Queueing Networks

Teletraffic Engineering and Network Planning

Dario Gjorgjevski[1]
gjorgjevski.dario@students.finki.ukim.mk

[1]Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University in Skopje

May 26, 2016

Outline

The motivation behind queueing networks

- In real systems, jobs often receive service from multiple successive nodes.
- The total service demand is composed of demand at all these nodes.
- Hence, we have a network of queues – a *queueing network*.
- Arrival process at one queue is a departure process at another queue and vice-versa.

Basic definitions to remember

- The queueing network is composed of individual queues, each called a *node*.
- The total number of jobs (incl. delayed and served jobs) at a node is called the *queue length*.
- Similarly, the *waiting (sojourn)* time includes both the *delay* and the *service* times.
- Two types of queueing networks:
  - *Closed* – fixed number of "customers;" and
  - *Open* – varying number of "customers."

Basic definitions to remember

- $M/M/n$ is a classical example of an open network.
- Palm's machine-repair model is a classical example of a closed network.
- More types of customers can lead to a network being *both* open and closed.
- Describe the state by $p(x_1, \ldots, x_K)$, x_k is the number of customers at node k ($k = 1, \ldots, K$).
- Every node is reversible $\implies$ $p(x_1, \ldots, x_K)$ can be written as a product form.

Considering multiple types of customers

- Multiple types of customers, each forming a *chain*.

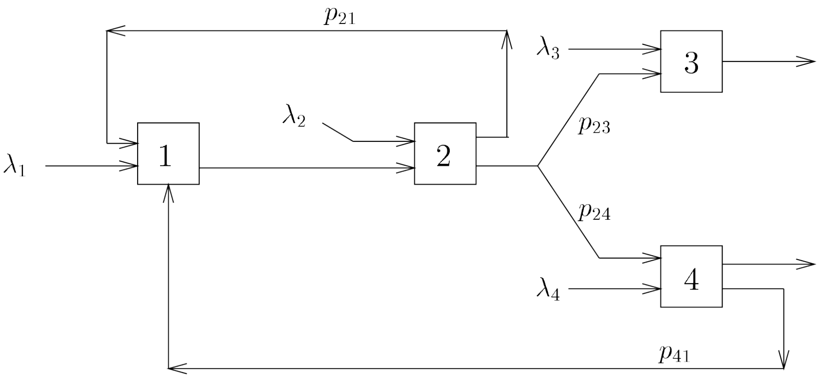

Figure: A queueing network with four chains

Outline

What is reversibility?

Running the queue "backwards in time" gives the same queue.

Definition (Reversibility)

A queue is reversible if and only if there is no circulation flow, i.e., the circulation flow among four neighboring states in a square equals 0.

Kolmogorov's cycle criterion proves that this is a necessary and sufficient condition for a queue (or more generally, a Markov process) to be reversible.

The importance of reversibility

- Reversibility means that the departure process is the same as the arrival process.
- In other words, if the arrival process is Poisson, so is the departure.

$\Longrightarrow$ This simplifies things greatly.

The $M/M/n$ case

Theorem (Burke)

The departure process of an $M/M/n$ system is a Poisson process. The state probabilities are given by

$$\frac{p(x)}{p(0)} = \begin{cases} \frac{A^x}{x!} & \text{if } 0 \leq x \leq n \\ \frac{A^x}{n! n^{x-n}} & \text{if } x > n, \end{cases}$$

where $A = \lambda/\mu$ and $p(0)$ is found by solving for normalization conditions.

Other cases

Other well known cases ($M/M/1$, IS, $M/G/1$-PS, $M/G/n$-GPS, $M/G/1$-LCFS-PR) have also been proven to have Poisson departure processes.

Example ($M/M/1$)

It may seem counter-intuitive that the departure process of an $M/M/1$ system with arrival rate λ and service rate μ is a Poisson process with rate λ. However, decomposing the departure process into Cox-distributions where one branch corresponds to idle, and the other to busy periods provides a nice graphical proof.

