User Manual for Projections (version 2.0)

1 Introduction

Projections is a performance analysis tool for Charm. Projections can be used to examine performance information for Charm programs. Charm is currently available on shared machines including Multimax and Sequent Symmetry, nonshared machines including Intel iPSC/2, i860, and Paragon, NCUBE/2, CM-5, and networks of workstations, and any UNIX based uniprocessor machine.

The tool provides the user with mechanisms to view:

1. *System specific performance information.* This includes properties of system, such as busy time, queue lengths, and quantities of various different types of messages — henceforth refered to as system attributes.

2. *Program specific performance information.* Projections allows the user to view information about the creation and processing of messages to particular entry points — henceforth refered to as user attributes.

3. *Analysis of system performance.* Projections also provides the user with a detailed analysis of the performance of the program through time. The analysis will point out the parts of the program responsible for poor performance. This is still being implemented, and may not be available for the version you are using.

This document should help the first-time user of Projections to get familiar with its use. A preliminary version of the tool was introduced in [1, 2]. A more developed version of the tool, and the design for the future version of the tool appears in [3].

2 Collecting trace information for Projections

Projections is a post-mortem tool: it needs traces of the program execution. Trace information for Projections can be obtained by linking the program with the *-execmode projections* option during *charmlink*. A program (say *pgm*) run in the projections mode of execution will produce one logfile for each processor called *pgm.X.log* (where X is the processor number) and a state file (which contains information about the user program) called *pgm.sts* [1]. Projections needs all the logfiles and the state file.

3 Running Projections

Projections is run using the projections system call.

NAME

projections

SYNOPSIS

projections $+l$ *pgm* [$+t$ *num*] [$+p$ *num*]

OPTIONS

$+l$ *pgm* Use the log-files with name *pgm*. This corresponds to the name of the program that was executed.

$+t$ *num* Specify the timestep (in microseconds) as *num*. The default timestep is 10,000 microseconds. The timestep is used to divide the execution of the program into stages of that length. Information is displayed in terms of stages. The user can adjust the timestep when projection is initially called, or later while examining the trace files through Projections.

$+p$ *num* Specify the number of processors as *num*. The default number of processors is 1.

[1]The name of the state file is borrowed from Upshot, a tool developed at Argonne National Laboratory by Rusty Lusk and Virginia Herrarte.

4 Displaying Data Using Projections

The nature of execution of a **Charm** program can be well understood by examining the values of the system and user attributes. These values provide an indication for when, where and what type of work is created during a program execution, and how this corresponds to the overall performance of the program in terms of the percent busy time.

The trace data obtained in the manner described above can be displayed in different ways to provide the user with many views of the performance of the program. The most basic views treat system and user attributes as a function of two variables: *stage* and *processor index*. Each attribute can be thought of as a three-dimensional object, and the views are merely projections of this object onto the coordinate axes defining the object space. The execution of the user program is divided into equal-length periods of time called *stages*. The length of the time period, called *timestep*, used to divide the execution time into stages is user-defined and can be changed interactively by the user program to define finer and coarser stages, as desired. The views provide different projections of this two-variable function. We can represent the function, F_a, for the attribute, a, as

$$a = F_a(s, p)$$

In the above equation s is the stage of program execution and p is the processor index. The stage, s, and the processor index, p, range over a *stage-set* and a *processor-set*, respectively. In the default case the *stage-set* ranges over the stages for the period of execution of the program, and the *processor-set* ranges over the processors used for execution.

