Multichassis MP (Part 2)

Document ID: 14945

Contents

Introduction

This document continues to describe support for Multilink PPP (MP) in a "stack" or multichassis environment (sometimes called MMP, for *Multichassis Multilink PPP*), on Cisco Systems's access server platforms.

This document is Part Two of a two−part document. Refer to Part One of this document for more information.

Prerequisites

The prerequisites for this document are given in Part One of this document.

Examples

AS5200 in a Stack (With Dialers)

When dialers are configured on the physical interfaces, there is no need to specify the virtual template interface at all. The virtual access interface acts as a passive interface, buttressed between the dialer interface and the physical interfaces associated with the dialer interface.

In short, you need only define the stack group name, the common password, and the stack group members across all the stack members. No virtual template interface is defined, as shown in the following example:

```
systema#config
  sgbp group stackq
  sgbp member systemb 1.1.1.2
  sgbp member systemc 1.1.1.3

  username stackq password therock
```

```
int dialer 1
ip unnum e0
dialer map .....
encap ppp
ppp authen chap
dialer-group 1
ppp multilink

controller T1 0
framing esf
linecode b8zs
pri-group timeslots 1-24

interface Serial0:23
no ip address
encapsulation ppp
dialer in-band
dialer rotary 1
dialer-group 1
```

The following example is from an E1 controller:

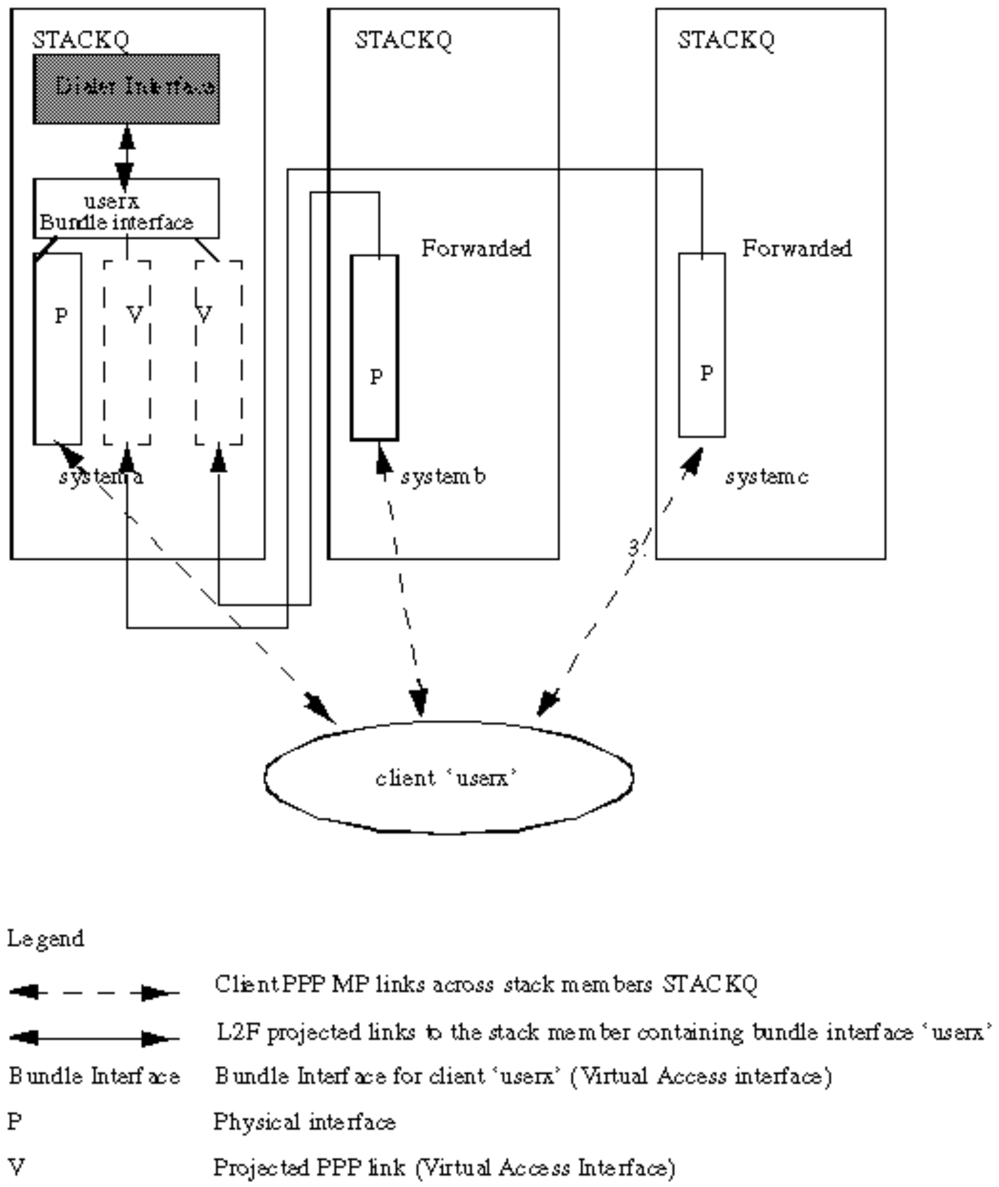
```
controller E1 0
  framing crc4
  linecode  hdb3
  pri-group timeslots 1-31
  interface Serial0:15
  no ip address
  encapsulation ppp
  no ip route-cache
  ppp authentication chap
  ppp multilink
```

