Troubleshooting EIGRP

Session 2207
Agenda

• Troubleshooting Common EIGRP Problems
  Neighbor Stability
  Stuck-in-Active Routes

• Troubleshooting Tools
  Event Log
  Debugs
  Topology Table
Troubleshooting Neighbor Stability

- Neighbor process—review
  Multicast hellos (by default)
  224.0.0.10 (0100.5e00.000a)

Neighbor timers
  Hello Interval—5 or 60 sec.
  Hold time—15 or 180 sec.

- EIGRP uses small Hello packets (74 bytes on Ethernet) to discover and retain neighbors.

- By default, hellos are sent to IP multicast address 224.0.0.10 (mac address 0100.5e00.000a.) Hellos are sent to unicast addresses when using NEIGHBOR statements (recent enhancement.) Neighbor statements will be covered at the end of this talk under new features.

- Hello and hold timers can be set independently on each interface of each router. Unlike OSPF, which requires the hello and dead timers to match on all routers sharing a subnet, EIGRP allows the timers to be set differently on different routers on the same subnet.

- Default values for the timers are:
  - Hello Interval - 5 seconds on point-to-point or high-speed links
    60 seconds on low-speed multi-point links
    (Note: low speed is <= T1)
  - Hold Time - 15 seconds on point-to-point or high-speed links
    180 seconds on low-speed multi-point links
Neighbor Process—Review

RTRA#show ip eigrp neighbors
IP-EIGRP neighbors for process 1

<table>
<thead>
<tr>
<th>H</th>
<th>Address</th>
<th>Interface</th>
<th>Hold (sec)</th>
<th>Uptime</th>
<th>SRTT (ms)</th>
<th>RTO</th>
<th>Q</th>
<th>Seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10.1.1.1</td>
<td>Et0</td>
<td>12</td>
<td>6d16h</td>
<td>20</td>
<td>200</td>
<td>0</td>
<td>233</td>
</tr>
<tr>
<td>1</td>
<td>10.1.4.3</td>
<td>Et1</td>
<td>13</td>
<td>2w2d</td>
<td>87</td>
<td>522</td>
<td>0</td>
<td>452</td>
</tr>
<tr>
<td>0</td>
<td>10.1.4.2</td>
<td>Et1</td>
<td>10</td>
<td>2w2d</td>
<td>85</td>
<td>510</td>
<td>0</td>
<td>314</td>
</tr>
</tbody>
</table>

• Most common method of determining state of health of neighbor relationships is with the command `show ip eigrp neighbors`

• **Hold Time** - length of time a router will wait to hear from a neighbor before declaring them down.

• **Uptime** - length of time we have been neighbors (this time!) A large value in this field means the neighbor relationship has been stable. A low value means the neighbor relationship has been reset at least once recently. You just don’t know how often or why. Yet.

• **Q count** - number of packets outstanding to send to this neighbor. If more than 0 consistently, there is a problem.
• Physical link state changes
• Hold timer expiring
• Exceeding the retry limit
• Manual changes
• Stuck-in-active routes

There are a number of reasons that EIGRP neighbors may have problems retaining stability. In this section we will review some of the most common. In the next, we will give examples of some more unusual problems seen between neighbors.
Physical Link State Changes

- Interface driver reports when a link goes down or comes up to EIGRP
- EIGRP takes neighbors down when the interface used to reach them goes down
- EIGRP (re)-initializes neighbors when a link comes up

- EIGRP will react to the notice from the lower layers that the interface has gone down. If you have a line continually bouncing up and down, EIGRP will work very hard to try to reflect the changing status to it’s neighbors, as well as trying to bring up and take down the neighbors that are reachable through the bouncing interface.

- Link problems are a normal part of life on a network (unfortunately!) but if you have continual link flapping, the routing protocol will spend so much time resolving the changes that the network could be impacted. Note that the same is true for OSPF or IS-IS, as well.
Hold Timer Expiring

- Hold time sent to neighbors inside the Hello packet
- Hold timer expires when an EIGRP packet is not seen for period of hold time

Usually caused by missing multicast Hello packets

Typically caused by congestion or physical errors

- The hold time value is included in the Hello packets a router sends to all of his neighbors. This allows you to have different hello/hold timers on different routers on the same network. Additionally, the hello/hold timers can be different on different interfaces on a router.

- When an EIGRP packet is received on a router from a neighbor, the hold timer for that neighbor begins decrementing, starting at the hold time value supplied by that neighbor. (If you do several `show ip eigrp neighbor` commands, you should see the value changing.)

- The Hold timer for each neighbor is then reset back to the hold time when another EIGRP packet is received from that neighbor. (At one time, it needed to be a Hello seen, but now any EIGRP packet will do.)

- Since Hellos are sent every 5 seconds on most networks, the Hold Time value in a `show ip eigrp neighbors` should normally be a value between 10 and 15 (resetting to hold time, decrementing to hold time minus hello interval or less, going back to hold time.)

- Why would a router not see EIGRP packets from a neighbor?
  - He may be gone (crashed, powered off, disconnected, etc.)
  - He (or we) may be overly congested (input/output queue drops, etc.)
  - Network between us may be dropping packets (CRC errors, Frame errors, excessive collisions)
Exceeding the Retry Limit

- Two types of packets in EIGRP—unreliable and reliable
  - Hellos and acks are unreliable
  - Updates, queries, and replies are reliable
- Reliable packets require an acknowledgement
  - If not acknowledged, packets are retransmitted, up to 16 times

- Exceeding the retry limit means that we aren’t getting reliable packets acknowledged from a neighbor. When a reliable packet is sent to a neighbor, he must respond with a unicast acknowledgement. If a router is sending reliable packets and not getting acknowledgements, one of two things are probably happening:
  - The reliable packet is not being delivered to the neighbor
  - The acknowledgement from the neighbor is not being delivered to the sender of the reliable packet
- These conditions are almost always due to problems with delivery of packets, either on the link between the routers involved or in the routers themselves. Congestion, errors, and other problems can all keep unicast packets from being delivered properly. Look for queue drops, errors, etc. when the problem occurs, and try to ping the unicast address of the neighbor to see if unicasts in general are broken or whether the problem is specific to EIGRP.
- You also see this symptom when a link with separate unidirectional connections is having a problem. For example, an ATM or frame-relay link with a PVC working one direction but not the other.
• 16 retransmits must occur and hold time period must expire before declaring the neighbor down

Retransmissions based on RTO, which is derived from SRTT

16 retransmits takes between 50 seconds and 80 seconds

• The Retransmit Timeout (RTO) is used to determine when to retry sending a packet when an acknowledgement has not been received, and is based on 6 X Smooth Round Trip Time (SRTT). The SRTT is derived from previous measurements of how long it normally takes to get acknowledgements from this neighbor. The minimum RTO is 200 msec and the maximum is 5000 msec. Each retry backs off exponentially.

