Local Area Network Cluster Interconnect Monitoring

Keith Parris
Systems/Software Engineer, HP Multivendor Systems Engineering

Overview

Local Area Network (LAN) technology is increasingly replacing older technologies such as Computer Interconnect (CI) and Digital Storage Systems Interconnect (DSSI) as the interconnect used for communications within the majority of OpenVMS Cluster configurations. This article describes the LAVC$FAILURE_ANALYSIS tool which is supplied with OpenVMS and which can be used to monitor and troubleshoot LANs that are used as cluster interconnects.

LAN Cluster Interconnect Monitoring using LAVC$FAILURE_ANALYSIS

Background

When support for OpenVMS Cluster communications over a Local Area Network (LAN) was first introduced, the resulting configuration was known as a Local Area VAX Cluster, or "LAVC" for short.

Support for multiple LAN adapters in a single system was introduced in VAX/VMS version 5.4-3. This allowed LAVC configurations with a significant amount of LAN redundancy to be configured, and allowed the cluster to continue operating and survive the failure of network adapters and various other network components.

One challenge that is always present when redundancy is configured is the need to monitor the redundant components to detect failures, or else, as components continue to fail over time, the entire system may fail when the last remaining working component fails.

To assist with this problem, supplied along with OpenVMS is a tool called LAVC$FAILURE_ANALYSIS. Its purpose is to monitor and report any LAN component failures. It reports these using Operator Communication (OPCOM) messages. It also reports when a failure is repaired.

Implementing LAVC$FAILURE_ANALYSIS

There is a template program for LAVC$FAILURE_ANALYSIS found in the SYS$EXAMPLES: directory on an OpenVMS system disk. The template program is called LAVC$FAILURE_ANALYSIS.MAR. The template program is written in Macro-32 assembly language, but you don't need to know how to program in Macro-32 just to use it.

To use the LAVC$FAILURE_ANALSYS facility, the program must be:

1. Edited to insert site-specific information
2. Compiled (on Alpha; assembled on VAX)
3. Linked, and
4. Run on each node in the cluster (preferably at boot time)

Maintaining LAVC$FAILURE_ANALYSIS

The program must be re-edited and rebuilt whenever:

1. The LAVC LAN is reconfigured
2. A node's MAC address changes (for example, when an HP Customer Services representative replaces a LAN adapter)
3. A node is added or removed (permanently) from the cluster

How Failure Analysis Is Done

PEDRIVER, the Port Emulator code which makes a LAN look and act like a CI (Computer Interconnect) port to the OpenVMS Cluster code, transmits multicast "Hello" packets every 2-3 seconds or so from each LAN adapter that is enabled for cluster communications. These "Hello" packets are sent to a multicast address that is associated with the cluster group number, and which has a MAC address of the form AB-00-04-01-xx-yy, where "xx-yy" is based on the cluster group number plus an offset. Each cluster member enables receipt of packets addressed to the MAC address associated with its own cluster group number, and uses the receipt of Hello packets to discover new communications paths and track the reachability of a node via a given path.

In the information added by editing the LAVC$FAILURE_ANALYSIS program to customize it for a given cluster, OpenVMS is told what the LAN configuration should be (in the absence of any failures). The LAN configuration is represented as a mathematical "graph" with "nodes" and "connections" between the nodes in the graph. From this information, OpenVMS infers which LAN adapters should be able to "hear" Hello packets from which other LAN adapters. By checking for the receipt of Hello packets as expected, OpenVMS can thus determine if a path is working or not.

By analyzing Hello packet receipt patterns and correlating them with the mathematical graph of the network, OpenVMS can tell which nodes in the mathematical network graph are passing Hello packets and which appear to be blocking Hello packets. OpenVMS determines a Primary Suspect (and, if there is any ambiguity as to exactly what specific component has failed because more than one failure scenario might cause the observed symptoms, also identifies one or more Alternate Suspects), and reports these via OPCOM messages with a "%LAVC" prefix.