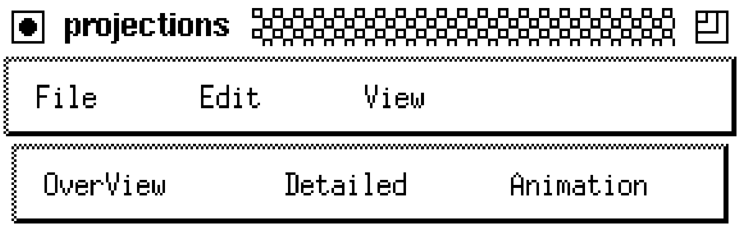

Figure 1: **Top-Level View of Projections**

The top-level window for Projections is shown in Figure 1. There are there types of views — *overview*, *detailed*, and *animation* — there is a button corresponding to each one of these views. The *Set Parameters* button in the *Edit* menu button in the top-level window can be used to globally change the value of the timestep, i.e., all views will be redrawn with this new value of timestep.

The *Show Legend* button in the *View* menu can be used to see the color and line type representation of different attributes.

We shall now describe the features of the three different types of views in Projections. In all the views, the stage and processor sets can be altered using the *Set Parameters* button in the *Edit* menu for that view. The change in these sets affect only that particular view.

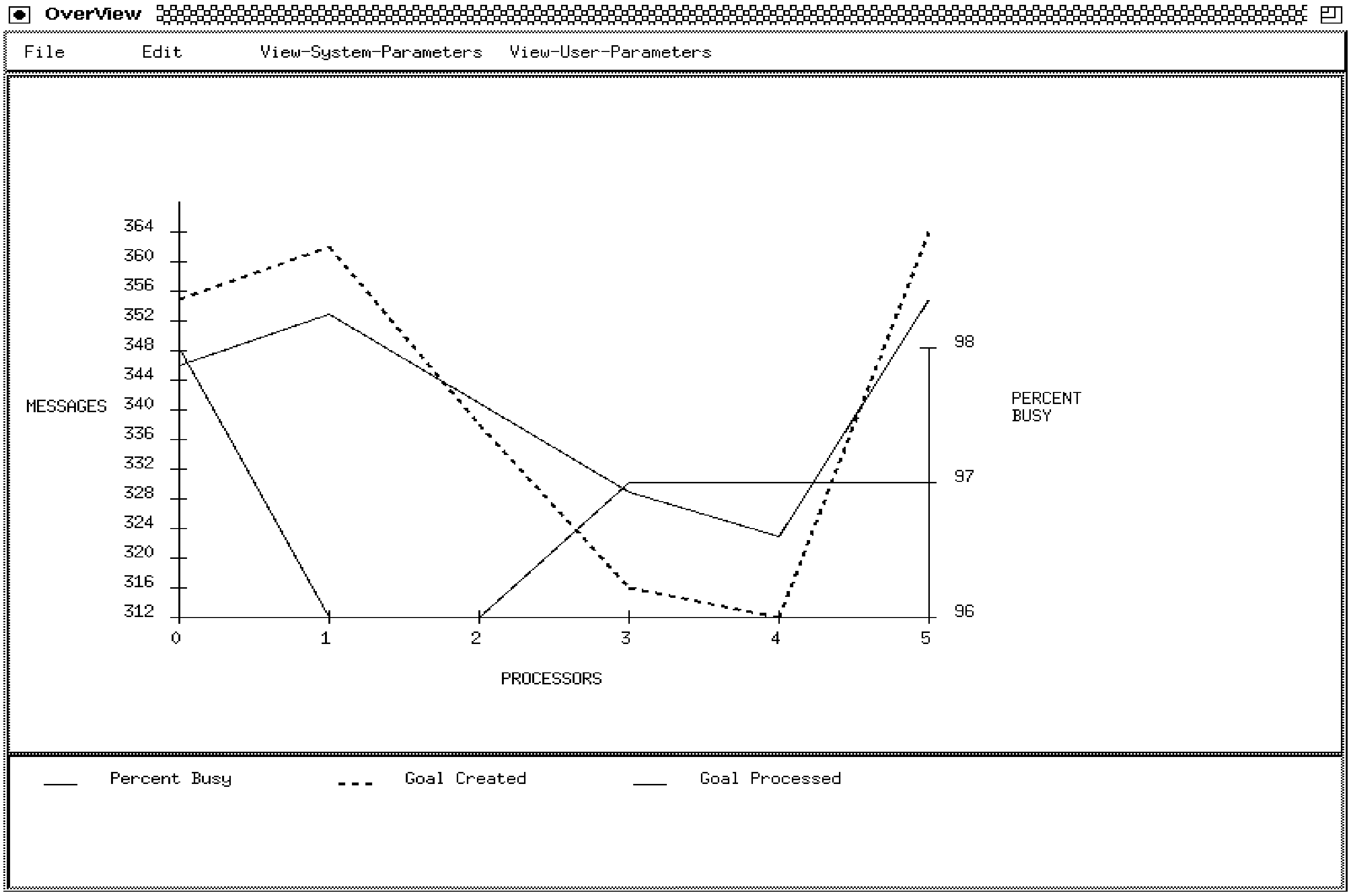

Figure 2: **This figure shows an Overview.**

The first set of views are called the *overview* views. Figure 2 shows an example overview. In an overview, the user can select the attributes to be viewed. The values of the selected program attributes over the selected *stage-sets* and *processor-sets* are shown as one screenful of data for an *overview* of the program execution. There are two types of overviews. In the first, the chosen program attribute is summed up over all processors in the processor-set for each stage, and this aggregate is displayed as a function of the stage number (so, the function displayed is $D_a(s) = \sum_p F_a(s,p)$). In the second, the chosen program attribute is summed up over all stages in the stage-set for each processor, and this aggregate is displayed as a function of the processor number (so, the function displayed is $D_a(p) = \sum_s F_a(s,p)$). These views are very useful in selecting interesting stages and processors for a more detailed view. In either of these two overviews, the user can interactively change the attributes to be selected, and the stage and processor sets for the views. By changing the stage and processor sets the user can selectively look at specific periods of program execution, or only at specific processors. Since the views are limited to a single screen full of information there is a consequent limit on the detail that can be presented in the view. This is the motivation for the second type of views.

