

Multichassis Multilink PPP (MMP)

Document ID: 14942

Introduction

Prerequisites

- Requirements
- Components Used
- Related Terms
- Conventions

Problem Definition

Functional Overview

- SGBP
- Virtual Access Interfaces
- L2F

End User Interface

- SGBP
- MP

Examples

Related Information

Introduction

This document describes 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.

Prerequisites

Requirements

There are no specific prerequisites for this document.

Components Used

This document is not restricted to specific software and hardware versions.

The information presented in this document was created from devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If you are working in a live network, ensure that you understand the potential impact of any command before using it.

Related Terms

This is a glossary of terms that this document uses:

- Access server Cisco access server platforms, including ISDN and asynchronous interfaces to provide remote access.
- L2F Layer 2 (L2) Forwarding Protocol (Experimental Draft RFC). This is the underlying link-level technology for both multichassis MP and VPN.
- Link A connection point that a system provides. A link can be a dedicated hardware interface (such as an asynchronous interface) or a channel on a multichannel hardware interface (such as a PRI or BRI).

- MP Multilink PPP Protocol (refer to RFC 1717).
- Multichassis MP MP + SGBP + L2F + Vtemplate.
- PPP Point-to-Point Protocol (refer to RFC 1331).
- Rotary Group A group of physical interfaces allocated to dial out or receive calls. The group acts like a pool from which you can use any link to dial out or receive calls.
- SGBP Stack Group Bidding Protocol.
- Stack Group A collection of two or more systems that are configured to operate as a group and support MP bundles with links on different systems.
- VPDN Virtual Private Dialup Network. The forwarding of PPP links from an Internet Service Provider (ISP) to a Cisco Home Gateway.
- Vtemplate Virtual template interface.

Note: For information on RFCs referenced in this document, see RFCs and Other Stds Supported in Cisco IOS Release 11.3–No. 523, a product bulletin; Obtaining RFCs and Standards Documents; or RFC Index for a link directly to InterNIC.

Conventions

Refer to Cisco Technical Tips Conventions for more information on document conventions.

Problem Definition

MP provides users with additional bandwidth on demand with the ability to split and recombine packets across a logical pipe (bundle) that multiple links form.

This reduces transmission latency across the slow WAN links, and also provides a method to increase the maximum receive unit.

On the transmitting end, MP provides for the fragmentation of a single packet into multiple packets to be transmitted across multiple PPP links. On the receiving end, MP provides packet reassembly from multiple PPP links back into the original packet.

Cisco supports MP to autonomous end systems, that is, multiple MP links from the same client can terminate at the access server. However, ISPs, for example, prefer to conveniently allocate a single rotary number to multiple PRIs across multiple access servers, and make their server structure scalable and flexible to business needs.

In Cisco IOS® Software Release 11.2 Cisco provides such functionality, so that multiple MP links from the same client can terminate at different access servers. While individual MP links of the same bundle can actually terminate at different access servers, as far as the MP client is concerned, this is similar to termination at a single access server.

In order to achieve this goal, MP uses multichassis MP.

Functional Overview

Figure 1 illustrates the use of MP on a single Cisco access server to support this feature.

Figure 1 MP on a single Cisco access server

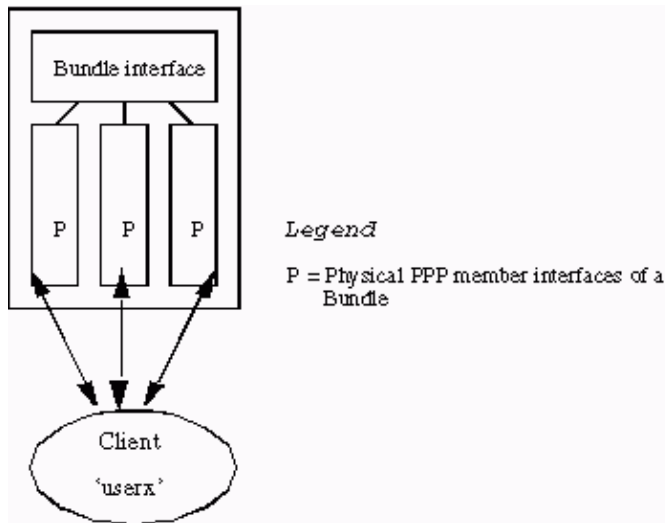


Figure 1 illustrates how MP member interfaces are connected to a bundle interface. In a standalone system without multichassis MP enabled, member interfaces are always physical interfaces.

In order to support a stacked environment, in addition to MP, these three additional subcomponents are necessary:

- SGBP
- Vtemplate
- L2F

The next few sections of this document explain these components in detail.

