Using `host-control`

Robert Olson

Technical Memo ANL/MCS-TM-154
Version 1.2: November 1, 1991

The utility `host-control` conveniently configures a set of machines for use with the networked version of PCN. It enables the user to specify exactly how PCN is to be run on each node, and it allows for easy control of runaway PCN nodes.

This document provides the information needed for a new user to get started with `host-control`. It also provides the advanced user with the information needed to get around more difficult problems in a networked PCN configuration.

Contents

1 The `host-control` utility

The utility `host-control` is an interactive Perl [1] script that manages the selection of hosts on which to run PCN. It eliminates the need to write a startup file, and makes PCN startup faster. In order to use `host-control`, it is necessary for Perl to be installed on your system and on each remote system on which you wish to run PCN processes. The Perl software is freely available from many anonymous FTP sites.

The `host-control` utility works by running a daemon program called `node-control` on each machine on which net-PCN nodes are to be started. The `node-control` daemons communicate with the `host-control` program via TCP socket connections. (We refer to `host-control` variously as a program and as a daemon since it acts as both: the user is always able to type commands at the `host-control` command line; however, `host-control` is in addition listening for messages from `node-control` daemons.)

The actual net-PCN node processes are created by the `node-control` daemons and are children of the `node-control` daemons. As such, the `node-control` daemons can monitor the execution status of the net-PCN nodes, collect their output, and kill them if necessary.

2 Use of `host-control`

To run `host-control`, first ensure that your environment is set up such that the `PATH` environment variable includes the PCN bin directory (e.g., `/usr/local/pcn/bin`). This is needed so that the `node-control` script can be executed. Then type `host-control` at the shell prompt. It will respond with a banner and prompt:

> % host-control
>
> PCN Host control version 2.5
>
> Listening on 1842
> HC>

Now `host-control` is ready to accept commands. The number that is printed is the TCP port number on which `host-control` listens to accept connections from new `node-control` daemons. This number can also be obtained by using the `listen-port` command.

The utility `host-control` accepts a large number of commands. To obtain a complete command list, use the `help` command with no arguments. For help on a particular command, use `help command-name`. For information on the usage of a given command, use `usage command-name`. For example:

> HC> **help**
> Available comands:

! abbrev alias apropos arch cd connect-to-node conns define-group
delete-group dirs disable dump-output edit enable enabled eval
flush-output help hosts kill kill-pcn kj list-group list-groups
listen-port load-config load-group node-rusage parray passoc popd
ps pushd pwd quit reconnect restart rset save-group set
start-group start-node status unalias undefine-group unset uptime
usage wait-for-pending write-nodes
Other help:
environment host-variables node-variables variables
HC> **help save-group**
Save the given group to the file
/Net/auriga/auriga5/olson/.pcn_control/groups/groupname.def.
Usage:
save-group groupname
HC> **usage save-group**
Usage:
save-group groupname

2.1 The `.pcn_control` Database

For a `node-control` daemon to connect to the `host-control` daemon, it must know the hostname and TCP port number on which the `host-control` daemon is listening. This information is stored in a database in the `host-control` user's home directory under the directory `.pcn_control`. Under this directory there are subdirectories `nodes` and `hosts`. For each `host-control` process (there can be several if desired, although not all operations are defined for multiple `host-control` daemons) running on host `hc-host` there is a file `hosts/hc-host` containing the TCP port number on which the `host-control` daemon is listening. Similarly, the file `nodes/nc-host` contains the TCP port on which the `node-control` daemon running on host `nc-host` is listening.

The database directory can also contain a subdirectory `groups` which contains saved host group definitions. A *host group* is simply a list of hosts, defined using the `host-control define-group` command. Host groups are used in starting `node-control` daemons when `rsh` fails. See §2.2.2.

2.2 Starting Nodes

There are several ways to start `node-control` daemons. The most straightforward is the `start-node` command. For each `start-node` argument `arg`, `host-control` first attempts to interpret `arg` as a valid Internet hostname (using `gethostbyname()`). If `arg` is not a valid hostname and a file named `arg` exists, this file is assumed to contain a list of host names, one per line. See §3.1 for some examples of the `start-node` command.

The `host-control` utility first invokes `rsh` to start `node-control` daemons. For this method to succeed, the remote machine must grant permission to the machine on which `host-control` is running to start processes. If this permission is not granted, the `node-control` daemons must be started by hand.

