CS 672
Alternative Modeling Approaches

Dr. Daniel A. Menascé

http://www.cs.gmu.edu/faculty/menasce.html

Department of Computer Science

George Mason University

Modeling Techniques

- Analytic
  - Queuing Networks
  - Stochastic Petri Nets
  - Generalized Stochastic Petri Nets
  - Markov Chains
  - Process Algebras
- Simulation
- Hybrid Models: combination of analytic and simulation models.

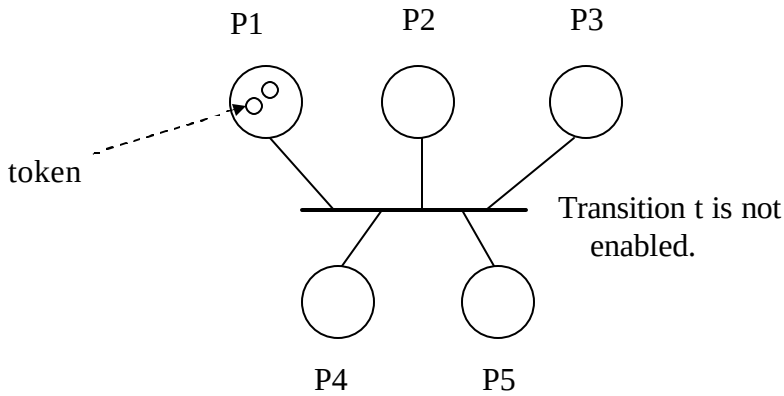
Petri Nets (PNs)

- Queuing Networks (QNs) are not good at expressing nor modeling concurrent events (e.g., fork and join situations, blocking, etc)
- Petri Nets (PNs) are good for representing concurrency but do not lend themselves to performance analysis.
- Adding time to PNs enable them to be used in modeling.

Review of Basic Petri Nets

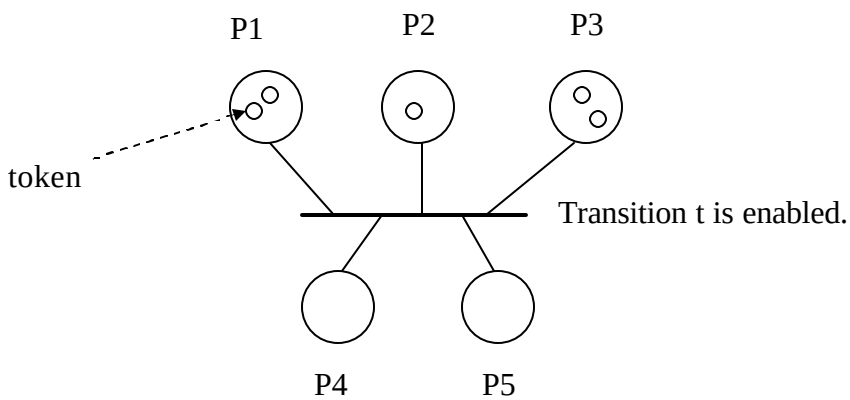
- Directed bi-partite graph with two types of nodes:
  - Places: used to hold tokens.
  - Transitions
- Firing Rule: a transition t is enabled if all input arcs have at least one token. When a transition fires, a token is removed from each input place and one token is placed in each output place.