Primary Suspects are reported with a message prefix of the form "%LAVC-W-PSUSPECT". Alternate Suspects are reported with a message prefix of the form "%LAVC-I-ASUSPECT". When the failure that caused a Suspect to be reported is repaired, a message with a prefix of the form "%LAVC-S-WORKING" is generated.

Getting Failures Fixed

Since notification is done via OPCOM messages, someone or something needs to be scanning OPCOM output and taking action. If no human is actively watching the OPCOM output, the error notification may be overlooked.

In one disaster-tolerant cluster there were two expensive DS-3 inter-site links configured, and LAVC$FAILURE_ANALYSIS was put into place to monitor the inter-site links, but the OPCOM message reporting a failure of one of the links one day was missed and the failure was not discovered until six days later. In the intervening time, a failure of the other link would have caused a loss of communications between the two sites of the disaster-tolerant cluster, which would likely have been noticed quickly, but would not have been very convenient.

Products such as TecSys Development Inc.'s ConsoleWorks, Computer Associates' Unicenter Console Management for OpenVMS (previously known as Console Manager), Ki Networks' Command Line Interface Manager (CLIM), or Heroix RoboMon can scan for the %LAVC messages generated by LAVC$FAILURE_ANALYSIS and take some appropriate action (sending e-mail, sending a notification via pager, etc.).

Gathering Information

To implement LAVC$FAILURE_ANALYSIS, data must be gathered about the LAN configuration. The data required includes:

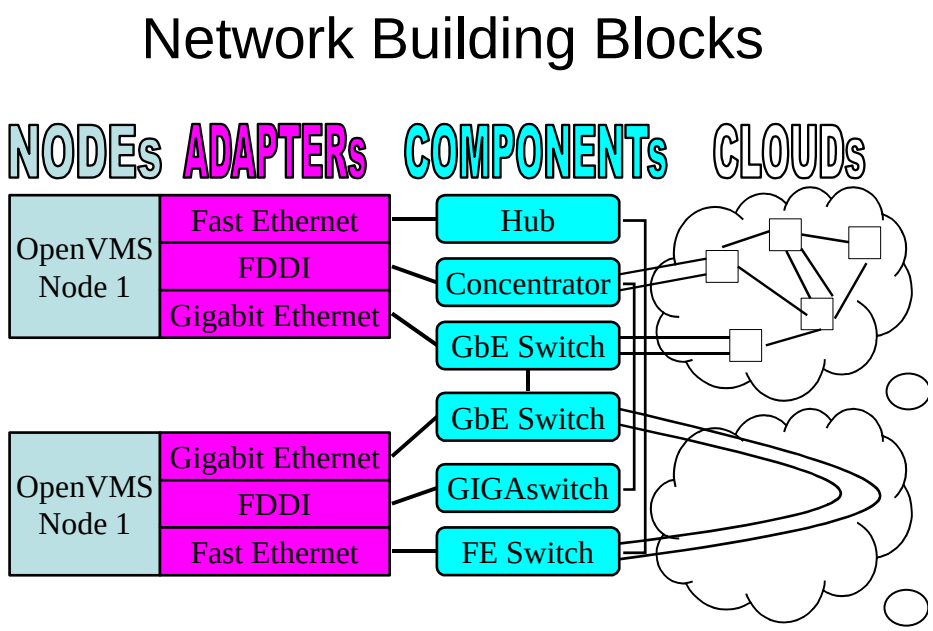
- OpenVMS nodes, and the LAN adapters in each node
- Hubs, switches, bridges, and bridge/routers (routers which have bridging enabled)
- Links between all of the above

Network Building Blocks

For the purposes of LAVC$FAILURE_ANALYSIS, OpenVMS considers LAN building blocks as being divided into 4 classes:

- NODE: An OpenVMS system
- ADAPTER: LAN adapters or Network Interface Cards (NICs) in each OpenVMS system
- COMPONENT: Hubs, switches, bridges, bridge-routers
- CLOUD: Combinations of components that can't be diagnosed directly

These relationships are illustrated in the following diagram:



Handling Network Loops

The algorithm used for LAVC$FAILURE_ANALYSIS can't deal with loops in the mathematical network graph. Yet redundancy is often configured among LAN components (and this is a good thing). The bridges' Spanning Tree algorithm automatically shuts off redundant (backup) links unless and until a failure occurs. But Hello packets don't get through these backup links, so LAVC$FAILURE_ANALYSIS can't track them directly.