2.2.1 Starting nodes manually

Assume that the host on which the `host-control` daemon is running is called `hc-host` and the host on which we desire to start a `node-control` daemon is called `nc-host`.

If accounts used on `hc-host` and `nc-host` have a common home directory (so that the `.pcn_control` database is available to the `node-control` daemon), a node can be started by simply typing

> % **node-control &**

which will display a banner such as

> PCN Node control version 2.4
>
> Listening on port 1879 on host donner.mcs.anl.gov
> Got host on donner.mcs.anl.gov/1874

If there is no mention of connections to any hosts, then `node-control` was not able to find a host to connect to. Two actions may be taken in this case. The failed `node-control` can be killed and reinvoked as

> % **node-control -h hc-host -p hc-port &**

where `hc-port` is the TCP port on which `host-control` is listening. Alternatively, `host-control` can be told where to find the `node-control` daemon by using the `host-control` command `connect-to-node nc-host nc-port`, where `nc-port` is the TCP port on which `node-control` is listening:

> HC> **connect-to-node donner 1874**
> Connected to donner.mcs.anl.gov/1874

2.2.2 Starting nodes using host groups

A common situation arises that causes problems in starting large groups of net-PCN nodes. A user may have accounts at several computing centers, each of which has several machines on which he wishes to run PCN. Each group of machines grants each other machine in the group permission for `rsh`, but no machine grants `rsh` permission for any machine outside of the group. The solution to this problem is to define a *host group* for each group of machines. A host group is simply a named list of *hostnames*. To define a group, use the `define-group` command:

> HC> **define-group anl-mcs**
> Enter group members and blank line to end
> define-group > **donner**
> Adding donner.mcs.anl.gov
> define-group > **ezra**
> Adding ezra.mcs.anl.gov
> define-group > **skeeve**
> Adding skeeve.mcs.anl.gov
> define-group >

After a group is defined, it can be saved in the `.pcn_control` database by using the `save-group groupname` command.

When the `start-node hostname` command fails because permission for the `rsh` command has not been granted, `host-control` attempts to find a host group in which `hostname` is a member. If it succeeds, it then determines whether another machine in the group is already running a `node-control` daemon. If this is the case, `host-control` requests the `node-control` daemon running in the group to start a `node-control` daemon on `hostname`. Presumably, this method will succeed, as the host group should be defined such that each machine in the group can `rsh` to each other machine in the group.

Note that a `node-control` daemon must be running on at least one machine in the host group for this method to work. This implies that a `node-control` daemon must be started by hand on one machine in each group for this method to work.

2.3 Killing Nodes

After a PCN session (encompassing possibly many PCN runs) is complete, the user will desire to kill off the node daemons. There are two ways to do this. If a node daemon sits idle for longer than a certain length of time (which defaults to one hour), the daemon will time out and exit. Alternatively, the user can use the `host-control kill` command:

> HC> **kill all**
> Remote command kill on locke.mcs.anl.gov:
> Node-control on locke.mcs.anl.gov/1726 exiting at 17:28:20 10/07/91
> HC>
> Disconnecting node locke.mcs.anl.gov

3 PCN and `host-control`

Once `node-control` daemons are running on the desired hosts, PCN can be started. The `-start-nodes` flag to PCN causes PCN to connect to the `host-control` daemon and request it to start PCN node processes. The number of processes started depends on the `-n` flag to PCN. If no `-n` flag is provided, PCN will start as many node processes as there are node daemons running. If a `-n` *nprocs* flag is provided, PCN will attempt to start $nprocs - 1$ node processes. If fewer than *nprocs* node daemons are running, more than one PCN node may be started on a given machine.

3.1 Examples

First we look at some straightforward `host-control` usage. The first example demonstrates starting `host-control`, then starting nodes on three hosts (ezra, skeeve, and lutra).

Example 3.1:

```
% host-control

PCN Host control version 2.5

Listening on 1842
HC> start-node ezra skeeve lutra
Starting node on ezra.mcs.anl.gov
Starting node on skeeve.mcs.anl.gov
Starting node on lutra.mcs.anl.gov
HC>
Got connection from host skeeve.mcs.anl.gov
HC>
Node control version 2.5 running on skeeve.mcs.anl.gov
HC>
Got connection from host ezra.mcs.anl.gov
HC>
Node control version 2.5 running on ezra.mcs.anl.gov
HC>
Got connection from host lutra.mcs.anl.gov
HC>
Node control version 2.5 running on lutra.mcs.anl.gov
```

The second example demonstrates querying `host-control` for the current list of connections and the architectures of all connected hosts.