Review of PNs



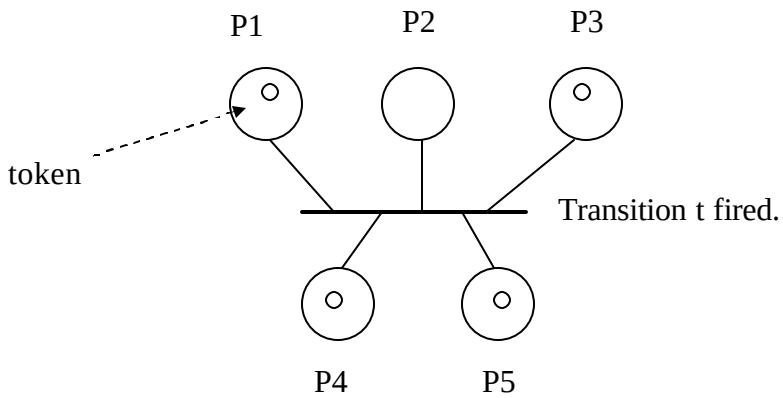
5

Review of PNs

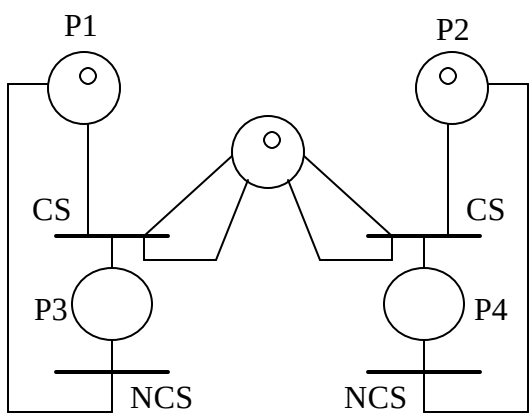


6

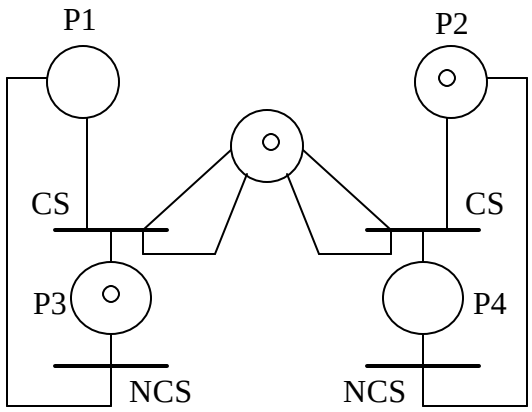
Review of PNs



PN for Mutual Exclusion

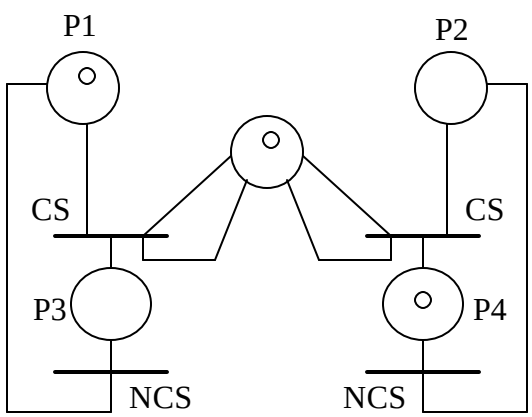


PN for Mutual Exclusion



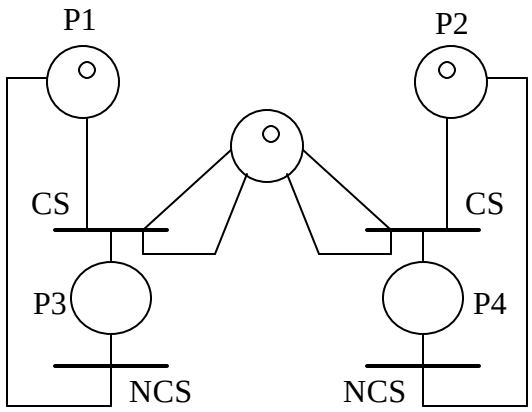
9

PN for Mutual Exclusion



10

PN for Mutual Exclusion



Reachability Set

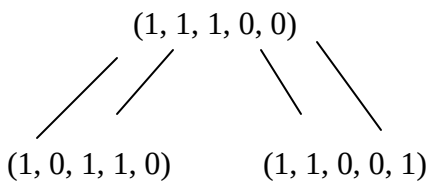
- A marking of a PN is a tuple of the form

$$M = (m_1, ..., m_P)$$

where m_i is the number of tokens in place i.

- R = set of all possible markings
- M_0: initial marking.

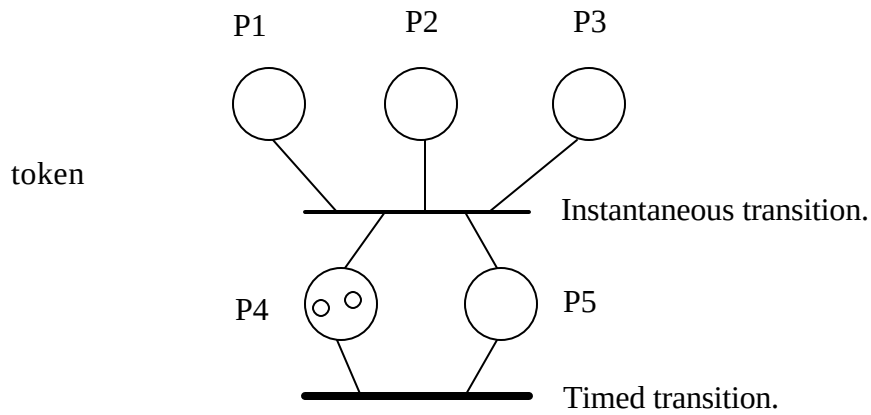
Reachability Set for Mutual Exclusion Example



