Performance Analysis

Peter Harrison

Imperial College, London

Course Details

- Lectures: Friday 3pm - 5pm, followed by tutorial until 6pm, in 144
- 20 lectures, 10 tutorials, 2-3 pieces of assessed coursework
- No coding
- Lecture notes at: `http://www.doc.ic.ac.uk/~uh/pgh/`
- This is the first year with typed notes
- Books:
  - *Performance modelling of communication networks and computer architectures* Peter G. Harrison, Naresh M. Patel. Wokingham : Addison-Wesley, 1992. ISBN: 0201544199, 14 copies in library, out of print
  - *Probabilistic Modelling* I. Mitrani, Paperback - 233 pages 2nd Ed (11 December, 1997) Cambridge University Press; ISBN: 0521585309, about £20

- The course aims to model real-world problems using analytical models
- Queueing theory
- *Uses* maths to solve problems
- Emphasis is on solving problems rather than making up hard sums
- Even if you do not choose to specialise in modelling, you will learn some "transferable" skills
- *Simulation and Modelling* by Tony Field, Wed. 11pm (145), Thu 9pm (343)

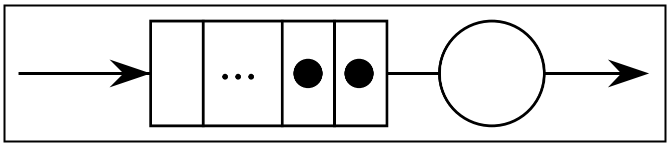
- Performance analysis
- …Modelling
- …Planning
- …Optimisation

- Real-world problem: Post Office (average length of queues, Utilisation, etc.)
- Abstraction, approximation, summary
- Assumptions
- Model
- Techniques for solution
- Answer to problem
- Abstraction helps to transfer solution to other problems (PO, computer)

1 Previews of selected examples and case studies...

1.1 Example 1: A Simple TP server

A TP system accepts and processes a stream of transactions, mediated through a (large) buffer:

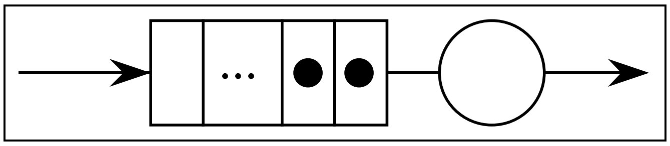


- Transactions arrive "randomly" at some specified rate
- The TP server is capable of servicing transactions at a given service *rate*

Q: If both the arrival rate and service rate are doubled, what happens to the mean response time?

1.2 Example 2: A simple TP server

Consider the same system as above:

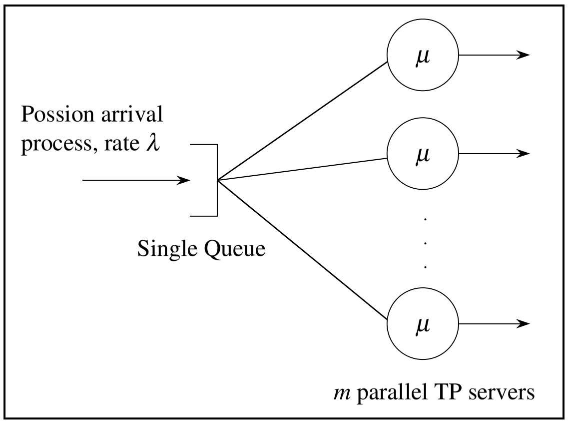


- The arrival rate is 15tps
- The mean service time per transaction is 58.37ms

Q: What happens to the mean response time if the arrival rate increases by 10%?

1.3 Example 3: A simple multiprocessor TP system

Consider our TP system but this time with multiple transaction processors

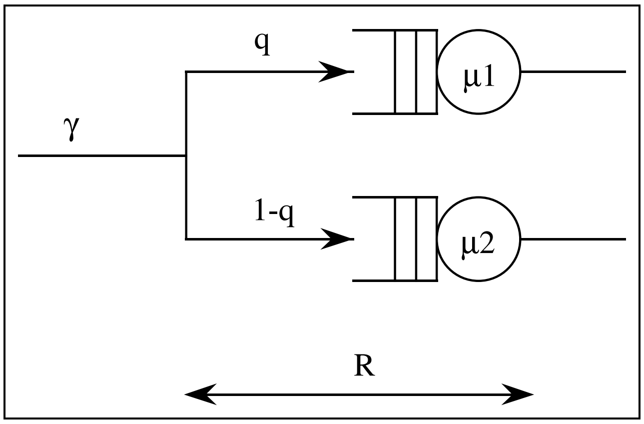


- The arrival rate is 16.5 tps
- The mean service time per transaction is 58.37ms

Q: By how much is the system response time reduced by adding one processor?

Example 4: File Allocation

What is the best way to allocate disk blocks to a heterogenous disk I/O system?

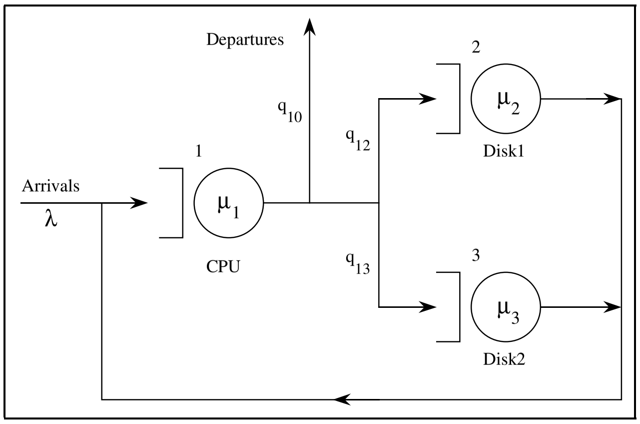


- Disk I/O requests are made at an average rate of 20 per second
- Disk blocks can be located on either disk and the mean disk access times are 30ms and 46ms respectively

Q: What is the optimal proportion of blocks to allocate to disk 1 to minimise average response time?

1.4 Example 5: A Simple Computer Model

Consider an open uniprocessor CPU system with just disks

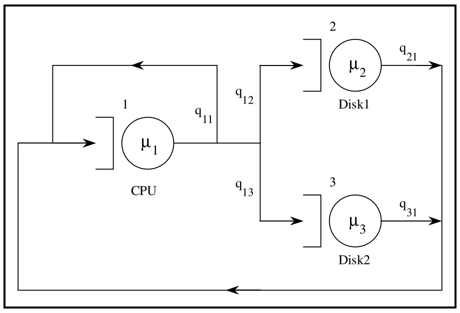


- Each submitted job makes 121 visits to the CPU, 70 to disk 1 and 50 to disk 2 *on average*
- The mean service times are 5ms for the CPU, 30ms for disk 1 and 37ms for disk 2

Q: What is the effect of replacing the CPU with one *twice* the speed?

Example 6: A Simple Batch Processor System

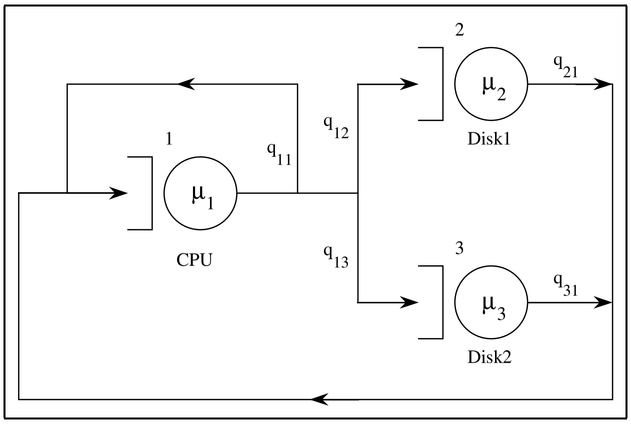
How does the above system perform in batch mode?



- Each batch job makes 121 visits to the CPU, 70 to disk 1, 50 to disk 2 *on average*
- The mean service times are 5ms for the CPU, 30ms for disk 1 and 37ms for disk 2

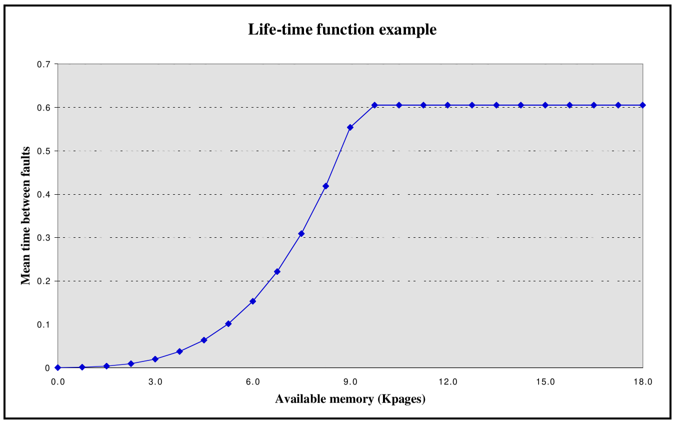
Q: How does the system throughput vary with the number of batch jobs and what is the effect of replacing Disk 1 with one (a) twice and (b) three times the speed?

Example 7: A multiprogramming system with virtual memory



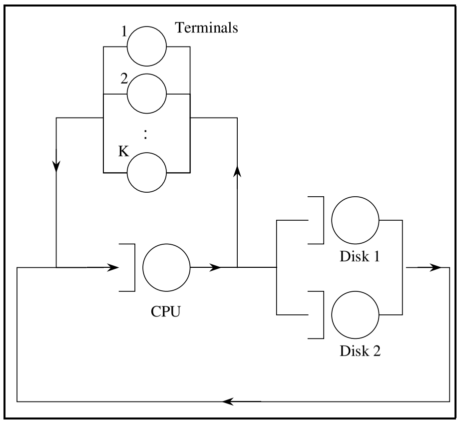
- Suppose now we add VM and make disk 1 a dedicated paging device
- Pages are 1Kbyte in size and the (usable) memory is equivalent to 64K pages

- Each job page faults at a rate determined by the following lifetime function:



Q: What number of batch jobs keeps the system throughput at its maximum and at what point does thrashing occur?

1.5 Example 8: A multiaccess multiprogramming system with virtual memory



- During the day the system runs in interactive mode with a number of terminal users
- The average think time of each user is 30 seconds

Q: How does the system response time and throughput vary with the number of terminals and how many terminals can be supported before the system starts to thrash?

2 Stochastic Processes

2.1 Introduction

Computer systems are

1. dynamic - they can pass through a succession of states as time progresses
2. influenced by events which we consider here as random phenomena

c.f. bank, supermarket, stock exchange

<u>Definition</u>: A stochastic process $\boldsymbol{S}$ is a family of random variables $\{X_t \in \Omega | t \in T\}$, each defined on some (the same for each) *sample space* Ω for a *parameter space* T.

- T, Ω may be either discrete or continuous

- T is normally regarded as time
  - real time: continuous
  - every month or after job completion: discrete
- Ω is the set of values each X_t may take
  - bank balance: discrete
  - number of active tasks: discrete
  - time delay in communication network: continuous

2.2 Example: The Poisson process

The Poisson process is a renewal process with renewal period (interarrival time) distribution and probability density function (pdf)

$$\begin{aligned} F(x) &= P(X \leq x) = 1 - e^{\lambda x} \\ \text{pdf: } f(x) &= F'(x) = \lambda e^{-\lambda x} \end{aligned}$$

λ is the parameter or rate of the Poisson process.

2.3 Memoryless Property of the (negative) exponential distribution

If S is an exponential random variable

$$P(S \leq t + s | S > t) = P(S \leq s) \quad \forall t, s \geq 0$$

(i.e. it doesn't matter what happened before time t)

Proof

$$
\begin{aligned}
P(S \le t+s | S > t) &= \frac{P(t < S \le t+s)}{P(S > t)} \\
&= \frac{P(S \le t+s) - P(S \le t)}{1 - P(S \le t)} \\
&= \frac{1 - e^{-\lambda(t+s)} - (1 - e^{-\lambda t})}{e^{-\lambda t}} \\
& \quad (\lambda \text{ is the rate of the exp. distribution}) \\
&= \frac{e^{-\lambda t} - e^{-\lambda(t+s)}}{e^{-\lambda t}} = 1 - e^{-\lambda s} \\
&= P(S \le s)
\end{aligned}
$$

2.4 More properties of the Poisson process

$$
\begin{aligned}
P(\text{arrival in } (t, t+h)) &= P(R \le h) = P(S \le h) \quad \forall t \\
&= 1 - e^{-\lambda h} \\
&= \lambda h + o(h)
\end{aligned}
$$

Therefore

1. Probability of an arrival in $(t, t+h)$ is $\lambda h + o(h)$ regardless of process history before t
2. Probability of more than one arrival in $(t, t+h)$ is $o(h)$ regardless of process history before t

In fact we can take this result as an alternative definition of the Poisson process. From it we can derive the distribution function of the interarrival times (negative exponential) and the Poisson distribution for N_t (the *arrival process*)

$$P(N_t = n) = \frac{(\lambda t)^n}{n!} e^{-\lambda t}.$$

2.5 Derivation of the Poisson process

The proof of the claim is by induction on the integers. Let
$P_n(t) = (P(N_t = n)$.
Base case $(n = 0)$

$$\begin{aligned}
P_0(t+h) &= P(N_t = 0 \text{ \& no arrivals in } (t, t+h)) \\
&= P_0(t)(1 - \lambda h + o(h)) \quad \text{(by independence)} \\
\implies \frac{P_0(t+h) - P_0(t)}{h} &= -\lambda P_0(t) + o(1) \\
P_0'(t) &= -\lambda P_0(t) \quad \text{as } h \to 0 \\
\implies P_0(t) &= Ce^{-\lambda t} \\
P_0(0) &= P(N_0 = 0) = 1 \text{ therefore } C = 1
\end{aligned}$$

Inductive Step ($n > 0$)

$$\begin{aligned} P_n(t+h) &= P_n(t)P_0(h) + P_{n-1}(h)P_1(h) + o(h) \\ &= (1-\lambda h)P_n(t) + \lambda h P_{n-1}(t) + o(h) \\ \Longrightarrow P_n'(t) &= -\lambda P_n(t) + \lambda P_{n-1}(t) \\ \Longrightarrow \frac{d}{dt}(e^{\lambda t}P_n(t)) &= \lambda e^{\lambda t} P_{n-1}(t) \end{aligned}$$

Inductive hypothesis:

$$P_{n-1}(t) = e^{-\lambda t}\frac{(\lambda t)^{n-1}}{(n-1)!}$$

This is OK for $n = 1$. Then

$$\begin{aligned} \frac{d}{dt}(e^{\lambda t}P_n(t)) &= \frac{(\lambda)^n}{(n-1)!}t^{n-1} \\ \Longrightarrow e^{\lambda t}P_n(t) &= \frac{(\lambda t)^n}{(n)!} + D \end{aligned}$$

For $t = 0$, $P_n(t) = 0$ for $n \geq 2$ and so $D = 0$. Thus

$$P_n(t) = e^{-\lambda t}\frac{\lambda t}{n!}$$

2.6 Derivation of the interarrival time distribution

1. Using the derived Poisson distribution

$$\begin{aligned} P(T \le t) &= 1 - P(0 \text{ arrivals in } (0,t)) \\ &= 1 - e^{-\lambda t} \end{aligned}$$

   (distribution for negative exponential random variable)

2. Directly from assumptions (exercise). Use the memoryless property to simplify the equation

$$P(T \le t+h) = P((T \le t) \vee [(T > t) \wedge (1 \text{ arrival in } (t, t+h))])$$

   hence derive and solve a differential equation for
   $P(T \le t) = F(t)$.

2.7 Superposition Property

If $A_1, \ldots, A_n$ are independent Poisson Processes with rates $\lambda_1, \ldots, \lambda_n$ respectively, let there be K_i arrivals in an interval of length t from process A_i $(1 \le i \le n)$. Then $K = K_1 + \cdots + K_n$ has Poisson distribution with parameter λt where $\lambda = \lambda_1 + \cdots + \lambda_n$. I.e. the superposition of PPs with rates λ_i is a PP with rate $\sum_i \lambda_i$.

Proof

The distribution of K is the convolution of the distributions of the K_i which are Poisson. E.g. if $n = 2$

$$\begin{aligned} P(K=k) &= \sum_{i=0}^{k} \frac{(\lambda_1 t)^i}{i!} e^{-\lambda_1 t} \frac{(\lambda_2 t)^{k-i}}{(k-i)!} e^{-\lambda_2 t} \\ &= \frac{e^{-(\lambda_1+\lambda_2)t}}{k!} \sum_{i=0}^{k} \binom{k}{i} (\lambda_1 t)^i (\lambda_2 t)^{k-i} \\ &= e^{-(\lambda_1+\lambda_2)t} \frac{[(\lambda_1+\lambda_2)t]^k}{k!} \end{aligned}$$

as required. The proof for arbitrary $n \geq 2$ is an easy induction on n.

2.8 Decomposition Property

If a Poisson Process is decomposed into processes $B_1, \ldots, B_n$ by assigning each arrival of A to B_i with independent probability q_i ($\sum q_i = 1$), then $B_1, \ldots, B_n$ are independent Poisson Processes with rates $q_1\lambda, \ldots, q_n\lambda$.

Example: Two parallel processors, B_1 and B_2. Incoming jobs (Poisson arrivals A) are directed to B_1 with probability q_1 and to B_2 with probability q_2. Then each processor "sees" a Poisson arrival for its incoming job stream.