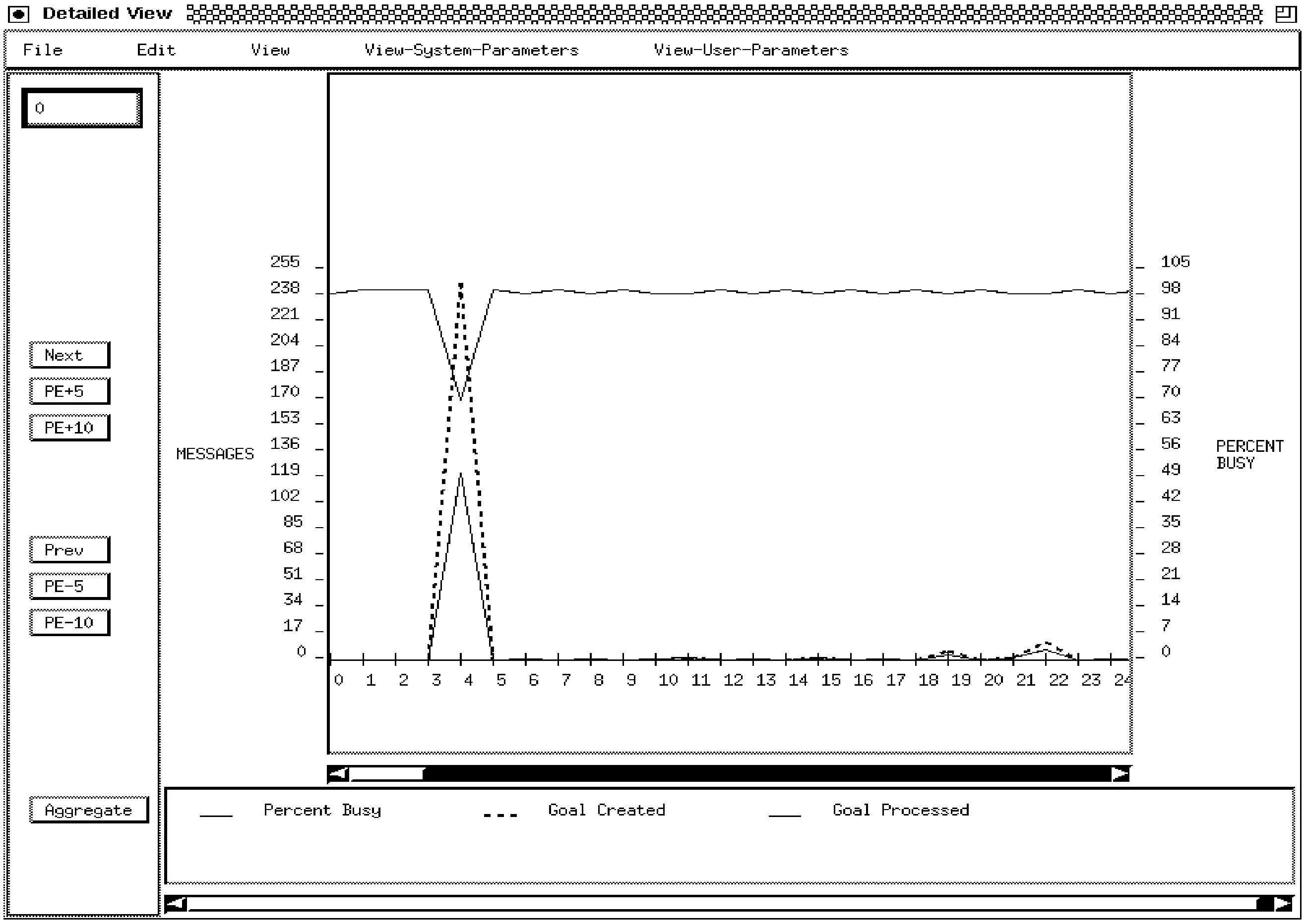

Figure 3: **This figure shows what the detailed view looks like. The timestep, the number of stages and the number of processors appear at the bottom.**

The second type of views are called *detailed* views. Figure 3 shows an example detailed view. There are two detailed views:

1. View showing the selected program attributes as a function of the processor-set for a selected set ($F_a(s_0, p)$), or aggregated over all stages in the the given stage-set ($\sum_s F_a(s, p)$).
2. View showing the selected program attributes as a function of the stage-set for a selected processor ($F_a(s, p_0)$), or aggregated over all processors in the the given processor-set ($\sum_p F_a(s, p)$).

