OpenVMS Technical Journal V13

Configuring TCP/IP Services for OpenVMS

Bart Zorn

Overview

A technique will be presented to configure TCP/IP Services for OpenVMS for multiple systems in a consistent way, without having to go through all of TCPIP$CONFIG.COM for every system. With this method, changes in the hardware configuration are also easy to handle. No changes are made to the standard TCP/IP software, only two DCL command procedures are added. These procedures do not use any undocumented feature of TCP/IP Services.

Introduction

TCP/IP Services for OpenVMS is not very flexible with regards to changes in the hardware configuration. The TCP/IP interfaces such as we0, ie0, and ie1 are based directly on the corresponding physical devices such as EWA0, EIA0, and EIB0. There is no way to change that relationship using logical names in a similar manner as we are accustomed to doing for most other devices in OpenVMS.

In a real life example, I had to add one Gigabit network adapter to each of four ES47 systems. These ES47 (model 4) systems consist of two 2P boxes and an I/O drawer. Logically, this I/O drawer appears to sit in between the two 2P boxes. (I am not a hardware expert, so I do not know if there are ways to change that.) The new network cards had to be placed in the I/O drawers, because there was no room in either of the two 2P boxes. The result was that the new adapters received a device name somewhere in between the existing ones, and some of the existing ones got a new name! The result of these changes was that I would have to reconfigure TCP/IP services quite extensively. I had anticipated that.

Implementation

Because TCP/IP Services do not allow the use of logical names to designate physical interfaces, a method had to be found to circumvent that.

Two DCL command procedures have been developed and both these procedures contain relevant information for all systems. Identical copies of the procedures can be used on all systems.

The first command should be executed before TCPIP$STARTUP.COM runs. It does the following:

- Sets up logical names to identify all LAN adapters (and their TCPIP alias names) by their hardware MAC address. This is done using the output of the "LANCP SHOW DEVICE/CHAR" command. Of course, this is the infamous technique of parsing the output of a utility, which is prone to cause problems when a new version of that utility provides a different layout. This has actually happened with the LANCP utility recently! On the other hand, TCP/IP services itself rely on parsing the output of both the TCPIP utility and the LANCP utility.
- Clears out the permanent TCPIP interface database and repopulates it with a default interface described below.
- Clears out any permanent default router information and supplies a new default route described below.

The logical names have the following format:

```
"ADAPTER_00-0F-20-2B-A1-38" = "EWA0","WE0"
```

The default interface and default route for each system are defined in DCL symbols like the following:

```
$ base_interface_<nodename> = -
      ”00-0F-20-2B-A1-38 10.5.50.11/24 10.5.50.3”
```

representing the MAC address, the IP address in CIDR format and the default route for this address. This information, combined with the corresponding logical name, is then used to assemble the “TCPIP SET CONFIGURATION INTERFACE” command for this interface. At least one interface needs to be defined because otherwise TCP/IP Services will not start.

Once this DCL procedure has been run, TCP/IP Services can be started and it will operate on one interface.

The second DCL procedure is called by TCPIP$SYSTARTUP.COM. This procedure does two things:

- Configures all interfaces and alias addresses
- Sets and resets the default route

The site where this configuration was developed makes extensive use of alias addresses. It appears that the ifconfig utility is much more flexible and powerful than the “TCPIP SET INTERFACE” command. Ifconfig does not create pseudo interfaces for alias addresses. The drawback is that the “TCPIP SHOW INTERFACE” command cannot display information about aliases which were created with ifconfig.

Another quirk is that ifconfig is supposed to create interface entities. I could not get it working. On the other hand, “TCPIP SET INTERFACE <ifname>” without further information creates the interface if it does not already exist. Conveniently, in that case, it does not issue an error message.

To sum it up, for every interface, a “TCPIP SET INTERFACE” command is issued, to make sure that it exists. All further configurations are done with ifconfig.

Next, for every interface, first the DCL symbol MAC is defined and then for each IP address, a call to a set_interface routine is made:

```
$ MAC := 00-0F-20-2B-A1-38
$ call set_interface 10.5.50.11/24
$ call set_interface 192.168.35.1/24 alias
```

The set_interface routine will figure out which interface is to be defined. This one gets IP address 10.5.50.11, and 192.168.35.1 as alias address.

Of course, DCL symbols can be used instead of constants. At the beginning of this procedure, symbols are defined for all IP addresses that are being used. Several IP addresses are being used more than once, for IP failover or cluster alias purposes.

A side effect of setting an address for an interface is that the default route may be erased. Therefore, once all the interfaces have been defined, the default route is set again. This default route is not necessarily the same as the one defined in the first DCL procedure described above.