• The minimum time required for 16 retransmits is therefore approximately 50 seconds (minimum interval of 200 ms with exponential backoff and a max interval of 5000 ms). For example, if there isn’t an acknowledgement after 200 ms, the packet is retransmitted and we wait 300 ms, then try again and wait 450 ms, then 675 ms, etc. until 5000 ms is reached. 5000 ms is then repeated until a total of 16 retransmissions have been sent.

• The maximum time for 16 retransmits is approximately 1 minute, 20 seconds, if the initial retry is 5000 ms and all subsequent retries are also 5000 ms.

• Since the Hold time is typically 15 sec on anything but low-speed NBMA, the hold time is normally not really a factor. NBMA links that are <T1 are really the only links which wait hold-time period before declaring a neighbor down due to retry limit exceeded. Later in this presentation we well show how this fact can be potentially harmful to network stability.
RTRA#show ip eigrp neighbors

<table>
<thead>
<tr>
<th>H</th>
<th>Address</th>
<th>Interface</th>
<th>Hold</th>
<th>Uptime</th>
<th>SRTT</th>
<th>RTO</th>
<th>Q</th>
<th>Seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10.1.1.1</td>
<td>Et0</td>
<td>12</td>
<td>6d16h</td>
<td>20</td>
<td>200</td>
<td>0</td>
<td>233</td>
</tr>
<tr>
<td>1</td>
<td>10.1.4.3</td>
<td>Et1</td>
<td>13</td>
<td>2w2d</td>
<td>87</td>
<td>522</td>
<td>0</td>
<td>452</td>
</tr>
<tr>
<td>0</td>
<td>10.1.4.2</td>
<td>Et1</td>
<td>10</td>
<td>2w2d</td>
<td>85</td>
<td>510</td>
<td>0</td>
<td>314</td>
</tr>
</tbody>
</table>

• This is typical output, with the first entry having a RTO (Retransmit Time Out) of 200, even though 6 X 20 (SRTT) is only 120. Again, the minimum is 200 ms.

• The second two neighbors each have an RTO of 6 X SRTT.
Manual Changes

- Manual changes which cause EIGRP neighbors to be reset:
  
  Summary changes
  Metric component changes
  Route filter changes

- Summary changes
  - When a summary changes on an interface, some of the old components may no longer be desired. Neighbors through that interface are reset to synch up topology entries.

- Metric component changes
  - If the delay or bandwidth value is manually changed on an interface, the neighbors known through that interface are bounced.

- Route filter changes
  - Similar to Summary explanation above. Neighbors are bounced if a distribute-list is added/removed/changed on an interface in order to synch up topology entries.
• **Stuck-in-active routes**  
  Often very complex problems  
  Will be covered in later section

• Stuck-in-active routes - Probably the most challenging problems to resolve in a large EIGRP network. In summary, when a route goes stuck-in-active, the router that declared the route stuck no longer trusts what he has learned from the downstream neighbor(s) that didn’t answer his query. (If he didn’t answer my question, what else isn’t he telling me?) In order to restore the neighbor relationship to one of trust, the neighbor is reset.

• Finding the cause of stuck-in-active routes will be covered in detail in a later section of this presentation.
Troubleshooting Tools for Neighbor Problems

RouterA#config terminal
Enter configuration commands, one per line. End with CNTL/Z.
RouterA(config) #router eigrp 1
RouterA(config-router) #eigrp log-neighbor-changes
RouterA(config-router) #logging buffered 10000
RouterA(config) #service timestamps log datetime msec
RouterA(config) ^Z
RouterA#

• EIGRP log-neighbor-changes is the best weapon you have to understand why neighbor relationships are not stable. It should be enabled on every router in your network. CSCdp89584 (12.1(2.1)) makes it the default behavior. As explained on an earlier slide, the uptime value from show ip eigrp neighbors will tell you the last time a neighbor bounced, but not how often or why. With log-neighbor-changes on and logging buffered, you keep not only a history of when neighbors have been reset, but the reason why. Absolutely invaluable!

• Logging buffered is also recommended, because logging to a syslog server is not bulletproof. For example, if the neighbor bouncing is between the router losing neighbors and the syslog server, the messages could be lost! It’s best to keep these types of messages locally on the router.

• It may also be useful to increase the size of the buffer log in order to capture a greater duration of error messages. You would hate to lose the EIGRP neighbor messages because of flapping links filling the buffer log! If you aren’t starved for memory, change the buffer log size using the command logging buffered 10000 in configuration mode.

• The service timestamps command above puts more granular timestamps in the log, so it’s easier to tell when the neighbor stability problems are occurring.
Log-Neighbor-Changes Messages

Neighbor 10.1.1.1 (Ethernet0) is down: peer restarted
Neighbor 10.1.1.1 (Ethernet0) is up: new adjacency
Neighbor 10.1.1.1 (Ethernet0) is down: holding time expired
Neighbor 10.1.1.1 (Ethernet0) is down: retry limit exceeded
Neighbor 10.1.1.1 (Ethernet0) is down: route filter changed
Neighbor 10.1.1.1 (Ethernet0) is down: interface delay changed
Neighbor 10.1.1.1 (Ethernet0) is down: interface bandwidth changed
Others, but not often...

• %DUAL-5-NBRCHANGE: IP-EIGRP 1: Neighbor 10.1.1.1 (Ethernet0) is down: peer restarted
  • The other guy reset our neighbor relationship. You need to go to him to see why he thought our relationship had to be bounced.

• %DUAL-5-NBRCHANGE: IP-EIGRP 1: Neighbor 10.1.1.1 (Ethernet0) is up: new adjacency
  • Established a new neighbor relationship with this neighbor. Happens at initial startup and after recovering from a neighbor going down. This may be the only neighbor message after SIA.

• %DUAL-5-NBRCHANGE: IP-EIGRP 1: Neighbor 10.1.1.1 (Ethernet0) is down: holding time expired
  • Went hold time period without hearing an EIGRP packet from this neighbor.

• %DUAL-5-NBRCHANGE: IP-EIGRP 1: Neighbor 10.1.1.1 (Ethernet0) is down: retry limit exceeded
  • This neighbor didn’t acknowledge a reliable packet after at least 16 retransmissions. (Actual number of retransmissions is also based on the hold time, but there were at least 16 attempts.)
Troubleshooting Tools for Neighbor Problems (Cont.)

rp-esc-2621b#debug eigrp packet hello
EIGRP Packets debugging is on (HELLO)
*Mar 16 19:08:38.521: EIGRP: Sending HELLO on Serial1/1
*Mar 16 19:08:38.521: AS 1, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
*Mar 16 19:08:38.869: EIGRP: Received HELLO on Serial1/1 nbr 10.1.6.2
*Mar 16 19:08:38.869: AS 1, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
*Mar 16 19:08:39.081: EIGRP: Sending HELLO on FastEthernet0/0
*Mar 16 19:08:39.081: AS 1, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
*Mar 16 19:08:39.749: EIGRP: Received HELLO on Serial1/2 nbr 10.1.7.2
*Mar 16 19:08:39.749: AS 1, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
*Mar 16 19:08:40.973: EIGRP: Sending HELLO on FastEthernet0/1
*Mar 16 19:08:40.973: AS 1, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
*Mar 16 19:08:43.409: EIGRP: Sending HELLO on Serial1/1
*Mar 16 19:08:43.409: AS 1, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0

• This debug should be non-disruptive (notice I said should!) since it will only produce output based on sending or receiving Hello packets.

