Red Hat Cluster Suite Networking

Revision History

0.1	17th January 2008	Christine Caulfield	Initial version
0.2	18th January 2008	Christine Caulfield	Mention RHEL4 cluster IDs
0.3	22nd February 2008	Christine Caulfield	Add a bit about bandwidth use
0.4	17nd March 2008	Christine Caulfield	Mention wireshark decoders
0.5	31st March 2008	Christine Caulfield	Add a bit about connection security
0.6	3rd April 2008	Christine Caulfield	Small changes & typo fixes, add a bit on ccsd
0.7	9th June 2008	Christine Caulfield	Mention version of wireshark for DLM
0.8	5th October 2010	Christine Caulfield	Mention Broadcast
0.9	8th March 2013	Christine Caulfield	Remove RHEL4, add mentions of rgmanager & ricci

This document describes how cluster suite in RHEL4 and RHEL5 uses networking. It does not describe in detail the internal protocols of the components, it is intended as an aid to understanding what sort of packets are in use and why, and as a troubleshooting document.

It is assumed that the reader has a working knowledge of the basic cman toolset (principally cman_tool, ccsd and the use of cluster.conf)

Contents:

1. Overview of protocols

There are several protocols in use by clustering. The most common is whatever is being used by openais/corosync. This will usually be multicast UDP but could also be broadcast or unicast UDP depending on the configuration. The DLM uses it's own TCP protocol that is independent of anything else and also ccsd (in RHEL5 only) uses its own unicast AND multicast UDP. Ricci has its own XML-based TCP protcol!

1.1 RHEL5 CMAN

cman in RHEL5 is layered on OpenAIS, in RHEL6 this is corosync, which is (for the intent of this document) essentially the same thing. This uses multicast UDP packets by default but unicast UDP (udpu) or broadcast can also be configured.

If no multicast address is configured in cluster.conf then one is generated using the cluster ID as the bottom 16 bits and 239.192 (IPv4) or FF15:: (IPv6) as the top 16 bits .

The openais 'totem' protocol very complicated so I won't attempt to summarise it here. There are academic papers that describe it in excruciating detail if that's what you need. Very simply, a rotating token visits all the nodes in a 'ring', this system ensures that messages arrive at nodes in a predictable order and provides useful features such that when a message arrives back at its originating node, that node knows that all other nodes have seen it.

Openais uses three UDP sockets for communications. A multicast receive socket which is bound to the multicast address. A multicast send socket which is bound to the local IP address but at the port number specified minus one, and a token socket which is bound to the local IP address. The default port number is 5405, so a normal system will have two sockets bound to port 5405 and one to 5404.

1.2 CMAN common bits

cman traffic is crucial to the liveness of the cluster. To ensure that it is not queued behind other bulk networking traffic it sets the socket priority to TC_INTERACTIVE (6). There are various traffic scheduling algorithms available in Linux to change the way the priorities are processed, but most will prioritize TC_INTERACTIVE above other (eg http or even DLM) traffic.

If you choose your own multicast address, make sure you do it carefully. A lot of people seem to choose things like 224.0.0.x which is “All hosts on the network” and might not be routed correctly, or even at all, by some hardware. A better solution is to use the 239.192.x.x address series that cman uses. Whichever you do choose, we strongly recommend that you double-check the configuration of any routers that the packets must pass through. Some can take a long time to learn addresses, stalling or breaking cluster formation.

I have, a couple of times, seen the statement 'cman works using a crossover cable, but behaves oddly on my routed network' ... check your router if this happens and/or adjust the TTL. This can be done in cluster.conf.

1.3 DLM

As mentioned, the DLM does not use cman as a transport protocol, it makes its own connections using TCP/IP. It gets the local IP address and the remote node IP addresses from cman (which gets them indirectly from cluster.conf). By default the DLM uses port 21064 for communication.

DLM expects the transport to be reliable and ordered, and, ideally, fast. Under heavy GFS load the DLM can generate a lot of lock traffic. Most DLM messages are very small, but they are coalesced internally as far as possible to maximise network bandwidth use. During recovery (eg when a node

joins or leaves the cluster, or a GFS filesystem is mounted) much larger packets can be generated.

All network traffic in the dlm is processed by the dlm_send and dlm_recv kernel threads (work queues in RHEL5). If you see dlm_recv high in the process CPU list then it is likely that a lot of work is coming in from remote notes. All remote locking operations happen in this process. dlm_send is responsible simply for sending the packets and coalescing messages together so should not use much cpu time.