Adding Time to PNs

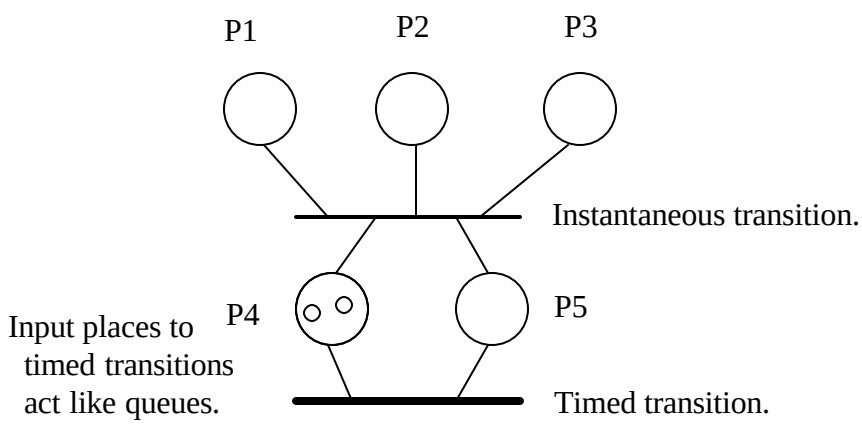
- Transitions correspond to actions and places to conditions.
- Actions do not occur in zero time in real life.
- Make transitions take a time to fire. If firing time is exponentially distributed, then PN becomes a Stochastic Petri Net (SPN).
- If the SPN also allows instantaneous transitions as well as exponential transitions, then it is a Generalized Stochastic Petri Net (GSPN).
- Instantaneous transitions are used to specify control.

GSPN Graphical Notation



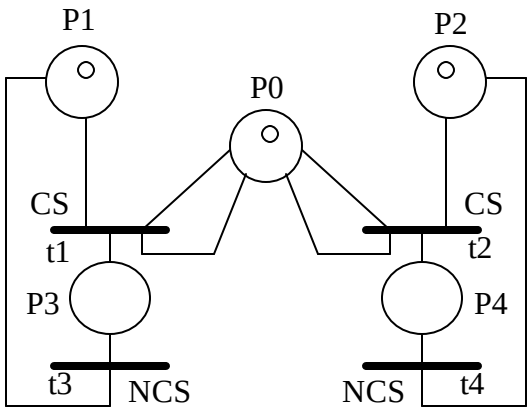
15

GSPN Graphical Notation



16

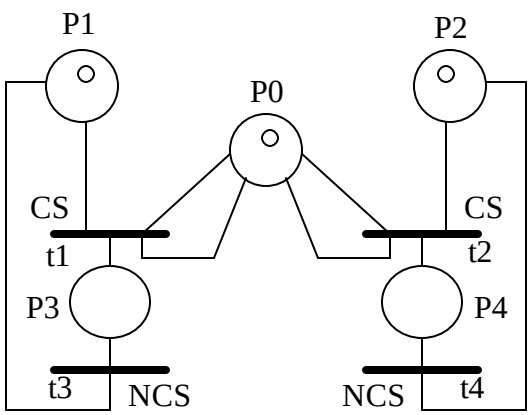
GSPN for Mutual Exclusion



M= (1,1,1,0,0)

17

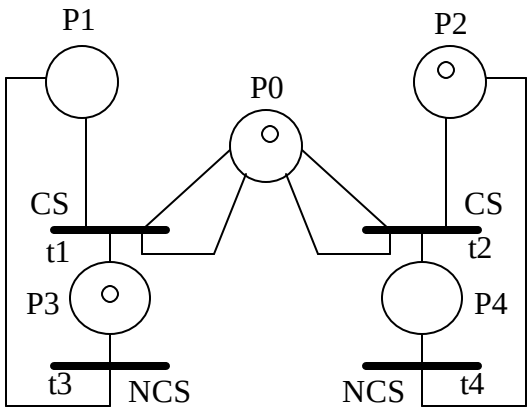
GSPN for Mutual Exclusion



M= (1,1,1,0,0)

