Probability and Statistics with Reliability, Queuing and Computer Science Applications

Second edition
by K.S. Trivedi
Publisher-John Wiley & Sons

Introduction

Dept. of Electrical & Computer Engineering
Duke University
Email: kst@ee.duke.edu
URL: www.ee.duke.edu/~kst

Copyright © 2006 by K.S. Trivedi
My Research Triangle

Theory
- Modeling methods, solution methods for: CTMC, NHCTMC, SMP/MRGP
- SRN, MRSPN, FSPN, FTREE
- Software Reliability
- Software Rejuvenation

Books
- Fault-tolerant Systems
- Software-based Systems
- Real-time Systems
- Computer Networks, Wireless Networks
  [GE, HP, Ericsson, IBM, Sun, EMC, Cisco, Lucent, Motorola, Boeing, ...]

Tools
- HARP, SAVE, SHARPE
- SPNP
- SREPT

Applications
- Fault-tolerant Systems
- Software-based Systems
- Real-time Systems
- Computer Networks, Wireless Networks
  [GE, HP, Ericsson, IBM, Sun, EMC, Cisco, Lucent, Motorola, Boeing, ...]

Copyright © 2006 by K.S. Trivedi
Textbooks


- Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package, Kluwer, 1996 (Redbook)


- Unless otherwise specified, chapter numbers will refer to bluebook second edition
Logistics of the course

- 10-11 home works, no late home works, most will be paper/pencil, some may involve writing a simulation program, some may involve using SHARPE software package
- 2 exams (midterm and a final)
- Grading: 40% HW, 30% each exam
- We cover chapters 1-5, part of 6 & 7& 9, all of 8, like to cover 10 & 11 but time usually does not permit
Outline

- Reliability, Availability, Security, Performance, Performability, Survivability
- Methods of Evaluation
- Evaluation Vs. Bottleneck Detection Vs. Optimization
- Model construction, parameterization, solution, validation, result interpretation
- An introduction to SHARPE Software Package
Use of SHARPE

- Instructors who choose not to use SHARPE may simply omit the slides pertaining the use of SHARPE for individual examples.
- We will have examples of use of SHARPE GUI as well as SHARPE textual inputs in most of the chapters.
- Instructors can request a copy of SHARPE by first downloading, completing and mailing the agreement from www.ee.duke.edu/~kst.
Program Performance Evaluation

- Worst-case vs. Average case
- Data-structure-oriented (Ch 2,7) vs. Control structure-oriented (Ch 2,3,4,5,7,8)
- Sequential vs. Concurrent
- Centralized vs. Distributed
- Restricted (Structured) vs. unrestricted transfer of control (Ch 7)
- Unlimited (hardware) resources vs. limited resources
- Software architecture: modules, their characteristics (execution time) and interactions (branching, looping)
- Business process flows (similar to programs)
- Measures: completion time & response time (mean, variance & dist.)
- Measurements, Models (simulation vs. analytic), or combination
- Analytic models: combinatorial (directed acyclic task precedence graph), DTMC, SMP, CTMC, SPN, Hierarchical
System Performance Evaluation

- Workload: Traffic arrival process, service time distributions, pattern of resource requests
- Hardware architecture and software architecture
- Resource Contention, Scheduling & Allocation
- Concurrency, Synchronization, Distributed processing
- Timeliness (may have to Meet Deadlines)
- Measures: Throughput, Goodput, loss (blocking) probability, response time or delay (mean, variance & dist (Sec 9.6))
- Low-level (Cache, memory interference: Ch. 7)
- System-level (CPU-I/O, multiprocessing: Ch. 8,9)
- Network-level (protocols, handoff in wireless: Ch. 7,8)
- Measurements, models (simulation or analytic), or combination
- Analytic models: DTMC (Ch 7), CTMC (Ch 8), PFQN (Ch. 9), SPN (Ch 8), Hierarchical (Ch 9), Approximation (Ch 9)
System Performance Evaluation (contd.)

- **Workload:**
  - Single vs. Multiple types of requests (classes, chains)
  - Items needed for each type of request:
    - Traffic arrivals types: one time vs. a stream (Ch. 6,7,8,9)
      Types of traffic stream: Poisson (Bernoulli), General renewal, IPP (IBP), MMPP(MMBP), MAP, BMAP, NHPP, Self-similar
    - Service time distributions: Exponential (geometric), deterministic, uniform, Erlang, Hyperexponential, Hypoexponential, Phase-type, general (with finite mean and variance), Pareto (Ch. 3,4,5,7,8,9,10)
    - Pattern of resource requests: Service time distribution (or the mean) at each resource per visit, branching probabilities- often described as a DTMC (Discrete-Time Markov Chain) and can also be seen as the behavior of an individual program (Ch. 7,8,9).
  - All this information should be collected from actual measurements (if possible) followed by statistical inference (Ch. 10,11).
Software Reliability Evaluation

- Black-box (measurements + statistical inference) vs. Architecture-based approach (Models)

- Black-box approach is called software reliability growth modeling (Ch. 3, 5, 8, 10)

- Black-box approaches treat software as a monolithic whole, considering only its interactions with external environment, without an attempt to model its internal structure

- With growing emphasis on reuse, software development process moves toward component-based software design

- White-box approach may be better to analyze a system with many software components and how they fit together
Software Architecture

- Software behavior describes the manner in which different components interact.
- May include the information about the execution time of each component.
- Control flow graph is used to represent architecture.
- Sequential program architecture is modeled by
  - Discrete Time Markov Chain (DTMC; Ch 7)
  - Continuous Time Markov Chain (CTMC; Ch 8)
  - Semi-Markov process (SMP)
  - Markov Regenerated Process (MRGP)
Failure Behavior of Components and Interfaces

- Failure can happen
  - during the execution of any component or
  - during the transfer of control between components.

