More Queuing Models

(Chapter 5)

1

Multiple Servers – M/M/s

- Add more servers to speed up service
  - Let s = # of servers
- If $n >= s$ then combined service rate = $s\mu$
- If $n < s$ then combined service rate = $n\mu$
- Assume the former: $\rho = \lambda/s\mu$
- The equations for P_0, $P(n>s)$, L_s, L_q now become the following:

2

M/M/s Queueing Models

$$P_0 = 1 \Big/ \left(\sum_{i=1}^{s-1} (s\rho)^i / i! + (s\rho)^s / s!(1-\rho) \right)$$

$$P(n \geq s) = (s\rho)^s P_0 \big/ \big(s!(1-\rho)\big)$$

$$L_s = s\rho + (s\rho)^{s+1} P_0 \big/ \big(s(s!)(1-\rho)^2\big)$$

$$L_q = P_0 \big(s^{s+1}\rho^{s+1}/s\big) \big/ s!(1-\rho)^2$$

$$W_s = L_s/\lambda \qquad W_q = L_q/\lambda$$

3

Grocery Store (Supermarket)

Better describes a Bank!

- *M/M/s* with 4 checkout lanes
  - $\lambda = 100$ customers/hr.
  - $\mu = 30$ customers/hr.
  - $\rho = \lambda/s\mu = .8333$
- From the previous equations we compute:
  - $L_s = 6.62$, $L_q = 3.29$
  - $W_s = .0662$ hrs. = 4 minutes
  - $W_q = .0329$ hrs. = 2 minutes

4

Bank vs. Supermarket

- *M/M/s* vs. Independent *M/M/1* queues
  - Balking
    - Not entering the queue
    - both
  - Reneging
    - Leaving the line from the middle
    - both
  - Jockeying
    - Entering a different line
    - Multiple *M/M/1's*

5

Non-Markovian Queuing Models

- *M/G/1* Model
  - Congestion is directly related to the variance in service time (σ^2 decreases, so does congestion)

$$P_0 = 1 - \rho$$
$$L_s = \rho + \rho^2(1 + \sigma^2\mu^2) / 2(1 - \rho)$$
$$L_q = L_s - \rho$$
$$W_q = L_q / \lambda$$

6

More Analytic Models

- *M/M/1/K*
  - The Barber Shop example
- *M/D/s, D/M/s*
  - *W.r.t.* E(wait time)/E(service time)
    - *M/M/x > M/D/x > D/M/x for x = 1, 2, 3, …s*
    - (pp. 162-163)
- *M/G/s/inf*
  - Hotel Lobby

7

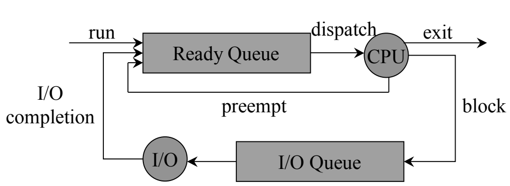
G/G/s – The Most General QM

- Little's Law: $L = \lambda W$
  - E(# in system) = λE(time in system)
- $W_s = W_q + 1/\mu$
  - E(time in system) = E(wait time) + avg. service time
- $L_s = L_q + \lambda/\mu$
  - E(# in system) = E(# in queue) + traffic density

8

Queuing Network Models

- Networks of queues best describe computing systems, and other systems



9

Queuing Network Models

- Departures from one queue become arrivals at another queue in the network
  - Very complex to analyze
  - Except in the case of *M/M/s* queues:
    - Where at steady state, times between departures are exponentially distributed! (Good News)
    - Surprisingly, the queuing discipline doesn't affect the distribution times between departures
    - So, each queue can be analyzed independently as *M/M/1* queues