The **Aggregate** button in the command menu in the *detailed* views allows the user to view performance data as an aggregate over all processors and stages in the processor and stage sets respectively.

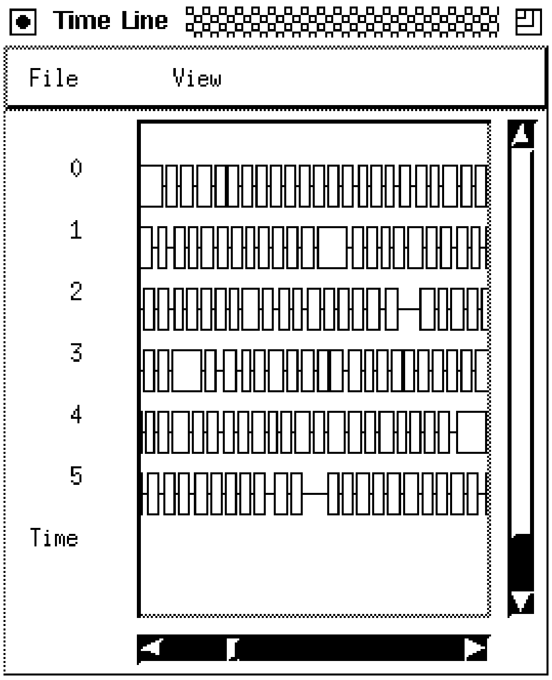

Figure 4: **This figure shows the timeline window.**

A timeline (depicting processing of various entry points on processors) can be seen for different pairs of points in the detailed view. One can view a timeline of activities by selecting the *See New Timeline* button in the *View* menu, and then choosing two points of interest by clicking on them. Figure 4 shows an example timeline showing events for processors 0-5 for a selected stage. The type of events in a timeline can be determined using the legend displayed at the top-level.

