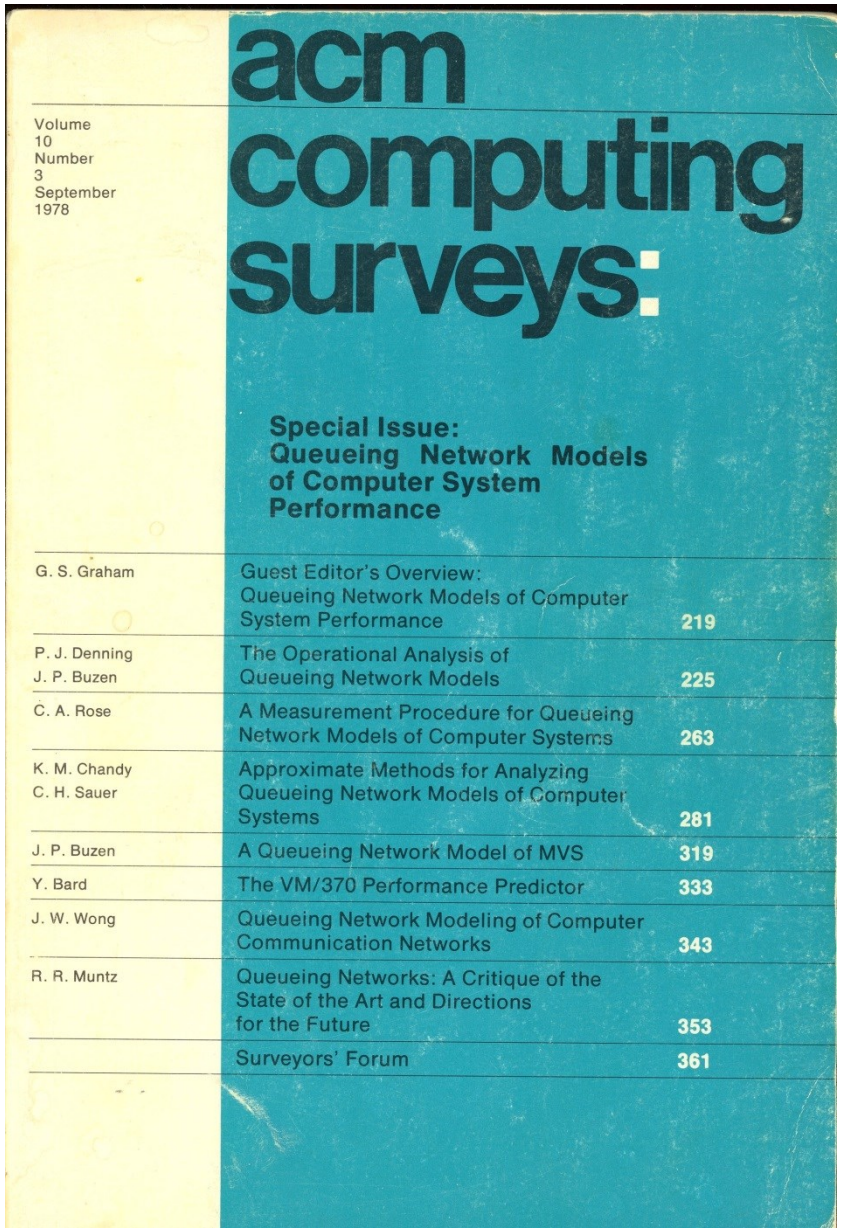

acm
computing
surveys:
Volume
10
Number
3
September
1978
Special Issue:
Queueing Network Models
of Computer System
Performance
G. S. Graham
Guest Editor's Overview:
Queueing Network Models of Computer
System Performance 219
P. J. Denning
J. P. Buzen
The Operational Analysis of
Queueing Network Models 225
C. A. Rose
A Measurement Procedure for Queueing
Network Models of Computer Systems 263
K. M. Chandy
C. H. Sauer
Approximate Methods for Analyzing
Queueing Network Models of Computer
Systems 281
J. P. Buzen
A Queueing Network Model of MVS 319
Y. Bard
The VM/370 Performance Predictor 333
J. W. Wong
Queueing Network Modeling of Computer
Communication Networks 343
R. R. Muntz
Queueing Networks: A Critique of the
State of the Art and Directions
for the Future 353
Surveyors' Forum 361

Predicting performance

- Operational Analysis:
  - Operational analysis of network models
- Also
  - Markov models

Numbers Everyone Should Know

L1 cache reference	0.5 ns	
Branch mispredict	5 ns	
L2 cache reference	7 ns	
Mutex lock/unlock	25 ns	
Main memory reference	100 ns	
Compress 1K w/cheap compression algorithm	3,000 ns	
Send 2K bytes over 1 Gbps network	20,000 ns	
Read 1 MB sequentially from memory	250,000 ns	250µs
Round trip within same datacenter	500,000 ns	
Disk seek	10,000,000 ns	
Read 1 MB sequentially from disk	20,000,000 ns	
Send packet CA->Netherlands->CA	150,000,000 ns	150ms

https://www.youtube.com/watch?v=modXC5IWTJI

The Operational Analysis of Queueing Network Models*

PETER J. DENNING

Computer Sciences Department, Purdue University, West Lafayette, Indiana 47907

JEFFREY P. BUZEN

BGS Systems, Inc., Box 128, Lincoln, Massachusetts 01773

Queueing network models have proved to be cost effective tools for analyzing modern computer systems. This tutorial paper presents the basic results using the operational approach, a framework which allows the analyst to test whether each assumption is met in a given system. The early sections describe the nature of queueing network models and their applications for calculating and predicting performance quantities The basic performance quantities—such as utilizations, mean queue lengths, and mean response times—are defined, and operational relationships among them are derived Following this, the concept of job flow balance is introduced and used to study asymptotic throughputs and response times. The concepts of state transition balance, one-step behavior, and homogeneity are then used to relate the proportions of time that each system state is occupied to the parameters of job demand and to device characteristics Efficient methods for computing basic performance quantities are also described. Finally the concept of decomposition is used to simplify analyses by replacing subsystems with equivalent devices. All concepts are illustrated liberally with examples

Keywords and Phrases: balanced system, bottlenecks, decomposability, operational analysis, performance evaluation, performance modeling, queueing models, queueing networks, response times, saturation.