• Pay attention to the amount of time between hellos sent or received. You will need to make sure you are recording timestamps in the debug output. You can do this by typing in the command `service timestamps debug datetime msec` in configuration mode.

• More information on debugs and ways to limit the scope of the output are covered later in this presentation.

• Another common method of troubleshooting EIGRP neighbor problems is to ping the multicast address 224.0.0.10. All neighbors should respond to this ping. If some neighbors do not respond to every ping to 224.0.0.10, there is a problem with their ability to receive multicast packets.
Unusual Neighbor Problems

- Unidirectional links
- Mismatched masks
- Mismatch of primary/secondary addresses

Next we’ll discuss some of the more unusual (but still seen fairly often) neighbor problems. Hopefully these examples will give you an idea of how to recognize how to find and fix neighbor problems.
• In this example, we see what happens when only one direction of traffic is working. This symptom is very similar to what you see when there is a duplicate IP address or wedged input queues on a router, or other reasons packets cannot be delivered in one direction on a link.

• The router on the left doesn’t even realize that the router on the right exists. RTRA is sending out his hellos, waiting for a neighbor to show up on the network. What he doesn’t realize is that the rtrB is already out there and trying to bring up the neighbor relationship.

• RtrB, on the other hand, sees the hellos from rtrA, sends his own hellos and then sends an update to rtrA to try to get their topology tables/routing tables populated. Unfortunately, since the updates are also not being received by rtrA, it of course isn’t sending acknowledgements. RTRB tries it 16 times and then resets his relationship with rtrA and starts over.

• You’ll spot this symptom by the “retry limit exceeded” messages on rtrB, rtrB having rtrA in his neighbor table with a continual Q count, and rtrA not seeing rtrB, at all.

• This is a pretty common symptom.
• In this example, we see what happens when addresses are misconfigured with the mask set differently on each end of a serial link.

• Both routers see each other in the neighbor table, but both show something continually in the Q count. We have reachability but EIGRP doesn’t like it, for some reason.

• What happens is that multicast traffic is received fine on both routers, but when unicast traffic from rtrA to rtrB is sent, it is sent off into the ozone layer. Why?

• Routing typically looks for most specific route, so when a unicast packet is sent from rtrA to rtrB (target address 10.1.1.1), it is received by rtrB, who looks in its routing table for the best route to that subnet.

• Believe it or not, the route learned through EIGRP for rtrA’s /25 interface is more specific than rtrA’s own /24 interface, so rtrB will forward it on out to rtrC!

• Both routers will show “retry limit exceeded” for updates sent to each other. Without the ability to deliver unicast traffic, all updates fail.

• Note that if the routers are not operating in fast-switching mode, this problem should not happen. (CEF or process will not forward packet addressed to the router.)
Another unusual, but sometimes seen problem is when a network uses secondary addresses and are inconsistent in the use of which address space is the primary. Often you'll see it when a new router is added to a network with primary and secondary addresses and the new router is only added to the secondary.

In this network, rtrC was added and starts whining about subnet problems! Both rtrA and rtrB see rtrC in their neighbor tables (but with the 50.x.x.x address). EIGRP verifies that a hello is received from a neighbor that belongs on this interface, but doesn’t really pay attention to whether the address is primary or secondary.

RtrC on the other hand, keeps getting hellos from someone who doesn’t even belong on this network! It lets you know in no uncertain terms by continually screaming out “not on common subnet”.

In earlier versions, rtrC would just silently discard the hellos and you would not be able to tell why the neighbors won’t come up. In the most recent versions, you can turn this error message off (if your network uses the same physical network for multiple IP subnets (not a great idea, but sometimes seen).
Agenda

• Troubleshooting Common EIGRP Problems
  Neighbor Stability
  Stuck-in-Active Routes
• Troubleshooting Tools
  Event Log
  Debugs
  Topology Table
Stuck-in-Active Routes (SIA)

%DUAL-3-SIA: Route 10.64.5.0 255.255.255.192 stuck-in-active state in IP-EIGRP 100. Cleaning up

- Indicates at least two problems
  A route went active
  It got stuck!

- Some of the most challenging problems to troubleshoot on an EIGRP network are when you start receiving “DUAL-3-SIA” messages and resetting neighbors.

- Stuck-in-active (SIA) routes are always an indication that a couple of bad things have happened. Routes went active and then got stuck. These two things will be explained on the following slides.
  - If you receive an SIA message very seldom, you should try to isolate the cause, but it is not a fatal condition. It is an error condition, however, and should be investigated.
  - If SIAs occur too frequently, they can completely cripple a network. I’ve seen networks which are totally unable to converge because they continually go SIA on routes and reset neighbors. More on this later.
Review of Active Process

• Going “active” is the normal process for resolving network topology changes

• Normal (stable) state of a route is passive

• Route becomes “active” if it is lost and no other successor or feasible successor exists

• When a network is stable and no routes are in transition, EIGRP is the quietest protocol known to man (except static routes, and they don’t count)
  • If routes are not in transition, they are considered “passive” and are never refreshed. EIGRP doesn’t use any type of refresh like RIP (30 seconds), IGRP (90 seconds), or OSPF (yes, even OSPF refreshes routes every 30 minutes!)

• The way that EIGRP can get by without refreshes is that it takes action whenever a change occurs to both notify others of the change as well as to look for alternative paths to bypass the failed path. This process of looking for an alternative path is known as “going active”.

• The route can also go active if the metric changes for the worse, because if the metric through the current successor becomes worse, another path may be better than the one that just changed. EIGRP actively goes looking to see if a better alternative exists.
• As this map shows, when the route on the left disappears, queries are sent trying to find an alternative path to reach the target network. RtrA sends queries to RtrB and RtrC and then waits for the replies to come back.

• When RtrB and RtrC receive the queries, they look in their topology table for an alternative path and don’t find one. RtrB and RtrC then send queries on to RtrD, RtrE, RtrF, and RtrG. And so on and so forth, to the ends of the earth (sort of).
Active Process (Cont.)

• Query process stops when:
  All queries are answered
  End of network is reached
  End of the autonomous system is reached (sort of)
  The lost component is unknown

• The query process will continue on and on, until each query path terminates due to one of these reasons:
  • A reply is returned for each query
  • The end of the network is reached (no one left to ask!)
  • The end of the autonomous system is reached (no one in our AS left to ask!)
    • This doesn’t mean multiple EIGRP AS’ really helps!
  • Where the component is unknown
    • If I don’t have a topology table entry for the component, I answer with an unreachable metric
    • I wouldn’t know about the component if:
      • It was blocked with a distribute list
      • It was blocked by summarization
When the Active Process Fails!