Proof

For $n = 2$, let K_i = number of arrivals to B_i in time t $(i = 1, 2)$, $K = K_1 + K_2$

$$\underbrace{P(K_1 = k_1, K_2 = k_2)}_{\text{joint probability}} = \underbrace{P(K_1 = k_1, K_2 = k_2 | K = k_1 + k_2)}_{\text{binomial}} \underbrace{P(K = k_1 + k_2)}_{\text{Poisson}}$$

$$= \frac{(k_1+k_2)!}{k_1!k_2!} q_1^{k_1} q_2^{k_2} \frac{(\lambda t)^{k_1+k_2}}{(k_1+k_2)!} e^{-\lambda t}$$

$$= \frac{(q_1 \lambda t)^{k_1}}{k_1!} e^{-q_1 \lambda t} \frac{(q_2 \lambda t)^{k_2}}{k_2!} e^{-q_2 \lambda t}$$

I.e. a product of two independent Poisson distributions

$(n \geq 2$: easy induction)

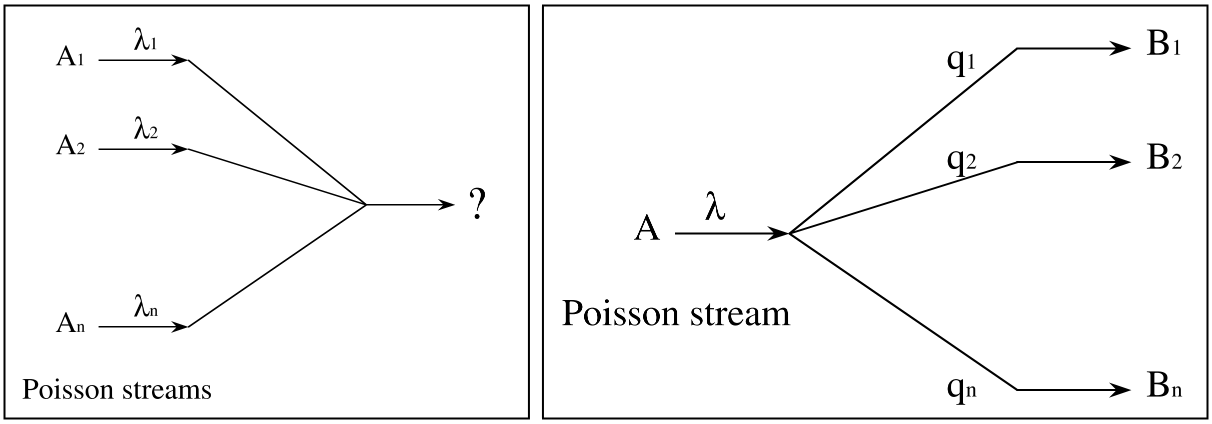


Figure 1: Superposition and decomposition of a Poisson process

2.9 Markov Chains and Markov Processes

- Special class of stochastic processes that satisfy the Markov property (MP): Given the state of the process at time t, its state at time $t+s$ has probability distribution which is a function of s only.
- i.e. the future behaviour after t is independent of the behaviour before t
- Often intuitively reasonable, yet sufficiently "special" to facilitate effective mathematical analysis
- We consider processes with discrete state (sample) space and:
  1. discrete parameter space (times $\{t_0, t_1, \dots\}$), a Markov chain or *Discrete Time Markov Chain*
  2. continuous parameter space (times $t \geq 0, t \in R$), a Markov Process or *Continuous Time Markov Chain*. E.g. number of arrivals in $(0, t)$ from a Poisson arrival process defines a Markov Process because of the memoryless property.

2.10 Markov Chains

- Let $X = \{X_n | n = 0, 1, \dots\}$ be an integer valued Markov chain (MC), $X_i \geq 0, X_i \in Z, i \geq 0$. MP states that:

$$P(X_{n+1} = j | X_0 = x_0, \dots, X_n = x_n) = P(X_{n+1} = j | X_n = x_n)$$

$$\text{for } j, n = 0, 1, \dots$$

- Evolution of a MC is completely described by its 1-step transition probabilities

$$q_{ij}(n) = P(X_{n+1} = j | X_n = i) \text{ for } i, j, n, \geq 0$$

Assumption: $q_{ij}(n) = q_{ij}$ is independent of time n (*time homogeneous*)

$$\sum_{j \in \Omega} q_{ij} = 1 \quad \forall i \in \Omega$$

- MC defines a transition probability matrix

$$Q = \begin{bmatrix} q_{00} & q_{01} & \cdots \\ q_{10} & q_{11} & \cdots \\ \vdots & & \\ q_{i0} & \cdots & q_{ii} \cdots \\ \vdots & & \end{bmatrix} \text{ in which all rows sum to 1}$$

  - dimension = # of states in Ω if finite, otherwise countably infinite
  - conversely, any real matrix Q s.t. $q_{ij} \geq 0$, $\sum_j q_{ij} = 1$ (called a *stochastic matrix*) defines a MC

2.10.1 Examples

1. <u>Telephone line</u>:
   Line is either idle (state 0) or busy (state 1).

$$\begin{aligned} \text{Idle at time } n &\implies \text{idle at time } (n+1) \text{ w.p. } 0.9 \\ \text{Idle at time } n &\implies \text{busy at time } (n+1) \text{ w.p. } 0.1 \\ \text{Busy at time } n &\implies \text{idle at time } (n+1) \text{ w.p. } 0.3 \\ \text{Busy at time } n &\implies \text{busy at time } (n+1) \text{ w.p. } 0.7 \end{aligned} \qquad Q = \begin{bmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{bmatrix}$$

   May be represented by a state diagram 2:

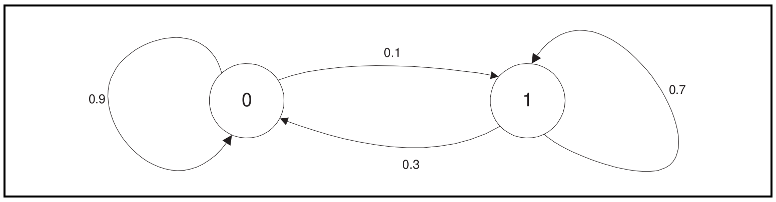


Figure 2: Telephone line example

2. Gambler
   Bets £1 at a time on the toss of fair die. Loses if number is less than 5, wins £2 if 5 or 6. Stops when broke or holds £100. If X_0 is initial capital $(0 \leq X_0 \leq 100)$ and X_n is the capital held after n tosses $\{X_n | n \leq 0, 1, 2, \ldots\}$ is a MC with $q_{ii} = 1$ if $i = 0$ or $i \geq 100$, $q_{i,i-1} = \frac{2}{3}$, $q_{i,i+2} = \frac{1}{3}$ for $i = 1, 2, \ldots, 99$ $q_{ij} = 0$ otherwise.

3. I/O buffer, capacity M records
   New record added in any unit of time w.p. α (if not full). Buffer emptied in any unit of time w.p β. If both occur in same interval, insertion done first. Let X_n be the number of records in buffer at (discrete) time n. Then, assuming that insertions and emptying are independent of each other and of their own past histories, $\{X_n | n = 0, 1, \ldots\}$ is a MC with state space $\{0, 1, \ldots, M\}$ and state diagram:

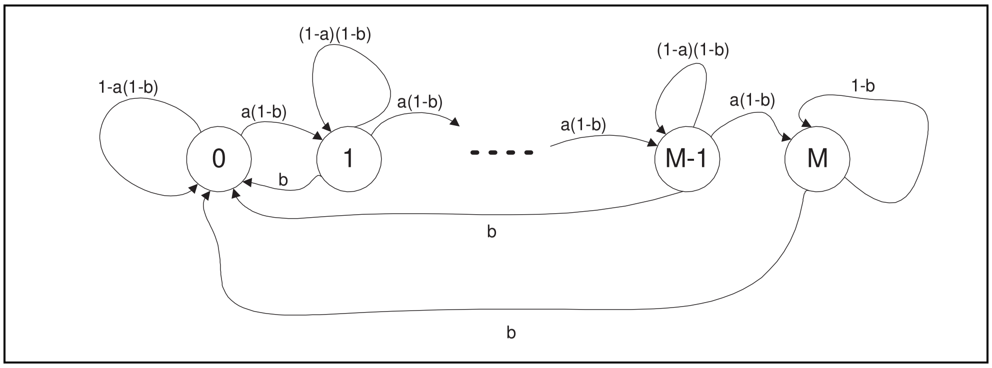


Figure 3: I/O buffer example

The transition matrix follows immediately, e.g.

$$q_{12} = \alpha(1-\beta) = q_{n,n+1}$$

$(0 \leq n \leq M-1)$

$$q_{MM} = 1-\beta$$

etc.

2.10.2 Multi-step transition probabilities

Let

$$\begin{aligned}
q_{ij}^{(2)} &= P(X_{n+2}=j|X_n=i) \\
&= \sum_{k\in\Omega} P(X_{n+1}=k, X_{n+2}=j|X_n=i) \text{ from law of tot. prob.} \\
&= \sum_{k\in\Omega} P(X_{n+2}=j|X_n=i, X_{n+1}=k)P(X_{n+1}=k|X_n=i) \\
&= \sum_{k\in\Omega} P(X_{n+2}=j|X_{n+1}=k)P(X_{n+1}=k|X_n=i) \text{ by MP} \\
&= \sum_{k\in\Omega} q_{ik}q_{kj} \text{ by TH} \\
&= (Q^2)_{ij}
\end{aligned}$$

Similarly, m-step transition probabilities :

$$\begin{aligned} q_{ij}^{(m)} &= P(X_{n+m} = j | X_n = i) \quad (m \geq 1) \\ &= (Q^m)_{ij} \end{aligned}$$

by induction on m. Therefore we can compute probabilistic behaviour of a MC over any finite period of time, in principle. E.g. average no. of records in buffer at time 50

$$E[X_{50}|X_0 = 0] = \sum_{j=1}^{\min(M,50)} j q_{0j}^{(50)}$$

Computationally intractable. Need to determine long-term behaviour: hope that asymptotically MC approaches a steady-state (probabilistically), i.e. that $\exists \{p_j | j = 0, 1, 2, \dots\}$ s.t.

$$p_j = \lim_{n\to\infty} P(X_n = j | X_0 = i) = \lim_{n\to\infty} q_{ij}^{(n)}$$

independent of i

2.10.3 Definitions and properties of MCs

Let v_{ij} = prob. that state j will eventually be entered, given that the MC has been in state i, i.e.

$$v_{ij} = P(X_n = j \text{ for some } n | X_0 = i)$$

- If $v_{ij} \neq 0$, then j is *reachable* from i
- If $v_{ij} = 0 \quad \forall j \neq i, v_{ii} = 1$, then i is an *absorbing* state

Example (1) & (3): all states reachable from each other (e.g. look at transition diagrams)
Example (2): all states in $\{1, 2, \ldots, 99\}$ reachable from each other, 0 & 100 absorbing

A subset of states, C is called *closed* if $j \notin C$, implies j cannot be reached from any $i \in C$. E.g. set of all states, Ω, is closed. If $\nexists$ proper subset $C \subset \Omega$ which is closed, the MC is called *irreducible.*
A Markov Chain is irreducible if and only if every state is reachable from every other state. E.g. example (1) & (3) and $\{0\}$, $\{100\}$ are closed in example (2).

2.10.4 Recurrent states

If $v_{ii} = 1$, state i is *recurrent* (once entered, guaranteed eventually to return). Thus we have:
i recurrent implies i is either not visited by the MC or is visited an infinite number of times no visit can be the last (with non-zero probability).
If i is not recurrent it is called *transient*. This implies the number of visits to a transient state is finite w.p. 1 (has geometric distribution).
E.g. if C is a closed set of states, $i \notin C$, $j \in C$ and $v_{ij} \neq 0$, then i is transient since MC will eventually enter C from i, never to return.
E.g. in example (2), states 1 to 99 are transient, states 0 and 100 are recurrent (as is any absorbing state)

Proposition: If i is recurrent and j is reachable from i, then j is also recurrent.
Proof: $v_{jj} \neq 1$ implies $v_{ij} \neq 1$ since if the MC visits i after j it will visit i repeatedly and with probability 1 will eventually visit j since $v_{ij} \neq 0$. This implies $v_{ii} \neq 0$ since with non-zero probability the MC can visit j after i but not return to i (w.p. $1 - v_{ji} > 0$) **qed.**

Thus, in an irreducible MC, either all states are transient (a transient MC) or recurrent (a recurrent MC).

2.10.5 Further properties of MCs

Let X be an irreducible, recurrent MC and let $n_1^j, n_2^j, \dots$ be the times of successive visits to state j, $m_j = E[n_{k+1}^j - n_k^j]$ $(k = 1, 2, \dots)$, the mean interval between visits (independent of k by the MP).

<u>Intuition</u>:

$$p_j = \frac{1}{m_j}$$

Because the proportion of time spent, on average, in state j is $1/m_j$. This is not necessarily true, because the chain may exhibit some *periodic* behaviour:

The state j is *periodic* with period $m > 1$ if $q_{ii}^{(k)} = 0$ for $k \neq rm$ for some $r \geq 1$ and $P(X_{n+rm} = j \text{ for some } r \geq 1 | X_n = j) = 1$. Otherwise it is aperiodic, or has period 1. (Note that a periodic state is recurrent)

Proposition: In an irreducible MC, either all states are aperiodic or periodic with the same period. The MC is then called aperiodic or periodic respectively.

Proposition: If $\{X_n | n = 0, 1, \dots\}$ is an irreducible, aperiodic MC, then the limiting probabilities $\{p_j | j = 0, 1, \dots\}$ exist and $p_j = 1/m_j$.

2.10.6 Null & positive recurrence

If $m_j = \infty$ for state j then $p_j = 0$ and state j is *recurrent null.* Conversely, if $m_j < \infty$, then $p_j > 0$ and state j is *recurrent non-null* or *positive recurrent.*

Proposition: In an irreducible MC, either all states are positive recurrent or none are. In the former case, the MC is called positive recurrent (or recurrent non-null).

A state which is aperiodic and positive recurrent is often called *ergodic* and an irreducible MC whose states are all ergodic is also called ergodic. When the limiting probabilities $\{p_j | j = 0, 1, \dots\}$ do exist they form the

$$\left.\begin{array}{l}\text{steady state}\\ \text{equilibrium}\\ \text{long term}\end{array}\right\} \text{probability distribution (SSPD)}$$

of the MC. They are determined by the following theorem.

2.10.7 Steady state theorem for Markov chains

An irreducible, aperiodic Markov Chain, X, with state space S and one-step transition probability matrix $Q = (q_{ij} | i, j \in S)$, is positive recurrent if and only if the system of equations

$$p_j = \sum_{i \in S} p_i q_{ij}$$

with $\sum_{i \in S} p_i = 1$ (normalisation) has a solution. If it exists, the solution is unique and is the SSPD of X.

Note: If S is finite, X is always positive recurrent, which implies the equations have a unique solution. The solution is then found by discarding a balance equation & replacing it with the normalising equation (the balance equations are dependent since the rows of homogeneous equations $\underline{p} - \underline{p}.\underline{Q} = \underline{0}$ all sum to zero).

2.11 Markov Processes

- Continuous time parameter space, discrete state space

$$X = \{X_t | t \geq 0\} \quad X_t \in \Omega$$

  where Ω is a countable set.

- Markov property

$$P(X_{t+s} = j | X_u, u \leq t) = P(X_{t+s} = j | X_t)$$

- Markov process is *time homogeneous* if the r.h.s. of this equation does not depend on t
  - $q_{ij}(s) = P(X_{t+s} = j | X_t = i) = P(X_s = j | X_0 = i)$ $(i, j = 0, 1, \dots)$
  - *Transition probability functions* of the MP
- Markov Property and time homogeneity imply:

If at time t process is in state j, time remaining in state j is independent of time already spent in state j: *memoryless property*

Proof:

$$\begin{aligned} P(S > t+s \mid S > t) &= P(X_{t+u} = j, 0 \le u \le s \mid X_u = j, 0 \le u \le t) \\ &\quad \text{where S} = \text{time spent in state } j \\ &\quad \text{state } j \text{ entered at time } 0 \\ &= P(X_{t+u} = j, 0 \le u \le s \mid X_t = j) \text{ by MP} \\ &= P(X_u = j, 0 \le u \le s \mid X_0 = j) \text{ by T.H.} \\ &= P(S > s) \\ \Longrightarrow P(S \le t+s \mid S > t) &= P(S \le s) \end{aligned}$$

- Time spent in state i is exponentially distributed, parameter μ_i.

2.11.1 Time homogeneous Markov Processes

The MP implies next state, j, after current state i depends only on i, j – state transition probability q_{ij}. This implies the Markov Process is uniquely determined by the products:

$$a_{ij} = \mu_i q_{ij}$$

where μ_i is the rate out of state i and q_{ij} is the probability of selecting state j next. The a_{ij} are the *generators* of the Markov Process.

- intuitively reasonable

2.11.2 Instantaneous transition rates

Consider now the (small) interval $(t, t+h)$

$$\begin{aligned} P(X_{t+h} = j | X_t = i) &= \mu_i h q_{ij} + o(h) \\ &= a_{ij} h + o(h) \end{aligned}$$

- a_{ij} is the *instantaneous transition rate* $i \to j$ i.e. the average number of transitions $i \to j$ per unit time spent in state i

2.11.3 Examples

1. Poisson process rate λ

$$a_{ij} = \begin{cases} \lambda \text{ if } j = i+1 \\ 0 \text{ if } j \neq i, i+1 \\ \text{not defined if } j = i \end{cases}$$

because

$$\begin{array}{rcl} P(\text{arrival in } (t, t+h)) &=& \lambda h + o(h) \\ i \to i+1 && a_{i,i+1} h \end{array}$$

2. The buffer problem
   - Records arrive as a P.P. rate λ
   - Buffer capacity is M
   - Buffer cleared at times spaced by intervals which are

exponentially distributed, parameter μ. Clearance times i.i.d. & independent of arrivals (i.e. clearances are an independent PP, rate μ)

$$a_{ij} = \begin{cases} \lambda \text{ if } j = i+1 \quad (0 \le i \le M-1) \\ \mu \text{ if } j = 0 \\ 0 \text{ otherwise } (j \ne i) \end{cases}$$

[Probability of >1 arrivals or clearances, or ≥ 1 arrivals <u>and</u> clearances in $(t, t+h)$ is $o(h)$]

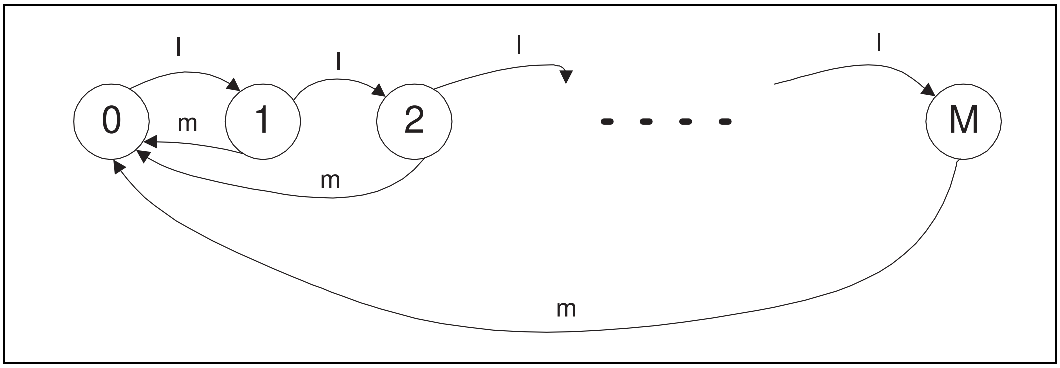


Figure 4: Buffer problem

3. <u>Telephone exchange</u>
Suppose there 4 subscribers and any two calls between different callers can be supported simultaneously. Calls are made by non-connected subscribers according to independent P.Ps, rate λ (callee chosen at random). Length of a call is (independent) exponentially distributed, parameter μ. Calls are lost if called

subscriber is engaged. State is the number of calls in progress $\in \{0,1,2\}$

$$\begin{aligned} a_{01} &= 4\lambda \quad \text{(all free)} \\ a_{12} &= \frac{2}{3}\lambda \quad \text{(caller has } \frac{1}{3} \text{ probability of successful connection)} \\ a_{10} &= \mu \\ a_{21} &= 2\mu \quad \text{(either call may end)} \end{aligned}$$

<u>Exercise</u>: The last equation follows because the distribution of the random variable $Z = \min(X, Y)$ where X, Y are independent exponential r.v.s. with parameters λ, μ respectively is exponential with parameter $\lambda + \mu$. Prove this.