18

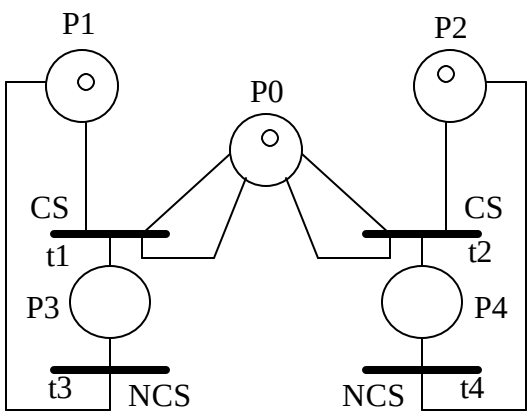
GSPN for Mutual Exclusion



M= (1,0,1,1,0)

19

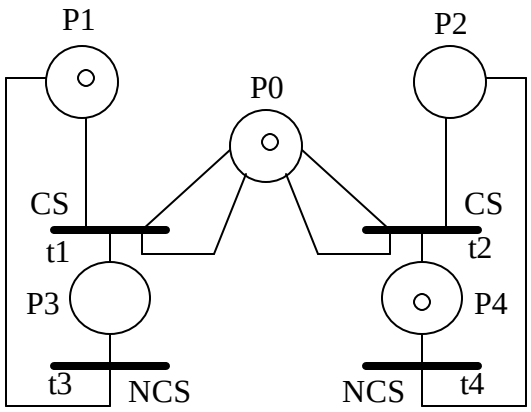
GSPN for Mutual Exclusion



M= (1,1,1,0,0)

20

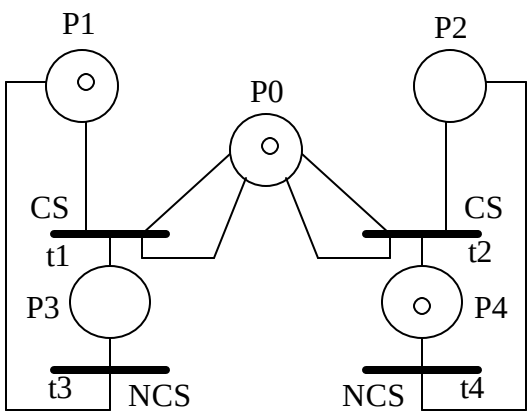
GSPN for Mutual Exclusion



M= (1,1,0,0,1)

21

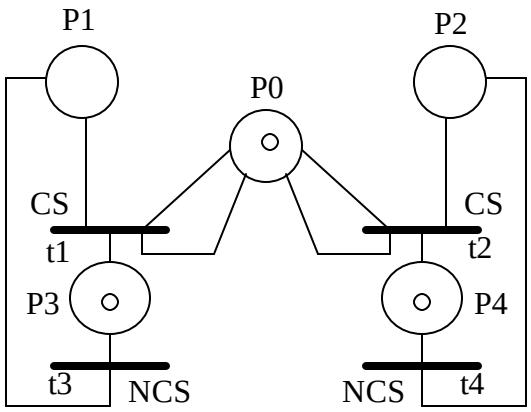
GSPN for Mutual Exclusion



M= (1,1,0,0,1)

22

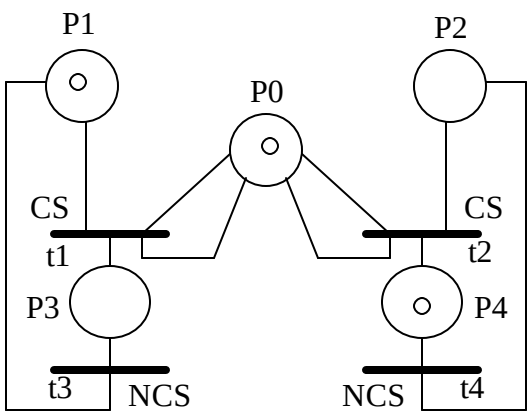
GSPN for Mutual Exclusion



M= (1,0,0,1,1)