Outline

The setup we will be looking at

- Each node is an $M/M/n$ system with n_k servers and μ_k service rate.
- Jobs arrive at node k according to a Poisson process with rate λ_k.
- Jobs may arrive at node k from other nodes.
- A job leaving node j is transferred to k with probability p_{jk} or leaves the network with probability $1 - \sum_{k=1}^{K} p_{jk}$.

Calculating the arrival intensity

- Solve the balance equations to calculate for the arrival rates at each node k:

$$\Lambda_k = \lambda_k + \sum_{j=1}^{K} \Lambda_j p_{jk}. \tag{1}$$

- Assume

$$\frac{\Lambda_k}{\mu_k} = A_k \leq n_k,$$

otherwise queue length $\rightarrow \infty$.

Jackson's theorem

Theorem (Jackson [3])

Under the aforementioned setup,

$$p(x_1, \ldots, x_K) = \prod_{k=1}^{K} p_k(x_k).$$

In other words, we can consider each node independently with state probabilities as in Erlang's delay system ($M/M/n$).

Performance measures

- Total throughput $= \Lambda = \sum_{k=1}^{K} \Lambda_k$.
- Average load at node $k = \dfrac{\Lambda_k}{\mu_k}$.
- Visits to node $k = \dfrac{\Lambda_k}{\Lambda}$.
- Average # of jobs at node $k = N_k = \sum_{n=0}^{\infty} n p_k(n)$.
- Mean sojourn time $= \sum_{k=1}^{K} \dfrac{N_k}{\Lambda}$.

Reducing multiple chains to a single chain

- Solve the flow balance equations for each chain, obtaining the arrival intensity from chain j to node k (Λ_{jk}).
- Exploit local balance to solve for the state probabilities.
- Apply Jackson's theorem.

Outline

Initial thoughts on closed networks

- Assume S customers are circulating within K nodes.
- Handling closed networks is very complex as we don't know the true arrival rate.
- We are again interested in $p(x_1, \dots, x_K)$.
- Knowing the arrival rate at a single node lets us solve the balance equations for the relative arrival rates to other nodes.
- Relative rates still need to be normalized: $$\binom{S+K-1}{K-1}$$ terms to sum over naively.
- Buzen's convolution algorithm lets us do the normalization in $\mathcal{O}(SK)$ time.

The Gordon–Newell theorem

Theorem (Gordon–Newell [2])

In a closed networks with S customers and K nodes with arrival rates Λ_k,

$$p(x_1, \ldots, x_K) = \frac{1}{G(S)} \prod_{k=1}^{K} \left(\frac{\Lambda_k}{\mu_k}\right)^{x_k},$$

where the Λ_k are found by solving the balance equations and $G(S)$ is a normalization constant:

$$G(S) = \sum_{(x_1,\ldots,x_K)} \prod_{k=1}^{K} \left(\frac{\Lambda_k}{\mu_k}\right)^{x_k}.$$

Calculating $G(S)$

The convolution algorithm

We can exploit the structure of the network to calculate $G(S)$ relatively efficiently [1].

1. Consider each node individually as if offered PCT-I traffic $a_k = \frac{\Lambda_K}{\mu_k}$, and calculate the relative state probabilities $q_k(x_k)$.
2. Convolve the probabilities of the nodes recursively, e.g.,

$$q_{1,\dots,i,\dots,k} = q_{1,\dots,i-1,i+1,\dots,k} * q_i.$$

The above procedure terminates as soon as $q_{1,\dots,K}$ has been computed. It can be shown that $G(S) = q_{1,\dots,K}$.

An example of applying the convolution algorithm

Example (Palm's machine-repair model)

Assume a computer system with S jobs and terminals, and one server. The *mean thinking time* is μ_1^{-1}, and the *mean service time* is μ_2^{-1}. In other words, there are two nodes: the terminals and the server. The relative loads are $a_1 = \Lambda/\mu_1$ and $a_2 = \Lambda/\mu_2$. By convolving $q_1(i)$ and $q_2(j)$, we get

$$\begin{aligned} q_{1,2}(S) &= (q_1 * q_2)(S) \\ &= a_1^0 a_2^S + a_1^1 a_2^{S-1} + \frac{a_1^2}{2!} a_2^{S-2} + \ldots + \frac{a_1^S}{S!} a_2^0. \end{aligned}$$

Palm's machine-repair model

The probability that all terminals are thinking is identified as $\frac{q_1(S)q_2(0)}{q_{1,2}(S)}$, which agrees with Erlang's B-formula.