- Failure behavior can be specified in terms of
  - Reliability
  - Constant failure rate
  - Time-dependent failure intensity
Motivation for Reliability: Dependence on Computer Systems
Motivation: High Availability

- Scott McNealy, Sun Microsystems Inc.
  - "We're paying people for uptime. The only thing that really matters is uptime, uptime, uptime, uptime and uptime. I want to get it down to a handful of times you might want to bring a Sun computer down in a year. I'm spending all my time with employees to get this design goal”

- Sun Microsystems – SunUP & RASCAL program for high-availability
- Motorola - 5NINES Initiative
- HP, Cisco, Oracle, SAP - 5nines:5minutes Alliance
- IBM – Cornhusker clustering technology for high-availability, eLiza, autonomic computing
- Microsoft – Trustworthy computing initiative
- John Hennessey paper in IEEE Computer raising importance of availability
- Microsoft – Regular full page ad on 99.999% availability in USA Today
- Service availability forum http://www.saforum.org/home
Motivation: High Availability

- Critical reliability applications (>6 NINES)
  (long-life/life-critical/safety-critical)
  - Space missions, aircraft control, defense, nuclear systems, telecommunications for hospitals,

- High availability applications (5 NINES)
  (non-life-critical/non-safety-critical)
  - E-commerce, airline reservation, telecommunication, financial systems

- Standard availability applications (4 NINES)
  (non-critical)
Motivation – High Availability

<table>
<thead>
<tr>
<th>Industry</th>
<th>Business Operation</th>
<th>Industry Cost Range (per hour)</th>
<th>Average Cost Per Hour of Downtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Financial</td>
<td>Brokerage Operations</td>
<td>$5.6M to $7.3M</td>
<td>$6.45M</td>
</tr>
<tr>
<td>Financial</td>
<td>Credit Card/Sales Authorization</td>
<td>$2.2M to $3.1M</td>
<td>$2.6M</td>
</tr>
<tr>
<td>Media</td>
<td>Pay-Per-View</td>
<td>$233K</td>
<td>$150K</td>
</tr>
<tr>
<td>Retail</td>
<td>Home Shopping Network (TV)</td>
<td>$87K to $140K</td>
<td>$113K</td>
</tr>
<tr>
<td>Retail</td>
<td>Home Catalog Sales</td>
<td>$60K to $120K</td>
<td>$90K</td>
</tr>
<tr>
<td>Transportation</td>
<td>Airline Reservations</td>
<td>$67K to $112K</td>
<td>$89.5K</td>
</tr>
<tr>
<td>Media</td>
<td>Tee-ticket Sales</td>
<td>$56K to $82K</td>
<td>$69K</td>
</tr>
<tr>
<td>Transportation</td>
<td>Package Shipping</td>
<td>$24K to $32K</td>
<td>$28K</td>
</tr>
<tr>
<td>Finance</td>
<td>ATM Fees</td>
<td>$12K to $17K</td>
<td>$14.5K</td>
</tr>
</tbody>
</table>
System Reliability/Availability

- Fault load: fault types, fault arrivals, repair/recovery procedures and delay time distributions
- Hardware architecture and software architecture
- Minimum Resource Requirements
- Performance/Reliability interdependence
- Measures: Reliability, Availability, MTTF, Downtime
- Low-level (Physics of failures, chip level)
- System-level (CPU-I/O, multiprocessing: Ch 1,3,4,5,6,8,9)
- Software and Hardware combined together (Ch 8).
- Network-level
- Measurements, models (simulation or analytic) or combination
- Analytic models types: RBD (Ch 1,3,4,5), FTREE (Ch 1,3,4,5), CTMC (Ch 8), SPN (Ch 8), Hierarchical (Ch 8)
Definition of Reliability

Reliability is defined in International Telecommunications Union (ITU-T) recommendations E.800 as follows:

"The ability of an item to perform a required function under given conditions for a given time interval."

In this definition, an item may be a circuit board, a component on a circuit board, a module consisting of several circuit boards, a base transceiver station with several modules, a fiber-optic transport-system, or a mobile switching center (MSC) and all its subtending network elements. The definition includes systems with software also.
Definition of Availability

Availability is closely related to Reliability, and is also defined in ITU-T Recommendation E.800 as follows:

"The ability of an item to be in a state to perform a required function at a given instant of time or at any instant of time within a given time interval, assuming that the external resources, if required, are provided."

An important difference between reliability and availability is that reliability refers to failure-free operation during an interval, while availability refers to failure-free operation at a given instant of time, usually the time when a device or system is first accessed to provide a required function or service.
Need for a new term

- **Reliability** is used in a generic sense as an umbrella term.

- **Reliability** is also used as a precisely defined mathematical function.

- To remove confusion, IFIP WG 10.4 has proposed *Dependability* as an umbrella term and **Reliability** is to be used a well defined mathematical function.
IFIP WG10.4

- **Failure** occurs when the delivered service no longer complies with the desired output.
- **Error** is that part of the system state which is liable to lead to subsequent failure.
- **Fault** is adjudged or hypothesized cause of an error.

Faults are the cause of errors that may lead to failures.
Dependability: Reliability, Availability, Safety, Security

- Redundancy: Hardware (Static, Dynamic), Information, Time, software (design/data diversity)
- Fault Types:
  - Permanent (needs repair or replacement) (Ch. 1, 3, 4, 5, 6, 8, 10)
  - Intermittent (reboot/restart) (Ch. 8)
  - Transient (retry),
  - Design: Mandelbugs, Resource exhaustion related (Ch. 8, 10, 11)
    - Bohrbugs
- Fault Detection, Automated Reconfiguration (Ch. 5, 8)
- Imperfect Coverage (Ch. 5, 8, 10)
- Maintenance: scheduled (Ch. 8), unscheduled (Ch. 6, 8)
Software Fault Classification

- Many software bugs are easy to find and fix during the testing and debugging phase.