- When a route goes active, timer started
  Approximately 3 to 3-1/2 minutes
- If timer expires without all queries being answered, “stuck” in the active process

- Since a router will not be able to take action to install a new successor until it has received replies from all neighbors it has asked, it sets a maximum amount of time it will wait. This is the “Active time” and defaults to 3 minutes (which is actually closer to 3 1/2 due to jittering of the timer, with a minimum value of 3 minutes.)

- If 3+ minutes has gone by without getting all of the replies, something is drastically wrong.
  - 3+ minutes is a very long time to a router - even on large networks, answers should be received well before 3 minutes.

- Since the network is now in an unknown state, EIGRP has to do something to restore it to a known state. This “correction” can be fairly intense.

- You can tell you are getting SIA routes by the following message appearing in the syslog, buffer log, and console:

- Oct 1 14:26:00: %DUAL-3-SIA: Route 10.64.5.0 255.255.255.192 stuck-in-active state in IP-EIGRP 100. Cleaning up
Stuck-in-Active (Cont.)

- On the router where timer expires:
  - Reinitializes neighbor(s) who didn’t answer
  - Goes active on all routes known through bounced neighbor(s)
  - Re-advertises to bounced neighbor all routes that we were advertising

- In order to restore a semblance of order to EIGRP’s world (since the unexpected happened and a reply wasn’t received in time), the router where the Active timer expired:
  - Reinitializes the neighbor(s) who didn’t send back replies in the allotted 3+ minutes.
  - Goes active on the routes which had the bounced neighbor as successor (assuming no alternative or feasible successors).
  - When re-initializing, re-advertise all of our routes to him, again.

- SIA’s occur on just about every network running EIGRP on the planet at some time or another. Just realize that if you see an SIA message, it is indicating to you that something isn’t right with your network. If it happens very occasionally, it probably isn’t anything to be overly concerned with. If it happens regularly, however, you need to track down the cause and resolve it.
Likely Causes for Stuck-in-Active

- Bad or congested links
- Query range is too long
- Excessive redundancy
- Router memory shortage
- Software defects (very seldom)

Remember that the location of the actual cause of the SIA route is likely to be far from the location where the SIA message and bounced neighbors happen! Some of the possible causes are:

- If a router has links that are either experiencing high CRC or other physical errors or are congested to the point of dropping a significant number of frames, queries, replies, or acknowledgements could be lost.

- If the time it takes for a query to go from one end of the network to the other is too long, the active timer could expire before the query process completes. I don’t think I’ve ever seen a network where this is true, by the way.

- If the complexity in the network is too great due to excessive redundancy, EIGRP could be required to work so hard at sending and replying to queries that it cannot complete them in time.

- If a router is low on memory, it may be able to send hellos, which are very small, but be unable to send queries or replies. This can certainly lead to SIAs

- CSCdi83660 identified a problem with EIGRP where misordered topology table entries caused SIAs. NOT the normal circumstance of an SIA, however.
Troubleshooting SIAs

- Two (probably) unrelated causes of the problem—**stuck and active**
- Need to troubleshoot both parts
  - Cause of active often easier to find
  - Cause of stuck more important to find

- If routes never went active in the network, we would never have to worry about any getting stuck! Unfortunately, in a real network there are often link failures and other situations that will cause routes to go active. One of our jobs is to minimize them, however.

- If there are routes that regularly go active in the network, you should absolutely try to understand why they are not stable. While you cannot ensure that routes will never go active on the network, a network manager should work to minimize the number of routes going active by finding and resolving the causes.

- Even if you reduce the number of routes going active to the minimum possible, if you don’t eliminate the reasons that they get stuck you haven’t fixed the most important part of the problem. The next time you get an active route, you could again get stuck.

- The direct impact of an active route is small. The possible impact of a stuck-in-active route can be far greater.
Troubleshooting the Active Part of SIAs

- Determine what is common to routes going active
  - Flapping link(s)?
  - From the same region of the network?
  - /32s from dial-in PPP?

- Looking at the syslog may or may not tell you which routes are going active, causing you to get stuck. Since the SIA message reports the route that was stuck, it seems rather straightforward to determine which routes are going active. This is only partially true. Once SIAs are occurring in the network, many routes will go active due to the reaction to the SIA. You need to determine which routes went active early in the process in order to determine the trigger.

- Additionally, you can do show ip eigrp topology active on the network when SIAs are not occurring and see if you regularly catch the same set of routes going active regularly.

- If you are able to determine which routes are regularly going active, determine what is common to those routes. Are they /32 routes created from PPP? This is a common cause of routes going active as dial-in users connect and disconnect.

- Are routes flapping (bouncing up and down) causing the route that is bouncing (and everything behind it!) to regularly go active?

- Are most or all of the routes coming from the same area of the network? If so, you need to determine what is common in the topology to them so that you can determine why they are not stable.
• Show `show ip eigrp topology active` 
  Useful only while the problem is occurring 
  If problem isn’t occurring at the time, it is difficult to find the source of route getting stuck

• Our best weapon to use to find the cause of routes getting stuck-in-active is the command `show ip eigrp topology active`. It provides invaluable information about routes that are in transition. Examples of the output of this command and how to evaluate it will be in the next several slides.

• Unfortunately, this command only shows routes that are currently in transition. It probably isn’t useful after the fact when you are trying to determine what happened earlier. If you aren’t chasing while the problem is occurring, there aren’t really any tools that will help you find the cause.
Why Is RTRA Reporting SIA Routes? Let’s Look at a Problem in Progress…

• In our example network, we’ve noticed dual-3-sia messages in the log of RTRA and we know the trigger is an unstable network off of this router. Instead of just shutting down the unstable link, we decide to try to determine the cause of the stuck part of stuck-in-active.

• In the above output, we see that RTRA is active on the route 20.2.1.0/24 (note the “A” in the left column) and has been waiting for an answer from 10.1.1.2 (RTRB) for 1 minute and 12 seconds. We know that we are waiting on RTRB because of the lower case “r” after the IP address. Sometimes, the lower case “r” comes after the metric in the upper part of the output (not under “Remaining replies”). Don’t be fooled. The lower case “r” is the key, not whether it’s under the “Remaining replies” or not.

• Since we know why we are staying active on the route because RTRB hasn’t answered us, we need to go to him (RTRB) to see why he’s taking so long to answer.
• We repeat the `show ip eigrp topology active` command on RTRB and we get the results seen above.

• We see that RTRB probably isn’t the cause of our stuck-in-active routes, since he is also waiting on another router downstream to answer his query before he can reply. Again, the lower case “r” beside the IP address of 10.1.3.2 tells us he is the neighbor slow to reply.

• We now need to go to 10.1.3.2 (RTRD) and see why he isn’t answering RTRB.
On RTRD we repeat the `show ip eigrp topology active` command and see what he thinks of the route.