Operational analysis

- Material drawn from:
- ACM Computing Surveys
  - Special issue on queuing network models
  - Volume 10, Number 3, September 1978
  - P.J. Denning & J. P. Buzen
  - The operational analysis of network models
- (Available from the CS446 home page)

Basic ABC of quantities

- A: Total number of [A]rrivals
- B: Total time system [B]usy
- C: Total number of [C]ompletions
- T: Total time spent monitoring above

Basic derived quantities

- Arrival rate λ
  - A/T (arrivals/second)
- Departure rate X (exit rate)
  - C/T (completions/second)
- Server utilization U
  - B/T (fraction)
- Mean service time S per task
  - B/C (seconds/completion)

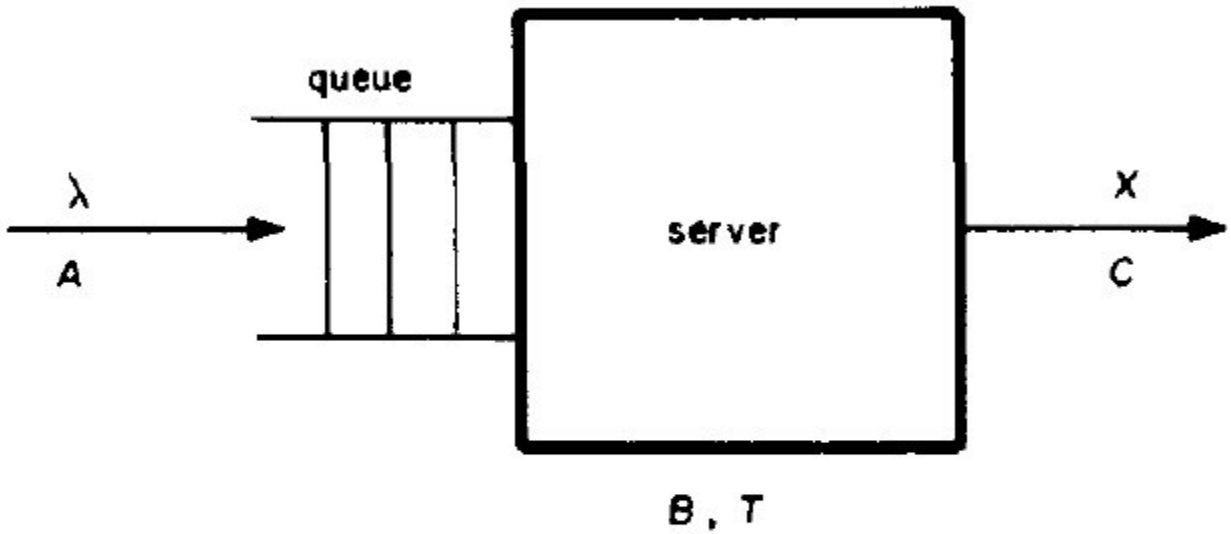


FIGURE 1 Single server queueing system.

Computing Surveys, Vol 10, No 3, September 1978

Utilization law

- Utilization = Completion Rate * Service Time
- U = B/T = (C/T) * (B/C) = XS
  - Example:
  - 3 jobs per second and each job needs 0.1 secs
  - Utilization of system = 3 * 0.1 = 0.3 (30%)
  - I.E. how much work is done per unit of time

Job flow balance assumption

- Total arrivals = Total Completions
- Reasonable since $(A-C)/C \rightarrow 0$ as $T \rightarrow \infty$
- $\lambda = X$
- $U = \lambda S$ (steady state limit theorem)
  - i.e. Can use arrival rate rather than departure rate
  - Have λ or X can assume the other is same

Generalizing to network

- Let there be $n \geq 1$ services (servers)
- Arrival rate $\lambda_i = A_i/T$ (at server i)
- Departure rate $X_i = C_i/T$ (from server i)
- Server utilization $U_i = B_i/T$
- Mean service time $S_i = B_i/C_i$
- Routing frequency (C_{ik} task goes $i \rightarrow k$)
  - $q_{ik} = C_{ik} / C_i$ if $i = 1..n$
  - $q_{0k} = A_{0k} / A_0$ if $i = 0$