After the bundle interface is created, it is cloned with only the PPP commands from the dialer interface. Subsequent projected PPP links are also cloned with the PPP commands from the dialer interface. Figure 3 shows how the dialer interface sits on top of the bundle interface. Compare it with Figure 2, in which there is no dialer interface.

PRIs and BRIs by default are dialer interfaces; a PRI configured without an explicit dialer (through the **dialer rotary** command) is still a dialer interface on Serial0:23, as shown by the following example:

```
interface Serial0:23
  ip unnum e0
  dialer map .....
  encap ppp
  ppp authen chap
  dialer-group 1
  dialer rot 1
  ppp multilink
```

Figure 3: A Stack Group stackq consisting of `systema`, `systemb`, and `systemc`. `systema`'s link is configured on dialer interface.


STACKQ
Dialer Interface
userx
Bundle interface
P
V
V
systema
STACKQ
Forwarded
P
systemb
STACKQ
Forwarded
P
systemc
3
client 'userx'
Legend
Client PPP MP links across stack members STACKQ
L2F projected links to the stack member containing bundle interface 'userx'
Bundle Interface
Bundle Interface for client 'userx' (Virtual Access interface)
P
Physical interface
V
Projected PPP link (Virtual Access Interface)

Using an Offload Server

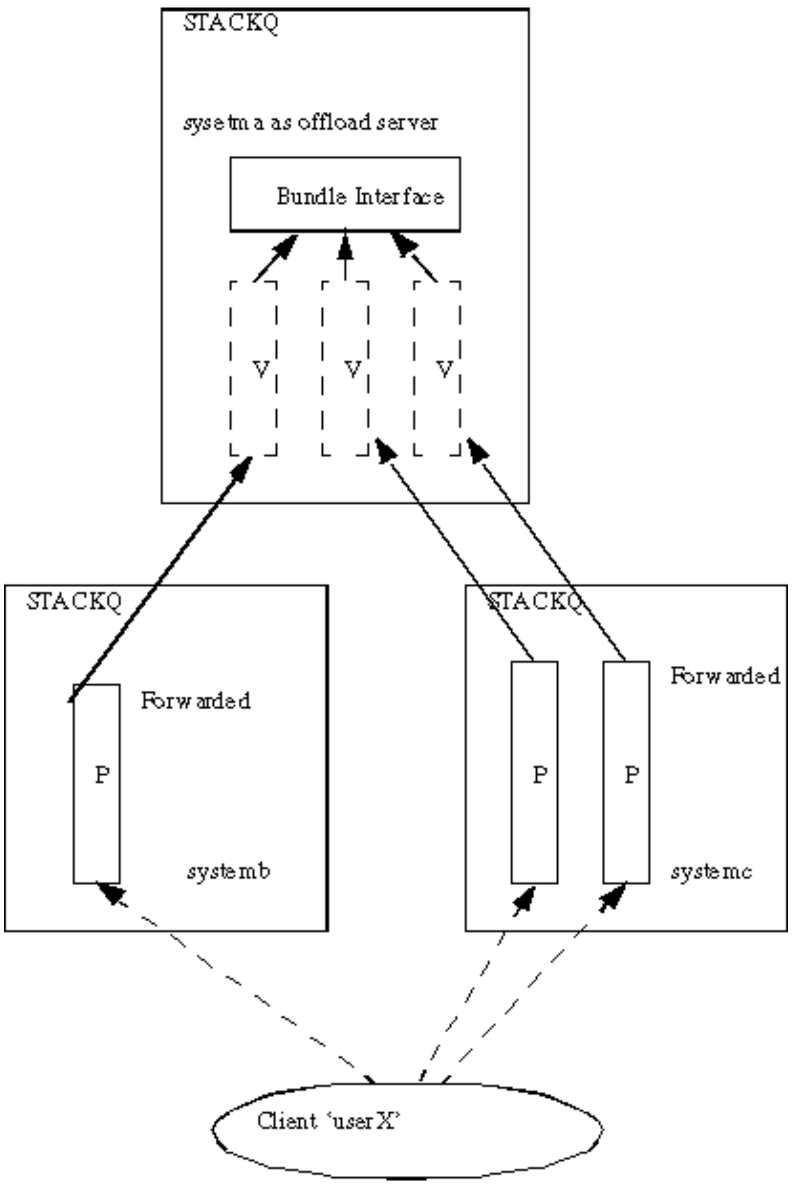
`systema` is designated an offload server (using the **sgbp seed−bid** command). All other stack group members must be defined with the **sgbp seed−bid default** command (or, if you do not define the**sgbp seed−bid** command, it defaults to this).

```
systema#config
  multilink virtual−template 1
  sgbp group stackq
  sgbp member systemb 1.1.1.2
  sgbp member systemc 1.1.1.3
  sgbp seed−bid offload
  username stackq password therock

  interface virtual−template 1
  ip unnumbered e0
  :

  ppp authen chap
  ppp multilink
```

Figure 4: `systema` as an offload server.



Offload Server with Physical Interfaces

If the designated offload server also has physical interfaces (for example, PRI) wishing to serve the same telco hunt group as the other stack members, you can configure it to do so by combining configurations shown in the sections of this document titled AS5200 in a Stack (With Dialers) and Using an Offload Server.