For these cases, you replace the redundant portion of the network with a "network cloud" that includes all of the redundant components. Then OpenVMS can determine if the network "cloud" as a whole is functioning or not, and doesn't have to worry about the internal details of which links are turned on or off by the Spanning Tree protocol at any given time.

Handling Multiple LANs

Note that multiple, completely separate LANs don't count as "loops" in the network, and OpenVMS can track each one separately, independently, and simultaneously.

Gathering Information

Let's look a bit more closely at the data required for LAVC$FAILURE_ANALYSIS:

- Node names and descriptions
- LAN adapter types and descriptions, and their:
    - Media Access Control (MAC) address (e.g., 08-00-2b-xx-xx-xx, 00-00-F8-xx-xx-xx)
    - plus DECnet-style MAC address for Phase IV (e.g., AA-00-04-00-yy-zz)
- Network components and descriptions
- Interconnections between all of the above

The names and descriptions supplied will be used in the OPCOM messages which are generated upon detection of failures and when failures are repaired.

Getting MAC address info

A DCL command procedure similar to the following can be used to help sift through the output from SDA> SHOW LAN/FULL and pick out the device names and MAC address data.

```
$! SHOWLAN.COM
$!
$     write sys$output "Node ",f$getsyi("nodename")
$     temp_file := showlan_temp.temp_file
$     call showlan/out='temp_file'
$     search 'temp_file' "(SCA)","Hardware Address" –
            /out='temp_file'-1
$     delete 'temp_file';*
$     search/window=(0,1) 'temp_file'-1 "(SCA)"
$     delete 'temp_file'-1;*
$     exit
$!
$ showlan: subroutine
$      analyze/system
show lan/full
exit
$     endsubroutine
```

Editing the Template Program

Once the necessary data has been gathered, you will need to edit a copy of the LAVC$FAILURE_ANALYSIS.MAR program found in the SYS$EXAMPLES: directory.

There are five sections to edit, as follows:

Edit 1

In Edit 1, you can give descriptive names to nodes, adapters, components, and clouds. These names become names of some macros. Later in the code, you will create invocations of these macros, calling them by the names defined here.

These macro names are for your convenience and reference inside the program, and do not appear external to the program.

Edit 1 example, taken from the template LAVC$FAILURE_ANALYSIS.MAR program:

```
;     Edit 1.
;
;           Define the hardware components needed to describe
;           the physical configuration.
;

      NEW_COMPONENT     SYSTEM      NODE
      NEW_COMPONENT     LAN_ADP     ADAPTER
      NEW_COMPONENT     DEMPR       COMPONENT
      NEW_COMPONENT     DELNI       COMPONENT
      NEW_COMPONENT     SEGMENT     COMPONENT
      NEW_COMPONENT     NET_CLOUD   CLOUD
```

Not very many networks today contain DELNI and DEMPR boxes or thickwire Ethernet cables, but you might create names like SWITCH, HUB, CISCO, DS3_LINK, and so forth here. Note that you can create multiple names for a given type of network building block, as we see done here with COMPONENTs.

Edit 2

In Edit 2, you create an "ASCII art" drawing to document the LAVC LAN configuration. This has no functional effect on the code, but helps you (and others who follow you) understand the information in the sections of data that follow.

In the drawing, you choose brief abbreviated names for each network building block (Node, Adapter, Component, or Cloud). These abbreviated names are only used within the program, and do not appear externally.