23

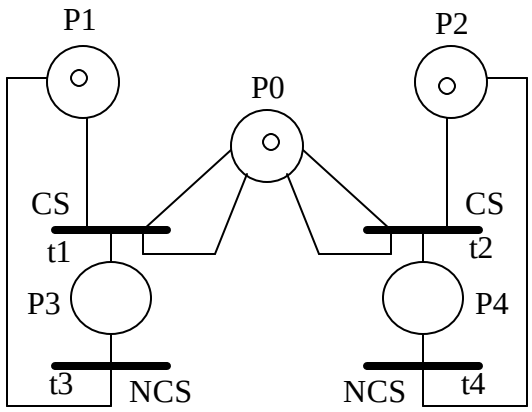
GSPN for Mutual Exclusion



M= (1,1,0,0,1)

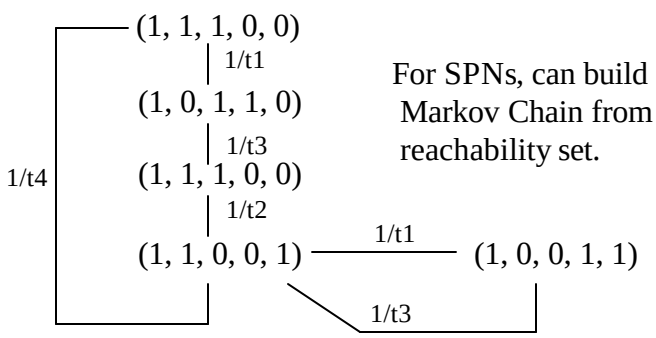
24

GSPN for Mutual Exclusion



M= (1,1,1,0,0)

Reachability Set for Mutual Exclusion Example



For SPNs, can build a Markov Chain from the reachability set.

Computing Performance Metrics from GSPNs

- Packages such as GSPN (http://www.di.unito.it/~greatspn/) allow one to:
  - Specify a GSPN using a graphic editor, language or combination.
  - Solve GSPN to obtain probabilities of each marking.
  - Simulate a GSPN.
  - Metrics of interest are associated with functions of marking probabilities.

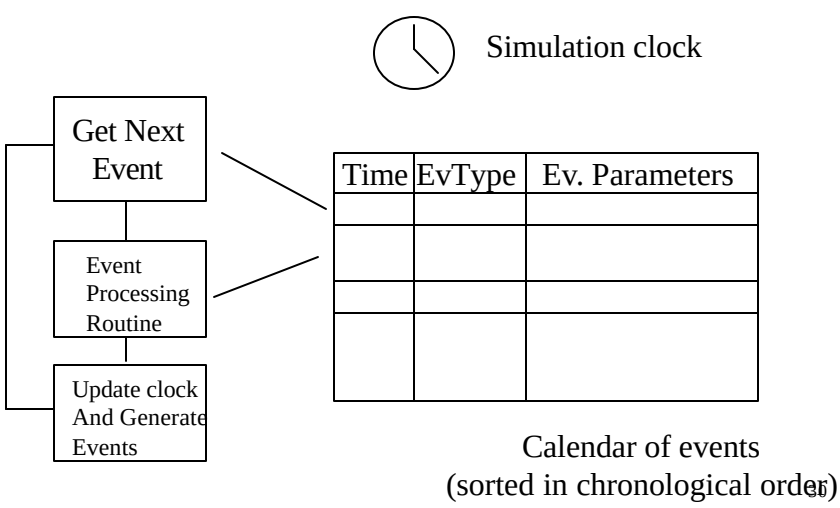
Simulation

- Discrete event simulation:
  - Event generation
  - Calendar of events
  - Event processing procedures
  - Clock (simulated clock)
- Trace-drive simulation: part of the input data comes from traces of execution (e.g., memory references, HTTP logs, etc).