SGBP

In a multiple access server environment, the network administrator can designate a group of access servers to belong to a stack group.

Suppose a stack group consists of System A and System B. A remote MP client called *userx* has the first MP link terminate at System A (*systema*). The bundle *userx* is formed at *systema*. The next MP link from *userx* now terminates at System B (*systemb*). SGBP locates that bundle where *userx* resides on *systema*. At this point, another component L2F projects the second MP link from *systemb* to *systema*. The projected MP link then joins the bundle at *systema*.

SGBP thus locates the bundle location of a stack member within a defined stack group. SGBP also *arbitrates* for a designated stack member for the bundle creation. In the example, when the first MP link is received on *systema*, both *systema* and *systemb* (and all other members of the stack group) actually bid for the bundle creation. The bid from *systema* is higher (because it accepted the first link), so SGBP designates it for bundle creation.

This description of the SGBP bidding process is somewhat simplistic. In practice, an SGBP bid from a stack member is a function of locality, a user-configurable weighted metric, CPU type, number of MP bundles, and so on. This bidding process allows for the bundle creation on a designated system even one that does not have any access interfaces. For example, a stacked environment can consist of 10 access server systems and two 4500s a stack group of 12 stack members.

Note: When the bids are equal, such as between two 4500s, SGBP randomly designates one as the winner of the bid. You can configure the 4500s so that they always outbid the other stack members. The 4500s thus become offload multichassis MP servers specialized on MP packets fragmenters andreassemblers a task

suited for their higher CPU power relative to the access servers.

In short, SGBP is the location and arbitration mechanism of multichassis MP.

Virtual Access Interfaces

Virtual access interfaces serve both as bundle interfaces (see Figures 1 and Figure 2) and projected PPP links (see Figure 2). These interfaces are dynamically created and returned to the system on demand.

Virtual template interfaces serve as repositories of configuration information from which virtual access interfaces are cloned. Dialer interface configurations serve as another source of configuration information. The method to choose the source of configuration from which to clone a virtual access interface becomes apparent in Multichassis Multilink PPP (MMP) (Part 2).

L2F

L2F provides for the actual PPP link projection to a designated end system.

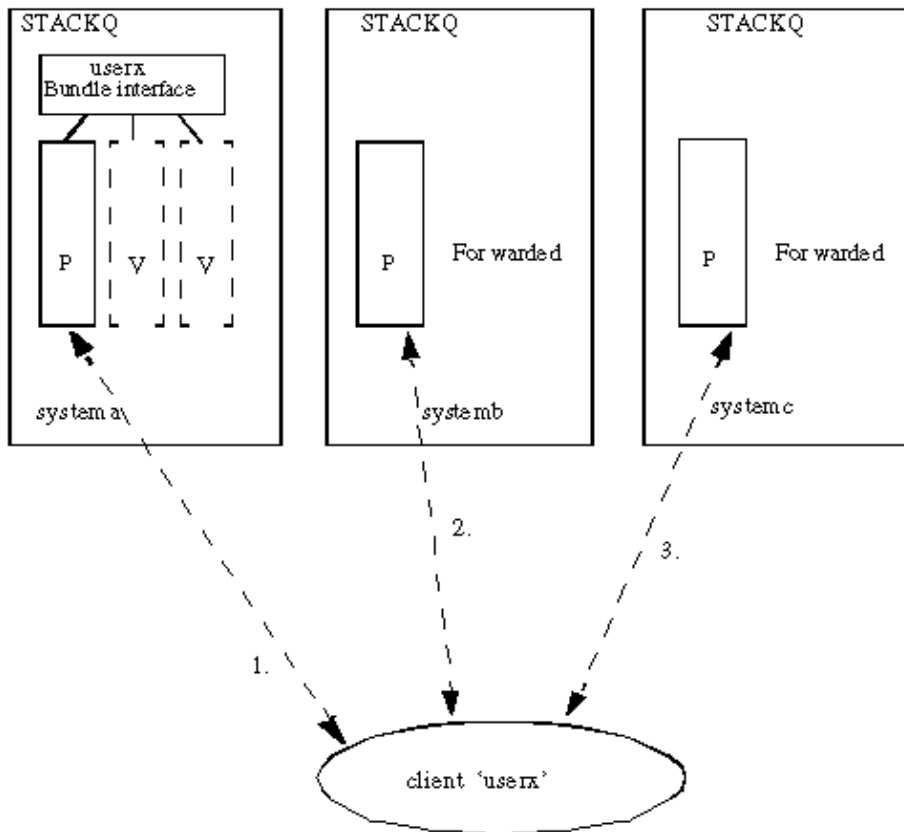
L2F performs standard PPP operation up to the authentication phase, where the remote client is identified. The authentication phase is not completed locally. L2F, provided with the target stack member from SGBP, projects the PPP link to the target stack member, where the authentication phase is resumed and completed at the projected PPP link. The final authentication success or failure is thus performed at the target stack member.

