Fairisle and Wanda
Start of Day and Management Issues

Simon Crosby

14th March 1994

1 Introduction

This document tries to set out some of the important issues regarding the maintenance of the Fairisle network and the Wanda experimental development environment. As new members join the systems group it is important to ensure that they understand how the current experimental environment works, so that they can properly use and maintain it.

This document does not claim to be authoritative. It merely contains my knowledge of the (admittedly complicated) set of procedures which are used by the current research students and research assistants to keep the experimental world running. Doubtless there will be errors in this document. If you are in doubt, please consult your nearest guru, who will put you on the correct track.

2 Basics

The wanda world is a network of experimental machines of various types which are used for systems research and networking research within the laboratory. Some of these machines are service machines and should not be used for research work. Examples include the Fairisle switches which are used to carry service (i.e. real data) traffic.

When a research machine is reset, it first determines its own identity using a process known as rarping. It reads a unique identifier from some part of its hardware, such as an Ethernet address or a Fairisle Serial Number (FSN) and sends a request over the network, requesting a rarp server to identify it and assign it an msnl identifier. The process of rarping is described in section 4 below.

Having obtained its address, the machine can boot. To do this it loads a kernel from a unix file server. In order for the booting process to work, the machine and the server which supplies the kernel (the boot server) must talk a common protocol, namely the Wanda Booting Protocol, which runs on msnl. A boot client on the booting machine requests the kernel from a boot server running on a unix workstation. This server, and other daemons, are described in section 3.2 below. They must be running in order for the experimental world to operate.

A user of a wanda machine or Fairisle switch may interact with the machine by using either `xwcons` or `xscons`. These programs allow the user to observe the console output from the machine or switch. The mechanisms by which the console output are brought to the user are described in section 6 below.

3 The Unix Daemons

Several software daemons are used to maintain the experimental system. These are described in more detail below. Note that the unix daemons run with the user identifier "wanda" and must be started as user wanda. If you do not have access to the wanda password, you cannot start the unix daemons.

3.1 The Modified Trader

The trader is based on the Ansa 3.1 trader, with an extended interface which allows it to be used as a general purpose database. Each host in the wanda world has an entry in the trader. This describes the properties of the host in terms of its physical devices etc, and specifies several properties which are used by other Unix daemons to boot, reset, and gather console output from the machine. The entries common to all hosts are:

- Aliases: A list of strings which are alternative names for the host
- BootFrom: This field indicates whether the machine uses serial line booting or mop booting, either of which require special action on the part of `xwcons`.
- BootTime: The time that the host was last booted.
- BootTimes: The number of times that the machine has been booted.
- CPU: Information regarding the CPU in the host.
- Console: Information for the use of a console program such as xscons or xwcons, which states which console line the host uses, and what protocol to use when connecting to it. For example, an entry "tel:maidenhead9-r" means "use telnet to contact the telnet service on maidenhead9-r".
- DFile: The default kernel file which should be booted into the machine when it is not being used for experimental research.
- DHost: The default hostinfo (environment variables etc) to supply to the host when it boots, when not being used for user research.
- InterfaceId: This field is inserted by the trader, and is a unique identifier for the trader entry for this host. Consult the Ansa manual for details on interface references. This field should not be modified under any circumstances. See the manual page for `tco, tci` for further details.
- MSNL: The primary msnl interface identifier for the host.
- MSNLAliases: Addresses of additional msnl interfaces for the host.
- MachType: The type of machine e.g. Fairisle-Switch.
- Memory: The amount of memory the host has.
- Name: The primary name of the host.
- NetName: The name of the host as used in the network – frequently the same as Name.

- Peripherals: A list of any peripheral devices attached to the host.
- ResetRelay: For all hosts attached to the reset server, this is the reset relay number which is attached to the host. The reset server uses the reset relay for the host to hard reset the host when it crashes and cannot be restarted in software.
- Type: The type of interface (entry) which this entry represents in the trader. For all wanda hosts this has the value “MachInfo”.
- UFile: The user specified boot kernel for the host. Used only if the host is currently locked by an xscons or xwcons session.
- UHost: The user specified hostinfo supplied to the machine when it boots.
- WConsId: This is the interface reference of the console program which is currently or which last locked this host and connected to the console output device. It is used by xscons, xwcons and the boot server.

Various other entries exist. These are used in a host-type specific way. For example, Fairisle switches have an entry “SwitchSize” which indicates the number of ports on the switch. This is used by xscons when sizing the window to display console output for the switch ports.