**Bohrbugs**

- Other bugs that are hard to find and fix remain in the software during the operational phase.
  - These bugs may never be fixed, but if the operation is retried or the system is rebooted, the bugs may not manifest themselves as failures.
  - Manifestation is non-deterministic and dependent on the software reaching very rare states.

**Mandelbugs**
Software Fault Classification

www.software-rejuvenation.com
Failure Classification

- Failures
  - Omission failures (Send/Receive failures)
    - Crash failures
    - Infinite loop
  - Timing failures
    - Early
    - Late (performance or Dynamic failures)
  - Response failures
    - Value failures
    - State-transition failures
Security

- Security intrusions cause a system to fail.
  - Security Failure
    - Integrity: Destruction/Unauthorized modification of information
    - Confidentiality: Theft of information
    - Availability: e.g., Denial of Services (DoS)
- Comparisons between:
  - Malicious vs. accidental faults
  - Security vs. reliability/availability
  - Intrusion tolerance vs. fault tolerance
- SITAR Project
The Need of Performability Modeling

- New technologies, services & standards need new modeling methodologies
- Pure performance modeling: too optimistic!
  Outage-and-recovery behavior not considered
- Pure dependability modeling: too conservative!
  Different levels of performance not considered
System measures besides performance

- **Reliability**: Probability of system being up throughout an interval without system-level repair.
- **Availability**: Fraction of time that the system is up.
- **Survivability**: System’s ability to operate under abnormal conditions.
- **Performability**: System performance under failure and repair.

R.A.S.-ability concerns grow. High-R.A.S. not only a selling point for equipment vendors and service providers. Regulatory outage report are required by FCC for public switched telephone networks (PSTN), and may soon apply to wireless.
Evaluation vs. Optimization

- Evaluation of system for desired measures given a set of parameters

- Sensitivity Analysis
  - Parametric (Blake et al. Sigmetrics 1988)
  - Bottleneck analysis
  - Reliability importance (Fricks, RAMS 2003)

- Optimization (Ch. 11 in 1st ed. white book)
  - Static: Linear, nonlinear, geometric, integer, multi-objective; constrained or unconstrained
  - Dynamic: Dynamic programming, Markov decision process, semi-Markov decision process
PURPOSE OF EVALUATION

- Understanding a system
  - Observation
    - Operational environment
    - Controlled environment
  - Reasoning
    - A model is a convenient abstraction
- Predicting behavior of a system
  - Need a model
  - Accuracy based on degree of extrapolation
PURPOSE OF EVALUATION (Contd.)

These famous quotes bring out the difficulty of prediction based on models:

- “All Models are Wrong; Some Models are Useful”
  
  *George Box* and *Albert Einstein*

- “Prediction is fine as long as it is not about the future”
  
  *Mark Twain*
Basic Definitions

- Reliability $R(t)$:
  
  $X$: Time to failure of a system
  $F(t)$: distribution function of system lifetime

  $$R(t) = P(X > t) = 1 - F(t)$$

- Mean Time To system Failure:

  $$MTTF = E[X] = \int_{0}^{\infty} tf(t)dt = \int_{0}^{\infty} R(t)dt$$

  $f(t)$: density function of system lifetime
Basic Definitions (Contd.)

Availability

- Instantaneous (point) Availability $A(t)$:

$A(t) = P \text{ (system working at } t)\)$

Let $H(t)$ be the convolution of $F$ and $G$:

- $g(t)$: density function of system repair time

$$H(t) = \int_0^t F(t-x)g(x)\,dx$$

Then:

$$A(t) = R(t) + \int_0^t A(t-x)dH(x)$$

Inst. Availability, $A(t) \geq R(t)$, Reliability
Availability

- System working at time $t$

  - Never failed in $(0,t)$, prob: $R(t)$
  - First failed and got repaired at time $x< t$ & UP at end of interval $(x,t)$, prob:

\[
\int_0^t A(t-x) dH(x)
\]

- First repair completed here
Availability (Contd.)

- MTTR: Mean Time to Repair
- \( Y \): repair period of the system

\[
MTTR = E[Y] = \int_{0}^{\infty} tg(t)dt
\]

- Availability and Reliability are related but different!
We can show from equation (1) that steady state Availability is:

\[ A_{ss} = \frac{MTTF}{MTTF + MTTR} \]

Also: \( downtime = (1 - A_{ss}) \times 8760 \times 60 \) (in minutes per year)
Availability (Contd.)

- **Steady-State Availability:**
  \[ A_{SS} = \lim_{t \to \infty} A(t) \]

- There are two kinds of Availabilities!
  - Instantaneous & Steady-state

- For a system with redundancy
  \[ A_{SS} = \frac{MTTF_{eq}}{MTTF_{eq} + MTTR_{eq}} \]

  where MTTF\(_{eq}\) & MTTR\(_{eq}\) must be carefully defined; they can be computed using SHARPE.
MEASURES TO BE EVALUATED

- **Dependability**
  - Reliability: $R(t)$, System MTTF (Ch. 3,4,5,8,10)
  - Availability: Steady-state, Transient (Ch. 6,8,10)
  - Downtime (Ch. 6,8)
  - Safety, security (Ch. 8)
  
  “Does it work, and for how long?”

- **Performance (Ch. 7,8,9,10,11)**
  - Throughput, Blocking Probability, Response Time
  
  “Given that it works, how well does it work?”
MEASURES TO BE EVALUATED (Contd.)

- Composite Performance and Dependability (Ch. 8,9) or Performability
  “How much work will be done(lost) in a given interval including the effects of failure/repair/contention?”

- Survivability can be seen as an aspect of composite performance and availability

- Need Techniques and Tools that can evaluate
  - Performance, Dependability and their combinations
Methods of EVALUATION

- Measurement-Based
  - More Accurate, most expensive
  - Not always possible or cost effective during system design.
  - Statistical techniques are very important here

- Model-Based
Methods of EVALUATION (Contd.)