The original physical interface that accepted the incoming call is said to be *L2F forwarded*. The corresponding interface that L2F creates dynamically (when PPP authentication succeeds) is a projected virtual access interface.

Note: The projected virtual access interface is also cloned from the virtual template interface (if defined).

Figure 2 describes a stack group stackq that consists of `systema`, `systemb`, and `systemc`.

Figure 2 Client calling into a stack



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)

1. Client userx calls. The first link on systema receives the call. SGBP tries to locate any bundle by userx existing among the stack group members. If there is none, and because MP is negotiated on the PPP, a bundle interface is created on systema.
2. systemb receives the second call from userx. SGBP helps to determine that systema is where the bundle resides. L2F helps to forward the link from systemb to systema. A projected PPP link is created on systema. The projected link is then joined to the bundle interface.
3. systemc receives the third call from userx. Again, SGBP locates that systema is where the bundle resides. L2F is used to forward the link from systemc to systema. A projected PPP link is created on systema. The projected link is then joined to the bundle interface.

Note: A bundle interface represents the bundle on systema. For every unique caller, MP member interfaces from that same caller terminate to or originate from one bundle interface.

End User Interface

The Vtemplate user interface is specified nominally here. Refer to the Virtual Template Functional Specification for details.

SGBP

1. **sgbp group** <name>

This global command defines a *stack group*, assigns a name to the group, and makes the system a member of that stack group.

Note: You can define only one stack group globally.

Define a stack group called *stackq*:

```
systema(config)#sgbp group stackq
```

Note: The PPP CHAP challenge or the PPP PAP request from *systema* now bears the name *stackq*. When you define the stack group name on the access server, the name generally supersedes the hostname defined for the same system.

2. **sgbp member** <peer-name> <peer-IP-address>

This global command specifies peers in the stack group. In this command, <peer-name> is the host name and <peer-IP-address> is the IP address of the remote stack member. Thus, you need to define an entry for every stack group member in the stack except yourself. a Domain Name Server (DNS) can resolve the peer names. If you have a DNS, you do not need to enter the IP address.

```
systema(config)#sgbp member systemb 1.1.1.2
```

```
systema(config)#sgbp member systemc 1.1.1.3
```

3. **sgbp seed-bid** {default | offload | forward-only | <0-9999>}

The configurable weight that the stack member bids with for a bundle.

If the `default` parameter is defined across all stack members, the stack member that receives the first call for the user `userx` always wins the bid, and hosts the master bundle interface. All subsequent calls from the same user to another stack member project to this stack member. If you do not define an **sgbp seed-bid**, the `default` is used.

If `offload` is defined, it sends the precalibrated per-platform bid that approximates the CPU power, minus the *bundle load*.

If <0-9999> is configured, the bid sent out is the user-configured value minus the *bundle load*.

The bundle load is defined as the number of active bundles on the stack member.

- a. When you have equivalent stack members stacked to receive calls in a rotary group across multiple PRIs, issue the **sgbp seed-bid default across all stack members** command. An example of equivalent stack members would be a stack group of four AS5200s. The stack member that receives the first call for the user `userx` always wins the bid, and hosts the master bundle interface. All subsequent calls to the same user to another stack member projects to this stack member. If the multiple calls come in concurrently over multiple stack members, the SGBP tie-breaking mechanism breaks the tie.
- b. When you have a higher-power CPU available as a stack member relative to the other stack members, you may want to leverage the relative higher power of that stack member over the rest (for example, one or more higher-powered CPUs available as a stack member relative to the other similar stack members; for instance, one 4500 and four AS5200s). You can set the designated high-power stack member as the offload server with the **sgbp seed-bid offload**

command. In that case, the offload server hosts the master bundle. All calls from other stack members are projected to this stack member. Actually, one or more offload servers can be defined; if the platforms are the same (equivalent), the bids are equal. The SGBP tie-breaking mechanism breaks the tie and designates one of the platforms as the winner.

Note: If you designate two dissimilar platforms as offload servers, the one with the higher CPU power wins the bid.