There is an option for DLM to use SCTP for communications rather than TCP. This is not currently supported but it works in the same way as TCP communications except that multiple interfaces are supported by SCTP, this is the only way to implement multi-homing.

The dlm starts up its communications when the first lockspace is created and shuts it down when the last lockspace is removed, so don't just '`modprobe dlm`' and expect to see sockets bound to port 21064.

When using TCP, the DLM might have two connections to another node. This is expected behaviour and happens because two nodes might try to establish communications with each other at the same time. It is not possible to close one of these connections without losing data on it. The overhead of having two connections is minimal. If there are two connections between nodes then each node will send on the connection it initiated, rather than the incoming connection from the other node.

2. Name resolution and cluster.conf

cman tries hard to match the local host name(s) to those mentioned in cluster.conf. Here's how it does it:

1. It looks up $HOSTNAME in cluster.conf
2. If this fails it strips the domain name from $HOSTNAME and looks up that in cluster.conf
3. If this fails it looks in cluster.conf for a fully-qualified name whose short version matches the short version of $HOSTNAME
4. If all this fails then it will search the interfaces list for an (ipv4 only) address that matches a name in cluster.conf

cman will then bind to the address that it has matched.

It should be noted that RHEL4 (until very recently) had a bug in stage 3 of this process which meant that it bound to the short name found, rather than the full name in cluster.conf. In fact this is what prompted this document. An example will, I hope clarify this step:

Suppose the host has the $HOSTNAME `myhost.mycompany.com` but also has interfaces accessible by names `myhost.internal.mycompany.com` and `myhost.remote.mycompany.com`. It is possible to use `myhost.internal.mycompany.com` as the node's name in cluster.conf and cman will find it.

In general I recommend that fully-qualified names are used for hostname and in cluster.conf as it avoids a lot of ambiguity. If there is any complexity in the configuration it might also help to add the nodename to the cman_tool join command line eg:

```
# cman_tool join -n myhost.internal.mycompany.com
```

If you are ever in any doubt about what IP address cman is using for communications, the command "`cman_tool status`" will enlighten you. It is worth mentioning here that cman does NOT lookup the address of remote cluster nodes using cluster.conf. It expects them to be visible using the broadcast or multicast address it was given to use. If the remote nodes are not available or only partially visible via that route then all hell will break loose and the cluster will likely fall apart. In particular, connecting two interfaces to the same piece of physical network can cause problems unless you explicitly set up the correct routes. If in doubt consult a real networking expert.

I mentioned that the DLM gets all of its IP addresses from the local cman. It's important to realise that cman only ever resolves it's *own* address – using the rules above. The other nodes' addresses are obtained from communication with the remote nodes themselves - using the broadcast or multicast messaging, and the DLM picks them up from cman. It does *not* resolve the names, ever.

3. Other Bits

I'll just briefly mention other parts of Cluster Suite in an attempt to clear up any confusions that have occasionally arisen.

1. GFS does not use any networking itself. It relies on the DLM to do its locking
2. ccsd uses networking to distribute the cluster.conf files. The following two bullet points are taken from the cluser wiki:
   - 50006/TCP - Cluster Configuration System front-end port. This is used to obtain configuration information from ccsd. Connections not from a reserved port (e.g. <1024) are dropped. This port is only bound on loopback interfaces; remote access is disabled.
   - 50008/TCP (ipv4) & 50009/TCP (ipv6) - Cluster Configuration System update ports. This port is used to handle updates and/or changes to the cluster configuration file (cluster.conf). Connections from outside the known-live cluster are dropped.

3. rgmanager, dlm_controld, groupd and clvmd all use CPG to communicate between nodes. CPG is an internal protocol of openais/corosync and will travel over the same low-level (multicast UDP usually) protocol of the heartbeat/token traffic. In addition rgmanager and clvmd also use the DLM, so will be responsible for TCP traffic too.

4. ricci has its own networking protocol based on XML packets and running over TCP port 11111. It reads the node names from cluster.conf and resolves them itself, independently of cman or openais/corosync

4. Routing

None of the daemons do any special routing, they do normal networking send and receive calls. If you have any special routing requirements the normal system routing utilities will work.