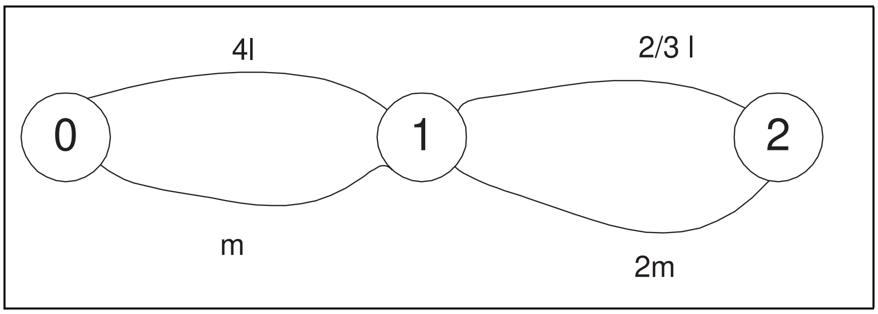


Figure 5: Telephone exchange

2.11.4 The Generator Matrix

$$\sum_{j \in \Omega, j \neq i} q_{ij} = 1 \implies \mu_i = \sum_{j \in \Omega, j \neq i} a_{ij}$$

(hence *instantaneous transition rate out of* i). Let $a_{ii} = -\mu_i$ (undefined so far)

$$(a_{ij}) = A = \begin{pmatrix} a_{00} & a_{01} & \dots & & \\ a_{10} & a_{11} & \dots & & \\ \dots & & & & \\ a_{i0} & a_{i1} & \dots & a_{ii} & \dots \\ \dots & & & & \end{pmatrix}$$

A is the generator matrix of the Markov Process. Rows of A sum to zero ($\sum_{j \in S} a_{ij} = 0$)

2.11.5 Transition probabilities & rates

- determined by A

$$\begin{aligned} q_{ij} &= \frac{a_{ij}}{\mu_i} = -\frac{a_{ij}}{a_{ii}} \\ \mu_i &= -a_{ii} \end{aligned}$$

- hence A is all we need to determine the Markov process
- Markov processes are instances of state transition systems with the following properties
  - The state holding time are *exponentially* distributed
  - The probability of transiting from state i to state j depends only on i and j
- In order for the analysis to work we require that the process in **irreducible** – every state must be reachable from every other, e.g.

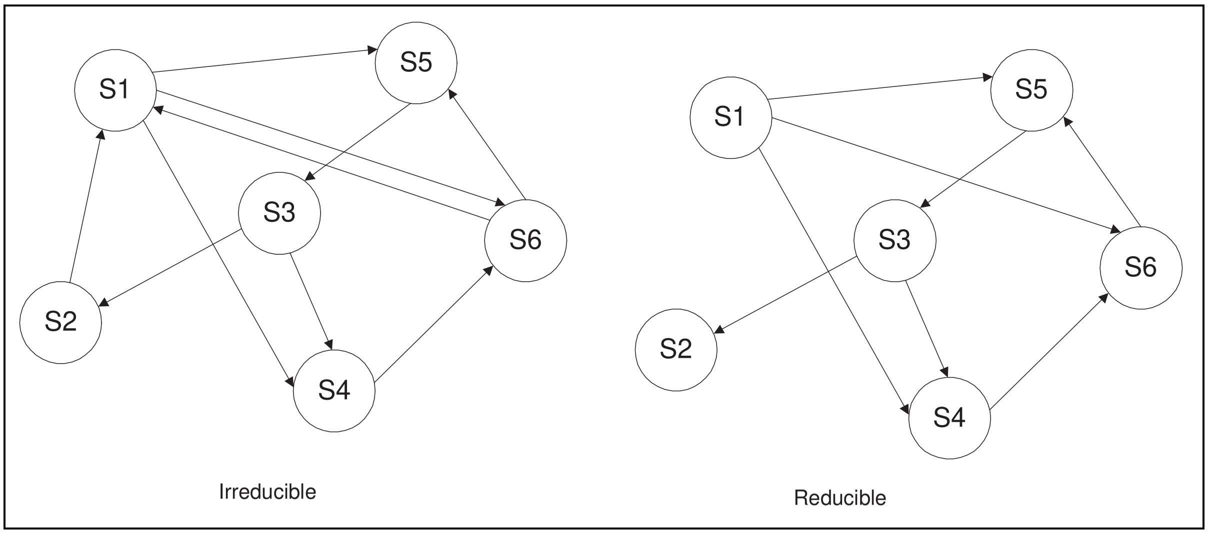


Figure 6: (Ir)reducible Markov chains

- Associated with each arc in a state transition diagram is a *rate*, e.g.:

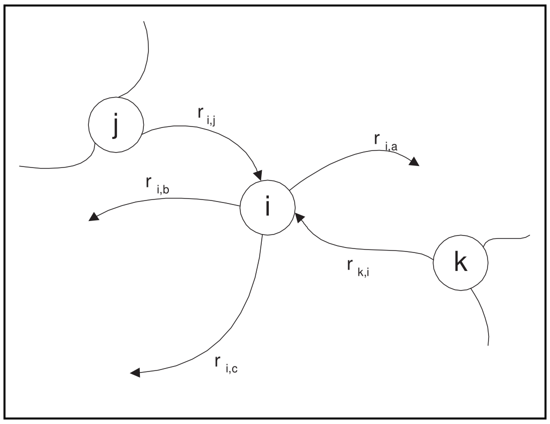


Figure 7: Transition rates in Markov chains

- The state holding time depends on the departure rates ($1/(r_{i,a} + r_{i,b} + r_{i,c})$ is the mean holding time for state i in the example).
- The *probability flux* from a state i to a state j is the average number of

transitions from $i \to j$ per unit time at equilibrium:

$$\text{Probability flux}(i \to j) = r_{i,j} p_i$$

- At equilibrium (if it exists) the total flux into each state must balance the total flux out of it
- For example, in the example, for state i alone:

$$(r_{i,a} + r_{i,b} + r_{i,c}) p_i = r_{j,i} p_j + r_{k,i} p_k$$

- Treating all states the same way leads to a set of *linear* equations:

$$\sum_{i \neq j} a_{ij} p_i = \sum_{i \neq j} a_{ji} p_j = -a_{ii} p_i \text{ for all } j \in S$$

- Called the *balance equations* and subject to the *normalising equation*

$$\sum_{i \in S} p_i = 1$$

- The **steady-state theorem** tells us that if these equations have a solution then there is a steady-state and that the solution is unique
- **Important generalisation**: At equilibrium the fluxes also balance into and out of *every closed contour* drawn around *any* collection of states
- This often provides a short-cut to the solution – see the M/M/1 queue later.

2.12 Steady state results

Theorem

(a) if a Markov process is transient or recurrent null, $p_j = 0 \quad \forall j \in S$ and we say a SSPD does not exist

(b) If a Markov Process is positive recurrent, the limits p_j exist, $p_j > 0, \sum_{j \in S} p_j = 1$ and we say $\{p_j | j \in S\}$ constitute the SSPD of the Markov Process.

2.12.1 Steady State Theorem for Markov Processes

An irreducible Markov Process X with state space S and generator matrix $A = (a_{ij})$ $(i, j \in S)$ is positive recurrent if and only if

$$\begin{aligned} \sum_{i \in S} p_i a_{ij} &= 0 \quad \text{for } j \in S \text{ [Balance equations]} \\ \sum_{i \in S} p_i &= 1 \quad \text{[Normalising equation]} \end{aligned}$$

have a solution. This solution is unique and is the SSPD.

2.12.2 Justification of the Balance Equation

The rate going from state i to $j(\neq i)$ is a_{ij}, the fraction of time spent in i is p_i

$$\begin{aligned}\sum_{i\neq j} a_{ij}p_i &= \sum_{i\neq j} a_{ji}p_j \\ \text{Avg. no. of transitions} & \\ i \to j \text{ in unit time} &= \text{Avg. no. of transitions} \\ & \quad j \to i \text{ in unit time} \\ \sum \text{flux}(i \to j) &= \sum \text{flux}(j \to i) \quad \forall j\end{aligned}$$

2.12.3 Buffer example (DTMC)

$\underline{1 \leq i \leq M-1}$

$$\begin{aligned}(\text{Net rate in}) &= (\text{Net rate out}) \\ (\alpha(1-\beta)+\beta)p_i &= \alpha(1-\beta)p_{i-1}\end{aligned}$$

$$\{= p_i = \alpha(1-\beta)p_{i-1} + p_i(1-\alpha)(1-\beta) \text{ from S.S. Theorem}\}$$

Therefore

$$p_i = kp_{i-1} = k^i p_0$$

where $k = \frac{\alpha(1-\beta)}{\alpha+\beta-\alpha\beta}$ $(1 \leq i \leq M-1)$

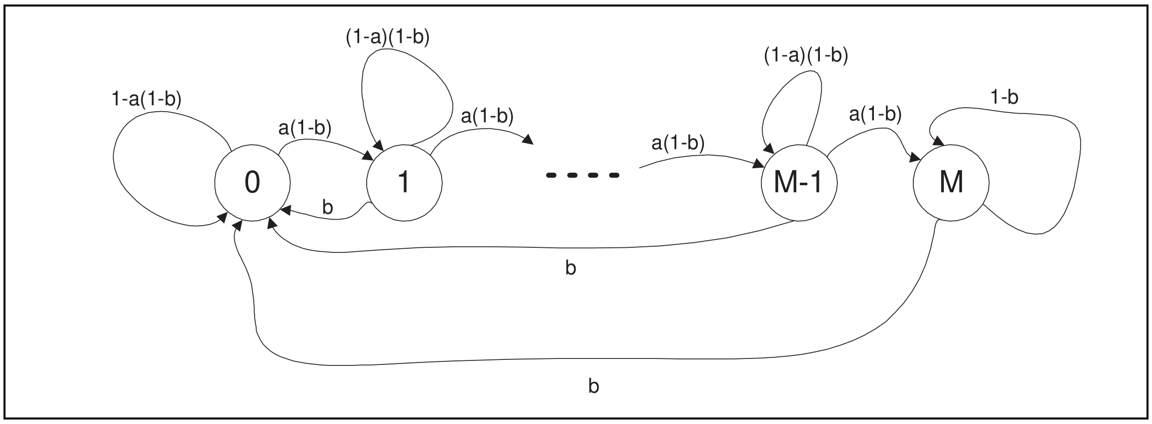


Figure 8: I/O buffer.

$\underline{i = M}$

$$\beta p_M = \alpha(1-\beta)p_{M-1}$$

therefore

$$p_M = \frac{\alpha(1-\beta)}{\beta}k^{M-1}p_0$$

$\underline{i = 0}$

Redundant equation (why?) - use as check

$$\begin{aligned}\alpha(1-\beta)p_0 &= \sum_{i=1}^{M}\beta p_i = \beta\sum_{i=1}^{M-1}k^i p_0 + k^{M-1}\alpha(1-\beta)p_0 \\ &= p_0\Big(\beta k\frac{1-k^{M-1}}{1-k} + \alpha(1-\beta)k^{M-1}\Big)\end{aligned}$$

rest is exercise

2.12.4 Examples (CTMC)

1. I/O buffer:
   By considering the flux into and out of each state $0, 1, \ldots, M$ we obtain the balance equations:

$$\begin{aligned}
\lambda p_0 &= \mu(p_1 + \cdots + p_M) \text{ (State 0)} \\
(\lambda + \mu)p_i &= \lambda_{i-1}p_{i-1} \quad (1 \le i \le M-1) \text{ State } i \\
\mu p_M &= \lambda p_{M-1} \text{ State } M
\end{aligned}$$

   Normalising equation:

$$\begin{aligned}
p_0 &+ \cdots + p_M = 1 \\
\implies p_j &= \left(\frac{\lambda}{\lambda+\mu}\right)^j \frac{\mu}{\lambda+\mu} \quad (0 \le j \le M-1) \\
p_M &= \left(\frac{\lambda}{\lambda+\mu}\right)^M
\end{aligned}$$

   Thus (for example) mean number of records in the buffer in the steady state $= M\alpha^M + \sum_{j=0}^{M-1} j\alpha^j \frac{\mu}{\lambda+\mu}$ where $\alpha = \frac{\mu}{\lambda+\mu}$

2. <u>Telephone network</u>

$$\begin{aligned}
4\lambda p_0 &= \mu p_1 \text{ (State 0)} \\
\left(\mu + \frac{2}{3}\lambda\right)p_1 &= 4\lambda p_0 + 2\mu p_2 \text{ (State 1)} \\
2\mu p_2 &= \lambda \frac{2}{3} p_1 \text{ (State 2)}
\end{aligned}$$

   Thus $p_1 = \frac{4\lambda}{\mu}p_0$, $p_2 = \frac{\lambda}{3\mu}p_1 = \frac{4\lambda^2}{3\mu^2}p_0$ with
   $p_0 + p_1 + p_2 = 1 \implies p_0 = \left(1 + \frac{4\lambda}{\mu} + \frac{4\lambda^2}{3\mu^2}\right)^{-1}$ Average number of calls in progress in the steady state $= 1.p_1 + 2.p_2 = \frac{\frac{4\lambda}{\mu}\left(1+\frac{2\lambda}{3\mu}\right)}{1+\frac{4\lambda}{\mu}+\frac{4\lambda^2}{3\mu^2}}$

2.12.5 Birth-Death processes and the single server queue (SSQ)

A Markov process with state space $S = \{0, 1, \dots\}$ is called a birth-death process if the only non-zero transition probabilities are $q_{i,i+1}$ and $q_{i+1,i}$ ($i \geq 0$), representing births and deaths respectively. (In a population model, q_{00} would be 1 since 0 would be an absorbing state.) The SSQ model consists of

- a Poisson arrival process, rate λ
- a queue which arriving tasks join
- a server which processes tasks in the queue in FIFO (or other) order and has exponentially distributed service times, parameter μ (i.e. given a queue length > 0, service completions form a Poisson process, rate μ)

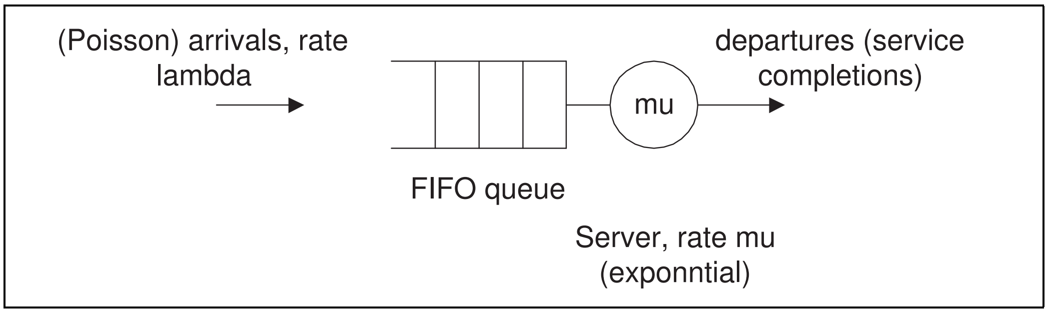


Figure 9: Single server queue.

- The state is the queue length (including the task being served if any), i.e. the state space is $\{0, 1, \dots\}$
- SSQ model is a birth-death process
- λ, μ are in general functions of the queue length (i.e. state dependent) and we write $\lambda(n), \mu(n)$ for state n.

2.12.6 The memoryless property in the M/M/1 queue

Notation: The SSQ with Poisson arrivals and exponential service times is called an M/M/1 queue

- the first M describes the arrival process as a Markov process (Poisson)
- the second M describes the service time distribution as a Markov process (exponential)
- the 1 refers to a single server (m parallel server would be denoted as M/M/m)
- Later we will consider M/G/1 queues, where the service time distribution is non-Markovian ("general")

SSQ therefore follows a Markov process and has the memoryless property that:

1. Probability of an arrival in $(t, t+h) = \lambda(i)h + o(h)$ in state i
2. Probability of a service completion in $(t, t+h) = \mu(i)h + o(h)$ in state $i > 0$ (0 if $i = 0$)
3. Probability of more than 1 arrival, more than one service completion or 1 arrival and 1 service completion in $(t, t+h) = o(h)$.

2.12.7 State transition diagram for the SSQ

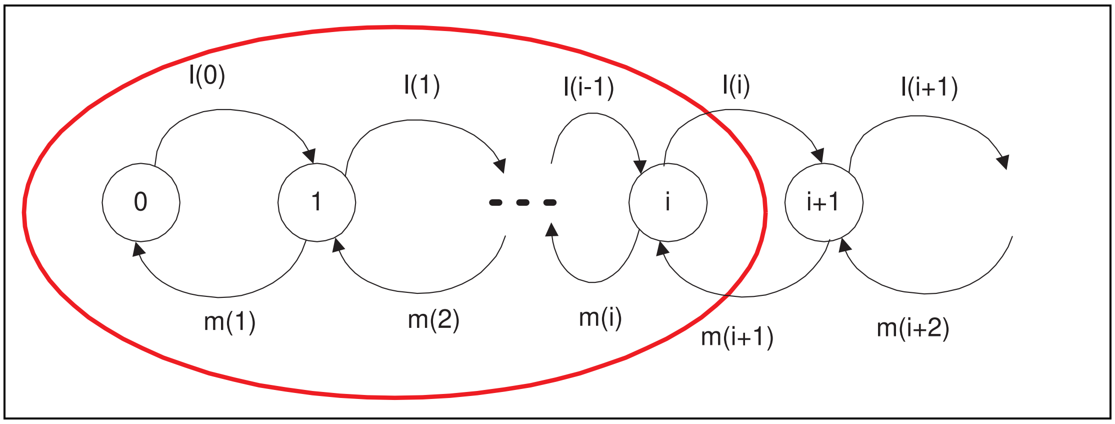


Figure 10: Single server queue state diagram.

- Consider the balance equation for states inside the red (thick) contour. Outward average flow (all from state $i+1$): $p_i\lambda(i) \quad (i \geq 0)$; Inward average flow (all from state i):

$p_{i+1}\mu(i+1) \quad (i \geq 0)$. Thus,

$$\begin{aligned} p_i\lambda(i) &= p_{i+1}\mu(i+1) \\ \text{so } p_{i+1} &= \frac{\lambda(i)}{\mu(i+1)}p_i = \Big[\prod_{j=0}^{i}\rho(j)\Big]p_0 \end{aligned}$$

where $\rho(j) = \frac{\lambda(j)}{\mu(j+1)}$

- Normalising equation implies

$$\begin{aligned} p_0 & \Big(1+\sum_{i=0}^{\infty}\prod_{j=0}^{i}\rho(j)\Big)=1 \\ \Longrightarrow \quad p_0 & =\Big[\sum_{i=0}^{\infty}\prod_{j=0}^{i-1}\rho(j)\Big]^{-1} \end{aligned}$$

where $\prod_{j=0}^{-1}=1$ (empty product). Therefore

$$p_i=\frac{\prod_{j=0}^{i-1}\rho(j)}{\sum_{k=0}^{\infty}\prod_{n=0}^{k-1}\rho(n)} \quad (i\geq 0).$$

- SSQ with constant arrival and service rates

$$\lambda(n)=\lambda, \quad \mu(n)=\mu, \quad \rho(n)=\rho=\lambda/\mu \quad \forall n\in S$$

implies

$$\begin{aligned} p_0 &= \Big[\sum_{i=0}^{\infty}\rho^i\Big]^{-1}=1-\rho \\ p_i &= (1-\rho)\rho^i \quad (i\geq 0) \end{aligned}$$

- Mean queue length, L (including any task in service)

$$\begin{aligned} L &= \sum_{i=0}^{\infty} i p_i=\sum_{i=0}^{\infty}(1-\rho)i\rho^i \\ &= \rho(1-\rho)\frac{d}{d\rho}\Big\{\sum_{i=0}^{\infty}\rho^i\Big\}=\rho(1-\rho)\frac{d}{d\rho}\Big\{(1-\rho)^{-1}\Big\} \\ &= \frac{\rho}{1-\rho} \end{aligned}$$

- Utilisation of server

$$\begin{aligned} U &= 1 - P(\text{server idle}) = 1 - p_0 = 1 - (1-\rho) = \rho \\ &= \lambda/\mu. \end{aligned}$$

However, we could have deduced this without solving for p_0: In the steady state (assuming it exists),

$\lambda = \text{arrival rate} = \text{throughput} = P(\text{server busy}).\text{service rate} = U\mu$

Argument applies for any system in equilibrium - didn't need MP - see M/G/1 queue later.