- Model-Based

  Less Accurate, Less expensive

1. Discrete-Event Simulation vs. Analytic
2. State-Space Methods (Ch. 7,8) vs. Non-State-Space Methods (Ch. 1-5,9)
3. Hybrid: Simulation + Analytic (SPNP)
4. State Space + Non-State Space (SHARPE)
Methods of EVALUATION (Contd.)

- Measurements + Models

  Vaidyanathan et al IEEE TDSC, 2005

  Hsueh et al IEEE TC April 1988
QUANTITATIVE EVALUATION TAXONOMY

- quantitative evaluation
  - measurement
  - abstract modeling

- discrete-event simulation
  - hybrid
  - analytic

Closed-form solution

Numerical solution using a tool
Both measurements & simulations imply statistical analysis of outputs
- Design of experiments
- Hypothesis testing (Ch 10)
- Statistical inference (Ch 10)
- Analysis of variance (Ch 11)
- Regression (linear, nonlinear) (Ch 11)

Distribution driven simulation requires generation of random deviates (variates). (Ch. 3, 4, 5)

Probability and Statistics are different but highly intertwined.

Probability models need inputs that generally come from measurement data (followed by statistical inference)

Statistics in turn uses probability theory to derive formulas.
MODELING TAXONOMY

abstract modeling

discrete-event simulation

hybrid

analytic

non-state space methods

state space methods
ANALYTIC MODELING TAXONOMY

NON-STATE SPACE MODELING TECHNIQUES

non-state space methods

performance models

Product form queuing models, Ch. 9
Task precedence graphs, Ch 2,3,8
SP reliability block diagrams, Ch 1,3,4,5
Non-SP reliability block diagrams, Ch 1
Fault trees w/o rep. events, Ch 1
Fault trees with rep. events, Ch 1

dependability models
State Space Modeling Taxonomy

Markovian modeling
- Discrete-time Markov chains, Ch. 7
- Continuous-time Markov chains, Ch. 8
- Markov reward models, Ch. 8
- Semi-Markov models, Ch. 8
- Markov regenerative models
- Non-Homogeneous Markov, Ch. 8

Non-Markovian modeling

State space methods
Modeling Steps (Sec. 2.2 White book)

- Model construction
  - Use of Stochastic Petri nets and model hierarchies
- Model parameterization
  - Measurements + statistical inference
- Model solution
  - Use of well known packages
- Result interpretation
- Model Validation
Measurements supply input parameters to Models (Model Calibration or Parameterization)
Confidence Intervals should be obtained (Ch. 10).
Model Sensitivity Analysis can suggest which parameters to measure more accurately: Ref. Blake, Reibman and Trivedi: SIGMETRICS 1988.
Model structure also derived from measurement data. Some references--

- Hsueh, Iyer and Trivedi; IEEE TC, April 1988
- Gokhale et al, IPDS 98;
- Vaidyanathan et al, IEEE TDSC, 2005
MODEL VALIDATION

- Model Validation
  1. Face Validation
  2. Input-Output Validation
  3. Validation of Model Assumptions
     (Hypothesis Testing)
- Rejection of a hypothesis regarding model assumption, based on measurement data leads to more accurate model.
MODELING THROUGHOUT SYSTEM LIFE CYCLE

- System Specification/Design Phase

Answer "What-if Questions"

- Compare design alternatives (Blake & Trivedi, IEEE TC Nov 1989)
- Performance-Dependability Tradeoffs (Ex. 8.32, p. 486)
- Design Optimization (optimizing the number of guard channels, Haring et al, IEEE VT, 2001)
MODELING THROUGHOUT SYSTEM LIFECYCLE (Contd.)

- Design Verification Phase
  
  Use Measurements + Models e.g. Fault/Injection + Availability Model
  
  - Union Switch and Signals, Boeing, Draper

- Configuration Selection Phase: DEC, HP

- System Operational Phase
  
  - Workload based adaptive rejuvenation

- It is fun!
MODELER'S DI LEMMA

Should I Use Discrete-Event Simulation?

- Point estimates and Confidence intervals.
- How many simulation runs are sufficient?
- What Specification Language to use?
  - C, SIMULA, SIMSCRIPT, MODSIM, GPSS, RESQ, SPNP v6, Bones, SES workbench, ns, opnet, Blocksim
MODELER'S DI LEMMA  (Contd.)

- Simulation (Pros and Cons):
  - Detailed System Behavior including non-exponential behavior.
  - Performance, Dependability and Performability Modeling Possible.
- Long Execution Time (Variance Reduction Possible)
  - Importance Sampling, importance splitting, regenerative simulation.
  - Parallel and Distributed Simulation.
- Many users in practice do not realize the need to calculate confidence intervals.
Should I Use Non-State-Space Methods?

- Also called Combinatorial Models.
- Model Solved Without Generating State Space
- Use: Order Statistics, Mixing, Convolution (Chapters 1-5)
- Common Dependability Model Types:
  - Series-Parallel Reliability Block Diagrams
  - Non-Series-Parallel Block Diagrams (or Reliability Graphs).
  - Fault-Trees Without Repeated Events
  - Fault-Trees With Repeated Events
Combinatorial analytic models

- Reliability block diagrams, Fault trees and Reliability graphs-
  - Commonly used for Reliability and Availability
  - These model types are similar in that they capture conditions that make a system fail in terms of the structural relationships between the system components.
Combinatorial (Non-state-space) Models

- Combinatorial modeling techniques like RBDs and FTs are easy to use and assuming statistical independence solve for system reliability/availability and system MTTF.

- Each component may have attached to it
  - A probability of failure
  - A failure rate
  - A distribution of time to failure
  - Steady-state and instantaneous unavailability
Non-State Space Modeling Techniques

- Following are possible to compute given component failure/repair rates:
  - System Reliability
  - System Availability (Steady-state, instantaneous)
  - Downtime
  - System MTTF
Non-State Space Modeling Techniques (Contd.)