Components of a Simulation Model

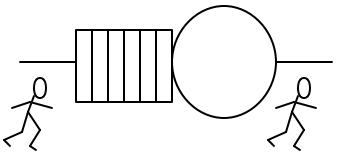
- Event Generation:
  - Trace-driven
  - Distribution-driven
  - Hybrid
- Event Processing
  - Calendar of Events
  - Event-handling procedures
- Transaction List (with parameters)
- Queues
- Simulation Clock
- Computation of Statistics

Simulation Basics



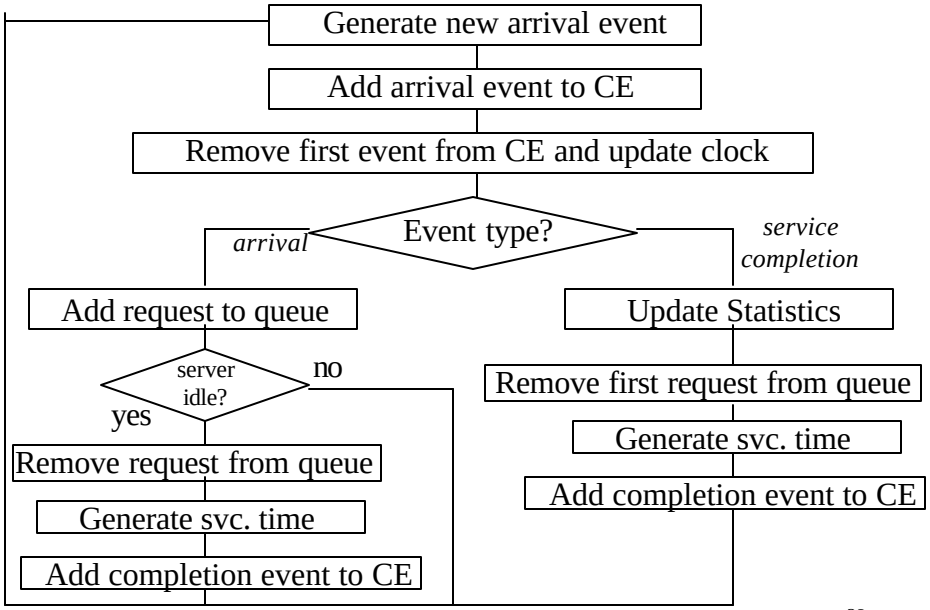
Simulation Model Example: Single Queue

- Events:
  - Arrival of a customer
  - Service completion
- Statistics:
  - Total number of arrivals
  - Total departures
  - Total server busy time
  - Total waiting time
  - Total departures from queue
  - Total squares of waiting time



31

Simulation Example



32

Calendar of Events

Event Type	Event Time	Event Parameters
arrival	10.5	…..
arrival	12.8	…..
completion	13.1	…..
…	…	…

- The calendar of events is ordered in increasing chronological order.
- Parameters may include the transaction Id associated with the event.

33

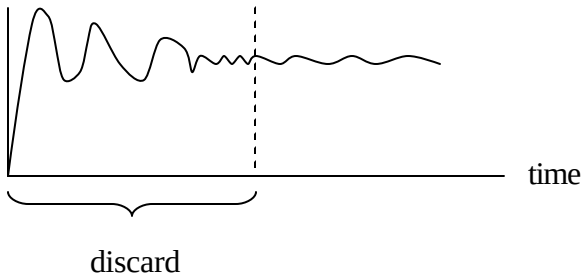
Common Mistakes in Simulation

- Inappropriate level of detail:
  - Too detailed: more development time and higher likelihood of bugs
  - Should start with a less detailed model first and increase complexity as needed.
- Unverified Models:
  - Simulation programs are usually large and complex programs and may have bugs that invalidate the results.
- Invalid Models:
  - Incorrect assumptions may be used. Need to validate through analytic models, measurements, and or intuition.

34

Common Mistakes in Simulation

- Improperly Handled Initial Conditions:
  - Should discard first part of run: transient behavior.



35

Common Mistakes in Simulation

- Improper simulation length.
- Poor Random Number Generator.
- Improper Selection of Seeds.

36

Verifying Simulation Models

- Trace Analysis: examine traces of a few transactions as they go through the system.
- Continuity Test: small variations in the input should show small variations in the output.
- Check Extreme Values: extreme values (e.g., low loads or very high loads) should be easy to verify by crude analytic models.

37

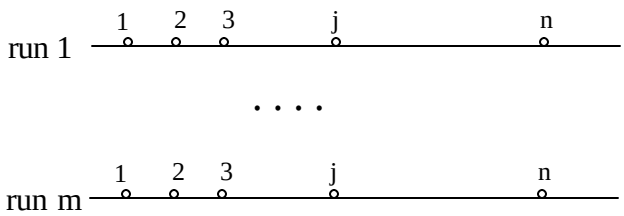
Verifying Simulation Models

- Check for Basic Relationships: verify if results satisfy basic laws (e.g., Little's Law).
- Bound validation: use, if possible, existing analytic models for situations that are known to be upper or lower bounds
- Trend verification: check if the trends shown by the model match your intuition.
- Numeric range validation: check if the numerical results are within expected numerical ranges.

