Contents

1 Introduction 3
   1.1 Structure of the Document 3

2 Overview of Design 4

3 Description of Components 4
   3.1 InnerRM implementation SimpleNetworkRM 4
      3.1.1 Network Description 4
   3.2 Path Computation Module 6
   3.3 Path Scheduler Module 6
1 Introduction

This document describes the current implementation of the HARC NRM. The initial NRM design document talked about the NRM having two parts:

- the “RM” part, which contained the generic RM code that implements the Paxos Commit, etc;
- the “N” part, which implements the Network-specific code.

Following the current RM documentation, which is more recent, the “N” part is just the InnerRM subclass. The “RM” part is everything else—the Paxos protocol code, the code that worries about timing out transactions which stay open too long, etc. and also the support classes that look after bookings. This latter part is documented pretty well in the RM Modules document, and won’t be mentioned further here.

The NRM has the same structure as any RM, shown in Figure 1.

1.1 Structure of the Document

The next section gives an overview of the structure of the NRM as implemented. Hopefully the basic structure of the NRM will be suitable for future versions; I’ve used O-O techniques to try to keep re-usable code in sensible places, and have tried to support the future directions we’ve discussed in the past. (But no doubt there will be things I haven’t thought of.)

When I’m talking about a Perl Module or class, I’ll use the name of the module or class, and not the filename.\footnote{The filename is always just the name of the module/class with the suffix ‘.pm’}
2 Overview of Design

The NRM implemented for the GLIF demo is very simple. Nonetheless, there is a basic structure in the components of the NRM which it should be possible to follow as the NRM is developed further. (I expect that the interfaces will require some tweaks, but I think that the basic structure is reasonable.) The NRM is divided into two key components:

1. A Path Computation Module, which works out all possible paths for a requested endpoint combination, and
2. A Path Scheduler Module, which gets to choose between the possible paths.

There is currently only one implemented Path Computation Module, NRM_PathCompTable, performs simple table lookups to determine the possible paths between two endpoints. Although it is simple, this class is designed so it can be configured with multiple paths for two endpoints.

The current Path Scheduler Module is simpler. It has no ability to choose between multiple possible paths, and does not support scheduling in such network topologies. It simply looks up the commands that must be sent to the switches to create/delete/monitor a path between the two given endpoints. There is some code which will be useful to re-use, in particular, the parts dealing with the control of the devices. This re-usable code has been placed in the class NRM_PathSched; the rest of the code, including the Timetable handling is in NRM_PathSchedTable. New Path Scheduler Modules can inherit from NRM_PathSched (they may wish to override some of the device control methods, which is fine of course).

The hope is that this division of components, and the code in SimpleNetworkRM will continue to be useful in the future. Perhaps the Path Computation module will survive, although it seems that a version which computed the paths based on a fuller description of the network topology would be a worthy replacement. The Path Scheduler will need to be replaced as soon as the network topology contains multiple paths between any two endpoints. But even before that, the scheduler module should be altered to build the commands using the endpoint and path information passed to it.

3 Description of Components

The structure of these components will now be discussed in depth.

3.1 InnerRM implementation SimpleNetworkRM

3.1.1 Network Description

Although different Path Computation and Path Scheduler modules are used to look after most of the “heavy lifting”, the SimpleNetworkRM module contains the Network Description. Figure 2 shows the network defined by the current NRM configuration files in CVS as of 4th October 2006. The cloud represents the part of the network where dedicated paths can be scheduled. There are three types of nodes in the network:

- “internal” nodes, i.e. UO1, UO2, UO3 and UO4;
Figure 2: HARC NRM Network Description.

- nodes on the edge of the bookable/schedulable network, i.e. X1U, CH1, RA1, VC1, etc.; and
- nodes which are outside of the bookable/schedulable network, but which are attached to it, i.e. BT1, BT2, BT3, LA1 and LA2.

The network is defined in five configuration files, the first three of which specify the devices in the network; the last two specify the topology of the network between those devices. These files are...

**nrm.pathable** A list of the devices which are inside or at the edge of the bookable part of the network. These are typically the devices that are attached directly to the Calient switches (a mix of compute resources and routers), and the Calient switches themselves.

**nrm.others** The other devices in the network. These are mainly computational resources that are attached to the network via a router connected to the Calient switch.

**nrm.aliases** A list of aliases that users can refer to the pathable/other devices by, e.g. BT2 can also be booked using the alias santaka.cct.lsu.edu

**nrm.links** A description of the links in the network. As well as defining the endpoints of the paths (both are pathable devices), the link type (TE or Trib) and the maximum bandwidth that the link supports (in Mb/s) are also defined. Some links are labeled on the diagram.

**nrm.paths** An exhaustive list of paths across the network, defined as a list of links, indexed by the start and end points of the path. If there is no path defined for a pair of endpoints, then the NRM will not accept attempts to schedule a path between them. In the current NRM configuration, nrm.paths contains no paths starting or ending at any of the internal nodes of the network (i.e. the Calient switches), e.g. UO1. As an example, the path from RA1 to X1U is defined as consisting of the links UO2-RA1, UO1-UO2 and UO1-X1U(RA).
3.2 Path Computation Module

3.3 Path Scheduler Module