Assuming:

- Failures are statistically independent
- As many repair units are assumed available, as needed.
- Relatively good algorithms are available for solving such models so that large no. of component systems can be handled.
Non-State Space Modeling Techniques (Contd.)

- Common Model Types: Performance analysis
  - Series-Parallel Task Precedence Graphs
  - Product-Form Queuing Networks
+ Easy specification, fast computation, no distributional assumption.
+ Can easily solve models with 100’s of components.
Combinatorial Modeling (Contd.)

- Combinatorial dependability models can be solved using fast algorithms assuming stochastic independence between system components.
  - Sum of Disjoint Products (SDP) algorithms.
  - Binary Decision Diagrams (BDD) algorithms.
  - Factoring (conditioning) algorithms.
  - Series-parallel composition algorithms.

- Failure/Repair Dependencies are often present; RBDs, FTREEs cannot easily handle these (e.g., shared repair, warm/cold spares, imperfect coverage, non-zero switching time, travel time of repair person, reliability with repair).
Markov chain

- To model more complicated interactions between components, use other kinds of models like Markov chains or more generally state space models.
- Many examples of dependencies among system components have been observed in practice and captured by Markov models.
State-Space-Based Models

- States and labeled state transitions.
  - State can keep track of:
    - Number of functioning resources of each type.
    - States of recovery for each failed resource.
    - Number of tasks of each type waiting at each resource.
    - Allocation of resources to tasks.
  - A transition:
    - Can occur from any state to any other state.
    - Can represent a simple or a compound event.
State-Space-Based Models (Contd.)

- Transitions between states represent the change of the system state due to the occurrence of an event.
- Drawn as a directed graph
- Transition label:
  - Probability: homogeneous Discrete-Time Markov chain (DTMC) (Ch. 7)
  - Rate: homogeneous Continuous-Time Markov chain (CTMC) (Ch. 8)
  - Time-dependent rate: non-homogeneous CTMC (Ch. 8)
  - Distribution function: Semi-Markov process (SMP) (Ch. 8)
  - Two distribution functions; Markov Regenerative Process (MRGP)
MODELER’S DILEMMA (Contd.)

Should I Use Markov Models?

State-Space-Based Methods

+ Model Fault-Tolerance and Recovery/Repair (Sec. 8.3, 8.4.1, 8.5)
+ Model Dependencies (Sec. 8.4.1)
+ Model Contention for Resources (Sec. 8.2, Sec. 8.4.2, Ch. 9)
+ Model Concurrency and Timeliness (Ch. 8)
+ Generalize to Markov Reward Models for Modeling Degradable Performance (Sec. 8.4.3).
MODELER'S DILEMMA (Contd.)

+ Generalize to Markov Regenerative Models for allowing Generally Distributed event times.
+ Generalize to Non-Homogeneous Markov Chains for allowing Weibull failure distributions (Ch. 8).
+ Performance, Availability and Performability Modeling Possible (Sec. 8.4).
- Large State Space (exponential in number of components.)
IN ORDER TO FULFILL OUR GOALS

- Modeling Performance, Availability and Performability
- For Modeling Complex Systems
  
  We Need
- Automated Generation and Solution of Large Markov Reward Models (Sec. 8.7)
IN ORDER TO FULFILL OUR GOALS (Contd.)

- Facility for State Truncation, Hierarchical composition of Non-State-Space and State-Space Models, Fixed-Point Iteration

- There are Two tools that potentially meet these goals-
  - Stochastic Petri Net Package (SPNP)
  - Symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE)
Model-based Performance/Dependability evaluation

Choice of the model type is dictated by:

- Measures of interest.
- Level of detailed system behavior to be represented.
- Ease of model specification and solution.
- Representation power of the model type.
- Access to suitable tools/toolkits.
Difficulty in Modeling using Markov chains

- Markov chains tend to be large and complex

  leading to:

  - Model generation problem

- Use automated means of generating Markov chains:
  Stochastic Petri Nets, Stochastic Reward Nets (Sec. 8.7)
Difficulty in Modeling using Markov chains (Contd.)

- Model solution problem (Sec. 8.6)
  - Use sparse storage for the matrices
  - Use “sparsity preserving” solution methods
    - Successive Overrelaxation,
    - Gauss-Seidel,
    - Uniformization,
    - ODE-solution methods.
Markov Reward Models (MRMs)

- Modeling any system with a pure reliability / availability model can lead to incomplete, or, at least, less precise results.

- Gracefully degrading systems may be able to survive the failure of one or more of their active components and continue to provide service at a reduced level.

- Markov Reward Models are commonly used for the modeling of gracefully degradable system.
State-Space-Based Models

- Use also the following model types:
  - Markov chains & Markov Reward models.
  - Semi-Markov & Markov Regenerative processes.
  - Stochastic Reward nets or Generalized Stochastic Petri nets.
    - SRN & GSPN models are transformed into Markov chains for analysis.
    - Only model types (in SHARPE) that requires a conversion to a different model (Markov chain) for solving.
Summary - Modeling Techniques

- Combinatorial techniques like RBDs and FTREEs are easy to use and solve.
- Combinatorial models cannot easily represent intricate dependencies.
- State space based models like Markov chains can handle dependencies.
- State space explosion problem
  - Use automated generation methods: Stochastic Petri Nets
  - Concurrency, contention and conditional branching easily modeled with Petri nets.
Hierarchy used

- State space explosion can be handled in two ways:
  - Large model tolerance must apply to specification, storage and solution of the model. If the storage and solution problems can be solved, the specification problem can be solved by using more concise (and smaller) model specifications that can be automatically transformed into Markov models.
  - Large models can be avoided by using hierarchical (Multilevel) model composition.
LARGENESS AVOID DANCE

- Non-State-Space methods
  - Reliability block diagrams
  - Fault-trees
  - Product-Form Queuing Networks

- Approximate solutions
  - State Truncation

SAVE, SPNP, ASSIST (Kantz and Trivedi: PNPM91)
Part II
An Introduction to SHARPE software tool
Overview of SHARPE

- **SHARPE**: Symbolic-Hierarchical Automated Reliability and Performance Evaluator
- **Well-known** modeling tool (Installed at over 350 Sites; companies and universities)
- Combines **flexibility** of Markov models and **efficiency** of combinatorial models
- Ported to most architectures and operating systems
- Used for Education, Research and Engineering Practice
Overview of SHARPE (contd.)