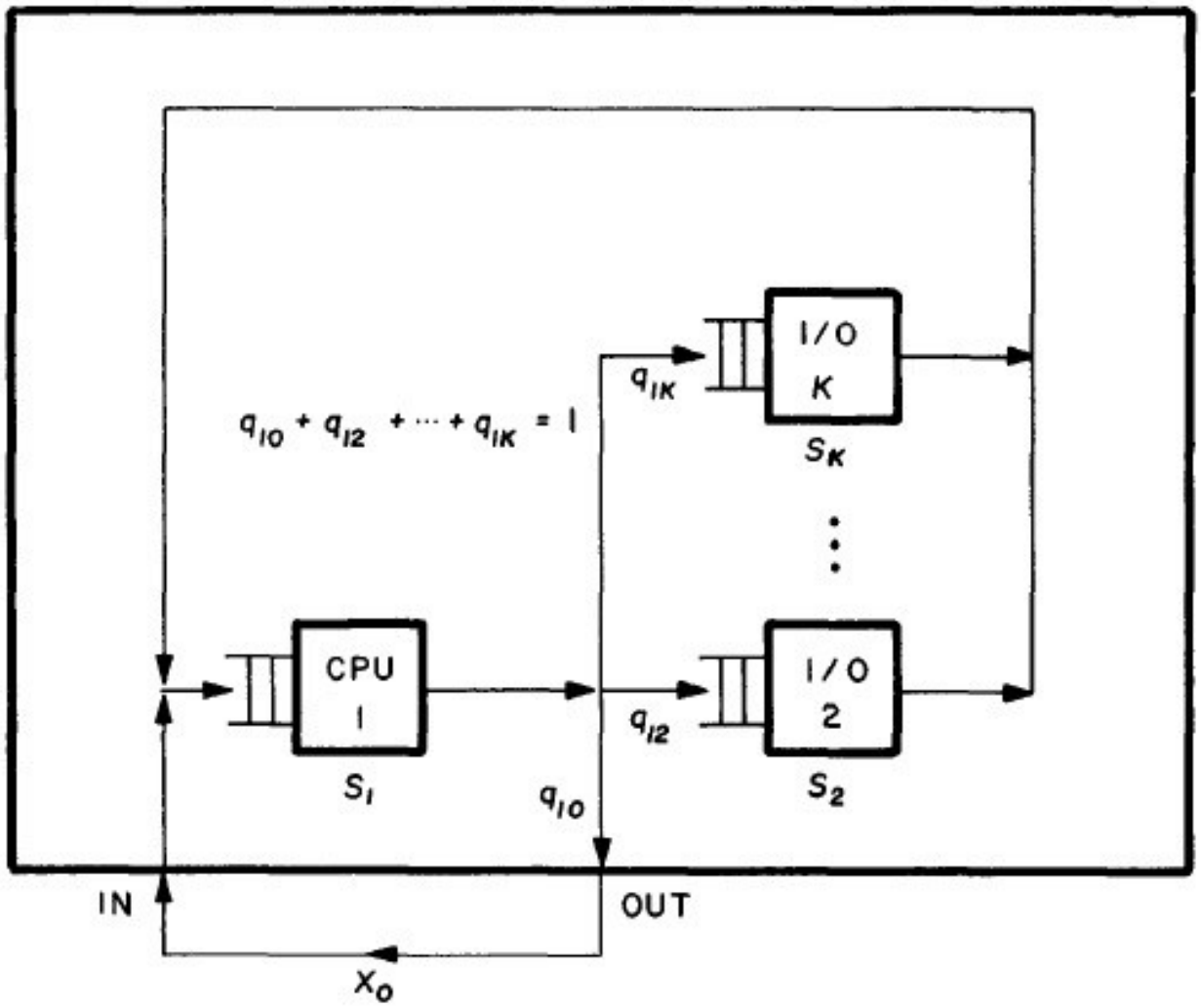


FIGURE 5. Central server network.

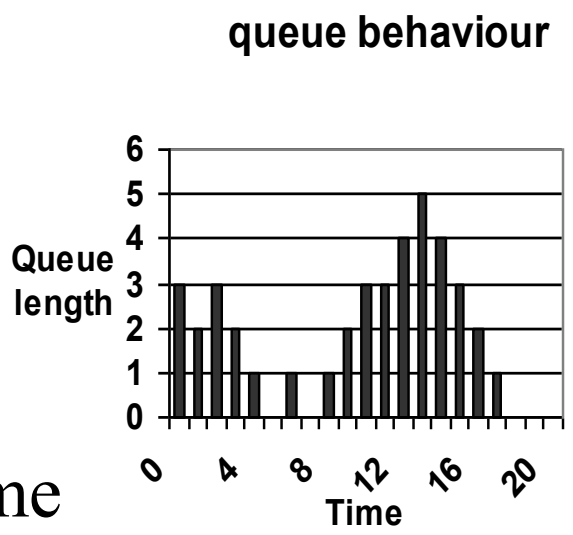
Observations

- $C_i = \Sigma_{k=0..n} C_{ik}$ (i=1..n)
- $C_0 = \Sigma_{k=0..n} C_{k0}$ (completions from system)
- $A_0 = \Sigma_{k=0..n} A_{0k}$ (arrivals at system)
- $X_0 = \Sigma_{k=1..n} X_k q_{k0}$
  - (Output flow law)

- $\Sigma_{k=0..n} q_{ik} = 1$ (for each i)
- Behaves like a probability estimate

Queuing at device

- $W_i = \Sigma_{t=0..n}$ queue lth(t)
  - ie. Area under the graph (total job seconds)
- $Q_i = W_i / T$
  - ie. Average queue length
- $R_i = W_i / C_i$
  - ie. mean waiting time at i
  - Also called the response time
- $Q_i = X_i R_i$ (Little's law)
  - $Q_i = W_i / T = (C_i / T) * (W_i / C_i) = X_i R_i$



