CS 780: Discrete State Models

Homework #1
Due Feb 1, 2008

The goals of this exercise are for you to become familiar with the mechanics of modeling using Möbius and analysis of systems using Fault Trees. Completing this exercise will assist you in completing the subsequent homework assignments and projects. This homework leads you through the construction of a model for a *reliable web server*. The design is divided into three sections. First, you will analyze the reliability of the web server's memory subsystem using methods learned in lecture and the Möbius tool. Second, you will analyze the web server's CPU subsystem, storage subsystem, and communications subsystem, as well as the entire web server. Lastly, you will optimize the system for reliability given a budget and component costs.

Note: The Möbius manual is a great reference for this assignment, e.g., Chapter 5.4 Fault trees.

Basic System Description

The web server you will model consists of four subsystems: memory, CPU, storage, and communication. The web server will function as long as each subsystem is functioning. The definition of "functioning" is given for each subsystem below.

1. **Memory Subsystem**

   The memory subsystem consists of one memory controller hub, channels, and DIMMS. Each hub can support up to two channels, each of which can support up to three DIMMS. An example of a maximal controller-channel-DIMM system is given in Figure 1.

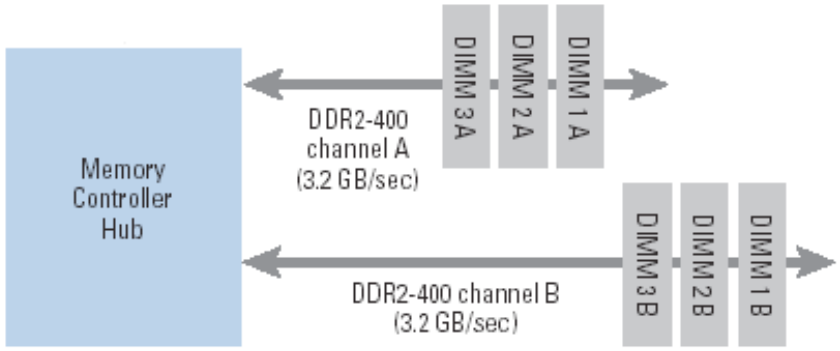

Figure 1: A maximally loaded memory subsystem.

   The subsystem is functioning if the controller is functioning, one channel is functioning, and a DIMM on that channel is functioning. In this system, all failures are independent of each other.

   From the MIL-HDBK-217 reference (Military Handbook for "Reliability Prediction of Electronic Equipment"), we find the following failure rates per hour:
   DIMM: 4.13608 E-06
   Channel: 8.67 E-07
   Memory Controller Hub: 4.433 E-09

2. **CPU Subsystem**
The CPU subsystem consists of one CPU controller, and up to 4 CPUs.

The CPU subsystem is functioning if it contains at least 1 functioning CPU, and the CPU controller is functioning. All failures are independent, with the following rates per hour:

CPU: 5.5 E-05
CPU Controller: 6.3 E -08

3. Storage Subsystem

The storage subsystem consists of hard drives, channels, and one hard drive controller. The controller can facilitate up to two channels, each of which can support up to 2 hard drives.

The storage subsystem is functioning if the controller is functioning, the controller is attached to 2 functioning channels, and each channel is connected to a functioning hard drive.

All failures are independent, with the following rates per hour:
Hard Drive Controller: 3.4 E-06
Channel: 2.23 E-07
Hard Drive: 1.667 E-06

4. Communications Subsystem

The communications subsystem consists of one communications controller and one or two network interfaces.

The system is functioning if the controller is functioning and at least one network interface is functioning.

All failures are independent, with the following rates per hour:
Communications Controller: 4.5 E-08
Network Interface: 3.333 E-06

Homework Instructions

Homework submission instructions are detailed at the end of this document. As a part of this submission, you will be required to record responses to the numbered parts that are in **bold face** throughout this document.

Part 1: Running Möbius

This will prepare you to begin work on the homework exercise.

1. To run Möbius from the command line, type "mobius". You will be presented with the Project Manager.
2. Select "Project -> New" from the menu. Name the Project "ReliableWebServer."

You are now ready to begin constructing models!

Part 2: Möbius Background

Below is a brief description of the functionality of each component presented in the Project Manager:

Atomic: Used to describe sub-models or entire models. Each atomic model can be represented in a different "formalism." A Fault Tree is just one of the formalisms.
Composed: Used to connect and replicate models.
Reward: Used to specify what we want to measure about the system (availability, reliability, latency, etc).
Study: Used to vary parameters of the system in order to study different system configurations.
Solver: Used to solve models (atomic or composed) for certain measures (rewards), either by simulation or state space generation.
Numerical: Used to analyze models solved by state space generation.

Part 3: Using Fault Trees to Analyze the Memory Subsystem

In this section, we will analyze the reliability of the memory subsystem over one year (8760 hours). We will consider the **minimal** functioning memory subsystem, i.e., 1 memory hub controller, 1 channel, and 1 DIMM.