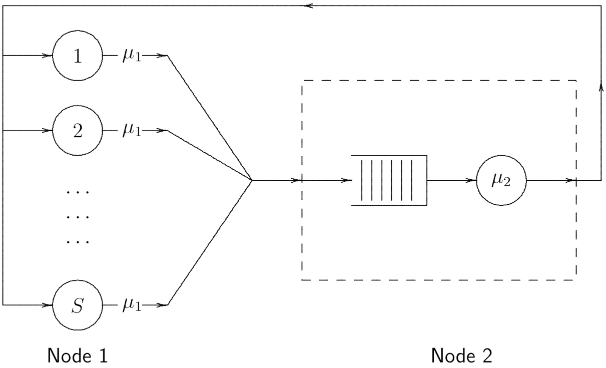

Figure: Palm's machine-repair model

The full calculation in Palm's machine-repair model

Table: The algorithm applied to Palm's machine-repair model

x	$q_1(x_1)$	$q_2(x_2)$	$q_{1,2}(x) = (q_1 * q_2)(x)$
0	1	1	1
1	a_1	a_2	$a_1 + a_2$
2	$\frac{a_1^2}{2!}$	a_2^2	$a_2^2 + a_1 a_2 + \frac{a_1^2}{2!}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
x	$\frac{a_1^x}{x!}$	a_2^x	$\vdots$
S	$\frac{a_1^S}{S!}$	a_2^S	$q_{1,2}(S)$

Preliminaries for the MVA

We will state two theorems that the MVA uses.

Theorem

For fully accessible systems with a limited number of sources, a random source will, upon arrival, observe the system as if the source itself does not belong to it.

Theorem (Little)

The mean queue length is equal to call intensity multiplied by the mean waiting time, i.e.,

$$L = \lambda W.$$

Definition of the MVA

- Let the average number of customers at node k be $L_k(S)$.
- Obviously, $\sum_{k=1}^{K} L_k(S) = S$.
- Now, proceed in two steps:
  1. Increase S to $S+1$. The average sojourn times become $W_k(S+1) = (L_k(S)+1)\mu_k^{-1}$ or μ^{-1}, for 1 and ∞ servers respectively.
  2. By Little's theorem, $L_k(S+1) = c\lambda_k W_K(S+1)$, where c is a normalizing constant.

These two steps allow us to compute performance measures efficiently. Note that the results are only approximate for multi-server systems.

Defining BCMP networks

- So far, when dealing with closed networks, we assumed a single chain.
- In 1975, Baskett, Chandy, Muntz, and Palacios showed that even closed networks with multiple chains admit product form solutions if they have either:
  - FCFS with exponential service times;
  - Processor sharing queues;
  - Infinite server queues;
  - LCFS with pre-emptive resume.
- In the last three cases, service time distributions must have rational Laplace transforms.
- Simple extensions to the convolution and mean-value algorithms.

References

- Jeffrey P. Buzen. "Computational Algorithms for Closed Queueing Networks with Exponential Servers". In: *Commun. ACM* 16.9 (Sept. 1973), pp. 527–531. ISSN: 0001-0782. DOI: 10.1145/362342.362345 (cit. on p. 21).
- William J. Gordon and Gordon F. Newell. "Closed Queuing Systems With Exponential Servers". In: *Operations Research* 15.2 (1967), pp. 254–265. DOI: 10.1287/opre.15.2.254 (cit. on p. 20).
- James R. Jackson. "Networks of Waiting Lines". In: *Operations Research* 5.4 (1957). DOI: 10.1287/opre.5.4.518 (cit. on p. 15).

FIN

Feel free to ask questions.