Using the Endace DAG 3.7GF Card With FreeBSD 7.0

Amiel Heyde, Lawrence Stewart
Centre for Advanced Internet Architectures, Technical Report 080507A
Swinburne University of Technology
Melbourne, Australia
amiel@swin.edu.au, lastewart@swin.edu.au

***Abstract*—The Centre for Advanced Internet Architectures (CAIA) recently acquired 4 Endace DAG 3.7GF high performance capture cards. Where possible, the centre's standard experimental environment utilises the FreeBSD operating system and commodity PC hardware to perform experimental research. This report details how to use DAG 3.7GF capture cards with FreeBSD 7.0 in a variety of commonly required experimental network configurations used in day-to-day research.**

I. Introduction

High performance packet capture and time stamping is a common requirement for experimental network research. A number of options exist, ranging in price, accuracy, scalability and features.

Endace [1] manufacture the DAG [2] range of hardware packet capture cards, which are one such option for researchers.

This report specifically focuses on documenting the use of the DAG 3.7GF dual-port PCI-X gigabit Ethernet network monitoring card with the FreeBSD 7.0 operating system. The report describes the procedure for configuring a capture system, documents possible usage scenarios for the cards and outlines the appropriate configuration required to achieve each scenario.

This report is not a substitute for the documentation provided by Endace in their software distribution.

II. Configuration

Being a specialised piece of hardware, the DAG 3.7GF card requires some non-standard configuration in order to function correctly. Before continuing with this guide, ensure you have a machine that meets the following requirements:

- Has a spare 3.3V PCI or PCI-X (preferred) expansion slot
- Has a basic install of FreeBSD 7.0-RELEASE including all system sources

The driver and software configuration process is a bit tedious, and does not utilise the FreeBSD package management system. This makes it difficult to manage the installation, upgrading and removal of the DAG drivers and software. To remedy this, we created a FreeBSD port to facilitate this. We aim to have the port included in the official FreeBSD ports tree. In the meantime, the port can be obtained from [3]. At the time of writing, the current version of the port is v1.0.

The steps required to configure a machine that has met the previously stated minimum requirements are as follows:

- Extract the dag3 port directory e.g. `tar -xzvf dag3-port.tar.gz`
- Change directory into the port e.g. `cd dag3-port-1.0`
- Install the port:
  - With PCAP support: `make install`
  - Without PCAP support: `make WITHOUT_PCAP=1 install`

After the installation has completed, you will be instructed to restart the machine. This is required to allow the low level kernel modules and loader tunables configured during the install to work correctly.

Assuming everything went well, running the `dagthree` utility from the command line should show the current configuration of the DAG card installed in the system.

If you built the dag3 port with libpcap support, you can test whether applications that use libpcap have DAG support by running `dagwatchdog -p &`, followed by `tcpdump -ni dag0`. The `dagwatchdog` command engages the fail-safe relays that allow the card to capture traffic.

A. *Tuning*

The `dagmem` kernel module is installed by the dag3 port. It uses the `dagmem_size` loader tunable in /boot/loader.conf to control how much system RAM (in bytes) will be allocated to the packet capture ring buffers used by the card. The port defaults to allocating 1/5th of the system RAM, but this variable can be adjusted according to requirements. A minimum of 8MB is required.

If you built the dag3 port with libpcap support, you should be aware of the requirements to ensure pcap enabled applications are able to use the DAG card as a capture device. The base FreeBSD system ships with its own version of libpcap, which is installed in /lib by default. Applications that are dynamically linked against libpcap will be linked against the /lib version of the library. The libpcap port installed as a dependency of the dag3 port installs itself into /usr/local/lib. The net result is that the system ends up with two separate pcap libraries available to the linker: one without DAG support (in /lib) and one with DAG support (in /usr/local/lib).

By default, the system will link against the main system pcap library in /lib. To change this behaviour, /etc/libmap.conf can be used to remap the base system pcap library to the DAG enabled pcap library from ports. By adding the line shown in Listing 1 to the file, dynamically linked executables using libpcap.so.5 (from /lib) will now link to libpcap.so.0.9.7 (from /usr/local/lib). If the dag3 port was build with libpcap support, this entry will have been added for you by the port. This allows all dynamically linked pcap applications to utilise the DAG card as a capture device without even a recompile.

Listing 1 /etc/libmap.conf library remapping

```
libpcap.so.5 libpcap.so.0.9.7
```

Note that statically linked pcap applications will not respond to the above configuration, and will need to be explicitly recompiled against the pcap library in /usr/local/lib in order to utilise the DAG card for capture. A simple way to ascertain if an executable is dynamically or statically linked is to use the `ldd` command. Running `ldd <path_to_binary>` on the command line will emit an error if the binary is statically linked.