2.12.8 Response times

To analyse response times, need to consider the state of the queue at the time of arrival of a task. We use the Random Observer Property of the Poisson process:
The state of a system at equilibrium seen by an arrival of a Poisson process has the same distribution as that seen by an observer at a random instant i.e. if the state at time t is denoted by S_t,

$$P(S_{t_0^-} = i \mid \text{arrival at } t_0) = P(S_{t_0^-} = i)$$

Mean response time, W
If the queue length seen by an arriving task is j,

$$\begin{aligned} \text{response time} = &\quad \text{residual service time of task in service (if } j > 0) \\ &+ \quad j \text{ i.i.d. service times} \end{aligned}$$

But for exponential service times, residual service time has the same distribution as full service time

$$\begin{aligned}
\Longrightarrow \text{Response time} &= \text{sum of } (j+1) \text{ service times} \\
\Longrightarrow \text{Mean response time, } W &= \sum p_j (j+1)\mu^{-1} \\
&= \Big(1 + \frac{\rho}{1-\rho}\Big)\mu^{-1} = \frac{1}{\mu - \lambda}
\end{aligned}$$

<u>Mean queueing time, W_Q (excluding service)</u>

$$W_Q = W - \mu^{-1} = L\mu^{-1} = \frac{\rho}{\mu - \lambda}.$$

2.12.9 Distribution of the waiting time, $F_W(x)$

By the random observer property (and memoryless property)

$$F_W(x) = \sum_{j=0}^{\infty} p_j E_{j+1}(x)$$

where $E_{j+1}(x)$ is the convolution of $(j+1)$ exponential distributions, each with parameter μ - called the Erlang-$(j+1)$ distribution with parameter μ. Similarly, for density functions:

$$f_W(x) = \sum_{j=0}^{\infty} p_j e_{j+1}(x)$$

where $e_{j+1}(x)$ is the pdf corresponding to $E_{j+1}(x)$, i.e.

$\frac{d}{dx}E_{j+1}(x) = e_{j+1}(x)$, defined by

$$\begin{aligned}
e_1(x) &= \mu e^{-\mu x} \\
e_{j+1}(x) &= \mu \int_0^x e^{-\mu(x-u)} e_j(u) du \\
& \text{(convolution of Erlang-}j\text{ and exponential)} \\
\Longrightarrow e_j(x) &= \mu \frac{(\mu x)^{j-1}}{(j-1)!} e^{-\mu x} \qquad \text{(Exercise)} \\
\Longrightarrow f_W(x) &= (\mu - \lambda) e^{(\mu-\lambda)x} \qquad \text{(Exercise)}
\end{aligned}$$

Example

Given m Poisson streams, each with rate λ and independent of the others, into a SSQ, service rate μ, what is the maximum value of m for which, in steady state, at least 95% of waiting times, $T \leq w$? (Relevant in "end-to-end" message delays in communication networks, for example)

$$\begin{aligned}
P(T \leq w) &\geq 0.95 \\
\text{i.e. } 1 - e^{-(\mu - m\lambda)w} &\geq 0.95 \\
\text{or } e^{-(\mu - m\lambda)w} &\leq 0.05 \\
\Longrightarrow \; e^{(\mu-\lambda)w} &\geq 20 \\
\mu - m\lambda &\geq \frac{\ln 20}{w} \\
\Longrightarrow \; m &\leq \frac{\mu}{\lambda} - \frac{\ln 20}{w\lambda}
\end{aligned}$$

$m < \mu/\lambda$ is equivalent to the existence of a steady state.

2.13 Multiple parallel servers - M/M/m queue

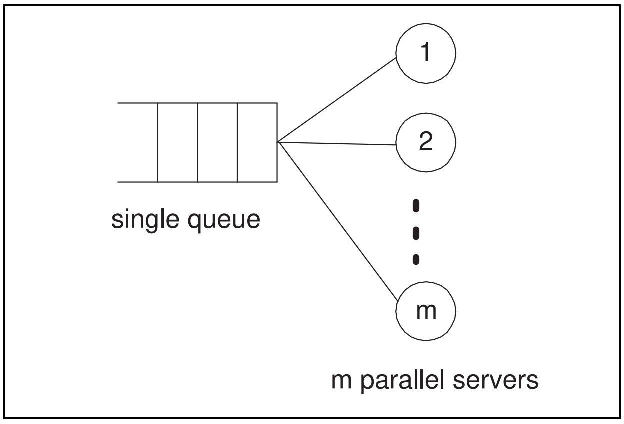


Figure 11: Multiple parallel servers.

SSQ representation:

$$\begin{aligned} \lambda(n) &= \lambda \quad (n \geq 0) \\ \mu(n) &= \begin{cases} n\mu & (1 \leq n \leq m) \\ m\mu & (n \geq m) \end{cases} \end{aligned}$$

Front of queue served by any available server.
Loading, $\rho = \lambda/\mu$ (average amount of service required in unit time).
By general result for M/M/1 queue:

$$p_j = p_0 \prod_{i=0}^{j-1} \frac{\lambda(i)}{\mu(i+1)} = \begin{cases} p_0 \frac{\rho^j}{j!} & (0 \leq j \leq m) \\ p_0 \frac{\rho^j}{m! m^{j-m}} & (j \geq m) \end{cases}$$

This implies

$$\begin{aligned} p_0 &= \frac{1}{\sum_{i=0}^{m-1} \frac{\rho^i}{i!} + \frac{\rho^m}{m!} \sum_{i=m}^{\infty} (\frac{\rho}{m})^{i-m}} \\ &= \frac{1}{\sum_{i=0}^{m-1} \frac{\rho^i}{i!} + \frac{\rho^m}{(m-1)!(m-\rho)}} \end{aligned}$$

<u>Average number of busy servers, S</u>

$$S = \sum_{k=1}^{m-1} k p_k + m \sum_{k=m}^{\infty} p_k = \cdots = \rho$$

<u>Steady state argument</u>

$$\begin{aligned} \text{arrival rate} &= \lambda \\ \text{average throughput} &= S\mu \quad (\mu \text{ per active server}) \\ \Longrightarrow S\mu &= \lambda \quad \text{in equilibrium.} \end{aligned}$$

<u>Utilisation</u> $U = S/m = \rho/m$, the average fraction of busy servers.

2.13.1 Waiting times

Waiting time is the same as service time if an arrival does not have to queue. Otherwise, the <u>departure</u> process is Poisson, rate $m\mu$, whilst the arrived task is queueing (all servers busy). This implies that the queueing time is the same as the queueing time in the M/M/1 queue with service rate $m\mu$.

- Let

$$\begin{aligned} q &= P(\text{arrival has to queue}) \\ &= P(\text{find} \geq m \text{ jobs on arrival}) \end{aligned}$$

- by R.O.P.

$$q = \sum_{j=m}^{\infty} p_j = p_0 \frac{\rho^m}{(m-1)!(m-\rho)} \quad \text{Erlang delay formula}$$

- Let Q be the queueing time random variable (excluding service)

$$\begin{aligned} F_Q(x) &= P(Q \le x) = P(Q=0) + P(Q \le x | Q > 0) P(Q > 0) \\ &= (1-q) + q(1 - e^{-(m\mu-\lambda)x}) \\ &= 1 - qe^{-(m\mu-\lambda)x} \end{aligned}$$

  $P(Q \le x | Q > 0)$ is given by the SSQ, rate $m\mu$, result for waiting time. Note that $F_Q(x)$ has a jump at the origin, $F_Q(0) \neq 0$.

- Mean queueing time

$$W_Q = \frac{q}{m\mu - \lambda}$$

- Mean waiting time

$$W = W_Q + 1/\mu = \frac{(q+m)\mu - \lambda}{\mu(m\mu - \lambda)}$$

- Exercise: What is the waiting time density function?

2.13.2 The infinite server

- In the M/M/m queue, let $m \to \infty$
  - "unlimited service capacity"
  - always sufficient free servers to process an arriving task - e.g. when the number tasks in the system is finite
  - no queueing $\implies$ infinite servers model delays in a task's processing

$$p_j = \frac{\rho^j}{j!} p_0 \implies p_0 = e^{-\rho}$$

  - balance equations have a solution for all λ, μ
  - this is not surprising since the server can never be overloaded
- This result is independent of the exponential assumption - a property of the queueing discipline (here there is no queueing - a pathological case)

2.14 M/M/m queues with finite state space

M/M/1/k (k is the max. queue length) queue

$$\mu(n) = \mu \quad \text{for all queue lengths } n$$

$$\lambda(n) = \begin{cases} \lambda & (0 \le n < k) \\ 0 & (n \ge k) \end{cases}$$

Hence, if $\rho = \lambda/\mu$.

$$p_j = \begin{cases} p_0 \rho^j & (j = 0, 1, \ldots, k) \\ \frac{\prod_{i=0}^{j-1} \lambda(i)}{\prod_{i=0}^{j-1} \mu(i+1)} = 0 & (j > k) \qquad \text{as } \lambda(k) = 0 \end{cases}$$

Telephone network with maximum capacity of k calls

$$\begin{aligned} \lambda(n) &= \lambda \quad (0 \le n < k), \quad \lambda(k) = 0 \\ \mu(n) &= n\mu \quad n \text{ calls in progress} \\ &\Longrightarrow \text{M/M/k/k queue} \end{aligned}$$

$$p_j = p_0 \frac{\rho^j}{j!} \quad j = 0, 1, \ldots, k \quad (\rho = \lambda/\mu)$$

Probability that a call is lost:

$$p_k = \frac{\rho^k/k!}{\sum_{j=0}^k \rho^j/j!} \quad \underline{\text{Erlang loss formula}}$$

$$\text{Throughput} = \lambda(1 - p_k) \quad \left[= \mu \sum_{j=1}^k j p_j \right]$$

2.14.1 Terminal system with parallel processing

N users logged on to a computer system with m parallel processors

- exponentially distributed think times, mean $1/\lambda$, before submitting next task
- each processor has exponential service times, mean $1/\mu$

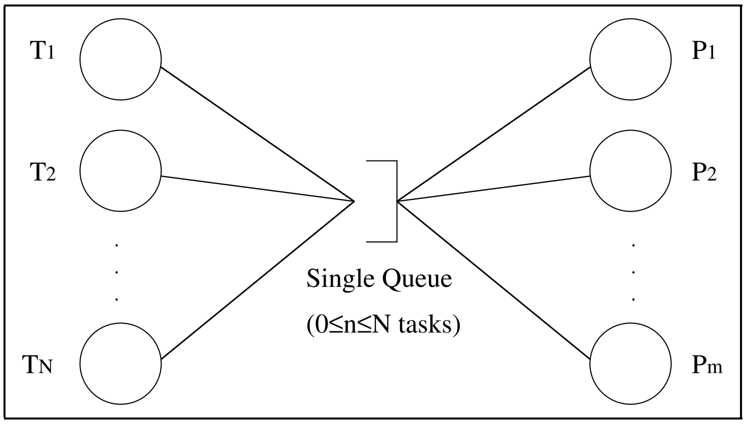


Figure 12: Terminals with parallel processors.

- tasks may use any free processor, or queue (**FIFO**) if there are none.
- state space $S = \{n | 0 \le n \le N\}$ where n is the queue length (for all processors)
- Poisson arrivals, rate $\lambda(n) = (N - n)\lambda$
- Service rate $\mu(n) = \begin{cases} n\mu & (1 \le n \le m) \\ m\mu & (m \le n \le N) \end{cases}$

Steady state probabilities $\{p_i | 0 \le i \le N\}$

$$p_i = \frac{\prod \lambda(i-1)}{\prod \mu(i)} = \frac{N(N-1)\ldots(N-i+1)}{i!}\rho^i p_0 \quad (0 \le i \le m)$$

$$\Longrightarrow p_i = \begin{cases} \binom{N}{i}\rho^i p_0 & (0 \le i \le m) \\ \frac{N!}{(N-i)!m!m^{i-m}}\rho^i p_0 & (m \le i \le N) \end{cases}$$

$$p_0 = \Big\{ \sum_{i=0}^{m-1} \binom{N}{i}\rho^i + \sum_{i=m}^{N} \frac{N!}{(N-i)!m!m^{i-m}}\rho^i \Big\}^{-1}$$

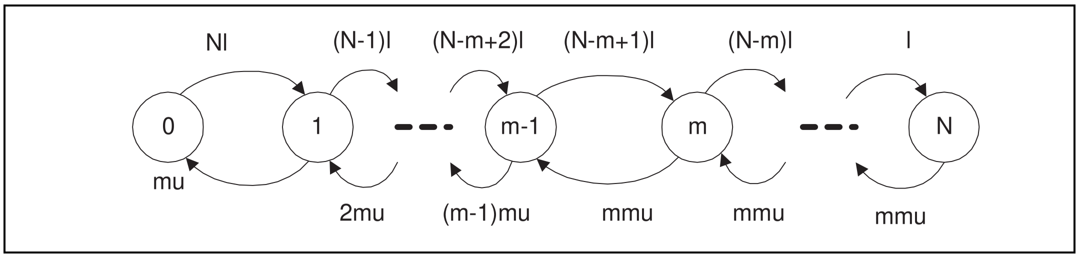


Figure 13: State transition diagram.

The throughput is either given by

$$\mu\Big\{ \sum_{j=1}^{m-1} j p_j + m \sum_{j=m}^{N} p_j \Big\}$$

(mean departure rate from processors) or

$$\lambda\Big\{N - \sum_{j=1}^{N} jp_j\Big\}$$

(mean arrival rate)

Other performance measures as previously

2.14.2 The case of "always sufficient processors" - $m \geq N$

- Here there is no case $m < i \leq N$

$$\begin{aligned}\Longrightarrow\ p_i &= \binom{N}{i}\rho^i p_0 \quad (0 \leq i \leq N)\\ \Longrightarrow\ p_0 &= \Big\{\sum_{i=0}^{N}\binom{N}{i}\rho^i\Big\}^{-1} = (1+\rho)^{-N}\\ \text{Thus } p_i &= \binom{N}{i}\Big(\frac{\rho}{1+\rho}\Big)^i\Big(\frac{1}{1+\rho}\Big)^{N-i}\\ &\quad \text{(Binomial distribution)}\end{aligned}$$

- Probabilistic explanation
  - p_i is the probability of i "successes" in N Bernoulli (i.i.d.) trials
  - probability of success = $\frac{\rho}{1+\rho} = \frac{1/\mu}{1/\lambda+1/\mu}$ = fraction of time

user is waiting for a task to complete in the steady state = probability (randomly) observe a user in wait-mode.

  - probability of failure = fraction of time user is in think mode in the steady state = probability (randomly) observe a user in think-mode
  - random observations are i.i.d. in steady state $\Longrightarrow$ trials are Bernoulli
  - hence Binomial distribution

- result is independent of distribution of think times.

2.14.3 Analogy with a queueing network

Regard the model as a 2-server, cyclic queueing network

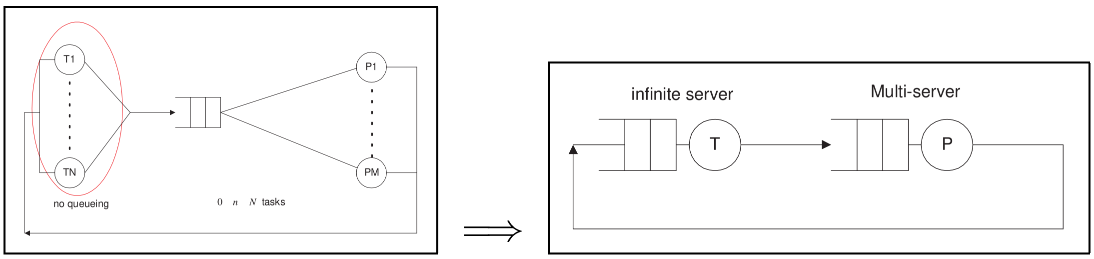


Figure 14: Original and equivalent network.

- As already observed, IS server is insensitive to its service time distribution as far as queue length distribution is concerned.
- Multi-server becomes IS if $M \geq N \implies$ 2 IS servers.

2.15 Little's result/formula/law (J.D.C. Little, 1961)

Suppose that a queueing system Q is in steady state (i.e. there are fixed, time independent probabilities of observing Q in each of its possible states at random times.) Let:

L = average number of tasks in Q in steady state

W = average time a task spends in Q in steady state

λ = arrival rate (i.e. average number of tasks entering or leaving Q in unit time)

Then

$$L = \lambda W.$$

Intuitive Justification

Suppose a charge is made on a task of £1 per unit time it spends in Q

- Total collected on average in unit time $= L$
- Average paid by one task $= W$
- If charges collected on arrival (or departure), average collected in unit time $= \lambda.W$

$$\implies L = \lambda W$$

- Example: M/M/1 queue

$$W = (L+1)/\mu \text{ by R.O.P.}$$

$$L = \lambda W \implies W = \frac{1}{\mu - \lambda}$$

2.15.1 Application of Little's law

Server utilisation U.

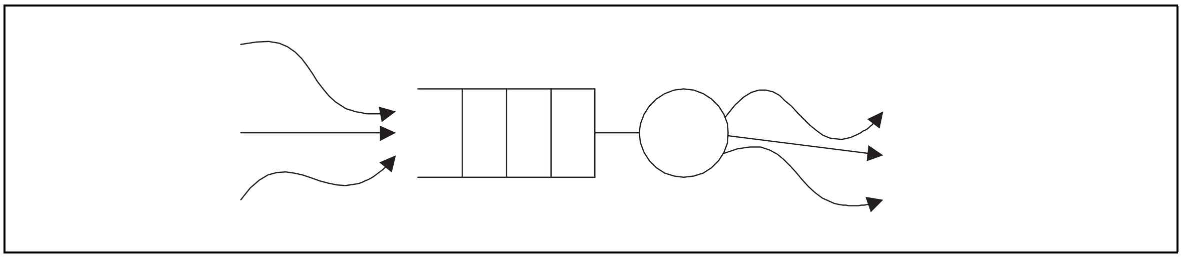


Figure 15: Little's law.