Edit 2 example:

```
;    Edit 2.
;
;              Diagram of a multi-adapter LAV cluster.
;
;
;   Sa   -----+---------------+---------------+-------------+-----   Sa
;             |               |               |             |
;             |             MPR_A             |             |
;             |          .----+----.          |             |
;             |         1|        1|         1|             |
;           BrA       ALPHA     BETA       DELTA          BrB
;             |         2|        2|         2|             |
;             |          `----+----'          |             |
;             |             LNI_A             |             |
;             |               |               |             |
;   Sb   -----+---------------+---------------+-------------+-----   Sb
```

Edit 3

In Edit 3, you name and provide a text description for each system and its LAN adapter(s), and the MAC address of each adapter. The name and text description will appear in OPCOM messages indicating when failure or repair has occurred. The MAC address is used to identify the origin of Hello messages.

For DECnet Phase IV, which changes the MAC address on all LAN adapters (which it calls Circuits) that it knows about from the default hardware address to a special DECnet address when it starts up, you provide both:

- The hardware MAC address (e.g., 08-00-2B-nn-nn-nn), and
- The DECnet-style MAC address, which is derived from the DECnet address of the node (AA-00-04-00-yy-xx)

This way, LAVC$FAILURE_ANALYSIS can work both before and after DECnet Phase IV starts up and changes the MAC address.

DECnet Phase V (DECnet/OSI) does not change the MAC address, so only the hardware address is needed.

Edit 3 example:

```
;      Edit 3.
;
;              Label Node                    Description                       LAN HW Addr          DECnet Addr
;              ----- -----   ----------------------------------------   --------------------  --------------------

      SYSTEM   A,    ALPHA,  < - MicroVAX II; In the Computer room>
      LAN_ADP  A1,   ,       <XQA; ALPHA - MicroVAX II; Computer room>,   <08-00-2B-41-41-01>,  <AA-00-04-00-01-04>
      LAN_ADP  A2,   ,       <XQB; ALPHA - MicroVAX II; Computer room>,   <08-00-2B-41-41-02

      SYSTEM   B,    BETA,   < - MicroVAX 3500; In the Computer room>
      LAN_ADP  B1,    ,      <XQA; BETA - MicroVAX 3500; Computer room>,  <08-00-2B-42-42-01>,  <AA-00-04-00-02-04>
      LAN_ADP  B2,    ,      <XQB; BETA - MicroVAX 3500; Computer room>,  <08-00-2B-42-42-02>

      SYSTEM   D,    DELTA,  < - VAXstation II; In Dan's office>
      LAN_ADP  D1,    ,      <XQA; DELTA - VAXstation II; Dan's office>,  <08-00-2B-44-44-01>,  <AA-00-04-00-04-04
      LAN_ADP  D2,    ,      <XQB; DELTA - VAXstation II; Dan's office>,  <08-00-2B-44-44-02>
```

Note that only NODE building blocks (named SYSTEM in this example) have a node name; all the other building block types have a null parameter in that location (indicated by a comma alone).

Edit 4

In Edit 4, you name and provide a text description for each Component and each Cloud. As with the nodes and adapters you described in Edit 2, the name and text description you provide here will appear in OPCOM messages indicating when failure or repair has occurred.

Edit 4 example:

```
;      Edit 4.
;
;               Label each of the other network components.
;

      DEMPR     MPR_A, , <Connected to segment A; In the Computer room>
      DELNI     LNI_A, , <Connected to segment B; In the Computer room>

      SEGMENT   Sa,  , <Ethernet segment A>
      SEGMENT   Sb,  , <Ethernet segment B>

      NET_CLOUD BRIDGES, , <Bridging between Ethernet segments A and B>
```

Here, you give an abbreviated symbolic name and a description for each of the COMPONENT and CLOUD building blocks. Again, the node name argument is null, indicated by a comma alone, for the second argument to the macros.