Example

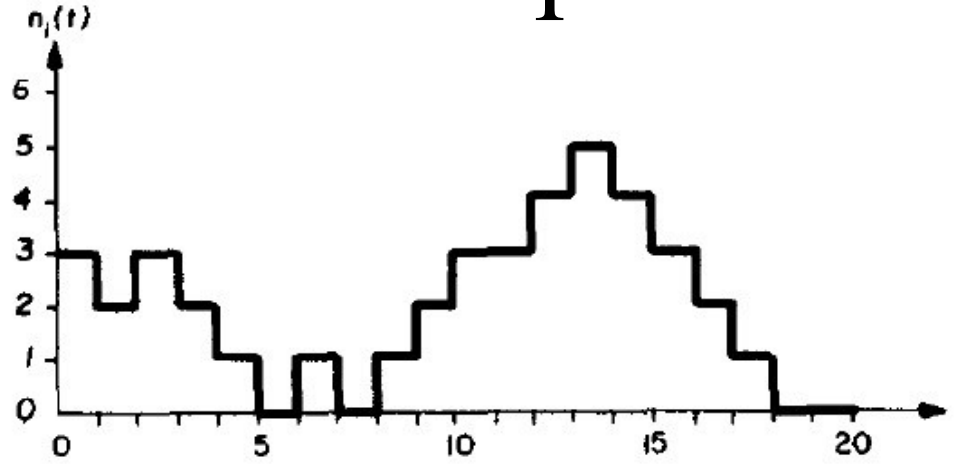


FIGURE 8. Example of a device's operation.

- A=7 jobs; B=16 sec; C=10 jobs (3 at start);T=20 secs
- W=40 job secs; Q = 40/20 = avg 2 jobs (load = 2.0)
- R = W/C = 40 / 10 = 4 secs
- U = B/T = 16 / 20 = 80%

Visit ratios

- Assuming $X_i = \Sigma_{k=0..n} X_k q_{ki}$
  - ie. balanced flow
- $V_i = X_i / X_o = C_i / C_o$
  - (mean requests per task for i)
  - (mean completions at i per task completion)
- $X_i = V_i X_o$ (Forced flow law)
  - Just rearranging above equation

Visit ratio example

- Tasks generate on average 5 disk requests
- Disk throughput is 10 requests/second
- What is system throughput..

Answer

- Tasks generate on average 5 disk requests
- Disk throughput is 10 requests/second
- What is system throughput..
- $X_o = X_i/V_i = 10(\text{request/sec})/5(\text{requests/job})$
- ie. 2 jobs per second
- N.B. Using “measured” disk throughput
- N.B. All other measures irrelevant….

Interactive Response Time

- Suppose M users are using terminals
- Mean wait time per user at terminal R
- Mean think time per user Z
- Mean thinking and waiting time (R + Z)
- $(R + Z)\ X_o = M$ = mean number of users
  - (by Little's law $Q_i = X_i\ R_i$)
- $R = (M/\ X_o - Z)$
  - (Interactive Response Time Formula)

Interactive Response Time

- $R = (\Sigma_{i=1..n} Q_i)/X_o$
  - (eg. Apply Little's law to system)
  - $\Sigma_{i=1..n} Q_i = \Sigma_{i=1..n} W_i / T$ (treat all queues as one)
- but $Q_i / X_o = V_i R_i$ since
  - $Q_i = X_i R_i$ (Little's law)
  - $X_i = V_i X_0$ (Forced flow law)
  - $Q_i = V_i X_0 R_i$
- Therefore $R = \Sigma_{i=1..n} V_i R_i$
  - (General response time law: Total delay is sum of each delay)

Example 1

- Parameters:
  - Each task generates 20 disk request
  - Disk utilization is 50%
  - Mean service time at disk 25 millisecs
  - 25 terminals (think time is 18 seconds)
- What is the terminal response time?

Example 1

- Parameters:
  - Each task generates 20 disk request
  - Disk utilization is 50%
  - Mean service time at disk 25 millisecs
  - 25 terminals (think time is 18 seconds)
- $X_o = X_d/V_d = (U_d/S_d)/V_d = (0.5/0.025)/20$
  - therefore $X_o = 1$ job/second
- $R = (M/X_o) - Z$
  - therefore R = (25/1) - 18 = 7 seconds

Example 2

- Parameters:
  - 40 terminals (think time is 15 seconds)
  - Interactive response time is 5 seconds
  - Disk mean service time is 40 milliseconds
  - Each interactive task generates 10 disk I/O
  - Each batch job generates 5 disk I/O
  - Disk utilization is 90%
- Want to calculate throughput of batch and lower bound on interactive response time if batch throughput is tripled.

Example 2 continued

- Interactive throughput
  - $X_{I,0} = M/(Z+R) = 40/(15 + 5) = 2$ jobs/second
- Disk j throughput {ie. interactive + batch }
  - $X_j = X_{I,j} + X_{B,j} = U_j / S_j = 0.9/0.040 = 22.5$ requests/second {by utilization law}
- $X_{I,j} = V_{I,j}\ X_{I,0} = 10 * 2 = 20$ req/sec
  - {by forced flow law}
- $X_{B,j} = X_j - X_{I,j} = 22.5 - 20 = 2.5$ req/sec
- $X_{B,0} = X_{B,j} / V_{B,j} = 2.5/5 = 0.5$ jobs/sec.