38

Transient Elimination with Independent Runs

- Run *m* runs of the simulation with a different seed for each run.
- Each run has *n* observations.
- Let $x_{i,j}$ be the j-th observation in the i-th run.



39

Transient Elimination with Independent Runs

Step 1: compute average of j-th observation over all runs.

$$\bar{x}_j = \frac{1}{m}\sum_{i=1}^{m} x_{i,j}$$

Step 2: compute the overall average.

$$\bar{\bar{x}} = \frac{1}{n}\sum_{j=1}^{n} \bar{x}_j$$

Step 3: Set the number of deleted observation, k, equal to 1.
Step 4: Compute the overall mean without the first k observations.

$$\bar{\bar{x}}_k = \frac{1}{n-k}\sum_{j=k+1}^{n} \bar{x}_j$$

40

Transient Elimination with Independent Runs

Step 5: compute the relative change Δ

$$\Delta = \frac{\bar{\bar{x}}_k - \bar{\bar{x}}}{\bar{\bar{x}}}$$

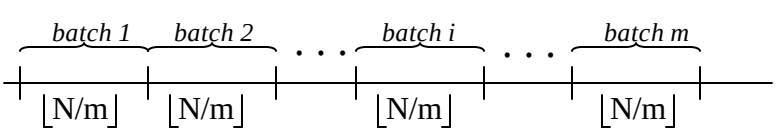
Step 6: If $|\Delta|$ > tolerance then do k $\leftarrow$ k + 1 and go to step 4.

Step 7: Remove the first k observations and use $\bar{\bar{x}}_k$ as the average.

41

Transient Elimination with Batch Means

- Single run with N observations.
- Divide the run into *m* sub-samples called batches of size $n = \lfloor N/m \rfloor$.
- Let $x_{i,j}$ be the j-th observation in the i-th batch.



42

Transient Elimination with Batch Means

Step 1: Set n = 10.

Step 2: compute the average of the i-th batch.

$$\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{i,j}$$

Step 3: compute the overall average.

$$\bar{\bar{x}} = \frac{1}{m}\sum_{i=1}^{m} \bar{x}_i$$

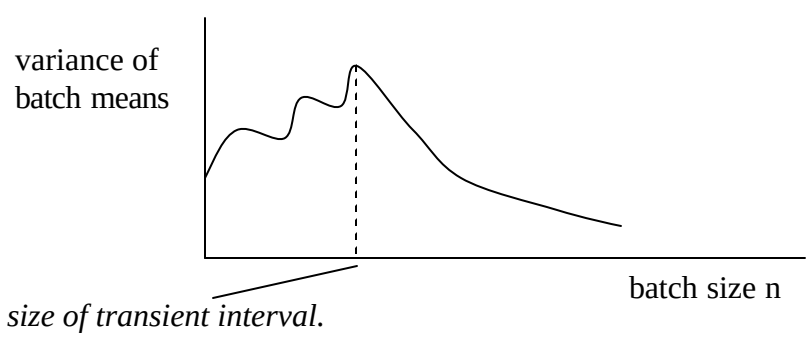
Step 4: Compute the variance of the batch means:

$$Var(\bar{x}) = \frac{1}{m-1}\sum_{i=1}^{m} (\bar{x}_i - \bar{\bar{x}})^2$$

43

Transient Elimination with Batch Means

Step 5: Increase n by 10 and repeat steps 2-4 and plot the variance as a function of n. The point at which the variance starts to decreases is the length of the transient interval.



44

Stopping Criteria
Independent Runs

- Run *m* runs of the simulation with a different seed for each run.
- Each run has $n + n_o$ observations where n_o is the size of the transient phase.
- The number *n* is increased until the precision in the confidence interval reaches a desired value.

45

Stopping Criteria
Independent Runs

Step 0: Initialization: n = 100.