In both the *overview* and the *detailed view*, the user can view system attributes using the *View-System-Parameters* menu and the user attributes using the *View-User-Parameters* menu.

The third type of views are *animation* views. Figure 5 shows an example of an animation view,

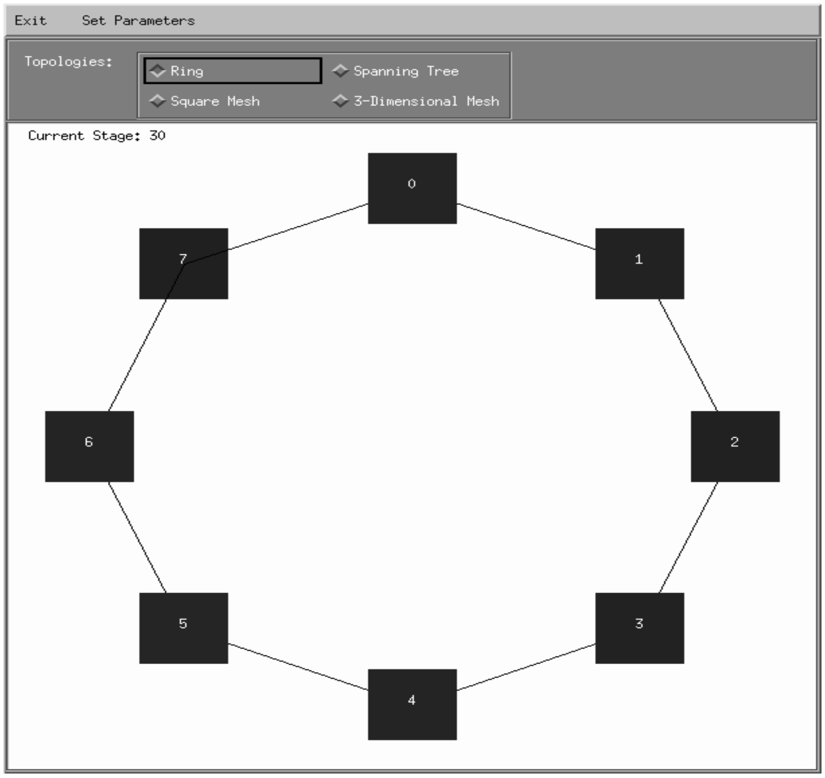

Figure 5: **This figure shows the animation window. The user can select to view the animation in one of the four modes — grid, ring, spanning tree and three-dimensional mesh**

where the ring topology has been selected. In an animation view each processor's state is represented as a color — blue for busy and red for idle, and intermediate shades between blue and red representing intermediate levels of busy and idle. The processors can be arranged in different topologies, and the user can understand from the colors of processors through the various stages of program execution the structure of the program. Currently, there are four available topologies — spanning tree, square mesh, ring and three-dimensional mesh. Here again, the user can interactively alter the topology of processor connections, length of a stage, and the stage and processor periods. In the case of the three dimensional mesh view, the user can specify the lengths of the three dimensions.

References

[1] A. B. Sinha and L. V. Kale. A load balancing strategy for prioritized execution of tasks. In *International Parallel Processing Symposium*, April 1993.

[2] L. V. Kale and A. B. Sinha. Projections: a preliminary performance tool for charm. In *International Parallel Processing Symposium*, April 1993.

[3] A. B. Sinha and L. V. Kale. A framework for intelligent performance feedback. submitted, August 1994.