- Graphical User Interface is available.
- Used for analysis of performance (traffic), dependability and performability.
- **Hierarchy** facilitates largeness & stiffness avoidance.
- Steady-state as well as transient analysis possible.
- Written in C language.
- Used as an engine by several other tools.
SHARPE - new features

- Many more built-in distributions.
- If and loop statements in model evaluations and function definitions
- Ability to easily specify structured Markov chains (Loop feature).
- Ability to print models and outputs.
New Features

- Equivalent Mean Time to System Failure (MTTF) and equivalent Mean Time to System Repair (MTTR) implemented for Markov chains and RBDs.
- BDD algorithms implemented for FTs and RGs.
- Steady-state computation of MRGP models.
- Stochastic Reward net is available as a model type.
- Fast MTTF algorithm implemented for Markov chain.
Architecture of SHARPE interface

Reliability Block Diagrams ➔ Fault tree ➔ MRGP/SMP ➔ Markov Chain and MRM

Hierarchical & Hybrid Compositions ➔ Reliability graph

Task graph ➔ PFQN, MPFQN

Reliability/Availability ➔ Performance ➔ Performability

Copyright © 2006 by K.S. Trivedi
SHARPE MENU OF MODEL TYPES

- Availability/Reliability:
  - Series-Parallel Reliability Block Diagram (block)
  - Fault Trees (ftree)
  - Reliability Graphs (relgraph)
SHARPE MENU OF MODEL TYPES
(contd.)

- Performance (traffic modeling):
  - Product-Form Queuing Networks (Pfqn, Mpfqn)
  - Series-Parallel Task Graphs (graph)
SHARPE MENU OF MODEL TYPES (contd.)

Both Availability and Performance

- *Markov Chains (markov)*
- *Semi-Markov Chains (semimark)*
- *Markov regenerative process (mrgp)*

*Reward Models*

- *Generalized Stochastic Petri Nets (gspn) and stochastic reward nets (srn)*

*Hierarchical & Hybrid Compositions of Above*

- Many solution algorithms for each model type; these algorithms are continually improving.
### Architecture of SHARPE

<table>
<thead>
<tr>
<th>Method</th>
<th>Reliability/Availability</th>
<th>Performance</th>
<th>Performability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault tree</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multistate fault tree</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reliability block diagram</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reliability graph</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phased-mission systems</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Markov chain</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semi-Markov chain</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GSPN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stochastic reward net</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MRGP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PFQN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPFQN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task Graph</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Copyright © 2006 by K.S. Trivedi 89
State space explosion can be handled in two ways:

- Large model tolerance must apply to specification, storage and solution of the model. If the storage and solution problems can be solved, the specification problem can be solved by using more concise model specifications that can be automatically transformed into Markov models (GSPN and SRN models).
- Large models can be avoided by using hierarchical model composition.

Ability of SHARPE to combine results from different kinds of models.

- Possibility to use state-space methods for those parts of a system that require them, and use non-state-space methods for the more “well-behaved” parts of the system.
Reliability models in practice

Creation of a model

Analysis

Finding bottlenecks

Breakdown of unreliability by causes

Reliability
Mean time to failure

Fully symbolic CDF
Fully symbolic MTTF
Fully symbolic PQCDF

Copyright © 2006 by K.S. Trivedi
Availability models in practice

Creation of a model

Analysis

- Availability
- Unavailability
- Downtime
- Cost of Downtime

Breakdown of Downtime
- Hardware
- Software
- Software Upgrades
- Preventive Maintenance
- MTTSF, MTTR

Expected interval availability
RBD example

File Server

Computer Network

Workstation 1  Workstation 2

Copyright © 2006 by K.S. Trivedi
Fault tree example
Performance models in practice

Creation of a model

Analysis

Steady-state probability of a node
Expected steady-state reward rate

Throughput
Average response time
Average queue length
Utilization
Markov chain model of a multiprocessor system
Markov Reward Model
GSPN model
GSPN model
Performability models in practice

Creation of a model

Analysis

Total loss probability
Capacity oriented availability
Throughput oriented availability
Possible outputs

- Availability, Unavailability and Downtime.
- Cost of Downtime.
- Mean Time to System Failure, Mean Time to System Repair.
- Downtime breakdown into Hardware, Software & Upgrade.
- Breakdown of Downtime by states, for Markov chain models, by blocks for Reliability block diagram models.
- Sensitivity Analysis, strategy to improve the Availability of desired systems.
SHARPE - references


ADVANTAGES OF THE APPROACH

- Pick a Natural Model Type for a Given Application
  (No Retrofitting Required)

- Use a Natural Model Type for a Portion of a Model
  (Encourages Hybrid and Hierarchical Composition)
ADVANTAGES OF THE APPROACH (contd.)

- Except for GSPN and SRN Models, No internal conversion done.
- Appropriate Solution Algorithm for Each Model Type i.e., hierarchy for Solution as well as Specification.
- Pedagogic Advantages.
- Multi-Version Modeling.
- Step-Wise Refinement in Modeling.
CASE STUDIES
CASE STUDY: AVAYA

- Modeling Swift: A combined hardware-software availability model of a real system being developed at Avaya labs

- Complete Reference

- Comprehensive model of 802.11
CASE STUDY: AVAYA (contd.)