Again, he’s waiting on another neighbor downstream to answer him before he can answer RTRB. You are probably getting the idea of how exciting this process can be! Of course, in a real network you probably have users/managers breathing down your neck making it a bit more exciting.

You may have also noticed that both RTRB and RTRC appear in the topology table with infinity metrics and no lower case “r”. Even though our chasing of the active route didn’t take us down that path, a query was also sent from RTRB to RTRC, who forwarded it on to RTRD. One of the rules of dual is that if you receive a query and you’re already active, you answer with infinity. Thus RTRD, who was active based on the query from RTRB, replied with infinity to RTRC, who replied to RTRB. Therefore that leg of the query path completed normally.

As I’m sure you suspect our next step should be to see why 10.1.5.2 (RTRE) isn’t answering RTRD’s query.
• And again, we look at the active topology table entries, this time on RTRE.
• Wait! RTRE isn’t waiting on anyone for any routes! Did the replies finally get returned and the route is no longer active? We need to go back to RTRD and see if he is still active on the route.
Chasing Active Routes (Cont.)

- Hmmm, RTRD still thinks the route is active and it’s gotten even older.
- There appears to be a problem, Houston! RTRD thinks he needs a reply from RTRE, yet RTRE isn’t active on the route. We need to take a look at the neighbor relationship between these two routers to try to identify what is going wrong.
• It appears that RTRD is having a bit of a problem communicating with RTRE! The neighbor relationship isn’t even making it completely up based on the Q count on RTRD. We also notice in the log that the neighbor keeps bouncing due to "retransmission limit exceeded".

• Now we need to use our normal troubleshooting methodology to determine why these two routers can’t talk to each other properly.
• How does basic connectivity look? A ping between RTRD and RTRE isn’t succeeding either! We’ll need to find out why they can’t talk to each other.

• Whatever is causing them to not talk to each other is undoubtedly a contributing factor to the SIAs we’re seeing in the network. We need to find and fix the problem with this link and remove the cause of the SIA routes.
Troubleshooting the Stuck Part of SIAs (Cont.)

- It’s not always this easy to find the cause

- Sometimes you chase the waiting neighbors in a circle

  If so, summarize and simplify!

- Our example of chasing SIA routes was intentionally made very easy in order to demonstrate the tools and techniques. In a real event on a network, there would probably be many more routes active, and many more neighbors replying. This can make chasing the waiting neighbors significantly more challenging.

- Usually, you will be able to succeed at tracking the waiting neighbors back to the source of the problem. Occasionally, you can’t. On highly redundant networks, in particular, you can find yourself chasing neighbors in circles without reaching an endpoint cause of the waiting. If you run into this case, you may need to reduce the redundancy in order to simplify the network for troubleshooting and convergence.
Minimizing SIA Routes

- Decrease query scope (involve fewer routers in the query process)
  
  Summarization (manual or auto)
  Distribute-lists
  Define remote routers as stubs

- We’ve now talked about the impact that SIA routes can cause on your network and how to track down the cause of SIA events. While you may not be able to completely rid your network of SIA routes, there are techniques you can use to minimize your exposure.

- Decrease query scope - In our example network, you saw the queries sent to each router in a chain. As explained earlier, if a router received a query on a route that it doesn’t have in its topology table, it immediately answers and doesn’t send the query onward. This is a very good thing! You do this through:
  
  - Summarization - auto-summary if not discontinuous. If discontinuous, readdress and then auto-summary! Use manual summary to summarize within a major network or to summarize external routes.
  
  - Distribute-lists - used to limit knowledge of routes. Particularly on dual-homed remotes, which tend to reflect all routes back to the other leg of the dual home connection.
  
  - Use hierarchy - if the network doesn’t have hierarchy, the two techniques above cannot adequately be used.
• In the sample network above, each dual-home remote router would be seen as an alternative path to 10.1.8.0 from RTRA unless information hiding techniques are used.

• In the above failure example, not only are the remote routers required to respond to queries from the distribution layer, they also continue the search by “reflecting” the queries back toward the distribution layer. This significantly complicates the convergence process on the network.

• With our example of only two distribution and three remotes, it’s not all that significant. On a real network with possibly hundreds of remotes, it can be brutal.

• And think about it; in most networks the designer put dual legs to remotes in order to improve their uptime reaching the remainder of the network. Rarely if ever does a designer desire for traffic to go from the distribution layer to the remote and back, so why is convergence acting as if this is a valid alternative path? We didn’t tell it any differently, that’s why.
• By adding summary-address commands on the outbound interfaces of RTRA and RTRB, components are not sent to the remote routers at all, so they will not reflect the queries back to the distribution layer. As we mentioned in an earlier slide, if a router doesn’t know a specific component it receives a query about, it immediately replies with unreachable.

• Likewise, distribute-list out commands would be installed at the remote routers to limit their advertisements to only those networks that exist at that remote site. Therefore, they won’t even reflect the summary route from RTRA back to RTRB, nor will they reflect the summary route from RTRB back to RTRA. This will minimize the part the remote routers play in the update and query process and will increase the stability and scalability of this network.
• In the example above, the remote routers are defined as Stub routers. When Router A and Router B see that the remotes are stubs, they flag them in their neighbor table and block all queries from flowing from the distribution layer to the remote routers.

• The downside of this approach is that it is only pertinent to sites which are truly stubbed; i.e., single or dual paths into the distribution layer but without any routers behind them. A site is a stub site if it is effectively the “end” of the network.

• Work is being done on expanding this definition of stubbing to support something like stub areas, but the kinks haven’t yet been worked out on how this will work. I wouldn’t expect to see this feature for quite a while (assuming we figure out how to actually make it work correctly!)
**Decreasing Query Scope**

**Stub Routers**

- Defined on remote routers
- Restricts route advertisement to connected, static, summary, or none
- Queries are not propagated to stub routers

---

- EIGRP Stub support allows you to define your remote routers so that they only advertise connected, static, summary, or none (depending on the configuration) back to the distribution layer. This will eliminate the problem described earlier with routes “reflecting” through the remote routers as if they were intended to be transit.
  - This would take the place of defining the “distribute-list out” on the remote routers advertising only local routes.
- Additionally, the distribution layer router will see in the received hello that the remote is a stub, so it will not send a query to him about any route loss in the remainder of the network. This is a major improvement, since there has not been any way up to now to stop queries from flowing to the remotes!
• Maintain reasonable redundancy
  Don’t make EIGRP’s job too difficult
  Use passive-interface
  Use hierarchy

• Redundancy is good, but excessive redundancy is not. If EIGRP has to converge across many, many paths with every routing change, the probability of a problem in the convergence is increased. An example of this is on the next slide.

• Some of the tools available to decrease redundancy are the passive-interface and implementation of topological hierarchy. The next slide shows an example of using the passive-interface command to decrease redundancy without hurting functionality.
• In the above example, the four Ethernets on the left (which could be VLANs through a switch or Token Ring networks or any other media, for that matter) are there to provide users with access to the network. There are two routes connected in order to provide redundancy (via HSRP) so that the users will have failover capability if there is a problem.