10

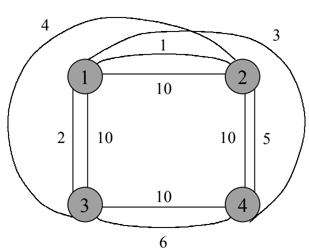
Queuing Network Example

- A communication network connects cities
- Every edge carries packets between 2 cities
- Every pair of nodes has some packet traffic
- On average, how long does it take a packet to travel from its source to its destination?
- Network delay can be modeled as an *M/M/s* queuing network model

11

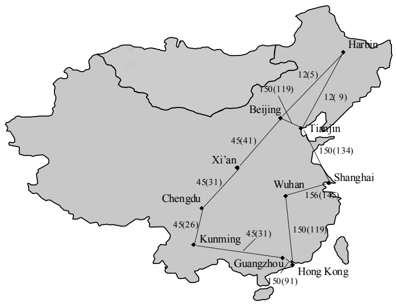
Multi-Commodity Flow

	1	2	3	4
1	0	1	2	3
2	1	0	4	5
3	2	4	0	6
4	3	5	6	0



12

Ten Chinese Cities



13

Modeling Packet Delay

- ❖Each edge is a single queue/single server
- ❖The network is modeled as an *M/M/s* QNM
  - ❖The path for each source/sink pair is fixed
  - ❖The output from one queue feeds the neighboring queue/server pair(s)
  - ❖Queuing theory tells us that each edge can be modeled as an *independent M/M/1* queue
  - ❖An analytic solution exists for such networks

14

Packet Delay Solution

- ❖Using independent *M/M/1* queues where:
  - ❖m = number of edges in the network
  - ❖γ = total of all demands between all nodes
  - ❖f_i = total packet flow in edge i
  - ❖c_i = capacity of edge i
  - ❖Kleinrock shows that network delay is:

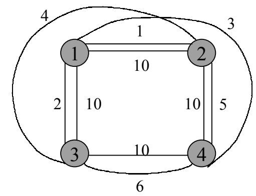
$$N_D = \frac{1}{\gamma}\sum_{i=1}^{m}\frac{f_i}{c_i - f_i}$$

The true inventor of the internet (not Al Gore)!

15

Kleinrock Packet Delay Calc.

	1	2	3	4
1	0	1	2	3
2	1	0	4	5
3	2	4	0	6
4	3	5	6	0



$$N_D = \frac{1}{\gamma}\sum_{i=1}^{m}\frac{f_i}{c_i - f_i}$$

$f_1 = 8;\quad f_2 = 6;\quad f_3 = 6;\quad f_4 = 8;\quad \gamma = 42$

$c_1 = 10;\quad c_2 = 10;\quad c_3 = 10;\quad c_4 = 10;$

$$N_D = \frac{1}{42}\left(\frac{8}{2}+\frac{6}{4}+\frac{6}{4}+\frac{8}{2}\right) = \frac{11}{42} = 26.19\%$$

16

Analytic Inventory Models

- ❖Queuing systems model congestion
- ❖Inventory systems model cost:
  - ❖C = unit *purchase* cost
  - ❖S = unit *sales* price
  - ❖C_H = holding cost (per unit time)
  - ❖C_R = replenishing cost
  - ❖C_P = penalty cost (backordering, etc.)
  - ❖C_S = salvage cost (value of scrap)

17

Analytic Inventory Models

- ❖Uncontrolled variables:
  - ❖D = customer *demand* for the item
  - ❖L = *lead time* required for reordering
- ❖Inventory policy variables:
  - ❖Q = replenishment order *quantity*
  - ❖Reorder Rule = rule used to trigger reorder
    - ❖Inventory level (automatic or by inspection)
- ❖Static vs. Dynamic Inventory Models

18

Static Inventory Model

- ❖A single-period inventory model (eg. 1 yr.)
- ❖An expected revenue model:
  - ❖E(Revenue) = E(Sales) – Purchase Cost – E(Penalty Cost) + E(Salvage Income)
  - ❖Demand is probabilistic and given by P_k

$$E(\$) = \sum_{k=1}^{Q} SkP_k + QS \sum_{k=Q+1}^{\infty} P_k - CQ - \sum_{k=Q+1}^{\infty} C_P (k-Q) P_k + \sum_{k=0}^{Q-1} C_S (Q-k) P_k$$

