From: Spitzer, Andras ([redacted])
Sent: Sunday, August 22, 2010 4:56 PM
To: [redacted]
Cc: [redacted]
Subject: Multicast issue on prod WLS Cluster between Solaris 10 zones in the same subnet [6377007, 6366182]

Hi,

Please find here the current state of my investigation regarding the multicast issue of the WLS clusters on Solaris 10 zones.

Symptom

The original case (#6377007 and #6366182) was about a Weblogic 9 cluster scenario, as WLS Cluster below version 10 supports only multicast for clustering functions the business tried to implement it. The **multicast** configuration seemed to work on their QA zones (fltalgaqqwls01v and fltalgaqqwls02v) but it **barely worked on their production zones** (fltciohppwls01v and fltciohppwls02v). Also our sysadmins discovered that the 224.0.0.0/24 network works perfectly, but any other multicast (class D) address will either work or not. Even these tests were incosistent as it happened that a test worked on one day while it stopped working on the other, etc.

A few facts about multicast

There are three basic type of packets from a distribution point of view on the Internet (IPv4), unicast, broadcast and multicast (and later on anycast). Unicast packets are the most common, the packet has only one destination (system with TCP/IP stack), broadcast which is a packet should be delivered to all the L2 (in our case ethernet) neighbours (we also call this subset of systems "subnet", or broadcast domain), and the multicast which is between the unicast and broadcast. **Multicast packets will be distributed to a selected number of hosts** (both on the local subnet plus it can be routed further depending on the multicast scope which is technically the TTL field in the IP header), hosts that are interested in a specific multicast group should join that group so they signal that they have an application which is interested in that group's traffic.

Multicast (class D) IPs are from 224.0.0.0 - 239.255.255.255, and this IP range has reserved (by IEEE) subnets for specific use case, and for different scope. **224.0.0.0/24 is special** because it's **link-local** (won't be routed outside of the local subnet) and because most of the networking protocols (routing, HA protocols, multicast signaling, etc.) are in that range, these are **privileged addresses** and should be not used by application. Also because many networking features rely on this subnet many filtering techniques treat this as special range and allows the flow of these packets even if the rest of the class D range is filtered for any reason (traffic limitation, policies, etc.).

This explains also why 224.0.0.0/24 addresses were passed on between the two prod zones, still, it's **not recommended to use this range for application** use. One additional detail to multicast, a multicast mac address always starts with 01:00:5e, and there is a basic rule of translating a multicast destination IP address to mac address, but because there is a 3 byte fixed part of the multicast mac address, only the lowest 23 bits are translated to the mac address. As a result, 32 IP addresses in the multicast range will share the same mac address, so the translation is not one - to one. That's why it's important that next to the ethernet frame's mac address the IP header contains the original destination IP address, so the receiving host can see which IP original was targeted. (if it would be a one - to - one mapping it could reverse translate the mac address to IP, but as it's not, it has to use the IP in the IP header) As a result when a UNIX receives a multicast mac address, the NIC layer goes through a so-called **imperfect filtering**, as it will be down to the application (called **perfect filtering**) as the packet flows further to decide if the subscribed IP is indeed one of the 32 addresses the mac translates to. An IP address within the multicast range is we call the multicast group, also want to note that Weblogic suggests to use the 239.192.0.X network for their cluster interconnect.

It's also important to know that the multicast traffic on the network side is much like broadcast, historically switches flooded the multicast packets to all their ports (as they did and do to broadcast packets), the difference appeared on the NIC first, as the NIC decided whether the system is interested in that multicast group or not (on a contrary to the broadcast where the NIC suppose to pass on/process every broadcast packet it receives). Later on switches got smarter, but I will write more about this later.

Tools I used and the beginning of the test

For my initial tests I used the WLS tool MulticasTest, I also used mtools which are a really simple multicast testing tool written in C. Obviously I used network sniffers to analyze traffic and later on I used packet generators to emulate

protocols (to prove my theory). Although the issue was reported on the production zones, **I ran my tests on their global zone** (same system**)**. These are giscvgh2sv**027** and giscvgh2sv**028**. I also resolved all the network connections on these systems, so without getting into the boring details 027 has an IPMP group of two NICs facing the default network (nxge4, nxge0), 028 has IPMP with 2 NICs as well (nxge0, e1000g0), both systems belong to VLAN 340 (called 3.24.140.0_Prod_SUN_Containers). 027 is connected to C4948 switch SNUsgXUScin[AB]25 and 028 is connected to a C3750 switch SNUsgXUScinh[ab]29, and I wanted to have a 3rd system (a control system) which is connected to one of these switches so I can double check if a symptom is specific to the OS or more suspected the network behind. The only system (within the same VLAN 340) I found connected to the 25 switch was giscvgh2sv024. Each of these systems are **running Solaris 10** with IPMP (**dual homed**).