Example 2 continued

- Tripling $X_{B,0}$ = 1.5 batch jobs per second
- Disk j batch throughput
  - $X_{B,j} = V_{B,j} * X_{B,0} = 5 * 1.5 = 7.5$ req/sec
- Maximum completion rate at disk
  - $1/S_i = 25$ requests/second (1 request = 0.04 sec)
  - $X_{I,j} \leq 25 - 7.5 = 17.5$ requests/second
- $X_{I,0} = X_{I,j} / V_{I,j} \leq 17.5/10 = 1.75$ task/sec
- $R_I = (M/X_{I,0}) - Z \geq (40/1.75)-15 = 7.9$sec
  - Assuming M,Z,V_i & S_i unchanged

Example 3

- System A
- 16 terminals
- 25 disk I/O per job
- 80% disk utilization
- service time 0.042sec
- Think time 15 secs

- System B
- 10 terminals
- 16 disk I/O per job
- 40% disk utilization
- service time 0.042sec
- Think time 15 secs

What are the response times for the current systems?
What if A & B's terminals and software are moved to a consolidate system C which uses the same disk drive?

Example 3 throughputs

- Since $U_i = X_i S_i$ & $X_i = V_i X_o$ => $X_o = U_i / V_i S_i$

- $X_{A,0} = .8/(25 * 0.042) \approx 0.77$ jobs/sec
- $X_{B,0} = .4/(16 * 0.042) \approx 0.60$ jobs/sec

- Why is $X_{B,0} < X_{A,0}$ if $V_{B,i} < V_{A,i}$?
- Because utilization lower, but why is this?

Why is utilization 40%

- Is a higher throughput possible?
- Minimum R is $V_i S_i = 16 * 0.042 = .672$ sec
- Max $X_{B,0} = M/(Z+R) = 10/(15+.672)$
  $= 0.638 > 0.6$ jobs per sec
- What is the highest $U_{B,i}$ possible?
- $U_i = X_o V_i S_i = 0.638*16*0.042 = 42.8\%$
- So in essence we can't do enough work at 10 terminals to keep the drive busy, given the 15 second think time.

Example 3 response times

- Since $R = (M/X_0) - Z$
- $R_A = (16/0.77) - 15 = 5.779$ sec
- $R_B = (10/0.60) - 15 = 1.666$ sec
- Fraction of I/O performed by A
  - $X_{A,i}T / (X_{A,i} + X_{B,i})T = X_{A,i} / (X_{A,i} + X_{B,i})$
    $= U_{A,I} / (U_{A,i} + U_{B.i}) = 80/120 = 2/3$
  - since $X_i = U_i/S_i$ & $S_{A,i} = S_{B,i}$

Example 3 consolidate machine

- $U_{A,i} + U_{B,i} = 0.8 + 0.4 = 1.2 = 120\% > 100\%$
- Can assume $U_{C,i} \approx 100\%$
- Therefore $X_{C,i} \approx 1/S_i = 1/.042 = 24$ IO/sec
- $X_{A,i} = 24 * (0.8/(0.8+0.4)) = (2/3) = 16$ IO/sec
- $X_{B,i} = 24 * (0.4/(0.8+0.4)) = (1/3) = 8$ IO/sec
- $X_{A,0} = X_{A,i}/V_{A,i} = 16/25 = .64$ jobs/sec
- $X_{B,0} = X_{B,I}/V_{B,i} = 8/16 = .5$ jobs/secs

Example 3 response times

- Since $R = (M/X_0) - Z$
- $R_A = (16/.64) - 15 = 10$ secs
- $R_B = (10/.5) - 15 = 5$ secs

- N.B. M is not 16+10 because looking at the throughputs $X_{A,0}$ and $X_{B,0}$ separately. We are viewing each subsystem as a closed system.

The useful equations

- Utilization law (Very useful)
  - $U_i = X_i S_i$ { $U_i = \lambda_i S_i$ if assume $\lambda_i = X_i$ }
- Little's law
  - $Q_i = X_i R_i$
- Forced flow law (Very useful)
  - $X_i = V_i X_o$
- General response time law
  - $R = \Sigma_{i=1..n} V_i R_i$
- Interactive response time law (Very useful)
  - $R = (M/X_0) - Z$

Bottleneck analysis

- Let N be the number of tasks/jobs running
- what happens as N increases
- Assuming V_i & S_i remain unchanged then:
  - $X_i/X_k = V_i/V_k$ and $U_i/U_k = V_i S_i/V_k S_k$ unchanged
- Device *i* is saturated if $U_i = 100\%$
- Since U_i/U_k constant as $N \rightarrow \infty$ U_i increase by the same fraction when $N \rightarrow N+1$
- The device with largest U_i (or equivalently the largest $V_i S_i$) will be the first to saturate.

Bottleneck response time

- Response time obviously a minimum when only one task.
- In this case $R_o = \Sigma_i V_i S_i$
- System saturation N^* occurs when N forces some task to be queued at some device b.
- If $U_b=1 \Rightarrow X_b S_b = 1 \Rightarrow X_0 V_b S_b = 1 \Rightarrow$
- $X_0 = 1/V_b S_b$
- $N^* = R_o X_0 = R_o/V_b S_b$
- $M = N^* + Z/ V_b S_b$ will saturate the system.