3.1.1 Applications which use the trader

Several application programs use trader as a database. These include:

- `xwcons`: A console program which displays the output a wanda machine in a window on the X display. It presents the user with an interface which allows the user to log and scroll through the console output, reset the machine, and view many of the trader entries for the machine. When xwcons starts it locks the trader entry for the host by filling in the field “ WConsId” with its own interface reference which is also exported to the trader. This “locks” the machine for the exclusive use of the person running xwcons, and ensures that when the machine boots it will use the “UFile” and “UHost” entries for the boot kernel.
- `xscons`: A console program for a Fairisle switch. This is very similar to xwcons, except that it uses the switch size to determine how many console windows to create. There is one window per port on the Fairisle switch. Xscons allows the user to soft reset and hard reset the switch. Soft resets are done using the wreset mechanism (see below). Hard resets make use of the reset server and the reset relay for the switch. Note that hard resets of a Fairisle switch reset all ports on that switch.
- `bootset`: This application modifies the “UFile” and “UHost” entries for the host, or the ‘DFile” and “DHost” entries if the appropriate command line option is given. Note that the default boot entries are set by the system managers to sensible values. Modifying these is strongly discouraged.
- `bootinfo`: This application retrieves the booting information for the host, including the number of times that the boot server has booted the host since the kernel entry was modified. It also prints the last boot time of the host.
- `wwho`: Lists all known xscons and xwcons sessions in existence.

- `wcheck`: Allows the user to check that a specified xscons or xwcons session is still active.
- `wunlock`: Removes the lock on a particular host. This is useful when the xscons or xwcons session has been killed without removing its interface reference from the trader.
- `wreset`: Allows the user to reset, either in software, or by physically resetting a host.
- `tco` and `tci`: These are applications which provide a friendly interface to the trader. Manual pages are available, and the user should take special note of the "BUGS" section!

In addition most of the servers comprising the unix daemons use the trader as well. Manual pages exist for all of the above applications.

3.2 The Boot Server

The boot server listens for boot requests from wanda machines. It listens for these requests at a well known address (consult the msnl services document for details). When a wanda machine or Fairisle port boots, it contacts the boot server whose msnl address is given to it in a rarp reply message by a rarp daemon. When it receives a boot request the boot server queries the trader to find out which kernel should be booted into the machine. If the machine is currently locked (i.e. somebody has an xscons or xwcons session on the machine) then the user booting information is used. Otherwise the default booting information is used. The boot server creates an image from the supplied kernel and hostinfo, and sends it to the booting machine using the boot protocol. Once the machine has loaded its new image, it begins to execute the new kernel.

3.3 The Grim Reaper

The Grim Reaper is an Ansa 3.1 service which removes old or out of date entries from the trader when the processes owning the interface references exported to the trader and the notification service are killed or exit. For details of this service see the Ansa manual.

3.4 The Request Server

The request server provides a Unix file service to the Ansa clients on Wanda. It has been largely replaced by the Wanda Dumb File System.

3.5 MSNL Internet Daemon

The MSNL Internet Daemon is a unix daemon which responds to msnl connection requests for the msnl echo and msnl tracing facilities, `mecho` and `mtrace`. It provides a mechanism for testing the connectivity of the msnl network and the routes used to reach a particular host through the network. For more information see the manual page.

3.6 MSNL Name service

The msnl name service is used to resolve name queries of the style of `gethostbyname(3n)`, for msnl addresses. For both unix and wanda, library functions called `getmsnlhostbyname()` and `getmsnlhostbyaddr()` (see the man pages) are available to enable application programs to resolve name queries. The name service uses the trader for some of its information, and queries the DNS for other information. See the msnl name service document elsewhere in this collection.

3.7 Wanda Dumb File System

The wanda dumb file system and wanda dumb directory service provide access to the unix file space from wanda machines. The interface to these services is the standard BSD file interface, and the underlying communication mechanism is a RPC protocol which uses msrpc to read and write blocks of a file. This service currently operates with msrpc rather than msrpc2. This should be modified in the near future.

3.8 Starting the Unix Daemons

At the time of writing, the unix daemons are started by a script which currently resides in `/usr/groups/wanda/info/unixdaemons` . This script must be executed as user wanda, and on the designated wanda service machine. Currently the default services run on the machine aspall, although many Fairisle switches boot from experimental boot services on other machines. This is likely to change in the near future, when the service machine aspall is upgraded to run the new MSNL code. At that time the wanda daemons will (should) get started automatically when the machine boots.

4 Rarping and Rarp Servers

When a wanda machine is reset, it needs to find out who it is, and how to boot. It does so using the rarp protocol. The format of rarp messages is described elsewhere in this collection. When a machine sends a rarp request it places in the message a unique identifier which it has read from its hardware. For some wanda machines this will be their Ethernet address, whilst for Fairisle port controllers this is their Fairisle Serial Number. This is a unique number which identifies each port controller. A rarp server is a server which, on receiving a rarp request, returns to the requester a message indicating its msnl address, and the msnl address of the host from which the requester should boot. It also supplies its own msnl address. The requester, on receiving a reply, assigns its msnl address from the information provided and then attempts to boot from the boot server at the supplied address.