Sales - Purchase - Penalty + Salvage

19

Newspaper Boy Problem

- ❖A newspaper boy can purchase papers for $0.15 and sell them for $0.25. Any unsold papers can be returned for $0.10
- ❖Demand curve for papers is:

Demand	Prob.
12	.05
13	.10
14	.20
15	.30
16	.20
17	.10
18	.05

20

Expected Profit Calculation

Q	Expected Sales	Expected Salvage	Purchase Cost	E(Profit)
12	$3.00	$0.00	$1.80	$1.200
13	3.24	0.01	1.95	1.293
14	3.45	0.03	2.10	1.380
15	3.61	0.06	2.25	1.418
16	3.70	0.12	2.40	1.420
17	3.74	0.21	2.55	1.393
18	3.75	0.30	2.70	1.350

21

Dynamic Inventory Model

- A *continuous review* is taken (electronically?) of the inventory level
- When inventory falls below R_p (reorder pt.) an order of size Q is placed
- The *demand* and the *lead time* for reordering are *deterministic* (constant)
- The reorder pt. Can be set so that no shortages occur and the inventory is exhausted just as the replenishment order arrives

22

Dynamic Inventory Model

- The total cost model for such a system is:
  - $C_H CQ/2 + C_R D/Q$
    - Where D is the annual rate of demand for the item
- Taking the derivative and setting it to zero, we get the optimal value for Q …
  - $Q = \text{sqrt}(2DC_R/C_H C)$
- …and the reorder point should be:
  - $L \times D$ (where L is the lead time to replenish)

23

Supplier Example 1

- An electronics parts supplier inventories a component that:
  - Has a yearly demand of 10,000 units
  - Each unit costs $1.00 to manufacture
  - Annual carrying costs are 12% of purchase price
  - Each order costs $10.00 to place
  - Lead time to fill an order is 10 days
  - Supplier currently orders 2,000 at once

24

Supplier Example 1

- Supplier wants to know:
  - What does it cost them to inventory the item?
  - What is the optimal order size?

$C_H = 0.12;\ C = \$1;\ Q = 2000;\ C_R = \$10;\ D = 10{,}000$

$0.12(\$1)(2000/2) + \$10(10{,}000/2{,}000) = \$170$

Optimal order size = sqrt(2(10,000)$10/.12($1)) = 1290

with a reorder level of (10,000/365)10 = 274

25

Supplier Example 2

- What if demand and lead time are random?
  - And follow a uniform dist'n
    - Between 200 and 400 units for demand
  - Then there's a penalty for being short
    - C_S = $0.50 per unit
  - Total cost model is now more complicated
    - E(U) = expected # of orders filled during lead time
    - E(S) = expected # of units short during the lead time
    - Let *f(x)* = prob. density fn. of demand during LT

26

Supplier Example 2

$$E(\$) = C_H C(Q/2 + R_P - E(U)) + C_R D/Q + C_P E(S) D/Q$$

$$E(U) = \int_0^{R_P} x\, f(x)\, dx, \text{ and}$$

$$E(S) = \int_{R_P}^{\infty} (x - R_P)\, f(x) dx$$

this system is optimally solved when :

$$Q = \sqrt{2E(D)(C_R + C_P E(S))/C_H C}, \text{ and}$$

$$P(D > R_P) = QC_H C / C_P E(D)$$

27

Supplier Example 2

- ❖Solve these equations iteratively
  - ❖First assume *E(S)* is zero and find *Q*
  - ❖Then solve for R_p
  - ❖Feed solution back into first eqn to find new *Q*
  - ❖Iterate until values reach steady state

28