An offloaded projected PPP link and its bundle interfaces rely on virtual templates for a configuration source. A connection that has the *first link* arrives at a physical device connected to a dialer interface, and the source of configuration for the bundle interface and all subsequent projected PPP links is the dialer interface configuration. Hence, these variations coexist, dependent upon the stack member on which the first link arrives.

This configuration is not recommended due to the complexity of configurations required on the dialer and virtual template interfaces.

Async, Serial, and Other Non−Dialer Interfaces

While you can configure asynchronous and serial devices as dialer interfaces (in which case it reverts to AS5200 in a Stack (With Dialers), as shown in that section of this document), you may choose to support multichassis MP without any dialer configuration for asynchronous, serial, and other non−dialer interfaces. The source of all configuration is then defined in virtual template interface, as shown below.

```
#config
  multilink virtual−template 1
  sgbp group stackq
  sgbp member systemb 1.1.1.2
  sgbp member systemc 1.1.1.3
  username stackq password therock
  interface virtual−template 1
  ip unnumbered e0
  :
  ppp authen chap
  ppp multilink

  int async 1
  encap ppp
  ppp multilink
  ppp authen chap
  :

  line 1
  login
  autoselect ppp
  autoselect during−login
  speed 38400
  flow hardware
```

Dialing Out from a Multichassis

Currently, multichassis configuration **does not support dialout**, because the Layer 2 Forwarding (L2F) protocol does not support dialout.

Consequently, there is no way for the offload server (where a route is spoofed, on a dialer profile, and so on) to initiate a dial on the front−end stack member in the same stack group. Any spoofed routes must be installed on the the front−end stack members, because these are the ones with the physical dial connections (such as PRI).

Some workarounds are as follows:

- When the **sgbp ppp−forward** command is issued on the front−end stack member, this means that all PPP and PPP multilink calls are automatically forwarded to the Stack Group Bidding Protocol (SGBP) bid winner, such as an offload server.