Example 3.2:

```
HC> conns
Current connections (3):
ezra.mcs.anl.gov/3922
lutra.mcs.anl.gov/1032
skeeve.mcs.anl.gov/1120
HC> arch all
Remote command arch on lutra.mcs.anl.gov:
sun4
Remote command arch on skeeve.mcs.anl.gov:
sun4
Remote command arch on ezra.mcs.anl.gov:
sun4
```

Note that `host-control` may write output even when no commands have been entered. This is a manifestation of the fact that commands may start a chain of events that executes in the background. For instance, `start-node` starts another process that performs the remote shell to the remote host. When the node daemon starts running, we get one message from the subprocess with the `node-control` startup banner (this is where the version information for `node-control` originates), and another message noting that a new node daemon has connected with the host.

The third example demonstrates starting PCN using `host-control`. The nodes started above by `host-control` are still running, and we see that one PCN node has been started on each for a total of four nodes.

Example 3.3:

```
% pcn -start-nodes
PCN: Version 1.1 beta4; 4 nodes, 512k heap.
(See the file: /home/olson/PCN/v1.1/install/DISCLAIMER)
*
exit(0)
%
```

When PCN nodes are started, `host-control` prints a number of diagnostic messages:

```
HC>
Got connection from host donner.mcs.anl.gov
Received PCN startup request from donner.mcs.anl.gov
HC>
Got request to start -2 PCN nodes

Started node on lutra.mcs.anl.gov

Started node on skeeve.mcs.anl.gov

Started node on ezra.mcs.anl.gov

Started 3 of 3

Disconnecting cmd donner.mcs.anl.gov:0
HC>
Output from node lutra.mcs.anl.gov:

PCN node exited on lutra.mcs.anl.gov

Output from node skeeve.mcs.anl.gov:

PCN node exited on skeeve.mcs.anl.gov
HC>
Output from node ezra.mcs.anl.gov:

PCN node exited on ezra.mcs.anl.gov
```

The request for starting -2 nodes means that `host-control` has been requested to start as many PCN nodes as there are `node-control` daemons.

Table 1: Environment variables

Environment variable	Default value	Meaning
PCN_CONTROL_DIR	$HOME/.pcn_control	Directory used for the database of hosts, nodes, and groups
EDITOR	/usr/ucb/vi	Editor invoked by the `edit` command

3.2 Process Control

One can display the status of PCN processes on the nodes by using the command `status hostname`. This will display the process ids of the PCN processes running on `hostname`. If necessary, one can kill a PCN process by using the `kill-pcn hostname pid` command.

4 Other `host-control` Features

The utility `host-control` has a number of other features that allow one to tailor the PCN computing environment more exactly.

4.1 Variables

Three flavors of variable affect `host-control` execution: environment variables, host variables, and node variables.

4.1.1 Environment variables

The `host-control` utility recognizes several shell environment variables. From the C shell, one can set environment variables with the `setenv` command:

% **setenv EDITOR /usr/ucb/vi**

Environment variables are set from the Bourne shell as follows:

$ **EDITOR=/usr/ucb/vi**
$ **export EDITOR**

Table 1 lists the environment variables that `host-control` recognizes.

4.1.2 Host variables

Host variables affect the execution of the `host-control` process only. Table 2 lists the currently implemented host variables and their default values. Host variables are set with the `set variable value` command. If `value` is not specified, the current value of `variable` is displayed. If neither `variable` nor `value` is specified, the values of all variables are displayed. The `unset variable` command will remove the definition for `variable`.

