RBDTool - Quick Start Guide

August 18, 2014

1 Introduction

RBDTool is a small utility for constructing and analyzing reliability block diagrams. It supports four standard types of reliability blocks:

Singletons
- Individual objects that either work or do not, according to some reliability.

Series Blocks
- Groups of two or more blocks, all of which must be operational for the block as a whole to be operational.

Parallel Blocks
- Groups of two or more blocks, at least one of which must be operational for the block as a whole to be operational.

K-of-N Blocks
- Groups of exactly $N$ blocks, at least $K$ of which must be operational.
  (Note that Series is a K-of-N with $K == N$, and Parallel has $K == 1$.)

These blocks can be used individually, or combined into arbitrarily complex arrangements. RBDTool also supports creating user-defined block groups, such as layouts corresponding to the various RAID configurations.

2 System Requirements

RBDTool is a Qt 4.8 application targeted at recent Linux, OS X, and Windows. Stand-alone binaries (i.e. ones which do not require a system Qt installation) are available for all three platforms running on x86_64 processors. Known-functional platforms include:

- Linux 2.6.32+, running Gnome (RHEL build in MTU labs) and XFCE (Debian)
- OS 10.9 (Mavericks) (Compiles but crashes as of 8 April)
- Windows 7 Enterprise (MTU lab build) and Professional

Other reasonable variants (e.g. other display managers, Windows versions, OS X to 10.6) should be supported by the existing binaries. 32-bit-only platforms will likely not be able to run the pre-compiled versions, though the program could be recompiled to target such machines.
3 Running RBDTool

Given a copy of the RBDTool binary, it can be directly invoked in whatever way is appropriate for the host OS. This will normally be a double-click through a graphical interface, or manually invoking the binary via the command line. RBDTool does not, at present, accept command line arguments, so there is no need to supply them. The tool may generated output on stdout / stderr, but this is limited to debugging information and can be safely ignored during normal operation.

4 Interface Overview

The RBDTool interface can be broken down into three major components:

Menus and Toolbar

Occupying their traditional location at the top of the interface (or in the system menu bar), these provide access to normal things like save/load, undo/redo, and some tools for manipulating how diagrams are being displayed.

Block Panel

The block panel occupies the left side of the interface, and is used to select blocks for insertion. At the minimum, the panel will include the basic RBD blocks: singleton, series, parallel, and K-of-N. If additional block groups (like the RAID blocks) are present, they are also presented in the block panel. Additional blocks are automatically grouped into their own sub-panels, which can be selected by clicking the panel heading (e.g. ”Basic RBD Blocks” or ”RAID Blocks”).

Immediately beneath the block panel is a short description of the currently-selected block. The description can be either text (as with the basic block types), or images (as with the network blocks). This description is updated automatically as different blocks are selected.

Diagram Area

The diagram area is the large, white space to the right of the block panel, and occupies most of the space in the interface. This area is used to display block diagrams, and can be scrolled and zoomed as needed. While the block panel is used to insert new blocks, most manipulation of existing blocks takes place in the diagram area.

5 Adding Blocks

RBDTool uses a two-clicks approach to block placement, rather than supporting click-and-drag placement at this time. To place a block, select it in the block panel (first click) and click in the diagram area to place the block (second click). Once the second click is registered, the tool will gather information needed for the block before actually placing it. What information is gathered, and how, is determined by whether the block is part of the standard set or being provided as a custom block. Clicking 'cancel' at any time during block creation with abort the creation of the block and cancel block creation mode. Once a block has been placed (or placement is cancelled) the user must once again make a selection in the block panel in order to continue placing blocks.
5.1 Basic Blocks
Singleton blocks will prompt for a single reliability before placement. This reliability may be in the range $[0.01 - 1]$, with a step size of 0.01. The default reliability is 0.5.

Compound blocks (series, parallel, and K-of-N) require information about the number of components present, the number required (for K-of-N blocks), and the reliability of individual components. A single dialog is used to retrieve all of this information. Once again, reliabilities may be in the range $[0.01 - 1]$, with a step size of 0.01 and a default of 0.5. At least two blocks must be present within a compound block, but no upper limit is directly enforced (computation time or memory may enforce a limit, however). The number of components required by a K-of-N block is constrained to be no greater than its number of components, however. Compound blocks are always created with singletons as their components, although these singletons may be replaced with other components as documented in Section 7.

5.2 User-defined (e.g. RAID or Network) Blocks
The system for creating user-defined blocks leaves information gathering to the individual block. At present, both the RAID and network blocks use a series of prompts to gather information before generating a block diagram. Reliabilities continue to work on the range $[0.01 - 1]$ in user-defined blocks, though other variables may or may not be limited.

More information about the user-defined types is provided at the end of this document.

6 Block Representations
RBDTool uses common representations to display block diagrams. Singletons, whether alone or as part of a compound block, are displayed as small rectangles with solid borders. Centered within a singleton is a string of the form “R: 0.5” which indicates the reliability of the block.

Compound blocks are presented with a dashed, rounded rectangle containing their components, which may be singletons or other compound blocks. These blocks present a string of the form “Reliability: 0.75” which gives the aggregate reliability of the system represented by the block. Series and parallel blocks present their components as, respectively, a horizontal or vertical set of components with lines representing the (logical) dependencies between them. K-of-N blocks have a representation similar to that of a parallel block, with the addition of a circle on the right side of the diagram presenting the number of components required out of the total available. These representations are fairly typical for reliability block diagrams.

When blocks are nested (that is, a compound block contains one or more other compound blocks), the component blocks are always presented with their normal borders and reliabilities displayed. This way, it is possible to see how the reliability of components is connected to overall reliability.