- Consider server, rate μ (constant), inside some queueing system: i.e. average time for one service $= 1/\mu$
- Let total throughput of server $= \lambda =$ total average arrival rate (since in steady state)

- Apply Little's result to the server only (without the queue)

$$\begin{aligned}
\text{mean queue length} &= 0.P(\text{idle}) + 1.P(\text{busy}) \\
&= U \\
\text{mean service time} &= \mu^{-1} \\
\Longrightarrow U &= \frac{\lambda}{\mu}
\end{aligned}$$

- Not the easiest derivation! This is a simple work conservation law.

2.16 Single Server Queue with general service times - M/G/1 queue

Assuming that arrivals are Poisson with constant rate λ, service time distribution has constant mean μ^{-1} (service rate μ) and that a steady state exists

$$\begin{aligned} \text{(1) Utilisation, } U &= P(\text{queue length} > 0) \\ &= \frac{\mu^{-1}}{\lambda^{-1}} = \lambda/\mu = \text{"load"} \\ &\quad \text{(average amount of work arriving in unit time)} \end{aligned}$$

$$\begin{aligned} \text{(2) Mean queue length} &= L \\ \text{Mean number in queue} &= L_Q \\ \text{Mean waiting time} &= W \\ \text{Mean queueing time} &= W_Q \\ \text{Little's law } \Longrightarrow L &= \lambda W \\ L_Q &= \lambda W_Q \\ \text{By definition, } W &= W_Q + 1/\mu \end{aligned}$$

3 equations, 4 unknowns

2.16.1 The fourth equation

- By the random observer property, queue faced on arrival has mean length L_Q (excluding task in service, if any)
- By "residual life" result for renewal processes, average service

time remaining for task in service (if any) is $\frac{\mu M_2}{2}$ where M_2 is the second moment of the service time distribution
($M_2 = \int_0^\infty x^2 f(x)dx$ where $f(x)$ is the pdf of service time)

- Thus, since $\rho = P(\exists$ a task being served at arrival instant)

$$W_Q \quad = \quad L_Q.1/\mu + \rho.\frac{\mu M_2}{2}$$

- Now

$$L_Q = \lambda W_Q \implies L_Q, W_Q, L, W$$

$$L \quad = \quad \rho + \frac{\lambda^2 M_2}{2(1-\rho)}$$

- Observe that if $\frac{\text{standard deviation}}{\text{mean}}$ (& hence the second moment) of service time distribution is large, L is also (not trivial as M_2 increases with μ^{-1} - but not difficult either!)

2.17 Embedded Markov Chain

- State of the M/G/1 queue at time t is $X(t) \in S$ where the state space $S = \{n | n \geq 0\}$ as in M/M/1 queue.
- M/G/1 queue is *not* a Markov process
  - $P(X(t+s)|X(u), u \leq t) \neq P(X(t+s)|X(t)) \; \forall t, s$
  - e.g. if a service period does not begin at time t
  - no memoryless property
- Consider times $t_1, t_2, \ldots$ of successive departures and let $X_i = X(t_i^+)$
- Given X_i, $X(t)$ for $t > t_i$ is independent of $X(t')$ for $t' < t_i$ since at time t_i^+
  - time to next arrival is exponential with parameter λ because arrival process is Poisson

  - instantaneously, no task is in service, so time to next departure is a complete service time or that plus the time to next arrival (if queue empty)

  $\Longrightarrow$ $\{X_i | i \geq 1\}$ is a Markov Chain with state space S ($\{t_i | i \geq 1\}$ are called "Markov times")

- It can be shown that, in steady state of E.M.C., distribution of no. of jobs, j, "left behind" by a departure = distribution of no. of jobs, j, seen by an arrival = $\lim_{n\to\infty} P(X_n = j)$ by R.O.P.

- Here we solve $p_j = \sum_{i=0}^{\infty} p_i q_{ij} \quad (i \geq 0)$ for appropriate one-step transition probabilities q_{ij}.

2.18 Balance equations for the M/G/1 queue

- Solution for $\{p_j\}$ exists iff $\rho = \frac{\lambda}{\mu} < 1$, equivalent to $p_0 > 0$ since $p_0 = 1 - U = 1 - \rho$ in the steady state.

- Valid one-step transitions are $i \to j$ for $j = i-1, i, \ldots$ since may have arbitrary no. of arrivals between successive departures.

- Let

$$r_k = P(k \text{ arrivals in a service period})$$

then

$$\begin{aligned} q_{ij} &= P(X_{n+1} = j | X_n = i) \quad (n \geq 0) \\ &= r_{j-i+1} \quad (i \geq 1, j \geq i-1) \\ & (i \to i-1+(j-i+1) = j) \\ q_{0j} &= r_j \quad (j \geq 0) \end{aligned}$$

[Eventually a job arrives, so $0 \to 1$, and then $1 \to j$ if there are j arrivals in its service time since then $1 \to 1 - 1 + j = j$]

$$\begin{aligned} \Longrightarrow p_0 &= 1 - \rho \\ p_j &= p_0 r_j + \sum_{i=1}^{j+1} p_i r_{j-i+1} \quad (j \geq 0) \end{aligned}$$

where $r_k = \int_0^\infty \frac{\lambda x)^k}{k!} e^{-\lambda x} f(x) dx$ if service time density function is $f(x)$ (known). This is because
$P(k$ arrivals in service time$|$service time $= x) = \frac{\lambda x)^k}{k!} e^{-\lambda x}$ and
$P($service time $\in (x, x + dx)) = f(x)dx$

2.18.1 Solutions to the balance equations

- In principle we could solve the balance equations by "forward substitution"

  - p_0 is known
  - $j = 0 \implies p_1$

    $\vdots$
  - $j = i : \quad p_0, p_1, \ldots, p_i \implies p_{i+1}$

  but computationally impractical in general
- Define generating functions

$$\begin{aligned} p(z) &= \sum_{i=0}^{\infty} p_i z^i \\ r(z) &= \sum_{i=0}^{\infty} r_i z^i \end{aligned}$$

Recap:

$$
\begin{aligned}
p'(z) &= \sum_{i=1}^{\infty} i p_i z^{i-1} \\
&\Longrightarrow\ p'(1) = \text{mean value of distribution } \{p_j | j \geq 0\} \\
p''(z) &= \sum_{i=2}^{\infty} (i^2 - i) p_i z^{i-2} \ \Longrightarrow\ p''(1) = M_2 - M_1
\end{aligned}
$$

- Multiply balance equations by z^j and sum:

$$
\begin{aligned}
p(z) &= p_0 r(z) + \sum_{j=0}^{\infty} \sum_{i=1}^{j+1} p_i r_{j-i+1} z^j \\
&= p_0 r(z) + z^{-1} \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} p_{i+1} z^{i+1} r_{j-i} z^{j-i} \\
&\quad \text{where } r_k = 0 \text{ for } k < 0 \\
&= p_0 r(z) + z^{-1} \sum_{i=0}^{\infty} p_{i+1} z^{i+1} \sum_{j=0}^{\infty} r_{j-i} z^{j-i} \\
&= p_0 r(z) + z^{-1} (p(z) - p_0) r(z)
\end{aligned}
$$

2.18.2 Solution for $p(z)$ and the Pollaczek-Khinchine result

$$\begin{aligned}
p(z) &= \frac{p_0(1-z)r(z)}{r(z)-z} \\
\text{where } r(z) &= \int_0^\infty \sum_{k=0}^\infty \frac{(\lambda xz)^k}{k!} e^{-\lambda x} f(x)dx \\
&= \int_0^\infty e^{-\lambda x(1-z)} f(x)dx \\
&= f^*(\lambda - \lambda z) \\
& \text{Laplace transform of } f \text{ at the point } \lambda - \lambda z
\end{aligned}$$

Recap: Laplace transform f^* of f defined by

$$\begin{aligned}
f^*(s) &= \int_0^\infty e^{-sx} f(x)dx \\
\text{so that } \frac{d^n}{ds^n} f^*(s) &= \int_0^\infty (-x)^n e^{-sx} f(x)dx \\
\implies \left.\frac{d^n f^*(s)}{ds^n}\right|_{s=0} &= (-1)^n M_n \quad n^{th} \text{ moment of } f(x)
\end{aligned}$$

E.g. $f^*(0) = 1$, $-f^{*\prime}(0) = \text{mean service time} = 1/\mu$. Thus,

$$p(z) = \frac{(1-\rho)(1-z)f^*(\lambda - \lambda z)}{f^*(\lambda - \lambda z) - z}$$

$p'(1) \implies$ P-K formula …

2.18.3 Derivation of P-K formula

$$\frac{p'}{1-\rho} = \frac{(f^* - z)(-\lambda(1-z)f^{*\prime} - f^*) + (1-z)f^*(1+\lambda f^{*\prime})}{(f^* - z)^2}$$

where $f^* = f^*(\lambda - \lambda z)$ etc.

- When $z = 1$, both denominator and nominator vanish
  $(\implies f^*(\lambda - \lambda.1) = f^*(0) = 1)$
- L'Hopital rule $\implies$

$$\frac{p'(1)}{1-\rho} = \lim_{z\to 1}\left\{\frac{\lambda((1-2z)f^{*\prime} - \lambda z(1-z)f^{*\prime}) - \lambda f^{*\prime} + 2\lambda f^* f^{*\prime}}{-2(f^* - z)(1+\lambda f^{*\prime})}\right\}$$

- Again, when $z = 1$, both denominator and nominator vanish

- L'Hopital rule now gives

$$\begin{aligned}\frac{p'(1)}{1-\rho} &= \frac{\lambda(-2f^{*\prime}(0) + \lambda f^{*\prime\prime}(0) - 2\lambda f^{*\prime}(0)^2)}{2(1+\lambda f^{*\prime}(0))^2} \\ &\quad (\text{since } f^*(0) = 1) \\ &= \frac{\lambda^2 M_2 + 2(\lambda/\mu)(1-(\lambda/\mu))}{2(1-(\lambda/\mu))^2} \quad \text{P-K formula!} \\ &\quad (\text{since } f^{*\prime}(0) = 1/\mu)\end{aligned}$$

- Hard work compared to previous derivation! But in principle we could obtain *any* moment ("well known" result for variance of queue length)

2.18.4 Waiting time distribution

- The tasks left in an M/G/1 queue on departure of a given task are precisely those which arrived during its waiting time

$$\begin{aligned}
\Longrightarrow p_j &= \int_0^\infty \frac{(\lambda x)^j}{j!} e^{-\lambda x} h(x) dx \\
&\quad \text{because } P(j \text{ arrivals}|\text{waiting time} = x) = \frac{(\lambda x)^j}{j!} e^{-\lambda x} \\
&\quad P(\text{waiting time} \in (x, x+dx)) = h(x)dx \\
\Longrightarrow p(z) &= h^*(\lambda - \lambda z)
\end{aligned}$$

by the same reasoning as before.

- Laplace transform of waiting time distribution is therefore (let $z = \frac{\lambda - s}{\lambda}$)

$$h^*(s) = p\Big(\frac{\lambda - s}{\lambda}\Big) = \frac{(1-\rho) s f^*(s)}{\lambda f^*(s) - \lambda + s}$$

- Exercise: Verify Little's Law for the M/G/1 queue:

$$p'(1) = -\lambda h^{*\prime}(0).$$

2.19 Example: Paging fixed head disk

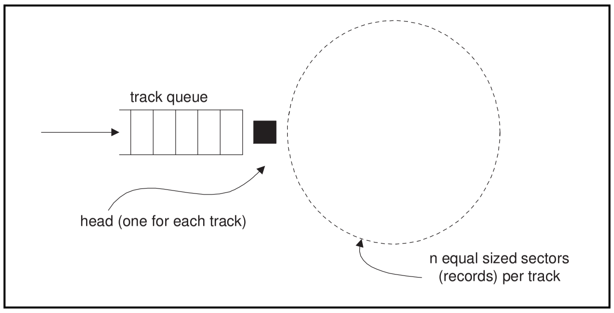


Figure 16: Paging fixed head disk.

Assumptions

- Tasks in the track queue require random sector
- arrivals to an empty queue always find the head at the beginning of a sector (as with the next task in the queue after a service

completion)

$$\Longrightarrow \text{ service times may be } \quad \begin{array}{ll} \frac{1}{nR} & \text{(requires next sector)} \\ \frac{2}{nR} & \text{(next but one)} \\ \vdots & \\ \frac{1}{R} & \text{(the one just gone)} \end{array}$$

(strictly, for arrivals to an empty queue, service times have continuous sample space $[1/nR, 1/nR + 1/R)$)

- Mean service time, $\mu^{-1} = \sum_{j=1}^{n} \frac{1}{n} \frac{j}{nR} = \frac{n+1}{2nR}$
- Second moment, $M_2 = \sum_{j=1}^{n} \frac{1}{n} (\frac{j}{nR})^2 = \frac{(n+1)(2n+1)}{6n^2R^2}$

2.19.1 Solution and asymptotic behaviour

- Load is $\rho = \lambda/\mu = \frac{\lambda(n+1)}{2nR} \implies \lambda < \frac{2nR}{n+1}$ if drum track is not to be saturated
- Mean queue length, $L = \rho + \frac{\lambda^2(n+1)(2n+1)}{12n^2R^2(1-\rho)}$
- As $n \to \infty$, i.e. many sectors on track
  - assumption about arrivals to an empty queue becomes exact (large $n \implies$ good approximation)
  - $\rho \to \frac{\lambda}{2R}$
  - $L \to \frac{\lambda}{2R} + \frac{\lambda^2}{3R(2R-\lambda)}$

3 Queueing Networks

3.1 Introduction

- Collection of servers/queues interconnected according to some topology

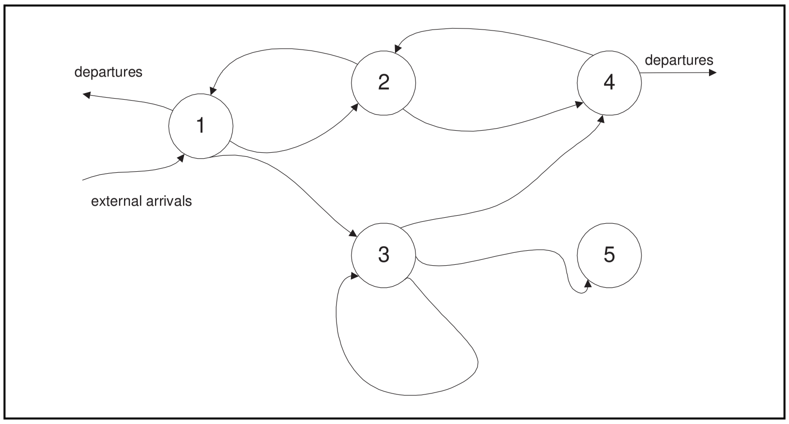


Figure 17: Network example

- Servers may be
  - processing elements in a computer, e.g. CPU I/O devices
  - stations/nodes in a communication network (may be widely separated geographically)
- Topology represents the possible routes taken by tasks through the system
- May be several different classes of tasks (multi-class network)
  - different service requirements at each node
  - different routing behaviours
  - more complex notation, but straightforward generalisation of the single-class network in principle
  - we will consider only the single class case

3.2 Types of network

- Open: at least one arc coming from the outside and at least one going out
  - must be at least one of each type or else the network would be saturated or null (empty in the steady state)
  - e.g. the example above
- Closed: no external (incoming or outgoing) arc
  - constant network population of tasks forever circulating
  - e.g. the example above with the external arcs removed from nodes 1 and 4
- Mixed: multi-class model which is open with respect to some classes and closed with respect to others – e.g. in the example above a class whose tasks only ever alternated between nodes 2 and 4 would be closed, whereas a class using all nodes would be open

3.3 Types of server

- Server defined by its service time distribution (we assume exponential but can be general for non-FCFS disciplines) and its queueing discipline (for each class)
  - FCFS (FIFO)
  - LCFS (LIFO)
  - IS (i.e. a *delay node*: no queueing)
  - PS (*Processor sharing*: service shared equally amongst all tasks in the queue)
- Similar results for queue length probabilities (in S.S.) for all

4 Open networks (single class)

- Notation:
  M servers, $1, 2, \ldots, M$ with FCFS discipline and exponential service times, mean $\frac{1}{\mu_i(n_i)}$, when queue length is n_i $(1 \le i \le M)$
  - *state dependent service rates* to a limited extent
  - $\mu_i(n_j)$ for $i \neq j \implies$ *blocking*: rate at one server depends on the state of another, e.g. rate $\to 0$ when finite queue at next is full
- External Poisson arrivals into node i, rate γ_i, $(1 \le i \le M)$ $(= 0$ if no arrivals)
- Routing probability matrix $Q = (q_{ij} | 1 \le i \le M)$
  - q_{ij} = probability that on leaving node i a task goes to node j independently of past history

- $q_{i0} = 1 - \sum_{j=1}^{M} q_{ij}$ = probability of external departure from node i
- at least one $q_{i0} > 0$, i.e. at least one row of Q sums to less than 1

- State space of network $S = \{(n_1, \ldots, n_M)\}|n_i \geq 0\}$
  - queue length vector random variable is $(N_1, \ldots, N_M)$
  - $p(\underline{n}) = p(n_1, \ldots, n_M) = P(N_1 = n_1, \ldots, N_M = n_M)$

4.1 Traffic equations

- Determine mean arrival rates λ_i to each node i in the network

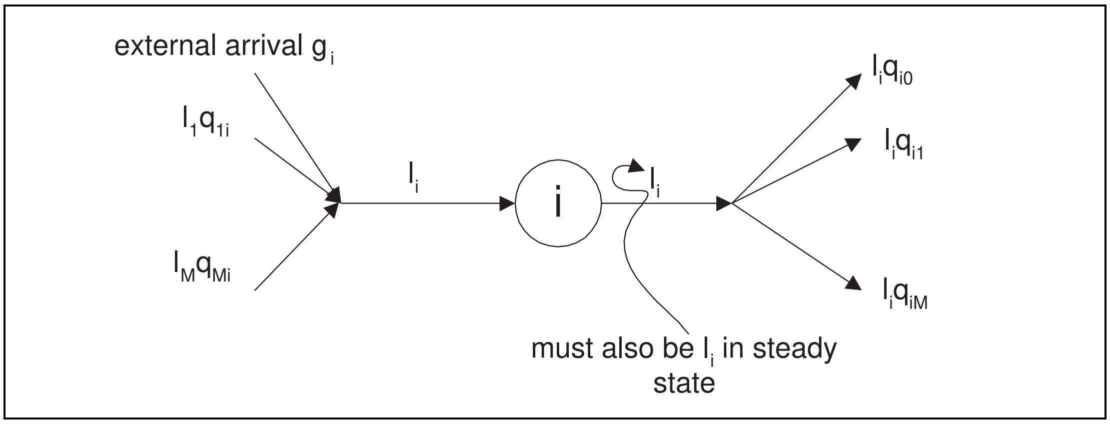


Figure 18: Traffic equations

- In the steady state, $\lambda_i = \gamma_i + \sum_{j=1}^{M} \lambda_j q_{ji}$ for $(1 \leq i \leq M) \implies$ unique solution for $\{\lambda_i\}$ (because of properties of Q)