- Network Survivability

- Several graduate student summer interns
CASE STUDY: AT & T

GSHARPE:

- A Preprocessor to SHARPE developed at Bell Labs by a Duke Student.
- User can specify Weibull Failure times and lognormal and other repair time distributions.
- GSHARPE fits these to phase type distributions and produces a Markov model that is generated for processing by SHARPE
- Several graduates hired by AT & T
- Many users of SHARPE at AT & T
CASE STUDY: BELLCORE/TELCORDIA

- Architecture-based software reliability:
  - proposed a methodology
  - applied the methodology to SHARPE
  - used Bellcore’s test coverage tool, ATAC, to parameterize the model
  - Bellcore is currently enhancing ATAC to incorporate our methodology
- Several summer interns, several graduates hired
- Complete Ref.
CASE STUDY: BOEING

- Several graduates hired; short courses offered; research contract for developing IRAP & SDM
- SHARPE, SPNP and HARP being used at Boeing
- SHARPE used to model a major subsystem of Boeing 787
- An Integrated Reliability Environment
- A working prototype
- Developed a high-level modeling language (SDM)
- Designed and implemented an intelligent interpreter
CASE STUDY: BOEING (Contd.)

- Interpreter determines which solution method is applicable
- Five different modeling engines are integrated:
  - CAFTA, SETS, EHARP, SHARPE and SPNP.
- Complete Reference

CASE STUDY: CISCO

- Conducted an availability comparison of a Cisco product with that of a competition using analytic models
- Hierarchical model with top level being a reliability block diagram and bottom level being a Markov chain
- Models solved using SHARPE
- Contained hardware-software-power supply, fans etc.
- A detailed report supplied to Cisco
- Reference
  - Internal to Cisco
CASE STUDY: DEC VAXCLUSTER

- Trivedi Sabbatical at DEC 1988-89
- Many sites for SHARPE and SPNP
- Developed three models of Processor Subsystem:
  - Two-Level Decomposition
    - Inner Level: 9-state Markov model; Outer level: n parallel diodes
    - Approximate Availability Analysis of VAXCluster Systems, Ibe, Howe, Trivedi,
      IEEE TR, April 1989
  - A Detailed SPN Model
    - O. Ibe, A. Sathaye, R. Howe, and K. S. Trivedi, “Stochastic Petri net modeling of
      VAXcluster availability,” Proc. Third Int. Workshop on Petri Nets and
      Performance Models (PNPM89), Kyoto, 1989, pp. 112–121.
  - A Detailed SPN model for Heterogeneous Cluster:
    - Dependability Modeling of a Heterogeneous VAXcluster System Using Stochastic Reward
      Nets, Muppala, Sathaye, Howe & Trivedi, in Avresky (ed.), Hardware and Software Fault
      Tolerance, Ellis Horwood, 1992
CASE STUDY: DEC VAXCLUSTER

- Storage Subsystem Model: A fixed-point iteration over a set of Markov submodels:
  

- Observed that availability is maximized with 2 processors:
  
  Should I Add a Processor?, Trivedi, O. Ibe, A. Sathaye and R. Howe, 23rd Annual Hawaii Conference on System Sciences, 1990

- Many interesting reliability, availability, performability measures computed:
  
CASE STUDY: DRAPER LAB

- Software reliability growth models developed for four different large software systems developed by Draper Lab
- Found Log-logistic based NHPP model the most suited
- Used SREPT tool
- In another project overall aim was Verification of system with very high reliability/availability specifications. Prototype under consideration was FTPP cluster 3.
CASE STUDY: DRAPER LAB

Hybrid approach proposed

- Fault injection based measurements.
- Statistical analysis of measured data to enable parameterization of analytical models.

Complete Reference

CASE STUDY: DRAPER LAB

- Reliability modeling of the prototype done: Parameterization done with the aid of existing reliability databases.
  - Analytical solution provided exact closed form expressions
  - Markov model solved using SHARPE
  - Petri net model solved using SPNP
  - Reliability bottlenecks found
CASE STUDY: GE

- Short courses offered
- summer interns
- many users of SHARPE
Case Study: HP

- Short courses offered; many users of SHARPE and SPNP
- Cluster Availability Modeling
- Server Availability
- Mass Storage Arrays Availability Modeling
- Started with Markov chains via SHARPE
- Progressed toward Stochastic Petri Nets and Stochastic Reward nets via SPNP
CASE STUDY: IBM

- Trivedi sabbatical in 1981 at IBM TJ W Res. Ctr.; worked with Phil Heidelberger and Phillip Yu and wrote the following papers:
  - Queueing Network Models for Parallel Processing with Asynchronous Tasks, Heidelberger and Trivedi, IEEE-TC, 1982
  - Analytic Queueing Models for Programs with Internal Concurrency, Heidelberger and Trivedi, IEEE-TC, 1983
CASE STUDY: IBM (contd.)

System Availability Estimator (SAVE):
- Duke-IBM Yorktown Joint Project; initial version of the software package delivered by Duke to IBM
- Worked with Steve Lavenberg and Ambuj Goyal and wrote the following papers:
  - Reliability Analysis of Systems with Limited Repairs, Goyal, Nicola, Tantawi & Trivedi, IEEE-TR, 1987
  - The System Availability Estimator (SAVE), Goyal, Carter, de Souza e Silva, Lavenberg & Trivedi, FTCS, 1986
  - Accelerating Mean Time to Failure Computations, Heidelberger, Trivedi and Muppala, Performance Evaluation, 1996

- The following Ph.D.s supervised by me are currently in IBM:
  - Steve Hunter, Bob Leech, Srini Ramani, Joe Rusnak, W. Earl Smith, Lorrie Tomek, Steve Woolet
CASE STUDY: IBM (contd.)