Markov models

- System viewed as a set of states $S_0 \ldots S_n$, S_{-1}
- S_0 is start state, and S_{-1} final state
- Probability p_{ik} transition between S_i and S_k
- Probabilities constant for lifetime of model
- $\Sigma_{i=-1..n} p_{ik} = 1$
- $p_{-1-1} = 1$ (Once finished keep finishing)

Markov modelling goals

- Determine probability of entering S_i
- Determine expected number of transitions before arriving in state S_i
- Flag certain state transitions as significant
- Counting achieved by multiplying state transition probability by dummy variables

Markov modelling reduction

- For all i,k reduce multiple transitions $S_i \rightarrow S_k$
  - Sum the probabilities of each such transition
- Select S_i : S_i does not move directly to S_i
- Eliminate S_i from the model using:
  - For all h,k : $S_h \rightarrow S_i \rightarrow S_k$ add $S_h \rightarrow S_k$ with $p_{hi} * p_{ik}$
- When all states loop eliminate p_{ii} if $i \neq -1$
  - For all $k \neq i$ $p_{ik} = p_{ik} * 1/(1-G)$ where G generating function describing p_{ii}
- Terminate when have G for $S_0 \rightarrow S_{-1}$

Why 1/(1-G)

- We can go through the loop 0 to ∞ times
- Each time with probability G
- Sum these probabilities
  - $G^0 + G^1 + G^2 + G^3 + G^4 \cdots G^\infty$
  - $1 + G + G^2 + G^3 + G^4 \cdots G^\infty = X$
  - $X * G + 1 = X$
  - $1 = (1 - G) * X$
  - $X = 1/(1 - G)$

Use of generating function

- Set all dummy variables of no interest to 1
- Probability that variable x visited ‘n’ times:
  - Expand G as polynomial in x
  - Coefficient associated with X^n is probability
- Probability at most ‘n’ is sum of first n+1 coefficients etc.

Markov example 1

- Software accesses a disk drive and a tape
  - accesses the disk drive $S_1$with probability 0.2
  - accesses the tape drive $S_2$with probability 0.1
  - Access pattern is independent of history
- Markov model:
  - $S_0 \rightarrow S_1$ $p_{01}=0.2$ $S_0 \rightarrow S_2$ $p_{02}=0.1$ $S_0 \rightarrow S_{-1}$ $p_{0-1} = 0.7$
  - $S_1 \rightarrow S_1$ $p_{11}=0.2$ $S_1 \rightarrow S_2$ $p_{12}=0.1$ $S_1 \rightarrow S_{-1}$ $p_{1-1} = 0.7$
  - $S_2 \rightarrow S_1$ $p_{21}=0.2$ $S_2 \rightarrow S_2$ $p_{22}=0.1$ $S_2 \rightarrow S_{-1}$ $p_{2-1} = 0.7$

Add in dummy variable

- Let d count disk accesses and t tape accesses
- Markov model:
  - $S_0 \rightarrow S_1$ $p_{01}=0.2d$ $S_0 \rightarrow S_2$ $p_{02}=0.1t$ $S_0 \rightarrow S_{-1}$ $p_{0-1} = 0.7$
  - $S_1 \rightarrow S_1$ $p_{11}=0.2d$ $S_1 \rightarrow S_2$ $p_{12}=0.1t$ $S_1 \rightarrow S_{-1}$ $p_{1-1} = 0.7$
  - $S_2 \rightarrow S_1$ $p_{21}=0.2d$ $S_2 \rightarrow S_2$ $p_{22}=0.1t$ $S_2 \rightarrow S_{-1}$ $p_{2-1} = 0.7$
- Eliminating $S_1 \rightarrow S_1$ and $S_2 \rightarrow S_2$:
  - $S_0 \rightarrow S_1$ $p_{01}=0.2d$ $S_0 \rightarrow S_2$ $p_{02}=0.1t$ $S_0 \rightarrow S_{-1}$ $p_{0-1} = 0.7$
  - $S_1 \rightarrow S_2$ $p_{12}=0.1t(1/1-0.2d)$ $S_1 \rightarrow S_{-1}$ $p_{1-1} = 0.7(1/1-0.2d)$
  - $S_2 \rightarrow S_1$ $p_{11}=0.2d(1/1-0.1d)$ $S_2 \rightarrow S_{-1}$ $p_{2-1} = 0.7(1/1-0.1d)$

- Eliminating S_1:
  - $S_0 \rightarrow S_2$ $p_{01} * p_{12} + p_{02} = .2d * .1t(1/1-.2d) + .1t$
  - $S_0 \rightarrow S_{-1}$ $p_{0-1} = 0.2d * 0.7(1/1-0.2d) + 0.7$
  - $S_2 \rightarrow S_2$ $p_{21} * p_{12} = .2d(1/1-.1d) * .1t(1/1-.2d)$
  - $S_2 \rightarrow S_{-1}$ $p_{2-1} = .7(1/1-.1d)$
- Eliminating $S_2 \rightarrow S_2$:
  - $S_2 \rightarrow S_{-1}$ $p_{2-1} = (1/1 -.2d(1/1-.1d) * .1t(1/1-.2d)) * .7(1/1-.1d)$

- Eliminating S_2 computing $S_0 \rightarrow S_{-1}$ i.e. G:
  - $p_{02}\, p_{2\text{-}1} + p_{0\text{-}1} = (.2d*.1t(1/1\text{-}.2d)+.1t) * (1/1 - .2d(1/1\text{-}.1d) * .1t(1/1\text{-}.2d)) *.7(1/1\text{-}.1d) + .7$
  - $S_0 \rightarrow S_2\ p_{02}=.1t\ S_0 \rightarrow S_{-1}\ p_{0\text{-}1} = .7$
  - $S_1 \rightarrow S_2\ p_{12}=.1t(1/1\text{-}.2d)\ S_1 \rightarrow S_{-1}\ p_{1\text{-}1} = .7(1/1\text{-}.2d)$
  - $S_2 \rightarrow S_1\ p_{11}=.2d(1/1\text{-}.1d)\ S_2 \rightarrow S_{-1}\, p_{2\text{-}1} = .7(1/1\text{-}.1d)$

In Practice

- Write a program (or ask me for one)
  - Input
    - state transitions
    - probabilities * dummy
  - Output G
    - In a format usable by Maple (or whatever)
    - Solve questions of interest using Maple etc.