Rarp servers are available for both unix and wanda. The msnl connection management daemon (see elsewhere in this collection) runs a rarp service if it is set up to do so. In addition, the master port on each Fairisle switch runs, by default, a rarp server which will respond to rarp requests from the other ports on the switch, when they boot. By convention a rarp server should only run on a wanda machine if that machine is also a gateway, and the gateway can route boot requests from the rarping machine to the boot server.

Each rarp server, on starting, loads a rarp table. For wanda, the rarp server uses a rarp table which is supplied in its hostinfo. For Fairisle switches, the rarp table is automatically constructed by the switch configuration tool `sc` (see below). For unix rarp servers, the rarp table is supplied to the msnl-master when it initialises. The rarp table specifies for each type of rarp, for each registered host, the msnl address of that host and the address of the boot server for that host. Two types of rarp requests are recognized: Ethernet-based rarp requests and FSN-based. For a Fairisle switch the rarp daemon responds with an address not based on the FSN number of the requester, but on the position of the requester in the switch. Likewise for LINKRARP the reply is based on the topology of the network, not the serial number of the board. The rarper has no idea that this is occurring but always puts its known information in place.

5 Fairisle Switches

The Fairisle switches form a large ATM LAN which is used for experimental networking and other projects. Currently they are being used for traffic control experiments for BT. At the time of writing there are 10 switches. Two of these switches are fpc2 switches and carry service traffic, linking file servers at the computer laboratory. Other hosts and sources of traffic will be connected to the network over the year. For details the design of the Fairisle switches, see other documents in this collection.

Each switch has its own entry in the trader. This permits applications such as xscons to obtain the information they require to access the switch. In addition, each port of a Fairisle switch has its own entry in the trader. This is because each port is an independent processor which can be configured to boot with a specific kernel. Thus each switch has 17 entries in the trader - one for the switch as a whole (and with entry name *switchname*-switch, where *switchname* is the name of the switch) and 1 for each port, whether that port is physically present or not. When `xscons` locks a switch it also locks all ports belonging to that switch.

5.1 Default Booting

Each Fairisle switch has a default boot configuration. This is defined by its default boot kernel. The default boot kernel is the kernel which is booted into the master port of the switch when the switch is reset. This can be determined by examining the trader entry for that switch.

The default boot kernel for the master port is kept in
`/usr/groups/wanda/boot/`name`.boot` where "name" is the switch name. The default kernel is built using the tool `sc` and `wboot`, described below.

5.2 User Booting

When a Fairisle switch is used for experimental work, the user can boot the ports of the switch with any kernel. To do this, it is necessary to lock the switch using `xscons`, and then to use `bootset` to set the new kernel and hostinfo for each port. In addition, the master port must run a gateway and rarp daemon, to enable the rest of the switch to access the boot server.

5.3 The Switch Configuration Tool

The switch configuration tool `sc` is used to define the topology of the Fairisle network and to automatically generate the routeing and hostinfo tables for Fairisle switches. It currently has a rather primitive interface which requires the user to hand code a file describing the connectivity between switches and between switches and hosts. This interface should be integrated with the other network management tools to ensure that there is only one source of information used when defining the network. `sc` Defines the topology of the whole network. As such, it should be used with great care and only by one person who understands the network. It is not a tool for use by users who simply want to experiment with Fairisle switches. It is vital that the switches all have a consistent view of the network topology otherwise routeing could become a nightmare. When the automatic topology discovery service has been completed, this will become easier.

`sc` Generates, given the connections between switches, and between switches and hosts, routeing tables for all Fairisle ports in the network. These take the form of files with the suffix `.auto`, and they are by default installed as part of the make process for `sc` in `/usr/groups/wanda/boot/hostinfo`. When the hostinfo for a particular port *portname* is executed by the boot server or `wbtool` by running it as a perl script, the *portname*`.auto` file is included in the hostinfo and the routeing information is made accessible to the booting machine in the form of environment variables which can be read by the kernel.

`sc` Also generates environment variables which permit Fairisle ports to use the network console (see below), and rarp tables for the switch-based rarp server on the master port. This form of rarp is called positional rarping, and is specified fully elsewhere in this collection.