Several projects in Performance modeling with IBM RTP working with Andy Rindos and Steve Woolet; wrote the following papers:

- Exact Methods for the Transient Analysis of Nonhomogeneous Continuous-Time Markov Chains, Rindos, Woolet, Viniotis & Trivedi, in a book edited by Stewart
- Analysis of a Realistic Bulk Service System, Wang, Rindos, Woolet, Groner & Trivedi, HiPC, 1995
CASE STUDY: IBM (contd.)

- Software rejuvenation technology transfer to IBM x-server family; the work is discussed in the following papers written jointly with IBM researchers
  
  Analysis and Implementation of Software Rejuvenation in Cluster Systems

- Proactive Management of Software Aging
  V. Castelli, R. E. Harper, P. Heidelberger, S. W. Hunter, K. S. Trivedi, K. Vaidyanathan and W. P. Zeggert.
CASE STUDY: IBM (current)

- BladeCenter Availability model: Earl Smith and K. Trivedi
- Availability Monitor for an Appliance: Marc Haberkorn & Trivedi
- Performance and Reliability analysis of Business Processes: Naoto Sato and K. Trivedi
CASE STUDY: LUCENT

- Short courses, graduates hired, summer internships, many users of SHARPE
- A Validated Model of Hardware-Software Availability.
- Worked with V. Mendiratta of Naperville.
- Model is semi-Markov; solved using SHARPE.
- Parameters collected form field data.
- Model results validated against actual measurements.
CASE STUDY: LUCENT/AVAYA

- Software Rejuvenation:
  - A technique to counter software “aging” and increase its availability to clients.
  - Evaluated optimum rejuvenation interval which maximizes steady state availability (minimizes expected cost).
  - Subsequently collected data from real systems to show aging and to determine proactive fault management strategies.

- Complete Reference
  - A Methodology for Detection and Estimation of Software Aging, Authors S. Garg, A. Van Moorsel, K. Vaidyanathan, K.S. Trivedi
CASE STUDY: MOTOROLA

- Short courses, summer internships, research contracts, graduates hired, several users of SHARPE and SPNP
- Availability & Performability Modeling:
  - Modeled several configurations of Communication Enterprise Common Platform.
  - Practical approaches for approximating steady state measures in large, repairable, and highly dependable system: model decomposition, state space truncation, etc.
  - Both SHARPE and SPNP used
- Complete Reference

CASE STUDY: MOTOROLA (contd.)

- Recovery strategies in wireless handoff:
  - proposed and modeled several strategies; SPNP was used
  - Hierarchy of two-level models used; Fixed-point iteration was used
  - Call Admission Control for Reducing Dropped Calls in CDMA Cellular Systems, Y. Ma, James J. Han, and K. S. Trivedi, *Computer Communications*, May 2002
  - A Method for Multiple Channel Recovery in TDMA Wireless Communications Systems, Ma, Han and Trivedi, *Computer Communications*, July 2001
  - Channel allocation with recovery strategy in wireless networks, Ma, Han, and Trivedi, *European Trans. on Telecommunications (ETT)*, 2000.
CASE STUDY: MOTOROLA (contd.)

- A soft handoff scheme for improving utilization efficiency of traffic channels, X. Ma, Y. Liu, K. S. Trivedi, Y. Ma and J. Han, IEEE Int. conf. on CSCC, Greece, July 2001.

- A New Handoff Scheme for Decreasing Both Dropped Calls and Blocked Calls in CDMA System, Xiaomin Ma, Yun Liu, Kishor S. Trivedi, Yue Ma and James J. Han, Proceedings of the International Conference on Trends in Communications (EUROCON’2001), Bratislava, Slovak Republic, July 4-7, 2001.

- Availability bounds with non-exp distribution
  - System availability with non-exponentially distributed outages
    Yonghuan Cao; Hairong Sun; Trivedi, K.S.; Han, J.J. IEEE TR-2002
CASE STUDY: MOTOROLA

- Software rejuvenation being *analyzed* in Motorola cable modem termination systems (CMTS) as a high availability option.

- Comprehensive Availability Modeling:
  - Overall implementation architecture proposed for adopting software rejuvenation in current CMTS.
  - Modeled hardware failures, Heisenbugs, aging-related bugs, failure detection coverage.
  - Several software rejuvenation strategies considered.
  - Computed optimum rejuvenation interval which maximizes system availability or minimizes downtime maintenance cost.
  - SPNP used.

- Complete Reference
  - Modeling and Analysis of Software Rejuvenation in Cable Modem Termination System, Yun Liu, Yue Ma, James J. Han, Haim Levendel, and Kishor S. Trivedi, Proceedings of the 13th Int'l. Symposium on Software Reliability Engineering, ISSRE2002, pages 159-170, Annapolis, Maryland, November 2002.
CASE STUDY: NOKIA

- Short course offered
- Helped solve an interesting modeling problem
- A short paper published based on this work

Complete Reference
- The Effect of Deferring the Repair on Availability
  Supriyo Bose & Veneet Kumar & Kishor Trivedi; Fast Abstract, DSN 2003
CASE STUDY: SOHAR

- Dependability Evaluation GUI called SDDS:
- The tool has been developed
- High-level modeling language related to SDDS
- Engine used: SHARPE
- Funded by Rome Lab under SBIR

Complete Reference

CASE STUDY: SUN Microsystems

- Short courses offered
- Helped model a fault tolerant system
- Hierarchical model using RBDs and Markov chains
- Hardware, software, different types of faults, power supply, fans, network cards etc.
- Many users of SHARPE
- Summer interns
- A graduate hired (Kalyan Vaidyanathan)
- Working together on software aging and rejuvenation
CASE STUDY: ZITEL

- Comparison of two different fault-tolerant RAMdisks.
- Stochastic Petri Net Package (SPNP) was used to model the two systems for their reliability.