  You have to rely on the Network Access Server (NAS) dialing out and let IP routing convergence (for IP only) take its course. For example, to dial 1.1.1.1, put this address in the dialer map statement on the NAS and put a static route on the NAS, as follows:

  ```
  ip route 1.1.1.1 255.255.255.255 serial0:23
  int serial0:23
  ip address 3.3.3.3 255.0.0.0
  dialer map ip 1.1.1.1 howard 7771234
  ```

  When the dial connects to the remote peer, the PPP connection is formed between the remote peer and the offload server. The front−end stack member is completely bypassed. PPP on the offload server then installs a host route to the peer¡.1.1.1. At this point, the IP routing protocol converges from to the host route at the offload server because the routing metric gravitates the route there.

Note: Routing convergence results in latency.

- When the **sgbp ppp−forward** command is not defined on the front−end stack member, this means that only PPP multilink calls are automatically forwarded to the SGBP bid winner, such as an offload server. Thus, a dialer from the front−end stack member to a remote peer spans the PPP connection between the front−end and the remote peer the same behavior as if the NAS were not part of a stack group.

  Note: This happens as long as the connection is straight PPP (and not PPP multilink).

Dialing to a Multichassis

If you have IP routing (such as Enhanced Interior Gateway Routing Protocol (EIGRP) and Open Shortest Path First (OSPF)) flowing between the client and the stack member that eventually wins the bid (such as the offload server), here are a few tips to follow:

Prevent Installing a Connected Route on the Client Side

Configure client 1.1.1.2 where 1.1.1.2 is the address of the NAS (the transparent frame−forwarder), as shown below.

```
int bri0

dialer map 1.1.1.2 ....
```

If you have EIGRP, for example, running between the client and the offload server, the routing table on the offload server indicates that to get to 1.1.1.2 the route should go through the virtual access interface. This is because PPP IP Control Protocol (IPCP) on the client side installs a connected route 1.1.1.2 to the BRI interface. EIGRP then advertises this route to the offload server over the PPP session (over L2F). EIGRP on the offload server thus indicates that to get to 1.1.1.2, it should route to the client the client's route 1.1.1.1 is to the virtual access interface.

Now, you have a packet destined for the client 1.1.1.1. IP routing sends the packet to the virtual access interface. The virtual access interface encapsulates IP/User Data Protocol (UDP)/L2F/PPP encapsulation and sends the packet to the L2F NAS¡.1.1.2. Everything is normal at this point. Then, instead of sending the packet out through (for example) the Ethernet interface, IP routing sends it through the virtual access interface again. This is because the routing table indicates that to get to the NAS, it must go through the client. This creates a routing loop and effectively disables input and output over the L2F tunnel.

To prevent this, do not allow IPCP to install a connected route on the client side.

Note: This pertains only when you have some IP routing protocol running between the client and Cisco Home Gateway.

The client configuration is as follows:

```
int bri0

no peer neighbor−route
```

Dialer Maps on the Client

When the client is dialing to a multichassis environment, always define the dialers to each potential winner of the multilink bundle. For example, if there are four offload servers in the multichassis stack, there should be four dialer maps defined in the client side.

For example:

```
client 1.1.1.1

int bri0

dialer map 1.1.1.3 ...
```

In this example, 1.1.1.3 is just one offload server.

A packet destined for 1.1.1.2 routes to the BRI, and the dialer dials the destination because there is a dialer map match. The offload server 1.1.1.4 actually wins the bid and the PPP session is projected there. EIGRP is exchanged between the client and the offload server. The IP routing table on the client is filled with a route 1.1.1.4 (offload server) to BRI0. Now, on the client, a packet destined for 1.1.1.4 is routed to BRI0. The dialer, however, cannot dial as there is no dialer match.

Note: Always define dialer maps for all potential SGBP bid winners on clients whenever accessing the offload servers is a requirement of the clients.

Configuration and Restrictions

- The enterprise j−image is required for multichassis MP.
- Only one stack group can be defined for each access server.
- High−latency WAN links between stack members, causing MP reassembly delays, may cause multichassis MP to be inefficient.
- Interfaces are supported for PRI, [M]BRI, serial, and asynchronous devices.
- Dialout is not supported.