Edit 5

In Edit 5, you indicate which network building blocks have connections to each other. This is a list of pairs of building blocks, indicating that the two components in a given pair are connected together.

Pairs of components which have no direct connection between them are simply not listed here.

Edit 5 example:

```
;     Edit 5.
;
;          Describe the network connections.
;

      CONNECTION  Sa,   MPR_A
      CONNECTION        MPR_A,    A1
      CONNECTION                  A1,     A
      CONNECTION        MPR_A,    B1
      CONNECTION                  B1,     B

      CONNECTION  Sa,             D1
      CONNECTION                  D1,     D

      CONNECTION  Sa,   BRIDGES
      CONNECTION  Sb,   BRIDGES

      CONNECTION  Sb,   LNI_A
      CONNECTION        LNI_A,    A2
      CONNECTION                  A2,     A
      CONNECTION        LNI_A,    B2
      CONNECTION                  B2,     B

      CONNECTION  Sb,             D2
      CONNECTION                  D2,     D
```

In this example, the author started with network segments, and indicated each component and LAN adapter that was connected to that segment. Note that it is necessary to explicitly indicate which LAN adapters are connected to a given node, even though that might be implied in the naming convention and thus perfectly obvious to a human being.

The formatting of columns adds to the readability here, with network segments, LAN adapters and nodes in their own columns.

My personal preference is to begin with the nodes at the left, adapters in the next column to the right, and so forth, moving outward from a node to the things to which it is connected. But the order of appearance here, either left to right or top to bottom, does not affect the functional behavior. Only which building blocks are connected in a pair is significant.

Handling MAC Address Duplications with Multiple LANs

If you are running DECnet Phase IV and have multiple independent LANs, you may end up with multiple LAN adapters on a single node with the same MAC address, connected to different LANs.

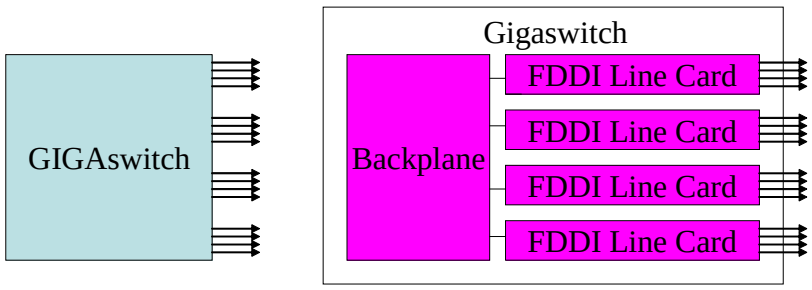
In this case, you will get error messages when you build LAVC$FAILURE_ANALYSIS because of duplicate symbol definitions unless you "comment out" (by putting a semi-colon in the first column) the first line of code after the following comment section:

```
; Within a single extended LAN address must remain unique.  The
; following line checks for unique LAN addresses in a single
; extended LAN configuration.  If your configuration is using
; multiple extended LANs then LAN address can be duplicated between
; the extended LANs.  In this case, the following line should be
; removed.
```

Level of Detail

There is a trade-off between the level of detail which is provided in diagnostic information generated and the amount of work required to initially set up the program and to maintain the program over time. More detail implies more work to set things up in the first place, and more maintenance work as things change, but provides more specific diagnostic information when failures occur and repairs are made.

For example, if a bridge/router or switch is represented as a single black box, notifications can only refer to the box as a whole. If an individual line card or port has failed, but the remainder of the box is working, it will appear to some nodes as if the entire box has failed, while to other nodes it will appear that a remote node's adapter has failed instead, and thus the error messages generated on different nodes will differ, depending on their view of the failure. If the extra effort is taken to identify individual line cards and ports within the box, and indicate to which line card and



port each LAN adapter is connected, then failure reports will be able to identify failures down to the line card level and even to the individual port level. The Level of Detail Example shows, on the left, a GIGAswitch represented as a black box, and, on the right, a GIGAswitch represented as line cards and the backplane connecting them in the box.