As a next step I started to resolve the path between these two switches, 25 and 29, and I found a **C65XX called SNUIbXUScingh0[34],** which has an MSFC1/2 module for L3 processing. switch 25 is connected directly to this C6500, while 29 is connected indirectly (via *cinh[ab]10), so I started to draw a network connectivity map (it became 2 x A4 papers). When I first started to test the multicast connectivity, I experienced random behaviour, as once it was working, second it was not, again later on the second was working, etc. The **test itself was pretty simple**, I started a multicast listener on one system, and I started to send multicast packets to the same multicast group, and in case of success the multicast listener was able to print the messages send by the other system, if I couldn't see a message, the test failed. My **priority was to have a series of consistent tests** what I can use to look investigate further. (investigating on an inconsistent base is a waste of time as the inconsistency will increase exponenially) As 024 and 027 were connected to the same switch (and same VLAN obviously), I started my mcast tests there. I decided that if I see stable results there, I can move on testing (024 and 027) against 028, as betwen these two groups I have more switches/features to count with.

Anyway, even the first tests randomly failed between 024 and 027, I spent a lot of time debugging what is behind that randomness. I used DTrace, traffic sniffers, and multicast test programs so I realized the following. nxge0 @ 024 and nxge4 @ 027 are both connected to the same switch cinhb25, while e1000g0 @ 024 and nxge0 @ 027 also connected to the same switch, cinha25. This is pretty much simple, nothing complicated here, if you want to send a packet from nxge0 @ 024 to nxge4 @ 027, it goes through only cinhb25, but if you want to send a packet from nxge0 @ 024 to nxge0 @ 027, it **has to travel through the C65XX switch** ! Not just because cinha25 and cinhb25 are physically not connected, even if they were because of the spanning tree (pvst+) the root bridge is the C65XX, which means that even if these two switches are connected physically the connection would be in a block state to provide a loop free tree structure for the data packets. In a STP the root bridge election goes by the bridge ID, which consist of a priority value (2 bytes, 0 to 65535) and the mac address of the switch. The priority value is set to default 32768, and the lower wins, and that's what I found in the C65XX's config :

```
spanning-tree vlan 301-304,340,346,400-411,495-503,598-599,717 priority 8192
```

This proves that the C65XX is most likely the root bridge in the STP, which means even if two switches are connected for redundancy, it doesn't necessarily means they can talk to each other that way, there is a good chance that the packet has to pass the root bridge first. That's nothing bad so far, it's a neat design, although **the features in the C65XX will heavily influence the whole broadcast domain**. Back to my test, the test was really simple, I started to multicast listen on 027, and started to send multicast packets from 024 to the same multicast group. I quickly realized that a multicast packet sent out on nxge0 @ 024 will only reach nxge4 @ 027, and a multicast packet sent out on e1000g0 @ 024 will appear only on nxge0 @ 027, which was bad, as the multicast packet should arrive everywhere within the broadcast domain (as I described above, until the NIC the multicast packet is like a broadcast packet). This test sounds really simple here but it was way more painful while I was doing it. The reason was how UNIX deals with multicasts.

UNIX and multicasts programming

There are **two sides** of multicasting, one side where you **listen to multicast packets**, the other side which **sends the multicast packets**. sending multicast packets is really simple, there is nothing multicast-specific about it, it's a simple UDP packet targeted to the multicast group ID (which is an IP address within the multicast range), the only trick here is that the system call which is responsible for "creating" the mac address will simply translate the IP address to the appropriate multicast mac address. The sender doesn't have to be in the multicast group, the only trick is the IP address -> mac address translation, and the UDP packet is ready to go. The listener is a bit more tricky. The multicast listening is pretty much similar to having an UDP listener created, but after you created the socket and before you start to listen, you have to issue a few setsockopt() commands to warn the kernel that this will be a multicast UDP socket. Whenever the multicast listener is complete, you can check with netstat that it will listen on port *.<port>, the * indicates that any interface are welcome. The tricky stuff comes at the setsockopt() part, where if you don't specify, it will use the INADDR_ANY interface for multicasting, which means the kernel will assign 1 (!) NIC to designate as a multicast interface! (so in case of a system with two NICs toward the main network you have to "bind" twice for each NICs to join the multicast group)

So even though you have your UDP socket bound to any, the setsockopt() system call will mark only one of the NICs as part of that multicast group! That was tricky, and I got tricked here until I found this. :-O

Back to my test. It was clear that even though all the 4 NICs (both IPMP NICs on 024 and 027) are in the same broadcast domain (and actually that's true as the broadcast ping goes through, etc.) the multicast packets only passing through if it goes through the directly attached C4948, if it has to go through the C65XX, the multicast packets won't

arrive, they disappear. As the case is pretty similar to 028, I started to suspect the **C65XX device as a possible device to influence our multicast traffic**.