7 Interacting with Blocks
7.1 Reliability Changes
It is possible to change the reliability of a singleton once it has been placed. To do this, double-click the singleton and enter a new reliability value in the dialog that appears. As usual, the new
reliability must be in the range [0.01 - 1] with a step size of 0.01. If a singleton is part of one or more compound blocks, updating its reliability will cause the reliability of the containing blocks to updated automatically. This way, it is possible to examine how changes in component reliability effect aggregate reliability.

7.2 Moving Blocks

Blocks can be moved around the diagram area using a standard drag-and-drop operation. Compound blocks always move as a unit, and the components within cannot be reordered at this time.

7.3 Block Replacement

It is possible to replace any singleton which is part of compound block with another simple or compound block. To do so, drag the replacement to the singleton, and release the mouse button with the cursor over the block to be replaced. The original single will be removed my from the compound block, and its place taken by the replacement block. Performing a replacement in this way will cause the reliability of the containing block to be updated automatically.

7.4 Block Deletion

Blocks can be deleted by selecting them and then pressing the delete key, or interacting with the delete options in the toolbar or item menu. There is no way, at present, to remove a component of a compound block without deleting the containing block.

8 Other Features

8.1 Undo and Redo

RBDTool provides support for unlimited undo and redo via toolbar buttons. The following actions can be undone (or redone):

- Inserting a block
- Changing singleton reliability
- Moving a block
- Combining / replacing blocks
- Deleting a block

For the purposes of undo, a block replacement counts a two actions: a block move followed by a block replacement. When reversing a replacement, the blocks are separated by the first undo, but only return to their original location after the second undo.

8.2 Save and Load

RBDTool is capable of saving and reloading diagrams. This functionality is accessible via the file menu, and generates a human readable / editable file at a location specified by the user. Loading a file is not an undoable action, and will clear the undo / redo history.
8.3 Block Ordering

Blocks are permitted to overlap within the diagram area. It is possible to use the Bring to Front / Send to Back options in the toolbar and item menu.

9 RAID and Network Blocks

9.1 RAID Blocks

RBDTool provides representations of the most common types of RAID array, including RAIDs 0, 1, 2-5, 6, 1+0 and 0+1. Inserting a RAID 0-6 block will prompt for the total number of drives in the array and the reliability of each drive. When a RAID 1+0 or 0+1 block is inserted, the system will ask for the number of stripes and the number of mirrors, and use these figures to compute the total number of drives. As before, the user is also prompted for the reliability to be applied to each drive. All six diagram types provide the option of including the RAID controller in the diagram. The reliability of the controller can be specified when the block is created.

9.2 Network Diagrams

Rather than allowing for fully general network topologies, the current version of RBDTool provides a small set of networks with configurable parameters. All of the networks are configured to include both the network nodes and the links connecting them, and these parameters have different default values. The network blocks also take advantage of RBDTool’s ability to display images in lieu of description text to display the network topology. RBDs are generated based on the reliability of communication between two nodes, indicated in the images by a double-circle.

9.2.1 Token Ring Networks

![Singly-linked Token Ring](image1.png) ![Doubly-linked Token Ring](image2.png)

The current set of blocks include representations of both singly- (above, left) and doubly-linked (above, right) token ring network. Normal token-ring networks provide an explicit ACK mechanism to allow senders to determine that their message was received. Singly-linked blocks can either account for this ACK-ing behavior or consider only the case of sending a message from one to the other without needing an ACK. Doubly-linked blocks only support the case where no ACK is required, because there are too many ways to handle dropping a network link to be accounted
for within the tool. For the purposes of the diagram, messages are sent by the the double-circle on the top of the image, and received by the double-circle at the lower right.

All token ring blocks prompt for the number of blocks in the left and right paths in the network. The right path runs from the start to end node in a singly-linked network, and the left path corresponds to the return trip. This is shown in the diagrams, and the same nodes are used in the doubly-linked case. The default settings for the networks produce a network equivalent to the one in the image.

9.2.2 (Fat) Tree Networks

Fat tree topologies are common in supercomputing clusters, where they generally connect distribution switches. (The name ‘fat tree’ comes from the tendency to use progressively-higher-bandwidth links as one gets closer to the root node, to deal with increased traffic.) When pairs of nodes communicate in such a tree, they generally have one of two arrangements: either the nodes share a common ancestor within the tree (above, left), or one is the ancestor of the other (above, right). These are referred to by the software as ”common ancestor” and ”ancestor path” arrangements, respectively.

Both types of tree will prompt for the height of the tree as part of creating the block. For the purposes of the network blocks, height is defined to include the root node, so the sample trees above have height 3. The system will then prompt for the depth of the ancestor (defined counting from the root, with the root node at depth 0), followed by the depths of the remaining nodes (left and right for common ancestor, or successor for an ancestor-path). The system automatically enforces a requirement that the ancestor node be ‘above’ the other nodes, and that everything fits within the given height. The root node may be specified as the ancestor, but this is not required.
9.2.3 Multi-Path Networks

It is not uncommon for large networks (such as the Internet) to provide multiple paths between nodes. This can either be an intentional feature for redundancy, a side-effect of the construction of the network, or both. The multi-path network blocks are intended to model simplified versions of these networks.

The Multi-Path A and B blocks share a common network layout (shown above, A on the left), but differ in which pair of nodes are selected for communication. Unlike the other blocks, only the reliability of nodes and links can be specified when these blocks are created. This helps to manage the complexity of the resulting block diagrams, which are already rather convoluted.