing the head of the queue—and one culling from the back, our drunk and disorderly customers from section 2.3. Notice that we are using ordinary negative customers, which have no effect on an empty queue. Thus if the final transition μ_{decay} fires and there is no-one waiting at the bar, the pint is wasted, see figure 6.

For a more complex system, we might not always fire the positive trigger from the bar queue, or speculatively pour some number of pints too (with rate λ_{spec}, which could reasonably be tied to the bar queue length).

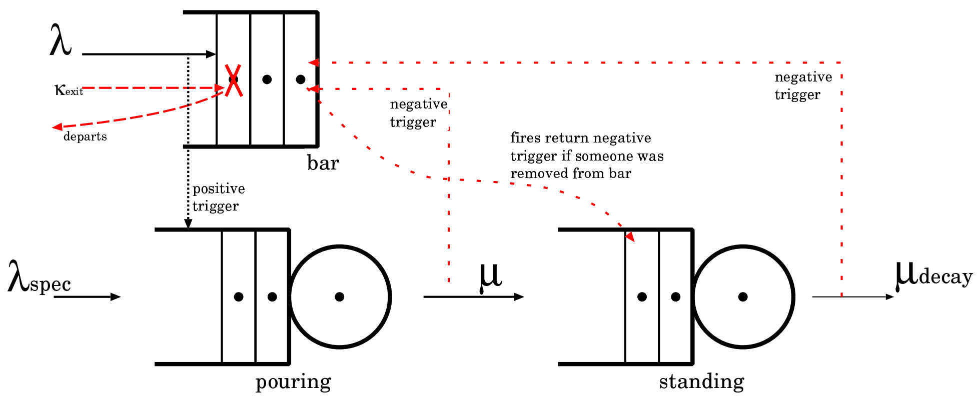

Figure 6: Speculative work and wastage

3 Dynamic pricing

Next, we modify the queue at the bar to add weights to each customer. A new arrival comes in with some weight—which represents how much they are willing to pay for their pint—and is inserted in order.

In [Ha02], customers join the queue with a certain weight and the whole queue ages over time. By extending this it should be possible to age each customer according to their personal strategy. This would behave like a mini-auction after each arrival or departure and the system could model quite complex bidding mechanisms—especially where a customer can modify their bid based on their position in the queue, and the bids of others in the queue.

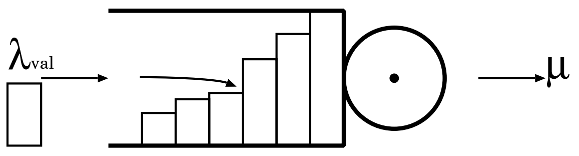

Figure 7: Weighted queue, inserting in order

4 Extensions

There are a number of ways to make the model richer, including:

- Typed orders—of a variety of different drinks—and proper orders each with a combination of drinks. Then, transfers could only be made to compatible orders.
- Allowing richer combinations of orders, and carrying forward only a proportion of work that is useful to a future customer

- Batched arrivals, as friends will often arrive together
- State-dependent rates, as people are more or less likely to come in to a crowded bar and service rates might suffer when the staff are overworked.
- Better represent the pool of people in the pub, rather than just those trying to get served.

5 Conclusions

By considering queueing theory in the pub, over some beer, we can find analogues of queueing mechanisms which make them easier to explain. Future work will hopefully develop a practical approach to dynamic pricing building on the initial ideas presented here.

Acknowledgements

Thanks go to my Ph.D supervisor, Prof. Pete Harrison, and the other members of the AESOP research group for their encouragement and pub-based research. The original motivation came over a pint of Guinness with Rob Chatley.

References

[Ge91] E. Gelenbe, P. Glynn and K. Sigmann *Queues with negative arrivals*, 1991

[Fo95] J.M. Fourneau, E. Gelenbe and R. Suros *G-Networks with multiple class negative and positive customers*, 1995

[Ha02] P. G. Harrison, *An M/M/1 Queue with Aging Priority*, 2002