III. General Usage

A. *Card Configuration*

The `dagthree` program is used to configure the card and show status information. Running the program with no arguements displays the current configuration of the card. Running `dagthree default` will restore the card to its default configuration. Running with the `-si` switch will show the current status of the card. Other options which can be configured with the `dagthree` utility include:

- reset: Reset the ethernet framers, set auto mode.
- default: Initialise the card and reinstate factory default settings.
- 10: Force 10BaseT mode.
- 100: Force 100BaseTX mode.
- 1000: Force 1000BaseT mode.
- slen=X: Capture X bytes of the packet content.
- varlen/novarlen: In variable length capture mode, packet records will match the size of the captured packet or be truncated to slen bytes. In fixed length capture mode (novarlen), packet records will always be slen bytes in length, with padding added if required.
- steer - Separate or combine the receive streams.

The logical path from the DAG hardware to the operating system is known as a stream. The DAG 3.7GF card supports two independent receive streams and one transmit stream. The `steer` option controls whether each physical port uses its own receive stream, or whether both ports use a single receive stream, as illustrated by Figure 1.

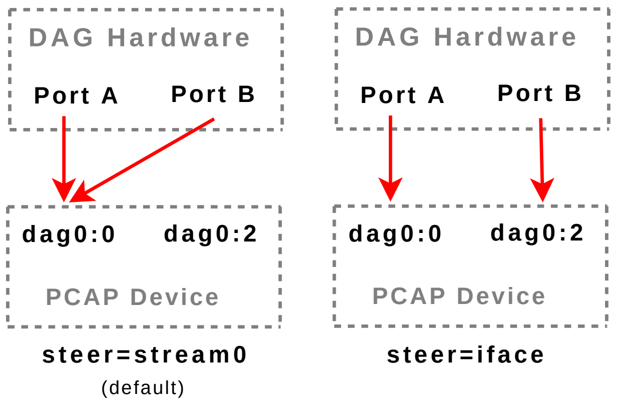

Fig. 1. Configuration of receive streams

Each utilised stream requires an individual allocation of memory from the system RAM allocation made to the DAG card (discussed in section II-A).

You may manually specify the amount of buffer memory (in megabytes) allocated to each stream with the `dagthree mem` option. The format is `mem=<rx_stream_0>:<tx_stream>:<rx_stream_2>`. For example, to set the first receive stream buffer to 48MB, the second receive stream buffer to 64MB and the transmit stream buffer to 16MB, run `dagthree mem=48:16:64` on the command line. Of course the total of these values must not exceed the amount of system RAM allocated to the DAG card.

Sensible memory allocation will depend on the configuration of the `steer` option. If `steer` is set to 'stream0' then all memory should be allocated to the first stream. If `steer` is set to 'iface', memory should be split between both receive streams.

The default configuration allocates 16MB to the transmit stream and divides the remaining amount between the two receive streams.

B. Capturing Data

The `dagsnap` utility can be used to capture all packets from the DAG card to a file in the Endace Extensible Record Format (ERF). To use the data from an ERF dump file with applications that require pcap formatted data, the `dagconvert` utility can be used to convert between formats.

If the dag3 port was built with libpcap support, you can also use any libpcap based application to capture packets from the dag card e.g. `tcpdump -ni dag0 -w dump.pcap`.

The DAG cards are built with inline forwarding applications in mind. As such, they are equipped with hardware fail-safe relays which automatically turn the DAG ports into an electrical passthrough in the event of a machine malfunction. Before any capturing can be performed, the fail-safe relays must be engaged using the `dagwatchdog -p &` command. Once started, the command can be left running in the background for all subsequent capture sessions, but must be restarted if the machine is rebooted or the command terminates.

IV. Common Experimental DAG Usage Scenarios

All configuration examples in this section assume the DAG card has been reset to default configuration as described in Section III-A.

A. Passive Taps

By using a hub or mirrored switch port, the DAG card can be supplied with a passive tap of network traffic between devices, as shown in Figure 2. No special configuration of the DAG host is required other than the steps outlined in Section III-B.

We can construct scenarios to test how an individual device affects packets flowing through it. A typical