Part 3a: Fault Tree Analysis by Hand

(1) On paper, draw the fault tree for the minimal functioning memory subsystem. What is the unreliability of this system over 1 year? What is the reliability of the system over 1 year?

Part 3b: Memory Subsystem Analysis Using Möbius

Create an Atomic Model

1. Select the "Atomic" branch from the project tree.
2. Push the "New" button; type in the name "MinMemSub;" and select the formalism you want to use, in this case a Fault Tree.
3. To create a top-level node, select the blue circle and left click on the panel and give it the name "MemSubFail."
4. Create an event to represent a failure of the memory controller by selecting the red circle and left clicking on the panel. Give it the name "ControllerFail" with the rate given in the description. Make sure to set the rate type to exponential.
5. Repeat step 4 for the channel ("ChannelFail") and the DIMM ("DIMMFail")
6. To represent the relationship between the three component failures and the system failure, select the appropriate logic gate and left click on the panel.
7. Connect the three events to the logic gate using the straight-line connection.
8. Connect the logic gate to the node "MemSubFail" using the straight-line connection.
9. Press "File->Save" to save the model, and "File->Close" to exit.

We now have a Fault Tree model representing the minimum memory subsystem.

Note:

- Möbius is picky about the order when connecting events, gates, and nodes. Events connect to gates and gates connect to nodes.
- Möbius builds your models by generating C++ code and compiling it. Numeric values that you enter in fields will be copied directly into a source code file. This means that an expression such as "1/2" will be interpreted as integer division and will be compiled into the value 0. Use "1.0/2" instead. In addition, scientific notation can be used, e.g., 1.34E-08.
- Events can only "fire" once. That is, an event can happen once and only once.

Create a Performance Variable

A performance variable will enable us to measure the reliability of the memory subsystem.

1. Select the "Reward" branch from the project tree.
2. Push the "New" button. Name the reward structure "MinMemSubReward."
3. When the "Select Reward Child" dialog box comes up, select the atomic or composed model you would like to analyze. In this case, it is the "MinMemSub" atomic model.
4. Now, we need to create a performance variable to measure the subsystem's reliability. To do this, type the name "reliability" as a variable name and press "Add Variable."
5. Select the "Time" tab. Since we are evaluating the 1-year reliability, we want to evaluate our performance variable at one instance of time, the 1-year mark. To do that, set the "Type" to "Instant of Time," the "Number of Time Points" to 1, and the "Start Time" to 8760.0.
6. Select the "Rate Rewards" tab. Here we place C/C++ code to evaluate the reward measure we want to analyze. To compute the 1-year reliability of the subsystem, we need to determine if the subsystem has failed at the one-year point. (Note that reliability is an interval of time measure, but since there are no repairs in our model, we can compute whether the system has failed at ANY POINT in the interval by looking at its status at the end of the interval using an instant of time measure.) In the previous step, we set the reward variable to be evaluated after 1 year. In this step, we need to determine if the system is functioning or not when the reward is evaluated.

   Hint:
   - The subsystem has failed if the high-level node in the Fault Tree is not 0.
   - The "Rate Reward" code should return a value.
   - The value of the high-level node in the Fault Tree can be obtained using the syntax `MinMemSub->MemSubFail->Mark()`, i.e., the model name, followed by the node name, followed by `Mark()`.
7. Ignore the other tabs for now. Press "File->Save" to save the model, then exit with "File->Close."

Note:

- A PV has a value of 0 by default.

Create a Study

A study allows us to vary the global variables in the model so that we can examine how different configurations affect the system. Since we are not using global variables in this model, we will create an empty study.

1. Select the "Study" branch from the project tree.
2. Push the "New" button. Select a Range Study with the name "MinMemSubStudy."
3. When the "Select Child Study" dialog appears, select the "MinMemSubReward" child. If the dialog does not immediately pop up, left click on the panel.
4. Since our study has no global variables, there is nothing left to do except to save the Study with "File->Save," and exit with "File->Close."

Create a State Space Generator

Now that we have described the model using the Möbius framework we are able to solve the model and our performance variable. Möbius supports two methods of solving a model: analytical solvers and simulation. In this exercise, we will only use analytic solvers. To use them, we must first generate a state space that a numerical solver can access.

1. Select the "Solver" branch from the project tree.
2. Press the "New" button. Select a State Space Generator with the name "MinMemSubSSG."
3. When the "Select Solver Child" dialog pops up, select the "MinMemSubStudy" since it is the study we would like to use.
4. Press the "Start State Space Generation" button. 8 States should be generated.
5. Save the Solver with "File->Save," and exit with "File->Close."

Create a Numerical Solver

Now that we have generated a state space, we can finally solve our model!

1. Select the "Numerical" branch from the project tree.
2. Press the "New" button. Select the Transient Solver and name it "MinMemSubTRS."
3. When the "Select Numerical Child" dialog appears, select the "MinMemSubSSG" since it is the state space we would like to use.
4. Press the "Solve" button. The results are stored in the `MobiusProject/ReliableWebServer/Result/MinMemSubTRS/` directory.