- independent of Poisson assumption since we are only considering mean numbers of arrivals in unit time
- assumes only the existence of a steady state

- Example

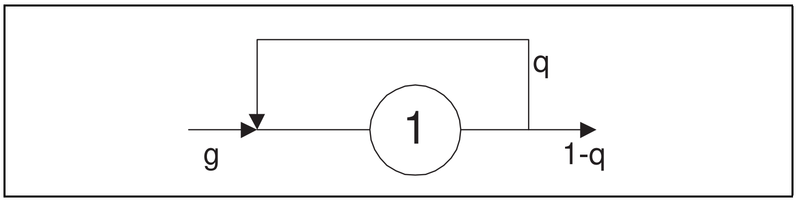


Figure 19: Traffic example.

$$\lambda_1 = \gamma + \lambda_1 q \implies \lambda_1 = \frac{\gamma}{1-q}$$

- Arrivals to a node are not in general Poisson, e.g. this simple example. If there is no feedback then all arrival processes are Poisson because
  1. departure process of M/M/1 queue is Poisson
  2. superposition of independent Poisson processes is Poisson
  3. Similarly, let R_i be the average interval between a task's arrival at server i and its departure from the network
     - R_i is the "average remaining sojourn time"
     -

$$R_i = W_i + \sum_{j=1}^{M} q_{ij} R_j$$

4.2 Steady state queue length probabilities

Jackson's theorem

1. The number of tasks at any server is independent of the number of tasks at every other server in the steady state
2. Node i behaves *as if* it were subjected to Poisson arrivals, rate λ_i $(1 \leq i \leq M)$

- Thus, even though arrivals at each node are not, in general, Poisson, we can treat the system as if it were a collection of M independent M/M/1 queues

$$\Longrightarrow \; p(n_1, \ldots, n_M) \propto \prod_{i=1}^{M} \left\{ \frac{\lambda_i^{n_i}}{\prod_{j=1}^{n_i} \mu_i(j)} \right\}$$

where $\mu_i(j)$ is the rate of server i when the queue length is j

- If service rates are constant, μ_i,

$$p(\underline{n}) \approx \prod_{i=1}^{M} \rho_i^{n_i} = \prod_{i=1}^{M} (1 - \rho_i) \rho_i^{n_i}$$

where $\rho_i = \frac{\lambda_i}{\mu_i}$

- $p(\underline{n}) \rightarrow$ usual performance measures such as mean queue lengths, utilisations, throughput by standard methods – mean waiting times by Little's result

N.B. The normalising constant for $\{p(\underline{n}) | \underline{n} \in S\}$ is not shown for the general case: it is the product of those for the M/M/1 queues.

4.3 Mean Value analysis

- Can exploit Jackson's theorem directly, together with Little's result, if only average quantities are required
  - values for mean queue lengths L_i are those for isolated M/M/1 queues with arrival rates λ_i $(1 \leq i \leq M)$
  - assuming constant service rates μ_i

$$L_i = \frac{\rho_i}{1-\rho_i} \text{ for } 1 \leq i \leq M$$

(average number of tasks at node i)

  - total average number of tasks in network

$$L = \sum_{i=1}^{M} L_i = \sum_{i=1}^{M} \frac{\rho_i}{1-\rho_i}$$

- Waiting times
  - Average waiting time in the network, $W = L/\gamma$ by Little's result where $\gamma = \sum_{i=1}^{M} \gamma_i$ is the total arrival rate
  - Average time spent at node i during each visit

$$W_i = L_i/\lambda_i = \frac{1}{\mu_i(1-\rho_i)}$$

  - Average time spent queueing on a visit to node i

$$W_{Qi} = L_{Qi}/\lambda_i = \frac{\rho_i}{\mu_i(1-\rho_i)}$$

4.4 An alternative formulation

- Let v_i be the average number of visits made by a task to server i

$$\gamma v_i = \text{average number of arrivals to node } i \text{ in unit time} = \lambda_i,$$

where γ is the average number of arrivals to the whole network in unit time and v_i the average number of visits a given arrival makes to node i

$$v_i = \frac{\lambda_i}{\gamma} \quad (1 \le i \le M)$$

- Let D_i be the total average service demand on node i from one task

$$D_i = \frac{v_i}{\mu_i} = \frac{\rho_i}{\gamma}$$

  $\rho_i = \gamma D_i =$ average work for node i arriving from outside the network in unit time

- Often specify a queueing network directly in terms of $\{D_i | 1 \le i \le M\}$ and γ; then there is no need to solve the traffic equations

-

$$L_i = \frac{\rho_i}{1-\rho_i} = \frac{\gamma D_i}{1-\gamma D_i}$$

  L and W as before

- Let $B_i =$ total average time a task spends at node i

$$B_i = v_i W_i = \frac{v_i}{\mu_i(1-\rho_i)} = \frac{D_i}{1-\gamma D_i}$$

  alternatively apply Little's result to node i with the external arrival process directly

$$B_i = \frac{L_i}{\gamma} = \frac{D_i}{1-\gamma D_i}$$

- However, D_i and γ cannot be used to determine μ_i and hence neither W_i nor R_i

- Specification for delay nodes (IS) i

- $L_i = \rho_i = \gamma D_i$
- $B_i = D_i$ (no queueing)
- $W_i = 1/\mu_i$
- $D_i = v_i/\mu_i$ (as for any work conserving discipline also)
- service time distribution arbitrary

4.5 Distribution of time delays

- Even though arrivals at each node are not, in general, Poisson, the Random Observer Property still holds: *waiting time distribution at node i is exponential, parameter* $\mu_i - \lambda_i$ again as expected from Jackson's theorem.
- Networks with no overtaking ("tree-like" networks) are easy to solve for time delay distributions:

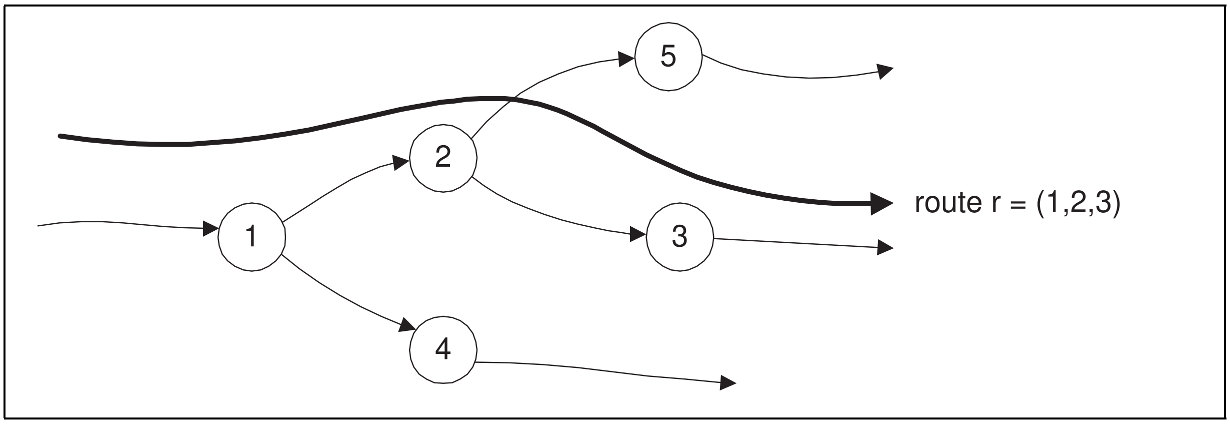


Figure 20: A network with no overtaking.

- sojourn time on any route is the sum of independent, exponential random variables
- this argument is independent of Jackson's theorem and one

proof uses the idea of *reversibility* of the M/M/1 queue

- time delay distribution is a convolution of exponentials, e.g. $f_1 \star f_2 \star f_3$ for route r where $f_i(t) = (\mu_i - \lambda_i)e^{-(\mu_i - \lambda_i)t}$ for $i = 1, 2, 3$.

4.6 Time delays in general networks

- mean sojourn time for any route is always easy because the mean of a sum of random variables is equal to the sum of the means of those random variables, whether or not they are independent
- in networks with overtaking, the distribution of route sojourn times remains an open problem. For example, in the network

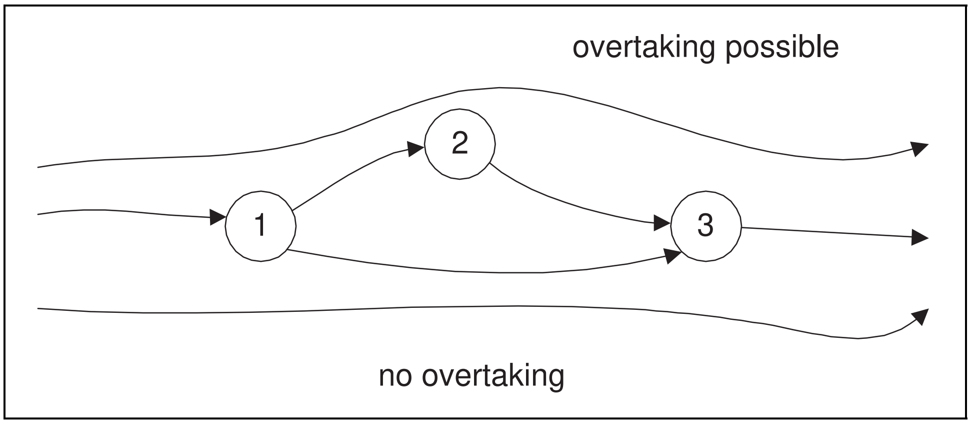


Figure 21: A network with overtaking.

- sojourn time distribution on route $1 \to 3$ is the convolution of 2 exponentials
- sojourn time distribution on route $1 \to 2 \to 3$ is *not* the convolution of 3 exponentials because the queue length faced at node 3 upon arrival depends on the number of departures

from node 1 during the sojourn time at node 2.
  - The Random Observer Property is not applicable since the arrival to node 3 is not random when conditioned on the previous arrival at node 1.

- Jackson's theorem does not apply because it is concerned only with steady state probabilities, i.e. the asymptotic behaviour of $p_t(\underline{n})$ at the single point in time t as $t \to \infty$.
- Subject of many research papers.

5 Closed Queueing Networks

- No external arrivals or departures (no γ_i terms).
- Routing probabilities satisfy

$$\sum_{j=1}^{M} q_{ij} = 1 \text{ for } 1 \leq i \leq M$$

- State space $S = \{(n_1, \ldots, n_M) | n_i \geq 0, \sum_{j=1}^{M} n_j = K\}$ for population K
  - $|S| =$ # of ways of putting K balls into M bags $= \binom{K+M-1}{M-1}$
  - finiteness of $S \to$ steady state always exists

- Traffic equations are

$$\lambda_i = \sum_{j=1}^{M} \lambda_j q_{ji} \text{ for } 1 \le i \le M$$

  - homogeneous linear equations with an infinity of solutions which differ by a multiplicative factor (because $|I - Q| = 0$ since rows all sum to zero)
  - let $(e_1, \ldots, e_M)$ be any non-zero solution (typically chosen by fixing one e_i to a *convenient* value, like 1)

  therefore $e_i \propto$ arrival rate , λ_i, i.e. $e_i = c\lambda_i$
  $x_i = \frac{e_i}{\mu_i} \propto$ load $= \rho_i$, i.e. $x_i = c\rho_i$

5.1 Steady state probability distribution for S

- Jackson's theorem extends to closed networks which have a product form solution

$$p(n_1, \ldots, n_M) = \frac{1}{G} \prod_{i=1}^{M} \frac{e_i^{n_i}}{\prod_{j=1}^{n_i} \mu_i(j)} \text{ where } \sum_{i=1}^{M} n_i = K. \tag{1}$$

  where $\mu_i(j)$ is the service rate of the exponential server i when its queue length is j.

- G is the normalising constant of the network

$$G = \sum_{\boldsymbol{n} \in S} \prod_{i=1}^{M} \frac{e_i^{n_i}}{\prod_{j=1}^{n_i} \mu_i(j)} \tag{2}$$

  not easy to compute (see later)

- Prove the result by using the network's steady state balance equations:

$$\begin{aligned}
\text{Total flux out of state } \boldsymbol{n} &= p(\boldsymbol{n})\sum_{i=1}\mu_i(n_i)\epsilon(n_i) \\
&= \text{Total flux into state } \boldsymbol{n} : \\
\sum_{i,j}\boldsymbol{n}_i^j \to \boldsymbol{n} &= \sum_{i=1}^{M}\sum_{j=1}^{M} p(\boldsymbol{n}_i^j)\epsilon(n_i)\mu_j((\boldsymbol{n}_i^j)_j)q_{ji}
\end{aligned}$$

where $\epsilon(n) = \begin{cases} 0 \text{ if } n = 0 \\ 1 \text{ if } n > 0 \end{cases}$

$\boldsymbol{n}_i^j = \begin{cases} (n_1, \ldots, n_i - 1, \ldots, n_j + 1, \ldots, n_M) \text{ if } i \neq j \\ \boldsymbol{n}_i^i = \boldsymbol{n} \text{ otherwise} \end{cases}$

note that $(\boldsymbol{n}_i^j)_j = \begin{cases} n_j + 1 \text{ if } i \neq j \\ n_i \text{ otherwise} \end{cases}$

- Try

$$p(n_1, \ldots, n_M) = \frac{1}{G}\prod_{i=1}^{M}\frac{e_i^{n_i}}{\prod_{j=1}^{n_i}\mu_i(j)} \text{ where } \sum_{i=1} n_i = K. \tag{3}$$

Then require

$$\begin{aligned}
\frac{1}{G}\prod_{i=1}^{M}\frac{e_i^{n_i}}{\prod_{j=1}^{n_i}\mu_i(j)}\sum_i \mu_i(n_i)\epsilon(n_i) &= \frac{1}{G}\prod_{i=1}^{M}\frac{e_i^{n_i}}{\prod_{j=1}^{n_i}\mu_i(j)}\sum_{i,j:i=j}\mu_i(n_i)\epsilon(n_i)q_{ji} \\
&+\frac{1}{G}\prod_{i=1}^{M}\frac{e_i^{n_i}}{\prod_{j=1}^{n_i}\mu_i(j)}\times \\
&\sum_{i,j:i\neq j}\frac{e_j}{e_i}\frac{\mu_i(n_i)}{\mu_j(n_j+1)}\epsilon(n_i)\mu_j(n_j+1)q_{ji}
\end{aligned}$$

i.e.

$$\sum_i \mu_i(n_i)\epsilon(n_i) = \sum_i \mu_i(n_i)\epsilon(n_i)\{\frac{\sum_j e_j q_{ji}}{e_i}\}$$

OK if $\boldsymbol{e}$ satisfies the traffic equations.

- Note that if $\boldsymbol{e}' = c\boldsymbol{e}$ is the correct solution to the traffic equations, the corresponding probabilities $p'(\boldsymbol{n})$ and G' are

$$p'(\boldsymbol{n}) = \frac{1}{G'} G c^{\sum n_i} p(\boldsymbol{n}) = \frac{Gc^K}{G'} p(\boldsymbol{n})$$

$$G' = c^K G = \sum_{\boldsymbol{n}\in S} G c^{\sum n_i} p(\boldsymbol{n})$$

and therefore $p'(\boldsymbol{n}) = p(\boldsymbol{n})$. This confirms the arbitrariness of $\boldsymbol{e}$ up a to multiplicative factor.

5.2 Computation of the normalising constant

- We consider only the case of servers with constant service rates to get an efficient algorithm.
- There are also algorithms for networks with servers having state dependent service rates, e.g. the convolution algorithm.
- Less efficient but important since the alternative *MVA algorithm* also requires constant rates.
- Define $G = g(K, M)$ where

$$g(n,m) = \sum_{\underline{n}\in S(n,m)} \prod_{i=1}^{m} x_i^{n_i}$$

where $S(n,m) = \{(n_1, \ldots, n_m) | n_i \geq 0, \sum_{i=1}^m n_i = n\}$ and $x_i = \frac{e_i}{\mu_i}$ $(1 \leq i \leq m)$.

- state space for subnetwork of nodes $1, 2, \ldots, m$ and population n.
- For $n, m > 0$

$$\begin{aligned}
g(n,m) &= \sum_{\underline{n} \in S(n,m), n_m = 0} \prod_{i=1}^{m} x_i^{n_i} + \sum_{\underline{n} \in S(n,m), n_m > 0} \prod_{i=1}^{m} x_i^{n_i} \\
&= \sum_{\underline{n} \in S(n,m-1)} \prod_{i=1}^{m-1} x_i^{n_i} + x_m \sum_{\substack{k_i = n_i (i \neq m) \\ k_m = n_m - 1 \\ \underline{n} \in S(n,m)}} \prod_{i=1}^{m} x_i^{k_i} \\
&= g(n, m-1) + x_m g(n-1, m)
\end{aligned}$$

because $\{\underline{k} | k_i \geq 0; \sum_{i=1}^{m} k_i = n - 1\} = S(n-1, m)$.

- Boundary conditions for

$$\begin{aligned}
g(0,m) &= 1 \text{ for } m > 0 \\
g(n,0) &= 0 \text{ for } n \geq 0
\end{aligned}$$

- Note

$$g(0,m) = \sum_{\underline{n} = (0,\ldots,0)} \prod_{i=1}^{m} x_i^0 = 1$$

and

$$g(n,1) = x_1 g(n-1,1) = x_1^n$$

5.3 Marginal queue length probabilities and performance measures

- Although $p(\underline{k}) \propto \prod p_i(k_i)$ it is *not* the case that $P(N_i = k_i) = p_i(k_i)$, as in the open networks. The use of M/M/1 factors is just a convenient mathematical device, there is no probabilistic interpretation.
- Probability that a server is idle ($= 1 -$ utilisation)

$$P(N_M = 0) = \frac{1}{g(K,M)} \sum_{\underline{n} \in S(n,m), n_m = 0} \prod_{i=1}^{M-1} x_i^{n_i} = \frac{g(K, M-1)}{g(K,M)}.$$

In general

$$P(N_i = 0) = \frac{G_i(K)}{G(K)} \text{ for } 1 \leq i \leq M$$

where $G(K) = g(K,M)$ and $G_i(k)$ is the normalising constant for

the same network with the server i removed and population k

$$G_i(k) = \sum_{\underline{n} \in S(k, M-1)} \prod_{j=1}^{M-1} y_j^{n_j}$$

where

$$y_j = \begin{cases} x_j \text{ for } 1 \leq j < i \\ x_{j+1} \text{ for } i \leq j \leq M-1 \end{cases}$$

- utilisation of node i

$$U_i(K) = 1 - \frac{G_i(K)}{G(K)}$$

5.4 Alternative Formulation: Cumulative Probabilities

- For $1 \leq k \leq K$ and $1 \leq i \leq M$

$$\begin{aligned}
P(N_i \geq k) &= \sum_{\underline{n} \in S(K,M), n_i \geq k} \prod_{j=1}^{M} \frac{x_j^{n_j}}{G(K)} \\
&= \frac{x_i^k}{G(K)} \sum_{\substack{m_i = n_i - k \\ m_j = n_j (j \neq i), n_i \geq k \\ \underline{n} \in S(K,M)}} \prod_{j=1}^{M} x_j^{m_j} \\
&= \frac{x_i^k}{G(K)} \sum_{\underline{m} \in S(K-k,M)} \prod_{j=1}^{M} x_j^{m_j} \\
&= x_i^k \frac{G(K-k)}{G(K)}.
\end{aligned}$$

Therefore the utilisation is given by

$$U_i = x_i \frac{G(K-1)}{G(K)}$$

- Equating two expressions for U_i yields the recurrence relation for $g(k,m)$ previously
- Throughput of server i,

$$T_i(k) = \mu_i U_i(k) = e_i \frac{G(K-1)}{G(K)}$$

proportional to visitation rate as expected.

