Configuring IP-in-IP Decapsulation

This module describes how to configure IP-in-IP Decapsulation.

-

IP-in-IP Decapsulation

Encapsulation of datagrams in a network is done for multiple reasons, such as when a source server wants to influence the route that a packet takes to reach the destination host. The source server is also known as the encapsulation server.

IP-in-IP encapsulation involves the insertion of an outer IP header over the existing IP header. The source and destination address in the outer IP header point to the end points of the IP-in-IP tunnel. The stack of IP headers are used to direct the packet over a predetermined path to the destination, provided the network adminstrator knows the loopback addresses of the routers transporting the packet. This tunneling mechanism can be used for determining availability and latency for most network architectures. It is to be noted that the entire path from source to the destination does not have to be included in the headers, but a segment of the network can be chosen for directing the packets.

The following illustration describes the basic IP-in-IP encapsulation and decapsulation model.

Figure 1: Basic Encapsulation and De-capsulation with an IP-in-IP
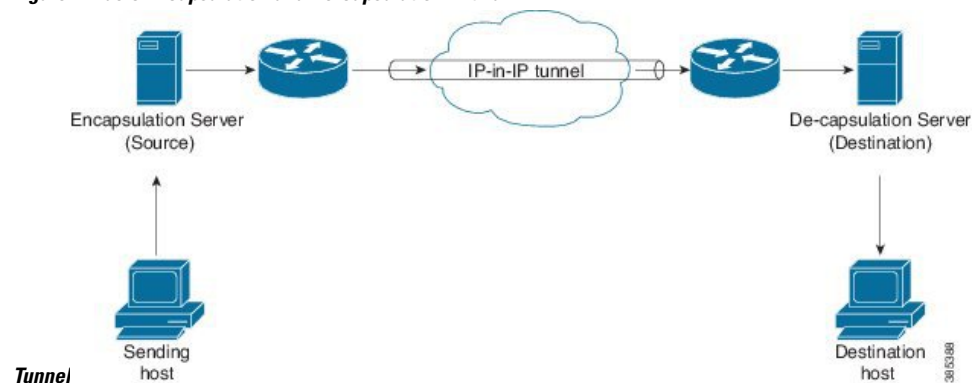
Tunnel

### Use Case: Configure IP-in-IP de-capsulation

The following topology describes a use case where IP-in-IP encapsulation and de-capsulation is used for different segments of the network from source to destination. The IP-in-IP tunnel consists of multiple routers used to de-capsulate and direct the packet through the data center fabric network.

Figure 2: IP-in-IP De-capsulation through a Data Center Network
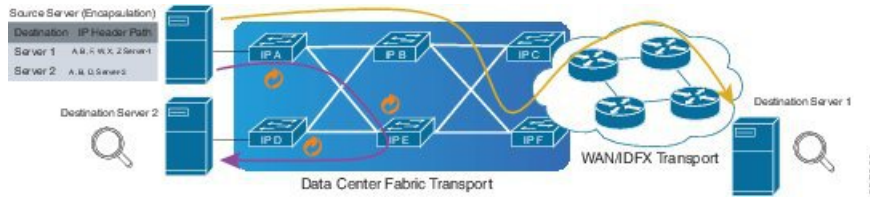
The following illustration shows how the stacked IPv4 headers are de-capsulated as they traverse through the de-capsulating routers.

Figure 3: IP Header De-capsulation
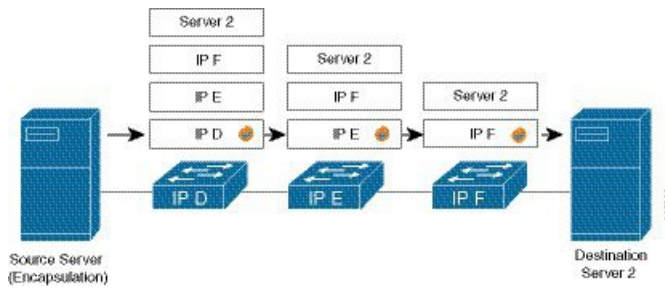
Stacked IP Header in an Encapsulated Packet

The encapsulated packet will have an outer IPv4 header stacked over the original IPv4 header, as shown in the following illustration.

Encapsulated Packet

Frame	
EthernetII	
Preamble (hex)	fb555555555555d5
Destination MAC	62:19:88:64:E2:68
Source MAC	00:10:94:00:00:02
EtherType (hex)	\<auto> Internet IP
IPv4 Header	
Version (int)	\<auto> 4
Header length (int)	\<auto> 5
ToS/DiffServ	tos (0x00)
Total length (int)	\<auto> calculated
Identification (int)	0
Control Flags	
Reserved (bit)	0
DF Bit (bit)	0
MF Bit (bit)	0
Fragment Offset (int)	0
Time to live (int)	255
Protocol (int)	\<auto> IP
Checksum (int)	\<auto> 33492
Source	192.xx.xx.xx
Destination	127.0.0.1
Header Options	
Gateway	192.0.2.10
IPv4 Header	
Version (int)	\<auto> 4
Header length (int)	\<auto> 5
ToS/DiffServ	tos (0x00)
Total length (int)	\<auto> calculated
Identification (int)	0
Control Flags	
Reserved (bit)	0

385413

Configuration

You can use the following sample configuration on the routers to decapsulate the packet as it traverses the IP-in-IP tunnel:

```
RP/0/RP0/CPU0:router(config)# interface tunnel-ip 10
RP/0/RP0/CPU0:router(config-if)# tunnel mode ipv4 decap
RP/0/RP0/CPU0:router(config-if)# tunnel source loopback 0
RP/0/RP0/CPU0:router(config-if)# tunnel destination 10.10.1.2/32
```

- **tunnel-ip**: configures an IP-in-IP tunnel interface.

- **ipv4 unnumbered loopback address**: enables ipv4 packet processing without an explicit address, except for loopback address.

- **tunnel mode ipv4 decap**: enables IP-in-IP de-capsulation.

- **tunnel source**: indicates the source address for the IP-in-IP decap tunnel w.r.t the router interface.

- **tunnel destination**: indicates the destination address for the IP-in-IP decap tunnel w.r.t the router interface.

Running Configuration

```
RP/0/RP0/CPU0:router# show running-config interface tunnel-ip 10
...
interface tunnel-ip 10
tunnel mode ipv4 decap
tunnel source Loopback 0
tunnel destination 10.10.1.2/32
```

This completes the configuration of IP-in-IP de-capsulation.