The DCL command procedures

The first one is called TCPIP_INIT_CONFIG.COM. It must be called before TCPIP$STARTUP.COM is invoked. I added this procedure to the CONFIG phase of SYSMAN STARTUP,
but it can be done in other ways.

```
$ set noon
$ nodename = f$getsyi("nodename")
$ if f$trnlnm("sys$pipe") .nes. "" then goto 'p1'
$!
$ call say_msg "-I- Executing CLUSTER_COMMON:TCPIP_INIT_CONFIG.COM"
$ nodename = f$getsyi("nodename")
$ debug = p1 .nes. ""
$ icalc := $sys$tools:icalc ! freeware tool, used for mask calculation
$ saved_parse_style = f$getjpi("","parse_style_perm")
$ set process/parse=traditional ! Needed for icalc, a ^ is being used
$ this_proc = f$environment("procedure")
$!
$! TCPIP_INIT_CONFIG.COM
$!
$! 31-Aug-2006, Bart Zorn
$!
$! This procedure is called before TCPIP$STARTUP.COM and does three things:
$!
$! 1. It sets up logical names to identify all lan adapters (and their TCPIP
$!    alias names) by their hardware MAC address. In addition, logical names
$!    that are common to all systems are defined.
$!
$! 2. It clears out the permanent tcpip interface database and repopulates
$!    it with the default interface defined below.
$!
$! 3. It clears out any permanent default router information and supplies
$!    a new default route defined below.
$!
$ base_interface_node01 := 00-0F-20-2B-A1-38 10.5.50.11/24   10.5.50.3
$ base_interface_node02 := 00-0B-CD-F4-E4-A8 10.5.50.12/24   10.5.50.3
$!
$! 1. Lookup all hardware MAC addresses
$!
$ pipe mcr lancp show device/characteristics | -
        search sys$pipe "Device Characteristics","Hardware LAN address" | -
            @ 'this_proc' do_define
$!
$! 2a. Delete current permanent interface configuration
$!
$ pipe tcpip show configuration interface/full –
      > sys$manager:tcpip_saved_configuration.txt 2> nl:
$ pipe tcpip set configuration nointerface */noconfirm > nl: 2> nl:
$!
$! 2b. Repopulate the configuration database
$!
$ lognam = "adapter_" + f$element(0," ",base_interface_'nodename')
$ interface = f$trnlnm(lognam,,1)
$ if interface .eqs. ""
$   then
$     call say_msg –
      "-F- TCP/IP default interface is not defined. TCP/IP will not startup."
$     goto exit
$   endif
$ ip_address = f$element(1," ",base_interface_'nodename')
$ mask_length = f$element(1,"/",ip_address)
$ ip_address = f$element(0,"/",ip_address)
$ call generate_mask mask_length mask_longword
$ call convert_longword_to_address mask_longword mask_address
$ vf = f$verify(1)
$ tcpip set configuration interface 'interface' –
      /host='ip_address' /network_mask='mask_address'
$ ! 'f$verify(vf)'
$!
$! 3a. Delete current permanent routing configuration, ignoring errors.
```

```
$!
$ pipe tcpip show route/permanent > sys$manager:tcpip_saved_routing.txt 2> nl:
$ pipe tcpip set noroute/permanent/noconfirm/gate=* > nl: 2> nl:
$!
$! 3b. Define permanent default route information
$!
$ def_route = f$element(2," ",f$edit(base_interface_'nodename',"compress,trim"))
$ vf = f$verify(1)
$ tcpip set route/gateway='def_route'/default/permanent
$ ! 'f$verify(vf)'
$!
$exit:
$ set process/parse='saved_parse_style'
$ exit
$!
$say_msg: subroutine
$ msg = f$fao("!8%T !AS",0,P1)
$ write sys$output msg
$ if f$trnlnm("sys$output") .nes. f$trnlnm("sys$error") then write sys$error msg
$ endsubroutine
$!
$do_define:
$!
$! Read the first line, it contains the device name
$!
$ read/end=eof sys$pipe line
$ device = f$element(2," ",line)
$!
$! The second line contains the physical address
$!
$ read/end=eof sys$pipe line
$ address = f$element(0," ",f$edit(line,"trim,compress"))
$!
$! Build the TCP/IP style interface name from the device name
$! This assumes that we have no more than 6 devices of any given type
$!
$ interface = f$extract(1,1,device) + f$extract(0,1,device)
$ unit = %X'f$extract(2,1,device)' - 10
$ interface := 'interface''unit'
$!
$ vf = f$verify(1)
$ define/system/exec/nolog adapter_'address' 'device','interface'
$ ! 'f$verify(vf)'
$ goto do_define
$eof:
$ exit
$!
$generate_mask: subroutine
$ i = 32 - 'p1'
$ pipe icalc 2^'i' > nl:
$ 'p2' == .not. ('ICALC_OUT' - 1)
$ endsubroutine
$!
$convert_longword_to_address: subroutine
$ hex_longword = f$fao("!XL",'p1')
$ a1 = %x'f$extract(0,2,hex_longword)'
$ a2 = %x'f$extract(2,2,hex_longword)'
$ a3 = %x'f$extract(4,2,hex_longword)'
$ a4 = %x'f$extract(6,2,hex_longword)'
$ 'p2' :== 'a1'.'a2'.'a3'.'a4'
$ endsubroutine
```