• If you don't put passive-interface statements under EIGRP, however, each of these user segments will be treated by EIGRP as a possible alternative path to reach every route throughout the network! It is rarely the network designers goal to have these user segments be treated as transit paths to reach other parts of the network for anyone other than the users that reside on that segment, but EIGRP doesn’t know what the designer intended, only what he/she configured.

• By putting passive-interface statements under the ROUTER EIGRP <AS> in the configuration, EIGRP will no longer establish a neighbor relationship across these subnets and will thus not treat them as transit paths.
Minimizing SIA Routes (Cont.)

• Multiple EIGRP AS’ are not the answer to minimizing query scope
  Terminates original query, but new one starts
  Adds redistribution complexity
  Requires distribute-lists to stop routing loops

• One technique used by some network designers to decrease query scope is to define multiple EIGRP autonomous systems (AS) and redistribute between them. In this way, they are treating EIGRP as if it were OSPF with areas and area borders. Unfortunately, this does not achieve the desired results.

• As explained earlier in this section, a query will stop at the end of the autonomous system. Unfortunately, a new query will start in the second autonomous system and will continue to flow using the same methods as if it were one AS.

• Mutual redistribution between multiple AS’ can create redistribution problems which are difficult to find and stop. Distribution-lists will need to be installed in order to stop routing loops and can be administratively burdensome.

• Summarization and Distribute-lists are far more effective at bounding query scope on an EIGRP network.
As you can see from the process described in this slide, the original query hits the boundary between AS 1 and AS 2, a reply is generated in AS1, but the query is restarted in AS 2. It then propagates across AS 2 until it reaches the boundary with AS 3, where it is again answered in AS 2 and a new Query is started in AS 3.

While this changes the timing significantly, and moves the problem toward the real cause of the “stuck”, it doesn’t resolve the problem nearly as well as using information-hiding techniques.
Impact of Low-Speed NBMA Links

- Retry limit = hold time
- Hold time = 180 seconds
- Active timer = 180 seconds
- One broken link can cause SIAs!

As explained in an earlier slide describing the problem of exceeding the retry limit for reliable packets, a packet is sent 16 times if it's not acknowledged and then we give up, if the hold time is also exceeded. Since low-speed NBMA networks typically have a hello interval of 60 and hold time of 180, the actual amount of time to take down a neighbor due to retry limit exceeded for low-speed NBMA is 180 seconds.

• Of course, the Active timer for SIA routes is also approximately 180 seconds (plus a little jitter factor). This means that if you have a low-speed NBMA link that is not getting packets across (or router out of memory, or any of the other possible causes already discussed), then the low-speed NBMA neighbor will not be killed until 3 minutes have passed.

• This could be enough to cause an SIA elsewhere in the network if a route happened to go active!
• Interface goes down on rtrA
  • Active timer starts
• Queries fan out through the network
• One Query isn’t getting acknowledged by rtrD
  • Active timer still ticking on rtrA
  • Retry timer started on rtrC
• Queries retried 16 times by rtrC
  • rtrC kills rtrD after retry limit (holdtime = 3 minutes)
• Meanwhile, Active timer expires on rtrA, so he kills rtrB!

• The moral is: if you use low-speed NBMA links in your network, you should manually change the hello interval and hold time to a lower value than 60/180. A Hello interval of 30 and hold time of 90 would be a good place to start!
Workarounds for Low-Speed NBMA

- Use point-to-point subinterfaces instead of multipoint
- Change the active timer to 4-5 minutes (not recommended)
- Change the hello/hold timers to 30/90

Since this situation only occurs if multipoint NBMA interfaces are used, it may be possible to change the network design to use point-to-point subinterfaces, instead. If point-to-points are used, however, more addresses are needed and more routes are propagated into the network, so there are certainly trade-offs.

Another alternative sometimes taken by customers is to change the active timer to a larger number (using the `timers active-time 5` under router eigrp.) Since this value defines the outer boundary for EIGRP to remain unconverged, we don't recommend using this tactic.

A better alternative is to change the hello and hold timers on the multipoint NBMA link to 30 second hello and 90 second hold timers. This should give a much better chance that the retransmit limit will be exceeded before the active timer expires elsewhere in the network.
EIGRP SIAs—Hope for the Future

• Development engineering working on significantly modifying the SIA process

  Would retry queries if no reply

  Give the neighbor a chance to say “I’m still working on it!”

  Should push failure point closer to where the problem actually exists

• In order to further EIGRP’s ability to provide stability in the real world, the EIGRP developers are working on a drastic change in the way the active process works. Assuming it works ;-) it should make life much better.

• Since the feature is currently being created, I don’t want to supply any details here, but in general it will work like this:
  • If a query isn’t replied to, send a follow-up query to the router that didn’t reply.
  • If he’s still working on it (waiting on an answer himself), don’t kill him.

• This should let those not near the cause of the “stuck” to stay up and working, and push the point of neighbor execution closer to the actual cause of the instability.

• No expected delivery date for the feature, but it certainly won’t appear first in mainline code.
Agenda

• Troubleshooting Common EIGRP Problems
  Neighbor Stability
  Stuck-in-Active Routes

• Troubleshooting Tools
  Event Log
  Debugs
  Topology Table
EIGRP Troubleshooting Tools

• Debugs versus the EIGRP event log
  
  On a busy, unstable network debugs can be hazardous to your health!
  
  Event log is non-disruptive—already running!
  
  Not for mere mortals to interpret!

• Two weapons at your disposal are debugs and the event log. Realize that the output of both debugs and the log are cryptic and probably not tremendously useful to you (so why am I telling you about them?)

• There are times when the output of debugs or the event log is enough to lead you in a direction, even if you don’t really understand all that it is telling you. Don’t expect to be an expert at EIGRP through the use of debugs or the event log, but they can help.

• Don’t forget, debugs can kill your router! Don’t do a debug if you don’t know how heavy the overhead is. I may tell you below about some debugs, but don’t consider this approval from the TAC to run them ;-)

• The event log is non-disruptive (except for one nasty bug seen in 11.2(15)) so it is much much safer. Just display it and see what’s been happening lately!
Event Log

- Always running (unless manually disabled)
- Default 500 lines (configurable)
- Most recent events at top of log

- Separate event log is kept for each AS.
- 500 lines are not very much! On a network where there is significant instability or activity, 500 lines may only be a second or two. You can change the size of the event log (if needed) by the command:
  - `eigrp event-log-size ##` (where ### are number of lines)
  - If number of lines set to 0, disables log.
- You can clear the event log by typing:
  - `clear ip eigrp event`
- Most recent at the top of the log, so time flows from bottom to top.
• Three different event types can be logged