Note:

- We used TRS (Transient Solver) to solve this model since we wanted to measure the failures at a specific time and not steady state.

(2i) What is the mean value of the "reliability" performance variable?
(2ii) Does it match your hand-calculated answer?
(2iii) How many iterations did the analytic solver take?

Document the Model

As part of the homework submission procedure, we need to document this model, which we can do using the Möbius documenter. For the atomic model "MinMemSub," and the performance variable "MinMemSubReward," perform the following steps.

1. Open the component.
2. Select "File->Document Model" from the menu.
3. Select the appropriate component from the component list.
4. Press the "Document" button.

Note:

- The documentation is stored in the `MobiusProject/ReliableWebServer/Documentation/`<component_name>`/` folder.

Part 4: Continued Analysis of the Minimal System

Part 4a: Analysis of the Other Minimal Subsystems

Congratulations on building and analyzing your first Fault Tree in Möbius! Hopefully the process of creating Atomic models, Rewards, Studies, Solvers, and Numerical Solvers has begun to make some sense.

In this section, we will analyze the reliability of the remaining minimal subsystems using the Möbius tool. Using the steps in Part 3b, create a model, performance variable, study, state space generator, and numerical solver for each of the minimal functioning CPU, storage, and communication subsystems. You do not have to draw or calculate the rewards by hand. However, please perform model documentation as you did in Part 3b.

(3,4,5) What is the mean value of each performance variable?

Part 4b: Analysis of the Entire System

We have analyzed each of the web server's subsystems. Now, we must piece together the subsystems so we can analyze the reliability of the entire web server. To do that, we use the Composed tool in Möbius to link the subsystems together.

Create a Composed Model

1. Select the "Composed" branch from the project tree.
2. Press the "New" button. Select the "Rep/Join" type with name "MinSys."
3. Select the J(oin) button and left-click on the panel. Enter the name "SysJoin."
4. Select the S(ub-model) button and left-click on the panel. Select the "MinMemSub" entry from the list.
5. Repeat step 4 for each of the CPU, storage, and communication sub-models.
6. Connect the "SysJoin" component to each sub-model.

Now, we need to create a shared variable between all of the sub-models. We do this by editing the Join structure and telling it states from each atomic model that should be joined into one state.

7. Right click on the "SysJoin" node and select "Edit."
8. Select the "Create New Shared Variable" button and type in the name "NumFailedSubsystems."
9. Each tab represents one of the four sub-models. Select the "SubFail" state for each subsystem, e.g., "MemSubFail," and "CPUSubFail."
10. Select the "OK" button, then save with "File->Save," and close with "File->Exit."

Now, we have effectively merged the top-level nodes of each of the subsystem Fault Trees. As with the atomic models, document the composed model.

Create a Performance Variable for the Entire System

Now, we must create a new performance variable for the reliability of the entire system. We

do this just as before, but now we select "MinSys" as the child. We create a reward named "MinSysReward," and a "reliability" performance variable within it.

The values for the "Time" tab are the same as in Part 3b. The code in the "Rate Rewards" tab is similar to that in Part 3b, except we now need to consider under what condition(s) the entire system fails. As before, document the performance variable.

Hint:
- The "NumFailedSubsystems" variable represents the number of failed subsystems.
- We can access the value of "numFailed Subsystems" by accessing any of the states that it joins.

Create a Study, State Space Generator, and Numerical Solver
Using the same technique as before, create the study, state space generator, and numerical solver. Solve using the Transient Solver.

(6i) What is the mean reliability for the "bare bones" web server?

Part 5: Creating a More Reliable Web Server

Up to this point, we have modeled the minimal functioning web server. The problem is that it is not very reliable.

The marketing department has guaranteed that our web server will be **greater** than .9 reliable.

Your task is to augment the web server so that it meets that requirement. However, your solution should minimize cost. The web server must still abide by the specifications in the System Description. For example, it must still only have 1 memory controller.

Components in addition to the minimal configuration cost the following:

DIMM:	$200
Memory Channel:	$100
CPU:	$200
Hard Drive:	$200
Network Interface:	$300

(7i) What additional components did you add?
(7ii) What reliability were you able to achieve?
(7iii) What was your additional cost?
(7iv) How many states did the state space generator generate?

Please document each new atomic model, composed model, and performance variable.

Part 6: Submission Instructions

- Paper submission
  - Responses to the numbered **bold face** problems contained within this document.
  - Model documentation in the order below. Use the "Document Model" feature

of Möbius; however, use this feature individually for each of the components below. That way, you can place them in the correct order. Also, you are responsible for making sure that all of the relevant information makes it onto the paper that you submit. If something trails off the side of your print out, then clearly write it on the documentation.

1. Atomic models from Parts 3 and 4a
2. Reward variables from Parts 3 and 4a
3. Composed model from Part 4b
4. Reward structure from Part 4b
5. Atomic models from Part 5
6. Composed model from Part 5
7. Reward structure from Part 5

- Electronic submission
    - At a later time, you will be informed how to electronically submit a **cleaned and archived** copy of your Möbius project.