Other tools for use with LAVC$FAILURE_ANALYSIS

A DCL command procedure called EDIT_LAVC.COM is available to automatically gather the required information and create a working example LAVC$FAILURE_ANALYSIS.MAR program customized for a given cluster. While the level of detail within the LAN that such an automated tool can provide is limited, it provides a working example and starting point for further customization for a given site. This tool may be included on the V6 Freeware CD in the [KP_CLUSTERTOOLS] directory, but it can also be obtained from http://encompasserve.org/~parris/edit_lavc.com. See the EDIT_LAVC_DOC.TXT file at the same location for more information on how to use this tool.

The DCL command procedure SIFT_LAVC.COM found at the same location can be used to gather all the %LAVC messages from the OPERATOR.LOG files on all cluster nodes, and sort and list them in timestamp order, to gather a better overall picture of the failures as indicated by the varying views from each of the different nodes in the cluster.

Turning On LAVC$FAILURE_ANALYSIS

Once the LAVC$FAILURE_ANALYSIS program has been customized, it needs to be compiled and linked. A command procedure LAVC$BUILD.COM provided in SYS$EXAMPLES: can assist with building the program. Once the program has been built, it should be run once on each member of the cluster. The program loads the network graph data into non-paged pool, and reports the amount of space used.

If you make changes to the program, you can simply run the new version, and it will replace the older network graph data in memory with the newer version.

Because the network graph data goes away when a system reboots, to implement the LAVC$FAILURE_ANALYSIS facility on an ongoing basis, the program should be run once each time on each system as it boots up. This can be done by adding a command to run the program into the SYS$STARTUP:SYSTARTUP_VMS.COM procedure.

Disabling LAVC$FAILURE_ANALYSIS

If you ever wish to turn off LAVC Failure Analysis after it has been started on a given node, you can use the LAVC$FAILURE_OFF.MAR program found at the same location as EDIT_LAVC.COM.

Using LAVC$FAILURE_ANALYSIS to Monitor Non-cluster LAN Components

Sometimes cluster system managers will disable the SCS protocol on certain LAN adapters, particularly if the adapter is to be dedicated to traffic for another network protocol, such as IP. The SYS$EXAMPLES:LAVC$STOP_BUS.MAR program can be used to turn off the SCS protocol on a given LAN adapter, as can the new SCACP utility available in OpenVMS version 7.3 and above.

If SCACP is used instead to lower the priority for SCS traffic on a given LAN adapter rather than turning SCS traffic off entirely on that adapter, then PEDRIVER will transmit Hello packets through the adapter. LAVC$FAILURE_ANALYSIS can then be used to monitor and track failures on those additional LANs as well, while preventing the sending of SCS cluster traffic through that LAN, except as a last resort in the event that all other LANs fail.

Summary

The LAVC$FAILURE_ANALYSIS tool allows monitoring of LAN components and reports failures and repairs in the form of OPCOM messages. This article covered the theory of operation behind the LAVC$FAILURE_ANALYSIS tool, and how to set it up for use in a given cluster.

For more information

Documentation on LAVC$FAILURE_ANALYSIS

LAVC$FAILURE_ANALYSIS is documented in Appendix D of the OpenVMS Cluster Systems Manual (http://h71000.www7.hp.com/doc/731FINAL/4477/4477pro_028.html#network_fail_analysis). Appendix E (subroutines in OpenVMS that support the programs in Appendix D) and Appendix F (general information on troubleshooting LAVC LAN problems) can also be very helpful.

Spanning Tree Algorithm

For more information on the Spanning Tree algorithm, I highly recommend the book "Interconnections" (2nd edition) by Radia Perlman, ISBN 0-201-63448-1.

Contact Information

E-mail: Keith.Parris@hp.com
Web: http://encompasserve.org/~parris/ and http://www.geocities.com/keithparris/