The mystery about the 224.0.0.0/24

It was pretty clear that whenever we use a 224.0.0.0/24 mcast address, it gets distributed correctly in the broadcast domain, but I needed more proof what feature does this, also I wanted proof that it's a conscious behaviour. I started to test multicast addresses randomly, when I realized the **all of the 32 addresses which translates to the 224.0.0.0's mac address actually passing by**! For example :

```
nxge0    224.0.0.130          255.255.255.255          01:00:5e:00:00:82
e1000g0  239.0.0.130          255.255.255.255          01:00:5e:00:00:82
nxge0    228.0.0.131          255.255.255.255          01:00:5e:00:00:83
```

You remember, I wrote before that each multicast mac address represents 32 IP addresses, as during the IP -> mac translation only the lowest 23 bits from the IP address is used. The 224.0.0.0/24 translates to 01:00:5e:00:00:XX, but as you see 32 x IPs are translates to these addresses as well, and those are passed too even though they were outside of 224! This was a real help, as I was able to tell that we have a L2 filtering in place, which makes decision based on L2 addresses.

Anyway, this really showed me that most likely we have a networking device feature which filters multicast traffic based on something (I had no idea what exactly was the logic behind it). So I narrowed down my list of possible issues to the C65XX and more specifically to 2 features of it. I want to emphasize that I **have no access to the networking devices** (I wish I have) so **what I write here is a theory** which can be used to diagnosing the issue further.

About SNUlbXUScingh0[34]

These two high-end catalyst switches are most likely the root bridge for this VLAN (I assume for all the VLANs), which means that the logical bridge tree structure depending on which two point is trying to communicate with each other, they have to pass this switch. This is especially true for multicast/broadcast packets, as those are intented to spread all over a broadcast domain. These 2 switches are configured in a redundant fashion using HSRP and they are connected via 2 x (2 x GE Etherchannel) connections, where **we have storm control configured** which is a simple technique **to limit broadcast/multicast storms**.

storm-control
(http://www.cisco.com/en/US/docs/switches/lan/catalyst6500/ios/12.2SXF/native/configuration/guide/storm.html)

This feature is a really simple feauture, it's a HW ASIC based filtering mechanism to **limit the rate of the broadcast/multicast packets on a specific port**. There is only one parameter to it, the percentage (comapred to the total bandwidth of an interface) of the traffic allowed to be broadcast/multicast (can be controlled separatedly). The monitoring tooks place on a per second basis, and if the amount of broadcast/multicast pass that threshold, the rest of the broadcas/multicast packets will be dropped until the end of the 1 sec period (where the measuring starts all over again). We have set this value for 10, which means we allow 10% of the traffic on each NIC to be broadcast/multicast, above that the switch drops these type of packets :

```
interface GigabitEthernet9/47
 description Routed Channel Interface to CGAlbMUScingh02 Gig 9/47
 no ip address
 storm-control broadcast level 10.00
 storm-control multicast level 10.00
 channel-group 1 mode desirable
!
```

(this is just an example, we have 4 interfaces in this scenario) Also would like two make **two notes**. **One** is that it **may worth having the multicast limit higher than the broadcast**, as L2 broadcasts are a subset of L2 multicasts, also my **second** not that even though we have etherchannel and storm-control together, **in this scenario Cisco does not recommend to configure storm-control at the physical interface level, but on the port channel interface** :

```
"Do not configure traffic storm control on ports that are members of an EtherChannel.
Configuring traffic storm control on ports that are configured as members of an
EtherChannel puts the ports into a suspended state. "
```

I may be wrong in some of these points, but what I'm really sure about is to **check if we had any multicast packets dropped because of the storm control**. (the interfaces store counters for dropped packets by storm control). As far as I know storm control can filter the 224.0.0.0/24 privileged multicast network as well in case the threshold has been reached (only the STP packets are passed through even above the threshold), and that's not something we experience in this case, so I would say storm control is less likely to be the source of our issue here, still, it's worth checking the storm control statistics for dropped multicast packets.