Configuration of Per−Protocol Interface Configurations

For all practical purposes, do not configure a specific protocol address on the virtual template.

```
interface virtual−template 1

ip address 1.1.1.2 255.0.0.0

:
```

The virtual template interface serves as a template from which any number of virtual access interfaces are dynamically cloned. You should not specify a per−interface protocol−specific address to the virtual template interface. Because an IP address must be unique for each network interface, specifying a unique IP address on the virtual template interface is erroneous. Instead, do the following:

```
interface virtual−template 1

ip unnum e0

:
```

Configuration of Global Protocol Configurations

A client that dials into a single access router and expects the access server to have a unique global address (such as DECnet) now actually dials to the multichassis multilink stack group consisting of several access servers. In this kind of situation, terminate the stack group deterministically at a single access server. To do that, issue the **sgbp seed−bid offload** command on the designated access server (or specify the highest bid).

Troubleshooting

The first thing to do if you have problems is to go back to a single stack member, disabling all other stack members. Then test your PPP multilink connections and go through the usual Challenge Handshake Authentication Protocol (CHAP) authentication and interface configuration for errors in configuration and so forth. When you are satisfied it works, enable the other stack members, then proceed as follows:

1. Make sure SGBP is up and running.
2. Debug PPP multilink.
3. Debug VPN and L2F.

Making Sure SGBP Is Up and Running Correctly

Issue the **show sgbp** command to make sure that all member states are ACTIVE. Otherwise, look out for IDLE, AUTHOK, or ACTIVE states. As mentioned previously, IDLE is a valid state for all remote stack members that are intentionally inactive.

If you find a problem as described above, turn on the **debug sgbp hellos** and **debug sgbp error** command. The authentication between two stack members, for example between `systema` and `systemb`, should be as follows (on `systema`):

```
systema# debug sgdp hellos

%SGBP−7−CHALLENGE: Send Hello Challenge to systemb group stackq
%SGBP−7−CHALLENGED: Hello Challenge message from member systemb (1.1.1.2)
%SGBP−7−RESPONSE: Send Hello Response to systemb group stackq
%SGBP−7−CHALLENGE: Send Hello Challenge to systemb group stackq
%SGBP−7−RESPONDED: Hello Response message from member systemb (1.1.1.2)
%SGBP−7−AUTHOK: Send Hello Authentication OK to member systemb (1.1.1.2)
%SGBP−7−INFO: Addr = 1.1.1.2 Reference = 0xC347DF7
%SGBP−5−ARRIVING: New peer event for member systemb
```

`systema` sends a CHAP−style challenge and receives a response from `systemb`. Similarly, `systemb` sends out a challenge and receives a response from `systema`.

If authentication fails, you see:

```
%SGBP−7−AUTHFAILED − Member systemb failed authentication
```

This means that the remote `systemb` password for stackq does not match the password defined on `systema`.

```
%SGBP−7−NORESP −Fail to respond to systemb group stackq, may not have password
```

This means that `systema` does not have a username or password defined locally or through TACACS+.

In general, define a common secret across all stack members for the stack group `stackq`. You can define them locally or through TACACS+.

A local username defined on each stack member is:

```
username stackq password blah
```

This common secret is to facilitate stack member SGBP bidding and arbitration.

Refer to the Debugging PPP Multilink section of this document for a discussion of PPP link authentication when a remote client dials in to the stack members.

In the case of wiring or routing problems, one common error is having the stack member's source IP address (which is actually received in the SGBP hello message) different from the locally defined IP address for the same stack member.

```
systema#debug sgbp error
%SGBP-7-DIFFERENT - systemb's addr 1.1.1.2 is different from hello's addr 3.3.4.5
```

This means that the source IP address of the SGBP hello received from `systemb` does not match the IP address configured locally for `systemb` (through the **sgbp member** command). Correct this by going to `systemb` and checking for multiple interfaces by which the SGBP hello can transmit the message.

Another common cause for errors is:

```
%SGBP-7-MISCONF, Possible misconfigured member routerk (1.1.1.6)
```

This means that you do not have `systemk` defined locally, but another stack member does.