\textit{EIGRP log-event-type}
\texttt{[dual][xmit][transport]}

Default is dual—most useful

Any combination of the three can be on at the same time

• I’ve never actually used the xmit or transport log to solve a problem. The dual log is normally what is useful for most problems.
RtrA#show ip eigrp events
Event information for AS 1:
1  01:52:51.223 NDB delete: 30.1.1.0/24 1
2  01:52:51.223 RDB delete: 30.1.1.0/24 10.1.3.2
3  01:52:51.191 Metric set: 30.1.1.0/24 4294967295
4  01:52:51.191 Poison squashed: 30.1.1.0/24 lost if
5  01:52:51.191 Poison squashed: 30.1.1.0/24 metric chg
6  01:52:51.191 Send reply: 30.1.1.0/24 10.1.3.2
7  01:52:51.187 Not active net/1=SH: 30.1.1.0/24 1
8  01:52:51.187 FC not sat Dmin/met: 4294967295 46738176
9  01:52:51.187 Find FS: 30.1.1.0/24 46738176
10 01:52:51.187 Rcv query met/succ met: 4294967295 4294967295
11 01:52:51.187 Rcv query dest/nh: 30.1.1.0/24 10.1.3.2
12 01:52:36.771 Change queue emptied, entries: 1
13 01:52:36.771 Metric set: 30.1.1.0/24 46738176
14 01:52:36.771 Update reason, delay: new if 4294967295
15 01:52:36.771 Update sent, RD: 30.1.1.0/24 4294967295
16 01:52:36.771 Update reason, delay: metric chg 4294967295
17 01:52:36.771 Update sent, RD: 30.1.1.0/24 4294967295
18 01:52:36.771 Route install: 30.1.1.0/24 10.1.3.2
19 01:52:36.767 Find FS: 30.1.1.0/24 4294967295
20 01:52:36.767 Rcv update met/succmet: 46738176 46226176
21 01:52:36.767 Rcv update dest/nh: 30.1.1.0/24 10.1.3.2
22 01:52:36.767 Metric set: 30.1.1.0/24 4294967295

• The above example shows how clear the information is conveyed! I particularly like the poison squash!
SIA Event Log

• **Show ip eigrp sia**
  
  Snapshot of event log when SIA occurs
  
  One-shot log—once taken, not taken again unless cleared
  
  Typically lets you know results of SIA, not the cause
  
  Seldom useful

• I’m always more interested in what caused the SIA, rather than what we did about it. By time we store the SIA log, the interesting stuff is long gone!
• **Remember—can be dangerous!**

  Use only in the lab or if advised by the TAC!

• **To make a little safer:**

  `logging buffered <size>`

  `no logging console`

• By enabling logging buffered and shutting off the console log, you improve your odds of not killing your router when you do a debug. Still no guarantees!

• We often change the scheduler interval when we do debugs in the TAC, as well. This command is version dependent, so I’m not going to give you syntax here.
• Use modifiers to limit scope of route events or packet debugs

  Limit to a particular neighbor

  `debug ip eigrp neighbor AS address`

  Limit to a particular route

  `debug ip eigrp AS network mask`

• Both packet debugs and route event debugs create so much output that you would have a hard time sorting through it for the pieces you care about. The two modifier commands above allow you to limit what the debug output will show.

• `debug ip eigrp AS <network><mask>` will limit the output to only those entries that pertain to the route identified.

• `debug ip eigrp neighbor AS address` will limit the output to those entries pertaining to a particular neighbor.

• Unfortunately, you have to enable the debug (packet or route events) prior to putting the modifier on, so you could kill your router before you are able to get the limits placed on the output. Sorry, but that’s the way it is.

• Examples of route events with modifiers will be in the next few slides.
• In this debug, we are looking at route events recorded when neighbors are cleared (in reality, the debugs produced were far, far more. This is only a snapshot of the debug.)  A modifier was included to limit the output to only the events related to a single EIGRP neighbor, 10.1.6.2.

• Notice that the debug output doesn’t identify which neighbors are involved in any of the events. Without the use of the modifier, you really can’t tell which neighbors you are interacting with in the debug output.

• This output is often useful when trying to determine what EIGRP thinks is happening when there are route changes in the network.
• Again, this is the output of debugging routing events, this time modified to only display output related to a single route in the network. This modifier can be very useful when trying to troubleshoot a single route (or representative route).
This debug is used often in a variety of problems and circumstances. Debug eigrp packet hello is used to troubleshoot neighbor establishment/maintenance problems.

Debug eigrp packet query, reply, update, etc are also often used to try to determine the process occurring when a problem occurs. BE CAREFUL! I've crashed/hung more than one router by doing a debug on a router that was too busy.

Probably the most commonly used debug eigrp packet option is “terse”, which includes all of the above except hellos. An example follows on the next page.
RtrA#debug eigrp packet terse
EIGRP Packets debugging is on
(UPDATE, REQUEST, QUERY, REPLY, IPXSAP, PROBE, ACK, STUB)

EIGRP: Enqueueing UPDATE on Serial1/0 iidbQ un/rely 0/1 serno 19707-19707
EIGRP: Enqueueing UPDATE on Serial1/1 iidbQ un/rely 0/1 serno 19707-19707
EIGRP: Enqueueing UPDATE on Serial1/2 iidbQ un/rely 0/1 serno 19707-19707
EIGRP: Enqueueing UPDATE on Serial1/0 nbr 10.1.1.2 iidbQ un/rely 0/0 peerQ un/rely 0/0 serno 19707-19707
EIGRP: Enqueueing UPDATE on Serial1/1 nbr 10.1.2.2 iidbQ un/rely 0/0 peerQ un/rely 0/0 serno 19707-19707
EIGRP: Enqueueing UPDATE on Serial1/2 nbr 10.1.3.2 iidbQ un/rely 0/0 peerQ un/rely 0/0 serno 19707-19707
EIGRP: Sending UPDATE on Serial1/0 nbr 10.1.1.2
  AS 1, Flags 0x0, Seq 2831/1329 iidbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1 serno 19707-19707
EIGRP: Sending UPDATE on Serial1/1 nbr 10.1.2.2
  AS 1, Flags 0x0, Seq 2832/1708 iidbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1 serno 19707-19707
EIGRP: Sending UPDATE on Serial1/2 nbr 10.1.3.2
  AS 1, Flags 0x0, Seq 2833/1680 iidbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1 serno 19707-19707
EIGRP: Received ACK on Serial1/0 nbr 10.1.1.2
  AS 1, Flags 0x0, Seq 0/2831 iidbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
EIGRP: Serial1/0 multicast flow blocking cleared
EIGRP: Received ACK on Serial1/1 nbr 10.1.2.2
  AS 1, Flags 0x0, Seq 0/2832 iidbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
EIGRP: Serial1/1 multicast flow blocking cleared
Debug IP EIGRP Notifications

rp-esc-2621b#debug ip eigrp notifications
IP-EIGRP Event notification debugging is on
rp-esc-2621b#clear ip route *
rp-esc-2621b#
*Mar 17 15:58:07.144: IP-EIGRP: Callback: reload_iptable
*Mar 17 15:58:08.148: IP-EIGRP: iptable_redistribute into eigrp AS 1
*Mar 17 15:58:12.144: IP-EIGRP: Callback: redist frm static AS 0 100.100.100.0/24
*Mar 17 15:58:12.144: into: eigrp AS 1 event: 1
*Mar 17 15:58:12.144: into: eigrp AS 1 event: 1

- Debug ip eigrp notifications is used to troubleshoot problems with redistribution into EIGRP. The callbacks above describe what is happening between EIGRP and the routing table as routes are redistributed.
• Debug eigrp fsm is very, very similar to dual event log. Since the dual event log is non-disruptive and this debug could certainly cause problems, I rarely use this debug.