- c. If you have assorted or exactly the same platforms and you want to designate one or more platforms as offload servers, you can manually set the bid value to be significantly higher than the rest with the **sgbp seed-bid 9999** command. For example, one 4700 (designated by the highest seed-bid), two 4000s, and one 7000. To determine the initial bid value associated with your particular platforms, use **show sgbp**.
- d. In a multichassis environment where the front-end stack members always offload to one or more offload servers, there are cases where the front-end stack member can actually not offload, such as when the multilink bundle is formed locally. This could happen, for example, when all the offload servers are down. If the network administrator prefers the incoming call to hang up instead, issue the **sgbp seed-bid forward-only** command.

4. **sgbp ppp-forward**

When **sgbp ppp-forward** is defined, both PPP and MP calls are projected to the winner of the SGBP bid. By default, only MP calls are forwarded.

5. **show sgbp**

This command shows the state of the stack group members. The states can be ACTIVE, CONNECTING, WAITINFO, or IDLE. ACTIVE on each stack group member is the best state. CONNECTING and WAITINFO are transitional states and you must only see them when in transition to ACTIVE. IDLE indicates that the stack group `systema` cannot detect the remote stack member `systemd`. If `systemd` is brought down for maintenance, for example, there is no cause for concern. Otherwise, look at some routing issues or other problems between this stack member and `systemd`.

```
systema#show sgbp
Group Name: stack Ref: 0xC38A529
Seed bid: default, 50, default seed bid setting

Member Name: systemb State: ACTIVE Id: 1
Ref: 0xC14256F
Address: 1.1.1.2

Member Name: systemc State: ACTIVE Id: 2
Ref: 0xA24256D
Address: 1.1.1.3 Tcb: 0x60B34439

Member Name: systemd State: IDLE Id: 3
Ref: 0x0
Address: 1.1.1.4
```

6. **show sgbp queries**

Displays the current seed bid value.

```
systema# show sgbp queries
Seed bid: default, 50

systema# debug sgbp queries
%SGBPQ-7-MQ: Bundle: userX State: Query_to_peers OurBid: 050
%SGBPQ-7-PB: 1.1.1.2 State: Open_to_peer Bid: 000 Retry: 0
%SGBPQ-7-PB: 1.1.1.3 State: Open_to_peer Bid: 000 Retry: 0
%SGBPQ-7-PB: 1.1.1.4 State: Open_to_peer Bid: 000 Retry: 0
%SGBPQ-7-MQ: Bundle: userX State: Query_to_peers OurBid: 050
```

```
%SGBPQ-7-PB:      1.1.1.2           State: Rcvd      Bid: 000  Retry: 0
%SGBPQ-7-PB:      1.1.1.3           State: Rcvd      Bid: 000  Retry: 0
%SGBPQ-7-PB:      1.1.1.4           State: Rcvd      Bid: 000  Retry: 0
%SGBPQ-7-DONE:    Query #9 for bundle userX, count 1, master is local
```

MP

1. **multilink virtual-template <1-9>**

This is the virtual template number by which the MP bundle interface clones its interface parameters. Here is an example for how an MP associates with a virtual template. A virtual template interface must also be defined:

```
systema(config)#multilink virtual-template 1
systema(config)#int virtual-template 1
systema(config-i)#ip unnum e0
systema(config-i)#encap ppp
systema(config-i)#ppp multilink
systema(config-i)#ppp authen chap
```

2. **show ppp multilink**

This command displays the bundle information for the MP bundles:

```
systema#show ppp multilink
Bundle userx 2 members, Master link is Virtual-Access4
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   (1.1.1.2)
```

This example shows, on stack group member `systema` on stack group `stackq`, that bundle `userx` has its bundle interface set as `Virtual-Access4`. Two member interfaces are joined to this bundle interface. The first is a local PRI channel and the second is a projected interface from stack group member `systemb`.

Examples

Refer to [Multichassis Multilink PPP \(MMP\) \(Part 2\)](#) to see these examples:

- AS5200 in a Stack with Dialers
- Using an Offload Server
- Offload Server with Physical Interfaces
- Async, Serial, and Other Non-Dialer Interfaces
- Dialing Out from a Multichassis
- Dialing to a Multichassis

And also refer to the sections on:

- Configuration and Restrictions
 - Troubleshooting
-

Related Information

- **Dial and Access Technology Support Pages**
 - **Technical Support & Documentation – Cisco Systems**
-

[Contacts & Feedback](#) | [Help](#) | [Site Map](#)

© 2009 – 2010 Cisco Systems, Inc. All rights reserved. [Terms & Conditions](#) | [Privacy Statement](#) | [Cookie Policy](#) | [Trademarks of Cisco Systems, Inc.](#)

Updated: Jan 29, 2008

Document ID: 14942