IGMP snooping
(http://www.cisco.com/en/US/docs/switches/lan/catalyst6500/ios/12.2SXF/native/configuration/guide/snooigmp.html)

My second suspect is **IGMP snooping**. Historically (as I wrote in my intro) switches used to flood multicast packets on all of their ports, so the NICs were able to decide whether to process the packet or just drop it. This behaviour is really nice but from a networking standpoint it's as stressful as a broadcast packet, so later on Cisco introduced the IGMP snooping. This is a feature where **switches will only forward multicast packets to a port if the switch saw an IGMP membership report on that port** (so it knows that behind that port the system is subscribed to a specific multicast group). IGMP stands for Internet Group Management Protocol, and it's a signalling protocol about multicast subscription/leave/query. IGMP snooping is enabled by default on C65XX, and I assume it's on in our case too on SNUIbXUScingh0[34]. This means that if the switch doesn't have knowledge about which ports are subscribed to which multicast group, the switch simply won't forward the multicast packet to that port.

Also, as this article about CGMP & IGMP proves
http://www.cisco.com/en/US/products/hw/switches/ps708/products_tech_note09186a00800b0871.shtml :

"As with CGMP, **GDAs that map to a MAC that falls in the range 01-00-5e-00-00-xx is never pruned by IGMP snooping.**"

If you read through my e-mail then you may remember when I found out that we have a L2 filter in place (but I couldn't identify which feature is responsible for it) passing all the multicast packets with the L2 address 01-00-5e-00-00-XX, even if the IP headers shows that the destination IP is above 224! That's exactly what we face here, so **IGMP snooping seems like a good match to our multicast issue**.

Now the question is, **where the IGMP membership reports are coming from**?

Solaris 10 and IGMP

Solaris 10 kernel supports IGMPv3 and by default does not send IGMP membership reports. It sends 1-2 packets whenever you join a multicast group (setsockopt()), but then it remains silent. Switches need constant updates about IGMP membership reports. If a system connected to a port stops advertising it's multicast membership, the switch will consider it as a leave from the multicast group, as such it will stop forwarding multicast packets to that port. The solution is the **IGMP membership query** (which will trigger IGMP membership reports from the OS), which should be coming from the IGMP enabled router or IGMP enabled L3 switch. If we don't have/use IGMP enabled routers, or the IGMP only needs to be L2 switched (which is the case here), there is a second option supported by C65XX, called **IGMP Snooping Querier.** The only major requirement to configure IGMP querier in a VLAN is to have an IP interface configured, and we have. If we enable **this feature** in this VLAN (340), the C65XX w**ill periodically send an IGMP membership query** to the whole broadcast domain (to 224.0.0.1 which is the all hosts multicast address), where systems will have the opportunity to reply (to 224.0.0.22) with a IGMP membership report indicating which multicast groups they are subscribed to. As both the query and the reply will spread in all over the broadcast domain, all the switches will know which ports are subscribed to which multicast groups, so they won't have problem forwarding them. Actually this IGMP Snooping Querier is just a trigger effect so the OS' can advertise what multicast groups they are subscribed to.

I **recommend to enable IGMP Snooping Querier** on SNUIbXUScingh0[34] for VLAN 340, so the Solaris systems can answer with their membership reports, and this will provide a perfect multicast flow within the broadcast domain. The CGMP & IGMP document I referred to earlier also mentions another option to us : if we don't want to enable IGMP Snooping Querier, we can still disable IGMP snooping. If we disable IGMP snooping, the switch will go dumb and will forward all the multicast packets on all the ports, as switches did before IGMP snooping was implemented. In short we either should enable IGMP Snooping Querier (as we have IGMP snooping enabled but nothing triggers the OS' to report multicast memership) or disable IGMP Snooping. Also, all of these options are available on a per interface/VLAN basis so we can make these changes only on VLAN 340 if we want to.

Also as a **last** suggestion -which is an **application suggestion**- as **WLS 10 supports unicast cluster communication**, it may with considering to use WLS 10 if possible (currently the business is working with WLS 9.2), as you can see multicast can be tricky sometimes, and unicast can make life easier. Obviously that's only a valid option if business supports it, other than that, we have to make an effort to provide a better environment for multicast applications.

Most of the things I wrote here are proved and tested, although as I had my limitations some are just suggestions/theories, especially the networking devices part as I have no access to those devices.

Summary

- recommend to **check the storm-control statistics** (if there are any multicast drops) on the interfaces between SNUIbXUScingh03 and SNUIbXUScingh04
- recommend to **check the etherchannel & storm-control configurations** (to have the storm-control on the port channel layer instead of the gig layer)
- recommend to **enable IGMP Snooping Querier** in VLAN 340 on SNUIbXUScingh0[34] (**or consider disabling**

IGMP Snooping on these switches)
- recommend to **evaluate the possibility of using Weblogic Cluster 10** with unicast cluster interconnect
- recommend to **check all the igmp & multicast stats** on the interfaces not only on the switches the systems are directly connected to but **on all switches part of the STP of VLAN 340**

Regards,
sendai