Probability of ‘n’ disk accesses

- t:=1; taylor(G, d);
- .7777777778 + .1728395062 d + .03840877915 d^2 + .008535284255 d^3 + .001896729834 d^4 + .0004214955188 d^5 + $O(d^6)$
- Truncate and set d=1 for ≤ ‘n’ accesses
- Could parameterize 0.2=p and 0.1=q and 0.7=(1-p-q)

Average number of disk accesses

- Let p_k be the probability of exactly k accesses
- $G = \Sigma_{k=0..\infty} p_k * d^k$
- G' = diff(G) w.r.t variable d
- $G' = \Sigma_{k=1..\infty} k * p_k * d^{k-1}$
  - $0*p(miss)*d^0+1*p(once)*d^1+2*p(twice)*d^2\ldots$
  - Then setting d = 1 to remove all d
  - This becomes average number of disk accesses
- G' = 0.2857142857 (Average # disk accesses)

Variance in number disk accesses

- V(accesses) = E(accesses^2) - E^2(accesses)
  - G'' = diff(G')
  - G'' = $\Sigma_{k=2..\infty}$ k * (k-1) * p_k * d^{k-2}
  - = $\Sigma_{k=2..\infty}$ k^2 * p_k * d^{k-2} - $\Sigma_{k=2..\infty}$ k * p_k * d^{k-2}
  - = E(accesses^2) - E(accesses)
  - So variance = (G'' + G') – G'* G'
- Now set d := 1
  - (G'' + G') - G'*G' = 0.3673469388 (Variance)

Detecting uncorrectable error in a Double Linked List

Double-linked list			
This State	Next State	Generating Function	Comments
0	0	$z(1-p)$	$N_{x+1} \cdot b_1 = N_x$
0	1	p	$N_{x+1} \cdot b_1 \neq N_x$
1	1	$z(1-p)$	$N_{x-1} \cdot f_1 = N_x$
1	-1	p	$N_{x-1} \cdot f_1 \neq N_x$

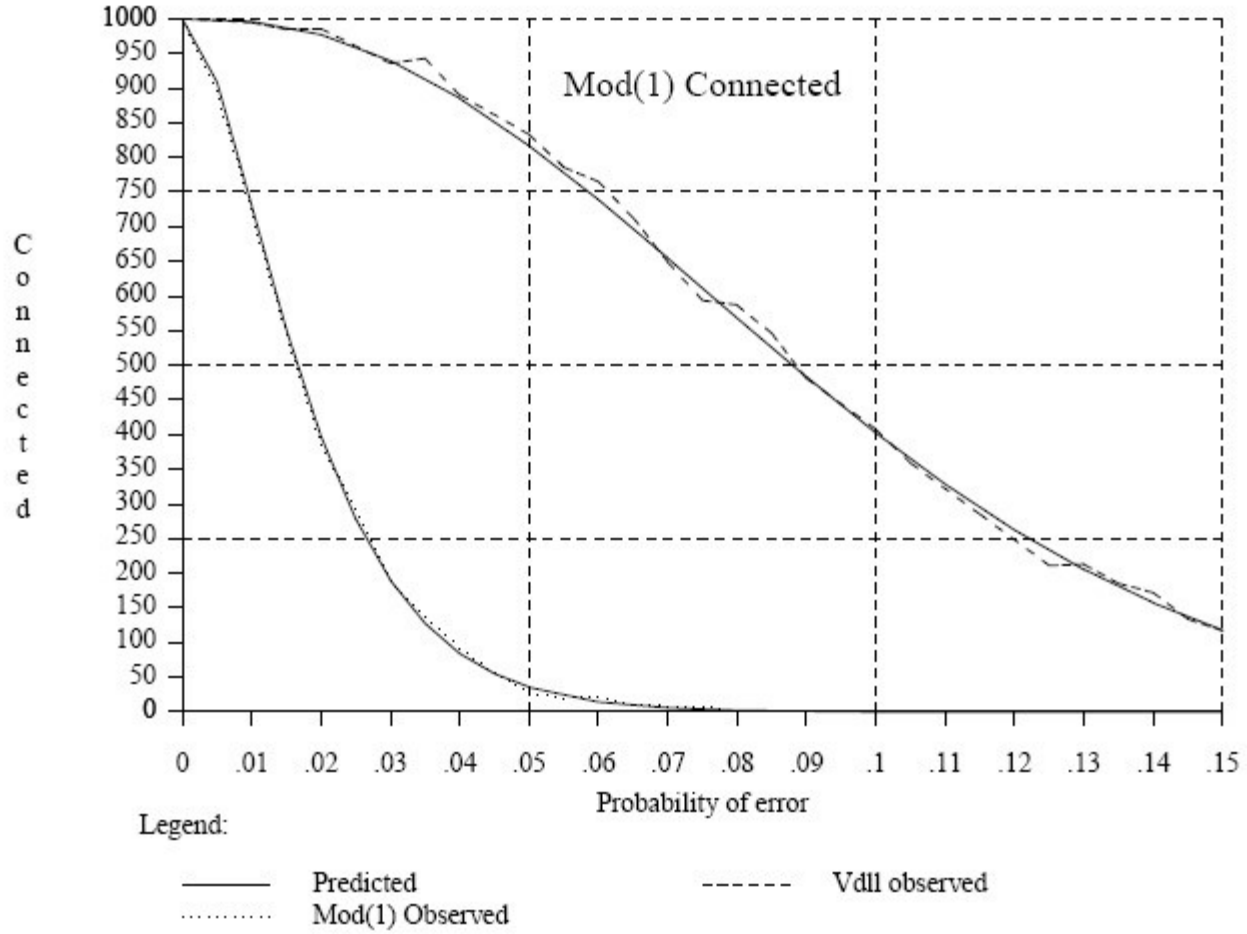
Probability of a pointer in error is p
z is a dummy variable