• FSM stands for Finite State Machine, which describes the behavior of DUAL, the path selection part of EIGRP.
• This debug describes the process taken when a summary (manual or auto) is changed on a router. Again, this debug is useful when you are having problems with summaries not being created (or deleted) when they are supposed to.
The topology table is probably the most critical structure in EIGRP.

- Contains building blocks used by DUAL.
- Used to create updates for neighbors/populate routing table.

Understanding topology table contents is very important to understanding EIGRP.

- One of the reasons that EIGRP is considered an advanced distance vector protocol is that it retains more information than just the best path for each route it receives. This means that it can potentially make decisions more quickly when changes occur, because it has a slightly more complete view of the network. The place this additional information is stored is in the topology table.

- The topology table contains an entry for every route EIGRP is aware of, and includes information about all neighbors that have reported this route to him. When a route is withdrawn by a neighbor, EIGRP will look in the topology table to see if there is a Feasible Successor, which is another downstream neighbor that is guaranteed to be loop-free. If so, EIGRP will use that neighbor and never have to go looking.

- Contrary to popular belief, however, the topology table also contains routes which are not feasible. These are called “possible successors” and may be promoted to feasible successors, or even successors if the topology of the network were to change.

- The following slides show a few different ways to look at the topology table and give hints on how to evaluate it.
The most common way to look at the topology table is with the generic “show ip eigrp topology” command. This command displays all of the routes in the EIGRP topology table, along with their successors and feasible successors.

In the above example, the “P” on the left side of the topology entry displayed means the route is Passive. If it has an “A”, it means the route is Active. The destination being described by this topology entry is for 10.200.1.0 255.255.255.0. This route has 1 successor, and the Feasible Distance is 21026560. The Feasible Distance is the metric that would appear in the routing table if you did the command “show ip route 10.200.1.0 255.255.255.0”.

Following the information on the destination network, the successors and feasible successors are listed. The successors (one or more) are listed first, then the feasible successors are listed (unordered). The entry for each next-hop includes the IP address, the Feasible Distance through this neighbor, the Reported Distance from this neighbor, and which interface is used to reach him.

As you can see, 10.1.2.2 is a feasible successor because his Reported Distance (21514560) is less than our current Feasible Distance (21026560). (Remember the Feasibility Condition?)
• If you want to display all of the information which EIGRP contains in its topology table, use the `show ip eigrp topology all-links` command.

• You’ll notice in the above output that not only are the successor (10.1.1.2) and feasible successor (10.1.2.2) shown, but another router that doesn’t qualify as either is also displayed. The Reported Distance from 10.1.3.2 (46228736) is far worse than the current Feasible Distance (21026560), so he certainly isn’t feasible.

• This command is often useful to understand the true complexity of network convergence. I’ve been on networks with pages of non-feasible alternative paths in the topology table because of a lack of summarization/distribution lists. These large numbers of alternative paths can cause EIGRP to work extremely hard when transitions occur, and can actually keep EIGRP from successfully converging.
• If you really want to know all of the information EIGRP stores about a particular route, use the command `show ip eigrp topology <network><mask>`.

• In the above display, you'll see that EIGRP not only stores which next-hops have reported a path to the target network, it stores the metric components used to reach the total (composite) metric.

• You also may notice that EIGRP contains a hop count in the vector metrics. The hop count isn’t actually used in calculating the metric, but instead was included to limit the apparent maximum diameter of the network. In EIGRP’s early days, developers wanted to ensure that routes wouldn’t loop forever and put this safety net in place. In today’s EIGRP, it actually isn’t necessary any longer, but is retained for compatibility.
If you perform the command `show ip eigrp topology <network><mask>` for an external route (one redistributed into EIGRP from another protocol), even more information is displayed.

The initial part of the display is identical to the command output for an internal (native) route. The one exception is the identifier of the route as being External. Another section is appended to the first part, however, containing external information. The most interesting parts of the external data are the originating router and the source of the route.

- The originating router is the router who initially redistributed the route into EIGRP. Note that the value for the originating router is router-id of the source router, which doesn’t necessarily need to belong to an EIGRP-enabled interface. The router-id is selected in the same way OSPF selects router-ids, starting with loopback interfaces, if any are defined, or using the highest IP address on the router if there aren’t loopback interfaces. Note that if a router receives an external route and the originating router field is the same as the receiver’s router-id, he rejects the route. This is noted in the event log as “ignored, dup router”.

- The originating routing protocol (where it was redistributed from) is also identified in the external data section. This is often useful when unexpected routes are received and you are hunting the source.
• Similar to the command `show ip eigrp topology active`, which was covered in the discussion on chasing Stuck-in-Active routes, the command `show ip eigrp topology pending` displays routes in transition.

• The output of the `show ip eigrp topology pending` command gives a list of all routes which are in the process of being sent, but which haven’t made it out of the interface, yet. I rarely use this command except to see if the network is stable or whether updates are happening regularly.
• The `show ip eigrp topology summary` command gives a brief view of what the topology table contains.

• The third line down is the most interesting, with a count of how many routes are in the topology table, and if any are pending replies. In contrast the command `show ip eigrp topology active`, which shows which routes are active (waiting on replies), the command `show ip eigrp topology summary` just supplies how many are waiting. Sometimes the number pending is larger than the total number of routes, since it is counted per neighbor.

• The “dummies” entry has to do with an internal structure and isn’t really significant for customers.

• Quiescent interfaces are those which do not have any activity presently happening on them (update, query, reply). Quiescent is quiet.
Show IP EIGRP Topology Zero

- And last, the **show ip eigrp topology zero** command is available to display the topology table entries that are not actually being used by the routing table!

- Typically, zero successor entries are ones that EIGRP attempted to install into the routing table, but found a better alternative there already. In our example above, when EIGRP tried to install its route (with an administrative distance of 90), it found a static route already there (with an administrative distance of 1) and thus couldn’t install it. In case the better route goes away, EIGRP retains the information in the topology table, and will try to install the route again if it is notified that the static (or whatever) route is removed.

- Routes that are active sometimes also show up as zero successor routes, but they are transient and don’t remain in that state.

- This command isn’t often used or useful.
Most problems seen in EIGRP networks are caused by factors outside of EIGRP, itself (congestion, lack of summarization, etc.)

There are many tools and techniques available for troubleshooting problems in EIGRP networks
Troubleshooting EIGRP

Session 2207
Please Complete Your Evaluation Form

Session 2207