Step 1: compute the mean for each replication.

$$\overline{x}_i = \frac{1}{n} \sum_{j=n_0+1}^{n} x_{i,j}$$

46

Stopping Criteria
Independent Runs

Step 2: compute the overall mean for all replications.

$$\bar{\bar{x}} = \frac{1}{m}\sum_{i=1}^{m}\bar{x}_i$$

Step 3: compute the variance of the replicate means.

$$\mathrm{Var}(\bar{x}) = \frac{1}{m-1}\sum_{i=1}^{m}(\bar{x}_i - \bar{\bar{x}})^2$$

Step 4: compute the confidence interval for the mean as:

$$\bar{\bar{x}} \pm t_{[1-a/2,m-1]}\frac{\sqrt{\mathrm{Var}(\bar{x})}}{\sqrt{m}}$$

47

Stopping Criteria
Independent Runs

Step 5: compute the accuracy r as.

$$r = \frac{\left(t_{[1-a/2,m-1]}\frac{\sqrt{\mathrm{Var}(\bar{x})}}{\sqrt{m}}\right)}{\bar{\bar{x}}}\times 100$$

Step 6: If r > desired value (e.g., 5) then n = n + 100 and go to Step 1, else STOP.

48

Stopping Criteria
Independent Runs

- Number of discarded observations: $m \times n_o$
- To reduce the number of wasted observations use a small value of *m*.

49

Stopping Criteria
Batch Means

- Single run with $N + n_o$ observations where n_o is the size of the transient phase.

Step 0: Start with a small value of n (e.g., 1).
Step 1: compute the mean for each batch.

$$\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{i,j}$$

50

Stopping Criteria
Batch Means

Step 2: compute the overall mean for all batches.

$$\bar{\bar{x}} = \frac{1}{m}\sum_{i=1}^{m}\bar{x}_i$$

Step 3: compute the variance of the batch means.

$$\mathrm{Var}(\bar{x}) = \frac{1}{m-1}\sum_{i=1}^{m}(\bar{x}_i - \bar{\bar{x}})^2$$

Step 4: compute the confidence interval for the mean as:

$$\bar{\bar{x}} \pm t_{[1-a/2;m]}\sqrt{\frac{\mathrm{Var}(\bar{x})}{m}}$$

51

Stopping Criteria
Batch Means

Step 5: compute the auto-covariance

$$\mathrm{Cov}(\bar{x}_i, \bar{x}_{i+1}) = \frac{1}{m-2}\sum_{i=1}^{m-1}(\bar{x}_i - \bar{\bar{x}})(\bar{x}_{i+1} - \bar{\bar{x}})$$

Step 6: Check for proper batch size: If $\mathrm{Cov}(\bar{x}_i, \bar{x}_{i+1}) << \mathrm{Var}(\bar{x})$ then stop. Otherwise, double n and go to step 1.

52

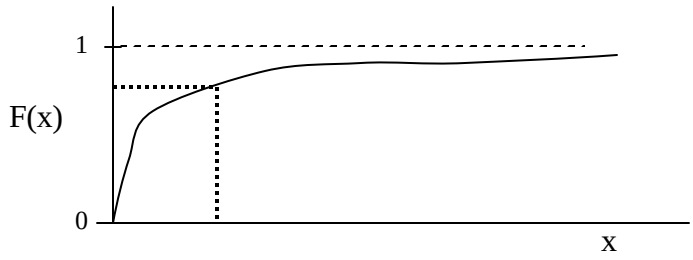
Seed Selection

- Never use zero as a seed.
- Avoid even values.
- Reuse seed for repeatability of experiments.
- Do not use random seeds (e.g., system time) if the simulation is to be repeated.

53

Generation of Random Variables

- Assume that *u* is a value uniformly distributed between 0 and 1.
- Method of the inverse of the CDF:



54

Generation of Random Variables

- Assume that *u* is a value uniformly distributed between 0 and 1.
- CDF for the exponential: $1-e^{-x/a}$
  - Inverse of the CDF: -a ln(u)
- CDF for the Pareto distribution: $1-x^{-a}$
  - Inverse of the CDF: $1/u^{1/a}$

55

Simulation Programs

- Written in general purpose programming languages (e.g., C, C++).
- Written in high-level programming languages with the help of simulation libraries (e.g., SMPL, Simpack, CSim).
- Using special purpose simulation languages (e.g., GPSS/H).
- Using simulation packages (e.g., SES Workbench, OPNET).

56