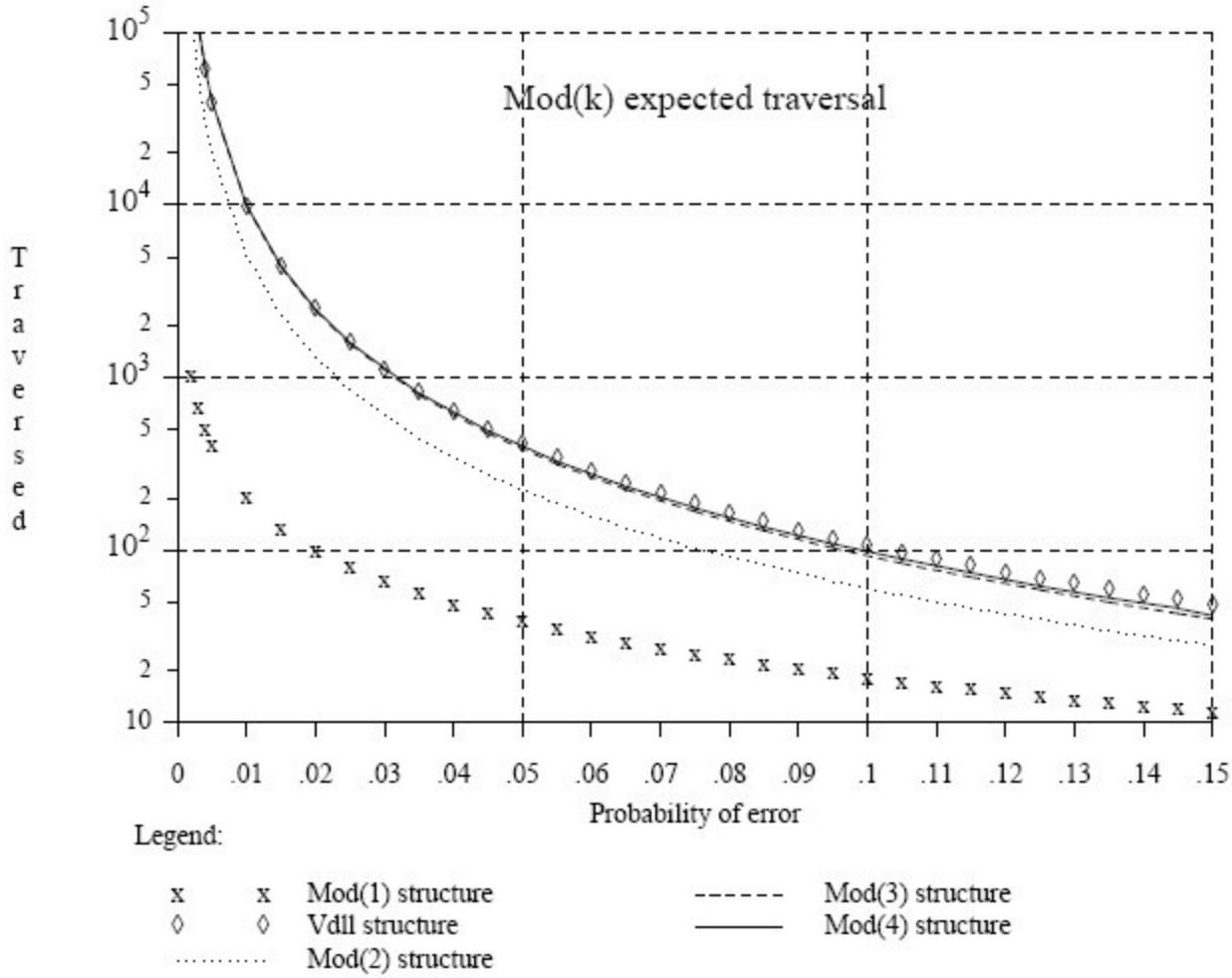
VDDL : XOR encoding strategy

Back pointer in every node $N_x.b \rightarrow N_{x+1} \oplus N_{x-1} = N_x.v$
Permits traversal forwards & backwards using only one pointer
Must always remember last two nodes visited
Forward pointer now provides additional redundancy

Virtual double-linked list (VDLL)			
0	0	$z(1-p)$	$N_{x+1} \cdot v \oplus N_{x+2} = N_x$
0	1	p	$N_{x+1} \cdot v \oplus N_{x+2} \neq N_x$
1	1	$z(1-p)$	$N_{x-1} \cdot f_1 = N_x$
1	2	p	$N_{x-1} \cdot f_1 \neq N_x$
2	1	$z(1-p)$	$N_{x-1} \cdot v \oplus N_{x-2} = N_x$
2	-1	p	$N_{x-1} \cdot v \oplus N_{x-2} \neq N_x$

Results for list of 100 nodes




Mod(k) expected traversal
Traversed
10^5
5
2
10^4
5
2
10^3
5
2
10^2
5
2
10
0
.01
.02
.03
.04
.05
.06
.07
.08
.09
.1
.11
.12
.13
.14
.15
Probability of error
Legend:
Mod(1) structure
Vdll structure
Mod(2) structure
Mod(3) structure
Mod(4) structure