Table 2: Host variables

Host Variable	Value	Action
startnode-query	**on**	Query the user before starting node daemon
	off	Do not query user
rsh-timeout	**30**	Time in seconds before a start-node times out
show-output	**yes**	Show all node output as it comes in
	no	Do not show node output
collect-output	**yes**	Save all node output in an internal buffer
	no	Do not save node output
kill-nodes-on-exit	**prompt**	Prompt user to kill `node-control` daemons upon exiting `host-control`
	yes	Kill `node-control` daemons upon exiting `host-control`without prompting user
	no	Do not kill `node-control` daemons upon exiting `host-control`
editor	**/usr/ucb/vi**	Editor invoked by the `edit` command
window-width	**70**	Try to wrap to this column on some `help` output

Table 3: Node variables

Node Variable	Value	Action
pcn	Default supplied by PCN host	Executable to start on this node
pcn-dir	Default supplied by PCN host	Directory in which to start PCN node
enabled	**yes**	Node can run PCN node processes
	no	Node cannot run PCN node processes

4.1.3 Node variables

Node variables affect the execution of the `node-control` daemons. Each node has a distinct set of node variables. Table 3 lists the currently implemented node variables and their default values. Node variables are set with the `rset host variable value` command. If `host` is `all`, all running nodes are affected. If `value` is not specified, the current value of `variable` is displayed. If neither `variable` nor `value` is specified, the values of all variables on the specified host are displayed.

The value of the `enabled` variable can be set using the `enable` and `disable` commands. The `enabled` command lists all currently enabled hosts.

4.2 Running on Many Machines

It can be difficult to manage the execution of PCN on a diverse set of machines. The PCN nodes must be able to find the compiled PCN code; they may expect to execute in a certain directory; or they may need to execute a custom PCN executable. It is possible to specify each of these parameters to `host-control` by using the `pcn` and `pcn-dir` node variables. For a simple network of machines, one can use the `rset` command to manually set these variables for each host. However, this procedure becomes tedious and must be repeated each time the node daemons are restarted.

The utility `host-control` provides a facility that automates the procedure. A *host configuration* file defines the values for node variables on a per-architecture, per-host-group (see §2.2.2), or per-node basis. Hence, to handle a large set of machines, one would define a host configuration file that defines the `pcn` and `pcn-dir` node variables for the appropriate architectures. For example, we know that our custom PCN executable is in /home/olson/<architecture>/mypcn and that we want to run PCN in /home/olson/proj. However, on the machine named `faraway.uni.edu` the directories are quite different. The configuration file would hence look like the following:

```
arch: sun4
pcn /home/olson/sun4/mypcn
pcn-dir /home/olson/proj

arch: dec5000
pcn /home/olson/dec5000/mypcn
pcn-dir /home/olson/proj

host: faraway.uni.edu
pcn /usr/guest/mypcn
pcn-dir /usr/guest/runit
```

Configuration files are loaded by using the `host-control load-config` command.

4.3 Seeing Node Output

By default, all node output is displayed by `host-control`. One can set the `show-output` host variable to `no` to disable the display. If the `collect-output` host variable is set to `yes`, `host-control` saves all node output (even if it does not display it). To see the saved output, type `dump-output hostlist`. This will dump the output from the specified hosts (or all hosts if none is specified). To flush the saved output, use the `flush-output hostlist` command.

4.4 Getting Help

The command `help command` prints a help message for the given command. If you are not sure which command to use, the `apropos string` command prints a list of commands in which the text `string` is found. `string` can be a regular expression. For the exact usage of a command, use the command `usage`.

4.5 Shell-like Commands

The `host-control` utility provides several features for interacting with the shell and environment:

- Each command that expects a filename performs tilde expansion on the filename.
- It is possible to change the current working directory of the `host-control` process by using the `cd`. The `host-control` utility also maintains a directory stack à la csh, defining the `pushd`, `popd`, and `dirs` commands.
- One can define a command alias using the `alias` command.
- One can run a shell command using the `!` command.
- One can edit a file using the `edit file` command. The editor used is determined by the `editor` host variable.

4.6 Startup File for `host-control`

When `host-control` begins execution, it reads the contents of the file `.host-control.rc` in the user's home directory and executes each line as a `host-control` command.

4.7 Exiting `host-control`

To exit `host-control`, use the `exit` or `quit` commands, or type the shell end-of-file character (usually control-D).

5 Summary

The `host-control` system provides a convenient framework in which to run the networked version of PCN on a large number of machines. It also provides more precise control of the PCN execution environment, something that is often difficult in the networked environment.

References

[1] Larry Wall and Randal L. Schwartz. *Programming perl*. O'Reilly & Associates, Inc., Sebastopol, California, 1990.