5. Diagnostic Tools

If you are experiencing networking problems with cluster suite, the first thing to check is that you are

using the correct IP address. `cman_tool` status will tell you this. Check it against cluster.conf and the list above to see if it is behaving as you expect. If you can't even get cman up and running because of suspected networking configuration problems then try starting the cluster with '`cman_tool join d`' which will give you more information about how cman is trying to resolve the host name.

Another handy utility is `netstat`, its -u and -t switches will show you which ports and addresses are bound to UDP and TCP sockets respectively. If running as root, adding -p will also show you the process ID and name.

For serious network diagnostic tasks, tcpdump or wireshark are what you need. The output from these can be very cryptic but also immensely useful in diagnosing problems, you don't need to understand all the output to find them helpful.

On a RHEL5 cluster

```
# tcpdump port 5405
```

will show the token packets being passed around.

If you are experiencing really odd problems *and are asked by a support or development engineer* to send in a tcpdump for diagnostic purposes it's important to add the -s0 switch to capture the whole packet, eg

```
# tcpdump -wmyfile.dmp -xs0 port 6809
```

will save traffic information to a file called myfile.dmp.

tcpdump can also be used to check DLM communications, however it's very important to make sure than cman is communicating correctly first. As the DLM sends packets only as needed, there will be nothing to see on a quiescent system, and a busy GFS system can soon overwhelm you with data. In general, if cman is communicating correctly then DLM will too. If that is not the case then it's probably a case of those pesky iptables rules again.

The ip command is helpful for determining which interfaces are bound to which multicast addresses, if you suspect something is wrong here.

```
# ip maddr ls
# ip link ls
```

The tc command shows and changes the network queueing discipline:

```
# tc qdisc show
qdisc pfifo_fast 0: dev eth0 root bands 3 priomap  1 2 2 2 1 2 0 0 1
1 1 1 1 1 1 1
```

Here you can see that priority 6 traffic is mapped internally to '0' (remembering to count the list from 0) which schedules it ahead of other packets. If you load another networking queueing discipline then make sure that it does something equivalent.

If you're serious about tracing DLM packets, then version 1.0.0 of wireshark has a dissector for the RHEL5 DLM and corosync packets, the corosync one will need to be told the encryption key if it is to make any sense of the contents though. Even if you don't have a recent enough version of wireshark to include these decoders, it's handier than tcpdump in many ways because it can separate out the protocol headers from the data. That can save a lot of effort and/or printouts with scribbled pencil marks on them!

6. Bandwidth Use

6.1 RHEL5 CMAN

Even when quiescent, a RHEL5 cluster will generate a measurable network load. This is the totem token being passed around the cluster members. In addition to this load, there are more daemons using the openais messaging system for communications. In particular plocks use openais messages and can add considerably to the networking traffic. In a future version of RHEL5, the totem multi-ring system should become supported and allow the traffic in large clusters to be split over several physical networks to help alleviate this.

Although it is possible to run several RHEL5 clusters on the same network, it is recommended that RHEL5 clusters at the same site are run on separated networks.

6.2 DLM

The DLM uses whatever it can get depending on what it needs. That is, if you have a very busy collection of GFS filesystems with lots of inter-node locking then you will see a lot of large DLM packets. With fast nodes and a lot of traffic, it is possible to flood a network with DLM traffic. It's impossible to put an exact figure on this because it depends on the GFS load, other programs doing locking, and where the master & directory nodes for a lock is.
The addition or removal of a node from a DLM lockspace can cause a great deal of lock traffic if lots of locks are active as locks are recovered and the lock directory is rebuilt

7. Connection Security

None of the cluster suite components implement much in the way of security on their connections. Red Hat recommends that all cluster traffic is routed over an internal, ideally isolated, LAN or VLAN. This not only ensures the security of cluster communications but should also allow for enough bandwidth for a busy cluster to operate at full efficiency.

The DLM will refuse to accept new connections from nodes that are not known to CMAN, and will only accept a maximum of two connections from any node that is known to CMAN. This is for reasons detailed in section 1.4.

CMAN in RHEL5 & RHEl6 uses openais/corosync and always enables packet encryption. If no security key is specified then cman will use the cluster name as the key. This does not provide good security but is probably enough to dissuade casual snooping or spoofing. If you are serious about preventing snooping of traffic then you should create a keyfile, securely distribute that around the

cluster and add its name to cluster.conf with a line similar to the following:

```
<cman keyfile=”/etc/cluster/cman_keyfile”/>
```