- Queue length distribution at server i is $P(N_i = k) = p_i(k)$ for

$0 \leq k \leq K$ and $1 \leq i \leq M$

$$\begin{aligned} p_i(k) &= P(N_i \geq k) - P(N_i \geq k+1) \\ &= x_i^k \frac{G(K-k) - x_i G(K-k-1)}{G(K)}. \end{aligned}$$

where $G(-1) = 0$ by definition.

- Notice that the previous formulation gives a more concise formulation for $p_i(k)$

$$\begin{aligned} p_i(k) &= \frac{1}{G(K)} \sum_{\underline{n} \in S(K,M), n_i = k} \prod_{j=1}^{M} x_j^{n_j} \\ &= \frac{x_i^k}{G(K)} \sum_{\underline{n} \in S(K-k,M), n_i = 0} \prod_{j=1}^{M} x_j^{n_j} \\ &= x_i^k \frac{G_i(K-k)}{G(K)} \end{aligned}$$

for $0 \leq k \leq K$ and $1 \leq i \leq M$. This is a generalisation of the result obtained for $U_i(k)$.

- Mean queue length at server i, $1 \leq i \leq M$, $L_i(k)$

$$\begin{aligned} L_i(K) &= \sum_{k=1}^{K} k P(N_i = k) \\ &= \sum_{k=1}^{K} P(N_i \geq k) \\ &= \frac{\sum_{k=1}^{K} x_i^k G(K-k)}{G(K)}. \end{aligned}$$

5.5 Equivalent open networks and the use of mean value analysis

- Consider an irreducible network – one in which every arc is traversed within finite time from a given time with probability 1 –

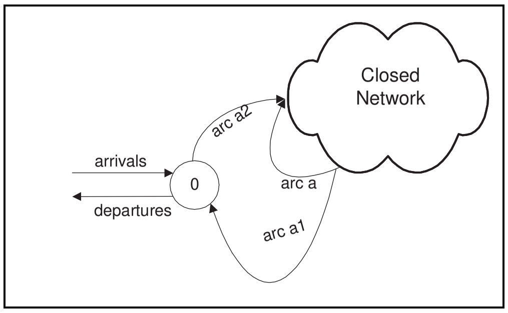


Figure 22: Equivalent open network.

- i.e. a network represented by an irreducible positive recurrent Markov process (finite state space)

- we introduce a node, 0, in one of the arcs and replace arc a by arc a_1, node 0 and arc a_2.
  - Source of a_1 is source of a
  - destination of a_2 is destination of a
- Whenever a task arrives at node 0 (along arc a_1), it departs from the network and is immediately replaced by a stochastically identical task which leaves node 0 along arc a_2
- Define the network's throughput, T, to be the average rate at which tasks pass along arc a in the steady state.
  - i.e. T is mean number of tasks traversing a in unit time.
  - One can choose any arc in an irreducible network.

5.6 Visitation rates and application of Little's result

- Let the visitation rate be v_i and the average arrival rate be λ_i for node i, $1 \leq i \leq M$, then $\lambda_i = Tv_i$ where T is the external arrival rate.
- The set $\{v_i | 1 \leq i \leq M\}$ satisfies the traffic equations, as we could have derived directly by a similar argument.
- Suppose arc a connects node α to node β in the closed network $1 \leq \alpha, \beta \leq M$, then $v_0 = v_\alpha q_{\alpha\beta}$ because all traffic from α to β goes through node 0 in the open network.
- But every task enters node 0 exactly once, hence $v_\alpha = \frac{1}{q_{\alpha\beta}}$ since $v_0 = 1$. This determines $\{v_i | 1 \leq i \leq M\}$ uniquely.

- Little's result now yields

$$L_i = \lambda_i W_i = T v_i W_i$$

$$\sum_{i=1}^{M} L_i = K = T \sum_{i=1}^{M} v_i W_i$$

since the sum of all queue lengths is exactly K in the closed network

$$\Longrightarrow \qquad T = \frac{K}{\sum_{i=1}^{M} v_i W_i}$$

5.7 Mean waiting times

- $$W_i = \frac{1}{\mu_i}[1 + Y_i]$$

  where the first term is the arriving task's mean service time and the Y_i is the mean number of tasks seen by new arrivals at server i.
- For an IS (delay) server $W_i = \frac{1}{\mu_i}$, otherwise ...
- Do *not* have the random observer property
  - number of tasks seen on arrival does not have the same steady state distribution as the queue length since K tasks are seen with probability 0. (arrival cannot "see itself")
  - do not have $Y_i = L - 1$ as in open networks
- Do have the analogous Task (or Job) Observer Property:

The state of a closed queueing network in equilibrium seen by a new arrival at any node has the same distribution as that of the state of the same network in equilibrium with the arriving task removed (i.e. with population $K - 1$)

- arriving task behaves as a random observer in a network with population reduced by one
- intuitively appealing since the "one" is the task itself
- but requires a lengthy proof (omitted)

5.8 Recurrence relations for throughput, mean waiting times and mean queue length

- Task observer property yields

$$Y_i(K) = L_i(K-1)$$

for $1 \leq i \leq M, K > 0$.

- Hence we obtain the recurrence relations

$$\begin{aligned} W_i(K) &= \frac{1}{\mu_i}[1 + L_i(K-1)] \\ T(K) &= \frac{K}{\sum_{i=1}^{M} v_i W_i(K)} \\ L_i(K) &= v_i T(K) W_i(K) \end{aligned}$$

for $1 \leq i \leq M, K > 0$ and the initial condition $L_i(0) = 0$

- $T(K), W_i(K), L_i(K)$ easily computed by a simple iteration

$$\{L_i(K-1)\} \rightarrow \underbrace{\{W_i(K)\} \rightarrow T(K)}_{\rightarrow \{L_i(K)\} \rightarrow \ldots}$$

- $2M + 1$ quantities computed each time round the loop

5.9 Alternative formulation

- Total average time spent at node i when population is K

$$B_i(K) = v_i W_i(K)$$

for $1 \leq i \leq M$

- Total average service time (demand) a task requires from node i

$$D_i = \frac{v_i}{\mu_i}$$

for $1 \leq i \leq M$ independent of population K.

- Therefore

$$\begin{aligned} B_i(K) &= D_i[1 + L_i(K-1)] \\ T(K) &= \frac{K}{\sum_{i=1}^{M} B_i(K)} \\ L_i(K) &= T(K)B_i(K) \end{aligned}$$

for $1 \leq i \leq M$

- Total average time in network spent by a task

$$W(K) = \sum_{i=1}^{M} v_i W_i(K) = \sum_{i=1}^{M} B_i(K) = \frac{K}{T(K)}$$

as expected by Little's law

- Utilisation of node i

$$U_i(K) = \frac{\lambda_i}{\mu_i} = T(K)\frac{v_i}{\mu_i} = T(K)D_i \leq 1$$

which implies

$$T(K) \le \min_{1 \le i \le M} \frac{1}{D_i}$$

which implies the maximum throughput is dictated by a bottleneck server (or servers) - that (those) with maximum D_i

5.10 A faster approximate algorithm

- Algorithms given above need to evaluate $2M+1$ quantities for each population between 1 and K
- Would be faster if we could relate $Y_i(K)$ to $L_i(K)$ rather than $L_i(K-1)$, which implies that we do not need to worry about populations less than K
- $$Y_i(K) = \frac{K-1}{K} L_i(K)$$

  is a good approximation, exact for $K=1$ and correct asymptotic behaviour as $K \to \infty$ (exercise)
- then

  $$B_i(K) = D_i\Big[1 + \frac{K-1}{K} T(K) B_i(K)\Big]$$

- $$B_i(K) = \frac{D_i}{1 - \frac{K-1}{K} D_i T(K)}$$

- Thus we obtain the fixed point equation

$$T(K) = f(T(K))$$

where

$$f(x) = \frac{K}{\sum_{i=1}^{M} \frac{D_i}{1 - \frac{K-1}{K} D_i x}}$$

there are many numerical methods to solve such equations

5.11 Example: Multiprogramming computer system

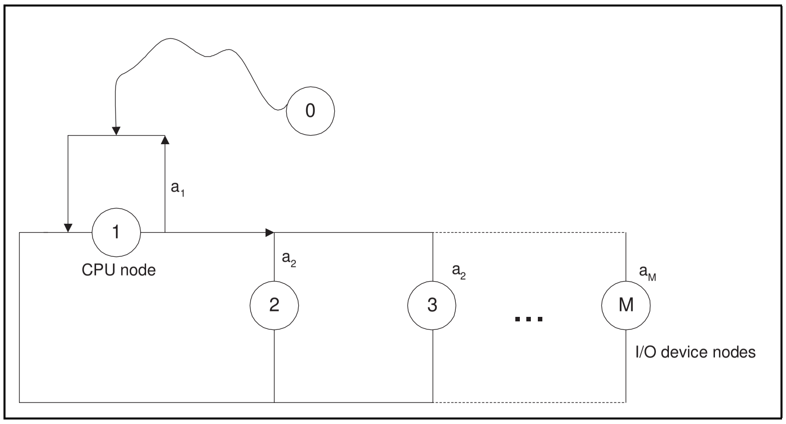


Figure 23: A multiprogrammed computer.

- Insert node 0 in route from CPU back to itself, *throughput* is the rate at which tasks are interrupted at CPU node, this is an arbitrary choice

- visitation rates

$$v_1 = a_1 v_1 + \sum_{j=2}^{M} v_j$$

$$v_i = a_i v_1 \text{ for } i \geq 2$$

$$v_1 = \frac{1}{a_1} \text{ with choice of position of node 0}$$

therefore

$$v_i = \frac{a_i}{a_1} \text{ for } i \geq 2$$

- Sanity check: is the first equation is satisfied?

$$D_i = \frac{a_i}{a_1 \mu_i} \text{ for } i \geq 2$$

$$D_1 = \frac{1}{a_1 \mu_1}$$

therefore $T(K)$ follows by solving recurrence relations

- total CPU throughput $= \frac{T(K)}{a_1}$ (fraction a_1 recycles)

5.12 Application: A batch system with virtual memory

- Most medium-large computer systems use virtual memory
- It is well-known that above a certain degree of multiprogramming performance deteriorates rapidly
- Important questions are
  - How does the throughput and turnaround time vary with the degree of multiprogramming
  - What is the threshold below which to keep the degree of multiprogramming?

5.12.1 Representation of paging

- Suppose node 2 is the paging device (e.g. fast disk)

- assume dedicated to paging
- easy to include non-paging I/O also

- We aim to determine the service demand of a task at node 2, D_2, from tasks' average paging behaviour
  - use results from '70s working set theory
  - consider the page fault rate for varying amounts of main store allocated to a task
- Let the population be K
- paging behaviour will be represented by a *lifetime function*:
  - $h(x)$ = average CPU time between consecutive page faults when a task has x pages of memory
  - $h(0) \approx 0$ (the first instruction causes a fault)
  - $h(x) = C$, the total CPU time of a task for $x \geq$ some constant value m, where m is the "size" of the task

- h is an *increasing function*: the more memory a task has the less frequent will be its page faults on average.
- Two possible types of lifetime functions:

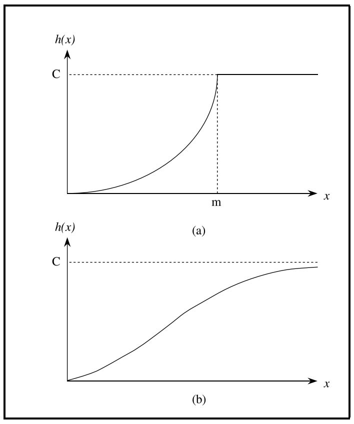


Figure 24: Lifetime functions

  - For (a) $h(x) = \begin{cases} ax^b \text{ for } x \le m \\ am^b = C \text{ for } x > m \end{cases}, b > 1$
  - For (b) $h(x) = \frac{C}{1+(a/x)^2}$
  - Note that for (b) there is no m s.t. $h(m) = C$

- Let each task have P/K pages of memory of size P pages
  - average CPU time between faults $= h(P/K)$
  - average number of faults during life of task $= \frac{D_1}{h(P/K)}$
  - average time at node 2 per fault $= \frac{1}{\mu_2}$

- Therefore average paging time of a task (from node 2)
  - $D_2 = H(K) = \frac{D_1}{\mu_2 h(P/K)}$
  - $D_2 = d_2 + H(K)$ if average non-paging requirement of a task from node 2 is d_2

- As $K \to \infty$, $T(K) \to \infty$ since $T(K) \le \min \frac{1}{D_i}$ and $D_2 \to \infty$

5.13 Solution

- We again use the alternative form of the MVA algorithm
- For population K and $1 \le k \le K$

$$\begin{aligned} B_i(k) &= D_i(K)[1 + L_i(k-1)] \quad (1 \le i \le M) \\ T(k) &= \frac{k}{\sum_{i=1}^{M} B_i(k)} \\ L_i(k) &= T(k)B_i(k) \quad (1 \le i \le M) \end{aligned}$$

- Notice that only D_2 is a function of $K - D_1$ and D_3 are constant
- Computation for a range of populations requires 2 nested loops:
  - outer loop for $K = 1, 2, \ldots, K_{max}$

- inner loop for $k = 1, 2, \ldots, K$
- need new value for $H(K)$ each time round the outer loop

5.14 Choice of lifetime function

- We assume (arbitrarily) the lifetime function

$$h(x) = \begin{cases} D_1(\frac{x}{10000})^{3.5}, & x \leq 10000 \\ D_1 \text{ otherwise} \end{cases}$$

for all tasks, i.e.

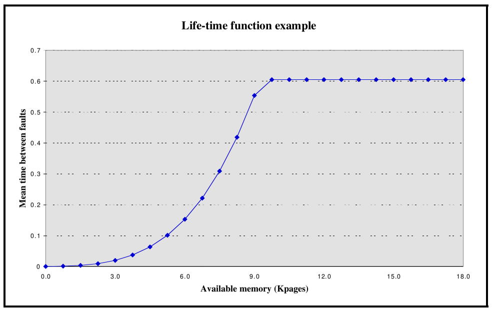


Figure 25: Lifetime function example.

5.15 Thrashing curves

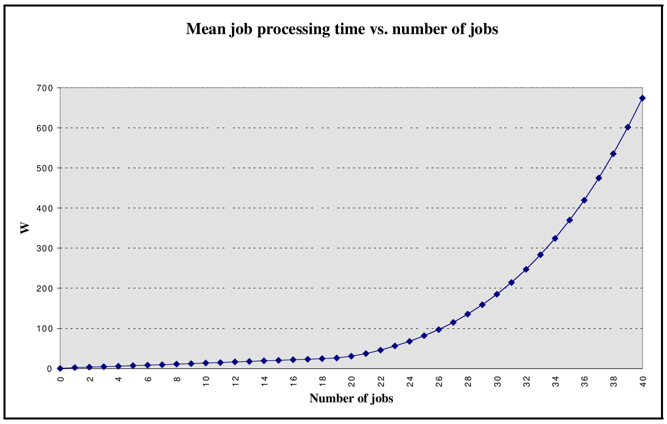


Figure 26: Mean job processing time vs. number of jobs.

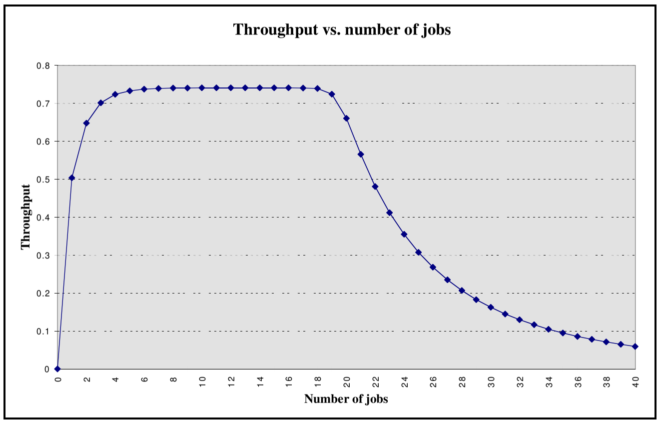


Figure 27: Throughput vs. number of Jobs.

5.16 Decomposition

- Separable queueing models are prone to break down in some situations:
  - There is usually a good model for small parts of any network
  - Composing the bits can be hard unless the criteria for separability are (reasonably) met
- Ideally, we need a method to:
  - Reduce complexity (breaking a large problem into smaller manageable ones)
  - Provide approximate solutions where the necessary assumptions do not hold
- The most powerful general method is *decomposition* (sometimes called *aggregation*)
- The idea is to replace a subnetwork of a (complex) system with a single *flow-equivalent server* (FES)

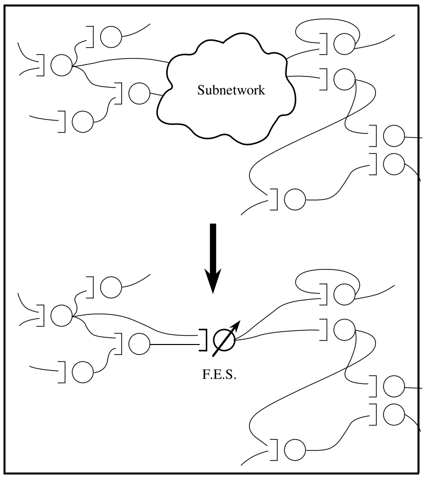


Figure 28: Flow equivalent server.

- We exploit the fact that servers in separable networks may have *state dependent* service rates, i.e. μ_n for queue length n
- To find the required μ_n, we compute the "throughput", τ_n, of the

subnetwork when the population is n and set

$$\mu_n = \tau_n, \quad n = 0, 1, \ldots$$

- The throughput can be determined by *any* method, e.g. an explicit Markov model, standard queueing network algorithm (e.g. MVA), or even simulation
- The technique is analogous to Norton's Theorem in electrical circuit theory.