The second procedure is called CREATE_INTERFACES.COM. It is called from TCPIP$SYSTARTUP.COM.

```
$!
$! CREATE_INTERFACES.COM
$!
$! 24-Mar-2003, Bart Zorn
$!
$ SET NOON
$ CALL SAY_MSG "-I- Executing CLUSTER_COMMON:CREATE_INTERFACES.COM"
$ ifconfig := $tcpip$ifconfig
$ nodename = f$getsyi("nodename")
$!
$! Define symbols to be used here
$!
$ NODE01 := 10.5.50.11/24
$!
$ GOTO SYSTEM_'NODENAME'
$ EXIT             ! Do not fall through
$!
$SYSTEM_NODE01:
$!
$ MAC := 00-0F-20-2B-A1-38
$ call set_interface ‘NODE01’
$ call set_interface 192.168.35.1/24 alias
$!
$ MAC := 00-0F-20-2B-A1-37
$ call set_interface ‘NODE01’
$ call set_interface 192.168.35.1/24 alias
$!
$ goto exit
$!
$SYSTEM_NODE02:
$!
$ MAC := 00-0B-CD-F4-E4-A8
$ call set_interface 10.5.50.12/24
$ call set_interface 10.5.50.18/24 alias          ! Cluster alias
$!
$ MAC := 00-0B-CD-F4-E4-A9
$ call set_interface 10.5.50.12/24
$ call set_interface 10.5.50.18/24 alias          ! Cluster alias
$!
$ MAC := 00-08-02-91-88-CA
$ call set_interface 10.5.50.13/24
$ call set_interface 10.5.50.33/24 alias
$!
$ call set_interface 10.5.52.1/24 home            ! Apllication alias
$!
$ goto exit
$!
$EXIT:
$ gosub reset_default_route
$ EXIT
$!
$say_msg: subroutine
$ msg = f$fao("!8%T !AS",0,p1)
$ write sys$output msg
$ if f$trnlnm("sys$output") .nes. f$trnlnm("sys$error") then –
      write sys$error msg
$ endsubroutine
$!
$set_interface: subroutine
$!
$! p1 - address/mask
$! p2 - optional parameters
```

```
$! p3 - additional parameters for ifconfig
$!
$ lnm = "ADAPTER_" + MAC
$ interface = f$trnlnm(lnm,,1)
$ if interface .eqs. ""
$   then
$     call say_msg "-E- ''lnm' logical name is missing"
$     exit
$   endif
$!
$! Create interface if it does not already exist
$!
$ tcpip set interface 'interface'
$!
$ if f$edit(p2,"lowercase") .eqs. "home" then p2 := home alias
$ params = f$edit(f$fao("!AS !AS !AS",p3,interface,p2),"trim,compress,lowercase")
$ sv = f$verify(1)
$ ifconfig 'params' 'p1'
$ ! 'f$verify(sv)'
$ endsubroutine
$!
$reset_default_route:
$ if "''new_default_route'" .eqs. "" then return
$ if f$mode() .eqs. "INTERACTIVE"
$   then
$     if f$trnlnm("tt") .nes. "OPA0:" then return
$   else
$     if f$mode() .nes. "OTHER" then then return
$   endif
$!
$ pipe tcpip netstat -rn | search sys$pipe default | -
      ( read sys$pipe line ; define/job/nolog line &line )
$ line = f$edit(f$trnlnm("line"),"trim,compress")
$ deassign/job line
$ default_present = f$element(0," ",line) .eqs. "default"
$ if default_present
$   then
$     current_default_route = f$element(1," ",line)
$     if current_default_route .eqs. new_default_route then return
$   endif
$ vf = f$verify(1)
$ tcpip set route /gate='new_default_route' /default
$ ! 'f$verify(vf)'
$ if default_present
$   then
$     if current_default_route .nes. new_default_route
$       then
$         vf = f$verify(1)
$ tcpip set noroute /gate='current_default_route' /noconfirm
$         ! 'f$verify(vf)'
$       endif
$   endif
$ return
```

Things yet to be done

When a new system needs to be configured, it is still necessary to run TCPIP$CONFIG.COM once. I have not yet reverse engineered what steps TCPIP$CONFIG.COM takes with regard to the host name and domain name settings. Also, the client and server configuration needs to be done with TCPIP$CONFIG.COM.

Summary

The techniques described here allow for a complete interface configuration for TCP/IP Services for OpenVMS with two DCL command procedures. These procedures are organized in such a way that they contain all relevant information for all systems to be configured. This makes it a lot easier to configure many systems and prevent duplicate IP addresses and other errors.

The author can be contacted at Bart.Zorn@Yahoo.com