Debugging PPP Multilink

The first thing to check is whether the client and the stack member authenticated on PPP correctly.

This example demonstrates CHAP authentication, as it is more involved. As a familiar example, it uses a Cisco platform as a client along with local usernames (Terminal Access Controller Access Control System Plus (TACACS+) is supported also, but is not shown here).

Client userx	Every member of stack `stackq`

`#`**`config`**
 	
`username stackq password blah`
	`#`**`config`**
	
	`username userx password blah`

No Dialer Interfaces Involved

Since there is no dialer interface on the offload server, there needs to be another *source of interface configuration* of virtual access interfaces. The answer is virtual template interfaces.

1. First make sure the multilink global virtual template number is defined on each stack member.

   ```
   #config
   Multilink virtual-template 1
   ```
2. If you have not configured any dialer interfaces for the physical interfaces in question (such as PRI, BRI, asynchronous, and synchronous serial), you may define:

   ```
   interface virtual-template 1
     ip unnumbered e0
     ppp authen chap
   ```

```
        ppp Multilink
```

Note: You do not define a specific IP address on the virtual template. This is because projected virtual access interfaces are always cloned from the virtual template interface. If a subsequent PPP link also gets projected to a stack member with a virtual access interface already cloned and active, you have identical IP addresses on the two virtual interfaces, causing IP to erroneously route between them.

Dialer Interfaces Involved

When dialers are configured on the physical interfaces, there is no need to specify a virtual template interface, because the interface configuration resides in the dialer interface. In this case, the virtual access interface acts as a passive interface, buttressed between the dialer interface and the member interfaces associated with the dialer interface.

Note: The dialer interface, Dialer 1, is displayed in the PPP multilink session as follows:

```
systema#show ppp Multilink
Bundle userx 2 members, Master link is Virtual-Access4
Dialer interface is Dialer1
0 lost fragments, 0 reordered, 0 unassigned, 100/255 load
0 discarded,  0 lost received, sequence 40/66 rcvd/sent
members 2
Serial0:4
systemb:Virtual-Access6    (1.1.1.1)
```

LCP and NCPs

The LCP states on all member interfaces must be UP. IPCP, ATCP, and other NCPs should be up only on the bundle interface.

Bundle interface `Virtual-Access4` **show int** output should read as follows:

```
router#show int  Virtual-Access4
Virtual-Access4 is up, line protocol is up
        :
    LCP Open, Multilink Open
    Open: ipcp
            :
```

All other member interfaces should have the following **show int** output:

```
router# show int Serial0:4
Serial0:4 is up, line protocol is up
        :
LCP Open, Multilink Open
Closed:  ipcp
```

Debugging VPN/L2F

Turn on the following:

```
debug vpn event
  debug vpn error
```

When the physical interface accepts the incoming call and is now forwarded to the target stack member, you see the following:

```
Serial0:21 VPN Forwarding
```

```
Serial0:21 VPN vpn_forward_user userx is forwarded
```

On the target stack member, if you see the following:

```
Virtual−Access1 VPN PPP LCP not accepting rcv CONFACK
Virtual−Access1 VPN PPP LCP not accepting sent CONFACK
```

Then check your definition of your virtual template interface. Generally, the virtual template interface must match the PPP interface parameters of the physical interface that accepted an incoming call.

Remember the minimum configuration (using CHAP as an example):

```
#config
multilink virtual template 4
int virtual−template 4
ip unnum e0
encap ppp
ppp authen chap
ppp Multilink
```

You may see the following:

```
Virtual−Access1 VPN PPP LCP accepted sent & rcv CONFACK
```

If you see the above message, it means that L2F has successfully projected the PPP link from the stack member that first took the incoming call to the stack member where the bundle interface for the same client resides (or will create, as in the offload scenario).

A common error is failure to define the username for the common stack name (stackq) or not matching the stack password on all stack members.

Issue the following command:

```
debug vpdn l2f−error
```

The following message results:

```
L2F Tunnel authentication failed for stackq
```

Correct the username and password match on each stack member in this case.

Related Information

- **Part 1 of This Document**
- **Virtual Access PPP Features in Cisco IOS Software**
- **Understanding VPDN**
- **Technical Support − Cisco Systems**