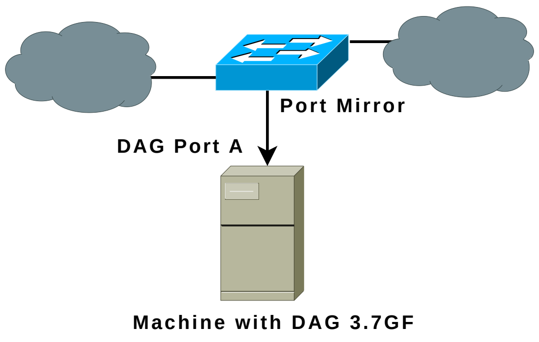

Fig. 2. Passive tap

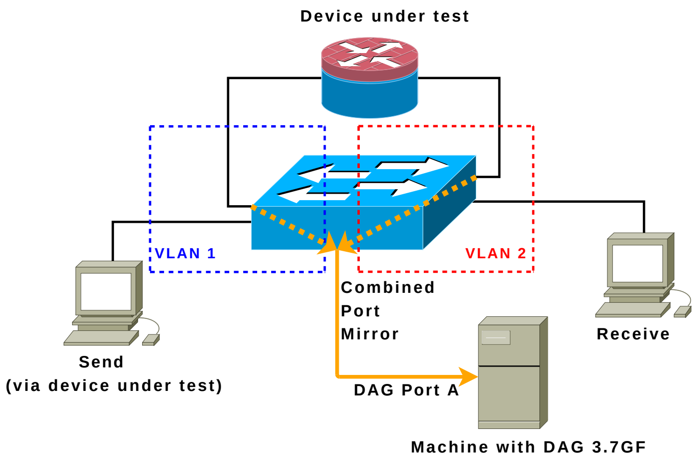

Fig. 3. Passive single stream double tap

configuration might involve a switch segmented into vlans, and the test device straddling the vlans. Figure 3 still uses a passive tap, but by virtue of a single port mirror on the switch replicating packets from our test device's interfaces to the DAG machine, we can capture traffic either side of the test device.

Whilst physically simple to configure, this scenrio has the drawback that we aggregate traffic from our test device's interfaces into a single fixed rate port mirror. This obviously reduces the peak throughput we can sustainably mirror at our switch and also introduces the possibility of slight queuing delays and jitter with packets being buffered at the switch's port mirror output queue. However, this scenario is perfectly suitable for small data rate tests.

As a result of the DAG 3.7GF's dual port configuration, we can further expand the scenario in Figure 3 to that shown in Figure 4. Here we utilse two separate port mirrors, each capturing only the traffic through one of the interfaces of our test device. This reduces the possible bottleneck at the port mirror's output queue which was

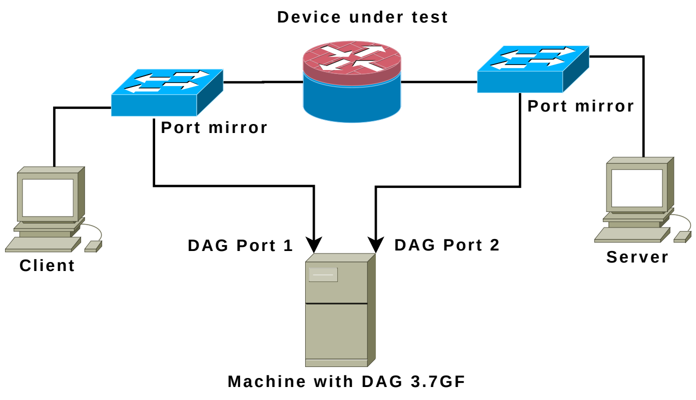

Fig. 4. Passive double stream double tap

present in the previous scenario. It also allows the peak throughput we can sustainably mirror at our switch to increase to the line rate of our port mirrors.

To configure the DAG card so that the two ports can be used independently, run `dagthree steer=iface` on the command line. This separates the ports into independent devices, allowing them to be individually accessed via dag0:0 and dag0:2 e.g. `tcpdump -ni dag0:0 -w dump.pcap` will capture from the first of the two DAG capture ports.

B. Active Taps With Passthrough

As alluded to in Section III-B, the DAG cards can be used inline to create an "active" tap in addition to the passive tap scenarios already discussed. Figure 5 illustrates a simple active tap with passthrough scenario.

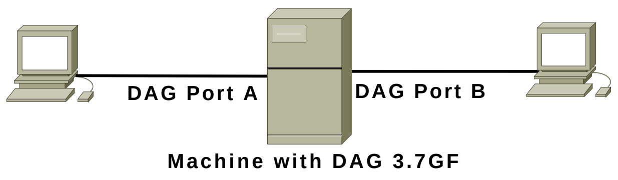

Fig. 5. Active tap with passthrough

A single overlapping buffer is used for both receiving and transmitting traffic, which must be configured by running `dagthree overlap` on the command line. One disadvantage of this technique is that this functionality may only be achieved using the DAG API directly and cannot be utilised by libpcap based applications.

The `dagfwddemo` demonstration utility can be used to configure the DAG card to forward packets. It requires a BPF filter argument which will record to disk and bidirectionally forward traffic matching the filter expression.

It has been observed that this utility introduces significant additional latency (in the order of multiple milliseconds) into the packet path. Further investigation is required to ascertain the cause of this delay, but we envisage that any serious use of the DAG forwarding capabilities would require the creation of a custom piece of software in C/C++ that utilised the DAG API directly.

V. Conclusion

This report describes how to configure a FreeBSD 7.0 machine to be an Endace DAG based high performance packet capture device. Some common usage scenarios and their associated configuration details were also discussed to provide a starting point for anyone wishing to use a DAG based capture solution for their research work.

References

[1] "Endace," accessed 29 April 2008. [Online]. Available: http://www.endace.com/
[2] "Endace - dag network monitoring cards," accessed 29 April 2008. [Online]. Available: http://www.endace.com/our-products/dag-network-monitoring-cards/
[3] "Freebsd dag3 port," accessed 29 April 2008. [Online]. Available: http://caia.swin.edu.au/urp/newtcp/tools/dag3-port.tar.gz