5.4 The Switch Boot Server

When a switch is booting its default configuration, it makes sense to reduce the network load of booting traffic, and boot the same kernel into each port. This is achieved by using the wanda boot server `wboot`. This is a wanda application which, when spliced onto a wanda kernel, will respond to boot requests from other ports on the same switch. A man page for both `sc` and `wboot` exists. Wboot is an application created by the tool `wbtool`, which builds the data area for the boot server from the image of the kernel to be booted into the ports of a switch and the hostinfo for each of the ports. When `wboot` receives a boot request from a port on its switch it will boot the port with its kernel and the correct hostinfo. Thus, by booting the master port with a kernel, gateway, rarp daemon, and wboot, the entire switch will be automatically booted by the master port. This means that switches boot much faster and that the boot server is not subject to as much load.

Note that when wboot is being used, if the hostinfo is modified then the master port kernel must be rebuilt, because the wboot image contains the hostinfo for the other ports in the switch as a static data area.

5.5 Routeing Tables

The wanda kernel and the user space gateway use routeing tables supplied as part of the hostinfo to the kernel when it boots. Each routeing table entry consists of a string of the form: `MSNLROUTE-`*pod-name*=*comparator*-*mask*, where *pod-name* is the name of the msnl interface through which the routeing will take place (for example: fair-transmission), *comparator* is a 4 octet value conventionally expressed using the Internet-style dotted notation (e.g. 128.232.133.32), and *mask* is a 4 octet value conventionally expressed using the Internet-style dotted notation (e.g. 255.255.255.248). Multiple comparator and mask pairs may be given provided they are separated by a comma. The algorithm used to select a route for a connection request for a given destination address A is as follows:

1. Sequentially process the list of routeing table entries, from most to least specific, until done. For each entry in turn:
2. Form the bitwise AND of A and the *mask* for `MSNLROUTE-`*pod-name*
3. Compare the result to *comparator*
4. If they are equal, then select that pod, and routeing is complete
5. Otherwise, go back to step 1.

5.6 Trader Entries

Each switch is represented in the trader by an entry for each port, and an entry for the switch as a whole. For example, for the switch manor, there is an entry for each port from `manor00` to `manor15` and also an entry entitled `manor-switch`. The port entries are used to specify user booting information, the msnl address of the port, and to lock the port for use by the owner of the current xscons session. The switch entry is used to obtain information about the switch as a whole, such as the number of ports, the master port number and the console output port address. At the time of writing the switch-specific entries in the trader contain the following additional fields:

- MasterPort: The port number of the master port for the switch.
- SwitchSize: The number of ports on the switch.

6 Console Output

Each trader entry for a wanda machine contains the address to which console output from the wanda kernel is directed, when it is written from the serial port on the host. Traditionally each wanda host has its own serial line which `xwcons` can access by using telnet (or occasionally some other protocol) and connecting to the address specified in the trader entry "Console". For example, the host shooga has a console entry "tel:selby3-r". This specifies that a telnet session to the address "selby3-r" will enable an xwcons session to receive all console output from the host.

For Fairisle switches the number of serial lines which which be required to monitor all ports of a switch is excessive. Fairisle ports therefore use a network console, which works as follows: each port on a switch checks to see whether it has been given in its hostinfo an environment variable which specifies the msnl address of a master port which has access to a serial line. If so, the port directs over an msnl connection all serial line output to that address. The master port, which has a serial line connected to it, is responsible for multiplexing the console output from all other ports on the switch and outputting it over its serial output device. The switch entry in the trader contains the console output address for the multiplexed console output of all ports on the switch. `xscons` Demultiplexes the output from each port and displays it in its own window. The protocol for console output is described elsewhere in this document collection.

The console output from each serial port is input into a bridge PAD, which also runs a telnet service. Each serial port is accessible via a different telnet address. The address of a particular port is given in the trader entry for the switch or host which uses it. The PAD serial lines must be set to the appropriate baud rate for the serial output device using them. For Fairisle ports, both fpc3 and fpc2, the line should be set at 4800 baud. The settings differ for other wanda machines. The current settings of the ports on the bridge PADS is stored in a file `/usr/groups/wanda/machines/TrojanPads`. This describes which wanda host or switch is connected to which bridge PAD line, and what settings should be used.

7 Hardware Status and History

There are currently 51 fpc3 ports in service, 12 fpc2 ports, and numerous other wanda machines. Keeping information on the performance of each physical device is difficult. For the fpc3 ports, at least, we have a full history for each card. Each fpc3 port has a unique identification number stored in its ID ROM. It is vital that the ROM is never removed from a board, otherwise we will have no means of identifying it. The full history of each board is stored in a text file in `/usr/groups/fairisle/BT/fpc3/fpc3_status.txt`. All modifications/repairs to a board should be noted in this file. For other components in Fairisle switches, such as clock boards and back planes, status information is kept in the file `/usr/groups/fairisle/BT/fpc3/switch_status.txt`. For some other wanda machines status information is stored in `/usr/groups/wanda/machines/cards/`.