5.17 Method

1. Identify the subnetwork to be aggregated

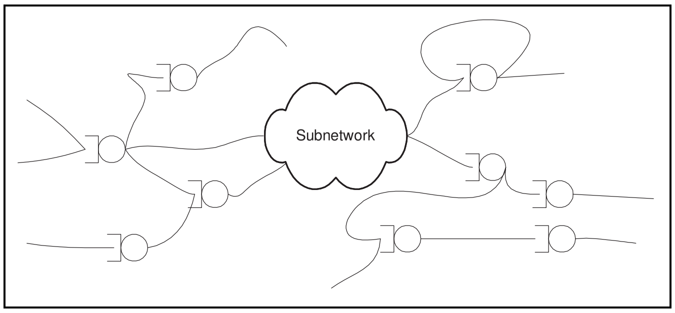


Figure 29: Flow equivalent server - step 1.

2. Short-circuit the rest of the network (i.e. use the same visitation rates as in the original subnetwork)

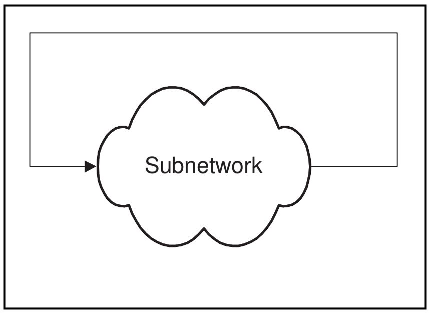


Figure 30: Flow equivalent server - step 2.

3. Compute the throughput τ_n, on the short-circuit arc for all populations $0, 1, \ldots$

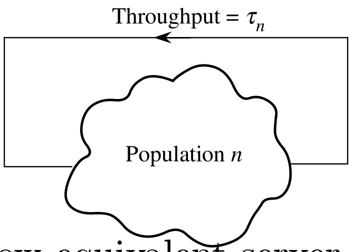


Figure 31: Flow equivalent server - step 3.

4. Define a single FES with service rate $\mu_n = \tau_n, n = 0, 1, \ldots$
5. Replace the subnetwork with the FES and re-solve

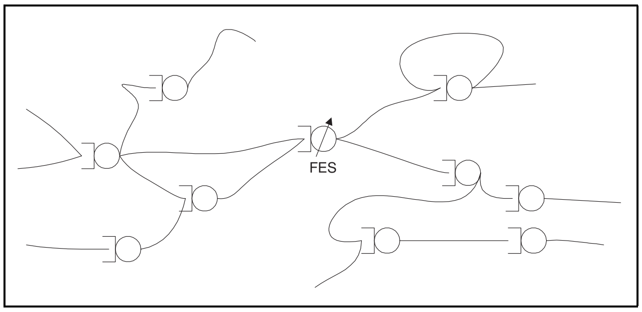


Figure 32: Flow equivalent server - step 5.

6. If necessary repeat the process (i.e. hierarchically)
7. Note: We can either aggregate the tricky part or aggregate the easy part leaving a smaller tricky network behind!

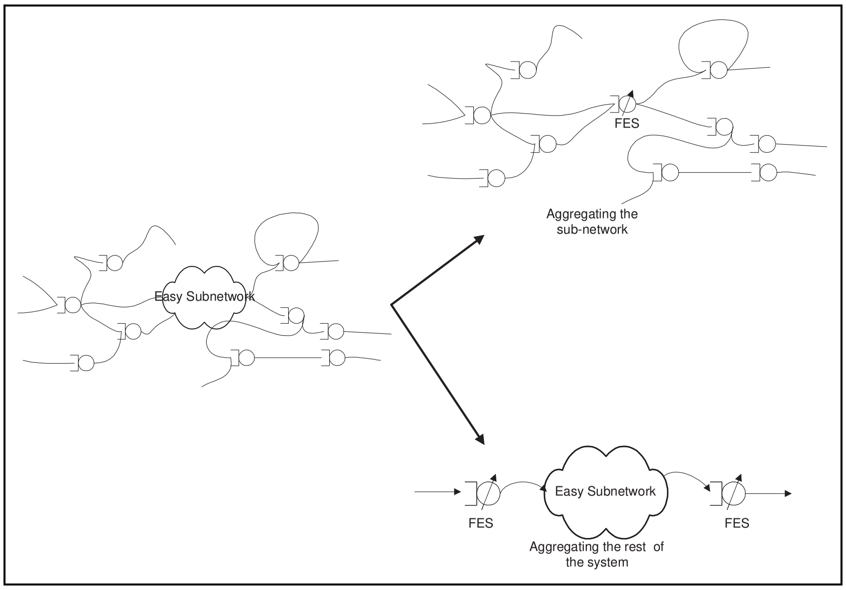


Figure 33: Flow equivalent server - trick.

5.18 Application: A multiprogramming system with virtual memory

- Let us now revisit the multiprogrammed virtual memory system, this time with terminals rather than batch processing:

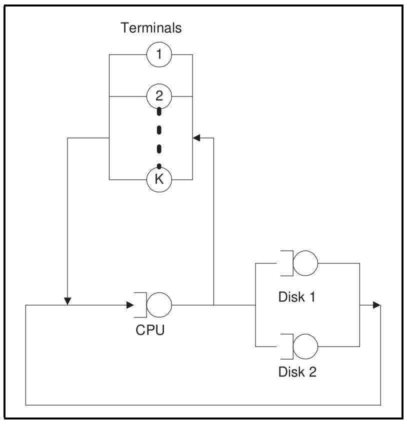


Figure 34: Multiprogramming system with virtual memory

- The average think time of each user is 30 seconds
- Q: How does the system throughput and response time vary with the number of terminals, K, and at what point does the system start to thrash?
- As users submit jobs, they increase the population at the computer system and consume memory
- The problem is that the performance of the computer subsystem depends on the number of jobs there
- But the number of jobs at the subsystem is governed by the number of terminals
- Specifically, the Disk 1 service rate depends on the population of the entire subnetwork: no simple (product-form) solution

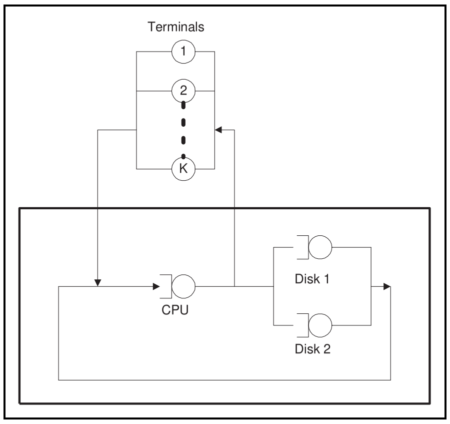


Figure 35: Multiprogramming system with virtual memory - step 1

- In the batch model we computed previously the throughput,τ_n, when there were n batch jobs being processed
- We can thus aggregate the server subsystem into a

flow-equivalent server whose service rate at population n is

$$\mu_n = \tau_n$$

- That is

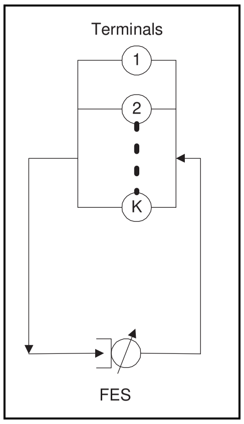


Figure 36: Multiprogramming system with virtual memory - FES

- approximation because of lack of product-form solution

- The resulting reduced network is a simple M/M/1 queue with state dependent arrival and service rates:

$$\begin{aligned} \lambda_n &= (K-n)\lambda \\ \mu_n &= \tau_n \end{aligned}$$

  where $0 \leq n \leq K$ and where $\lambda = 1/30$ is the submission rate for a single terminal in the "think" state

- Let p_n be the equilibrium probability that there are n jobs at the subsystem (i.e. $K-n$ users in "think" mode)
- The magic formula for the M/M/1 queue gives

$$\begin{aligned} p_n &= \frac{(K-n+1)\lambda}{\mu_n}\frac{(K-n)\lambda}{\mu_{n-1}}\cdots\frac{K\lambda}{\mu_1}p_0 \\ &= \lambda^n \frac{K!}{(K-n)!}\prod_{i=1}^{n}\frac{1}{\mu_i} \end{aligned}$$

with the usual constraint that

$$p_0 + p_1 + \cdots + p_k = 1$$

- The algorithm requires one loop to compute p_0 and a further loop to find the p_n, $1 \leq n \leq K$
- We are interested here in the throughput and mean response time for a given value of K
- The throughput is:

$$\tau = p_0\mu_0 + p_1\mu_1 + \cdots + p_K\mu_K$$

- We next need to find the mean population of the subsystem:

$$L = 0 \times p_0 + 1 \times p_1 + \cdots + K \times p_K$$

- The mean waiting time then follows from Little's law

$$W = L/\tau$$

- With the same paging model as the batch system earlier, this is what happens to τ as we increase K

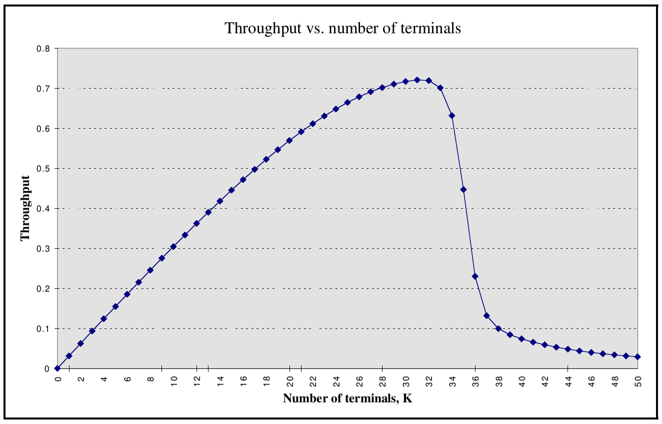


Figure 37: Throughput vs. number of terminals.

- The saturation point is clear! Beyond this the mean population of the computer subsystem is such that thrashing occurs
- Here is the behaviour of the mean waiting time as we increase W

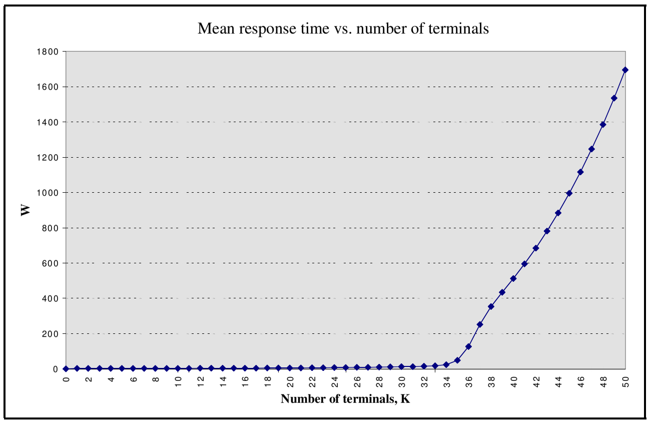


Figure 38: Mean response time vs. number of terminals.

- And finally (for the record), here is L:

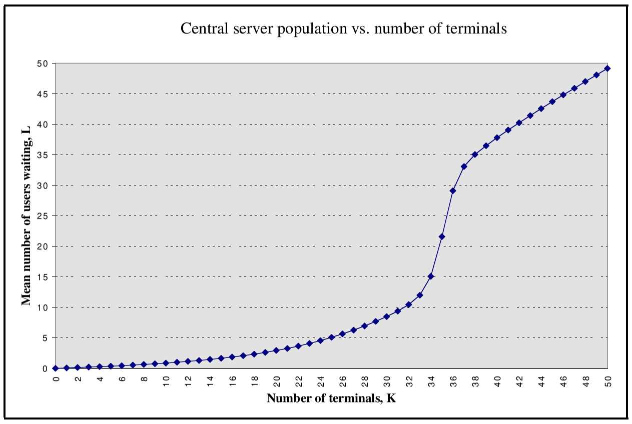


Figure 39: Central server population vs. number of terminals.

6 The Markov Modulated Poisson Process (MMPP)

6.1 Introduction

- Poisson process is a good model for many arrival streams
  - more short interarrival times than long
  - superposition of many sparse independent renewal processes is asymptotically Poisson
  - right intuitive properties given minimal information (i.e. arrival *rate*)

- But cannot model many characteristics seen in modern networks

  - multiple traffic types with known switching pattern between types
  - correlated traffic, i.e. non-zero correlation between interarrival times
  - self-similarity

- MMPP accounts for some of these limitations
  - models stochastically changing modes with different rates
  - has non-zero autocorrelation which can be calculated
  - integrates well into queueing models – the MMPP/M/1 queue

6.2 Definition

- The MMPP is a Poisson process with rate that depends on the state (called *phase*) of an independent continuous time finite Markov chain $X(t)$
  - N is the number of phases
  - λ_k is the rate of the Poisson process in phase k
  - so the probability of an arrival in a small interval of length h in phase k is

$$\lambda_k h + o(h)$$

- The modulating Markov chain $X(t)$ is assumed irreducible and so has a steady state

  - let the generator matrix (instantaneous transition rate matrix) be Q
  - let the equilibrium probability of being in state k be

$$\pi_k = \lim_{t \to \infty} P(X(t) = k)$$

- Average arrival rate in steady state is then

$$\overline{\lambda} = \sum_{k=1}^{N} \pi_k \lambda_k$$

6.3 State at Arrival Instants

- Important for response time distributions
- Do *not* have the Random Observer Property
- In a long period of time T at equilibrium
  - expected time spent in state k is $\pi_k T$
  - expected number of arrivals that see state $k = \lambda_k \pi_k T$
  - total number of arrivals has mean $\overline{\lambda} T$
- So probability that an arrival sees state k in steady state is

$$\pi'_k = \frac{\lambda_k \pi_k}{\overline{\lambda}}$$

6.4 The MMPP/M/1 queue

- Suppose the queue has MMPP arrival process as defined above and server with exponential service times of constant mean μ^{-1}
- Simple CTMC
  - easy to write down the balance equations
  - apply steady state theorem
  - conditions harder to derive rigorously but expect that the queue has a steady state iff

$$\overline{\lambda} < \mu$$

- The CTMC has a 2-dimensional state space

- phase of the MMPP horizontally (say)
- queue length vertically
- infinite lattice strip for unbounded queue

- Let the steady state probability for phase k and queue length j be denoted by π_{jk} and let the vector $\mathbf{v}_j = (\pi_{j1}, \pi_{j2}, \ldots, \pi_{jN})$

- Balance equations are, for state $(i, j+1)$

$$\pi_{j+1,i}\left(\lambda_i + \mu + \sum_{k \neq i} q_{ik}\right) = \sum_{k \neq i} \pi_{j+1,k} q_{ki} + \lambda_i \pi_{j,i} + \mu \pi_{j+2,i}$$

6.5 The MMPP/M/1 queue (continued)

- This may be written in matrix form as

$$\mathbf{v}_j \Lambda + \mathbf{v}_{j+1}(Q - \Lambda - \mu I) + \mu \mathbf{v}_{j+2} I = \mathbf{0}$$

where I is the unit $N \times N$ matrix and Λ is the diagonal matrix with the arrival rates λ_k down the diagonal, i.e.
$\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$

- For states $(i, 0)$ we get

$$\mathbf{v}_0(Q - \Lambda) + \mu \mathbf{v}_1 I = \mathbf{0}$$

- The normalising equation is

$$\sum_{j=0}^{\infty} \mathbf{v}_j \cdot (1, 1, \ldots, 1) = 1$$

- These equations give the unique equilibrium probabilities if $\overline{\lambda} < \mu$

6.6 Exact solution methods

- We have a matrix recurrence relation of the form

$$\mathbf{v}_j Q_0 + \mathbf{v}_{j+1} Q_1 + \mathbf{v}_{j+2} Q_2 = \mathbf{0}$$

for $j \geq 0$ and constant matrices Q_0, Q_1, Q_2

- **Spectral Analysis** (Chakka and Mitrani; see Mitrani's book, last chapter)
  - find the eigenvalues ξ_k and eigenvectors $\mathbf{e}_k$ of a certain $N-$dimensional matrix given by Q and $(\lambda_1, \ldots, \lambda_N)$
  - solution can then be shown to be, for a queue of unbounded capacity

$$\pi = \sum_{k=1}^{N} a_k \xi^k \mathbf{e}_k$$

  where the a_k are constants determined by the equations at queue length 0 and normalisation
  - more complex result for queues of finite capacity

- **Matrix Geometric Method**
  - Closely related to Spectral Analysis but a completely different approach
  - Try a solution of the form $\mathbf{v}_j = M^j \mathbf{u}$ for some matrix M and vector $\mathbf{u}$
  - Then solve

$$Q_0 + MQ_1 + M^2 Q_2 = 0$$

    for M
  - $\mathbf{u}$ from the other equations

6.7 Approximate solution methods

- It often happens that phase changes are rare in comparison to arrivals and departures
  - rates in Q-matrix $<< \lambda_1, \ldots, \lambda_N, \mu$
  - e.g. phase changes represent change of traffic type, such as data, video, voice

- Often then a good approximation to use (an approximate) decomposition, cf. queueing networks

- For each phase k, solve the M/M/1 queue assuming the arrival rate is constant, λ_k, giving equilibrium probability $p_j^{(k)}$ for queue length j, $1 \leq k \leq N$

  - i.e. solve all the ‘columns’ in the 2-D state space
  - can only work if every $\lambda_k < \mu$

- Then solve the phase process for the equilibrium probability r_k of being in phase k, i.e. solve

$$\mathbf{r} \cdot Q = \mathbf{0}$$

- Approximate the solution by

$$\pi_{jk} = p_j^{(k)} r_k$$

6.8 Fitting MMPPs

- MMPP can be fitted to observed data by matching:
  - average arrival rate $\overline{\lambda}$ (or mean interarrival time)
  - higher moments of interarrival time (variance etc.)
  - autocorrelation function (ACF)
  - Hurst parameter, a measure of self-similarity

- Matching moments is easy but often gives a poor match on correlation, the main reason for using the MMPP in the first place!

- Can use ACF to model correlated traffic directly
  - next slide

  - computationally difficult and expensive
  - possibly use spectral methods, cf. Fourier transforms

- Hurst parameter gives a good representation of 'how correlated' traffic is, but only contributes one parameter in the matching process
- Better used to validate another matching process

6.9 Autocorrelation function of the MMPP

- The autocorrelation function at lag $k \geq 1$ for the above MMPP is

$$\rho_k = \frac{\pi^* R^{-2}\Lambda[\{I - \mathbf{e}\pi^* R^{-1}\Lambda\}\{R^{-1}\Lambda\}^{k-1}]R^{-2}\Lambda\mathbf{e}}{2\pi^* R^{-3}\Lambda\mathbf{e} - (\pi^* R^{-2}\Lambda\mathbf{e})^2}$$

  where the matrix $R = \Lambda - Q$, $\mathbf{e} = (1, 1, \ldots, 1)$ and $\pi^* = \frac{\pi\Lambda}{\pi.\lambda}$.

- Autocorrelation function is easily calculated by the above formula, *given* the parameters of the MMPP

- But the converse, to determine the parameters from the ACF, is hard!
  - gives non-linear equations in several variables (the sought-after parameters)
  - fixed point methods of solution tend to be unstable
  - research area

References

[IM] Isi Mitrani, *Probabilistic Modelling*, CUP 1998, ISBN 0 521 58530 9