Translational Bridging of Various IPX Frames on FDDI

Document ID: 10673

Introduction

This document will give you information about how Cisco bridges translate IPX frames from Ethernet to FDDI and back to Ethernet, as well as background on how Cisco Catalyst switches work. This information will help you to design IPX bridged networks that incorporate both products.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

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

Conventions

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

Packet Formats

DA SA − Destination Address, Source Address

SAP − Service Access Point

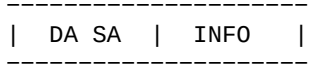
Ethernet 802.2 (also called Ethernet_II)

```
_______________________________
|  DA SA  |  TYPE   |  INFO   |
_______________________________
```

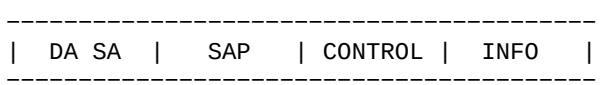
Ethernet 802.3 (also called Ethernet_SNAP)

```
_________________________________________________
```

FDDI 802.2 (also called FDDI_802.2 or FDDI_RAW)

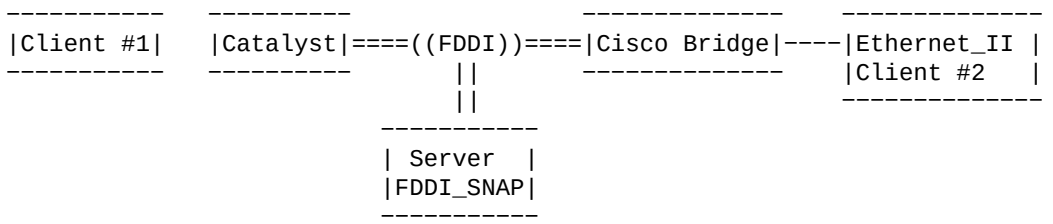


FDDI 802.3 (also called FDDI_SNAP)



Catalysts are Ethernet/FDDI switches that perform translational bridging differently than Cisco routers. A Catalyst translates packets between Ethernet and FDDI depending on the encapsulation styles that each client and server uses − it remembers each device's particular encapsulation style. This allows you to place Ethernet_802.3 or Ethernet_SNAP clients on Ethernet while a server on FDDI is using FDDI_SNAP or FDDI_802.2. Cisco bridges, on the other hand, follow different rules. For instance, a Cisco bridge will always translate FDDI_SNAP to Ethernet_II and vice versa, and it will always convert FDDI_802.2 to Ethernet 802.2 and vice versa.

Cisco Routers vs. Catalyst Bridging



This diagram shows a network that uses both a Catalyst switch and a Cisco router bridging IPX. Since Catalyst switches maintain the encapsulation type for each device, Client #1 can use any encapsulation style while client #2 must use Ethernet_II. If client #2 were to used Ethernet_SNAP instead, the packet from client #2 (Ethernet_SNAP) would be translated to FDDI_SNAP − that's no problem. But the response packet from the server (FDDI_SNAP) would be translated to Ethernet_II which client #2 wouldn't understand.

Similarly, if the server on the FDDI is using FDDI_802.2, then client #2 must use the Ethernet_802.2 encapsulation style since the Cisco bridge will always convert FDDI_802.2 to Ethernet_802.2, and vice versa. Again, the Catalyst maintains a list of the encapsulations, so it isn't bound by this restriction.

Remember, Catalysts perform translational bridging and do not route packets. Using Catalysts, you get only a flat−bridged network, not a routed one, and all the broadcast packets will be flooded across the entire network any time a packet is broadcasted. Between different media, use routers to isolate traffic more locally and avoid the complication of mixed−media bridging.

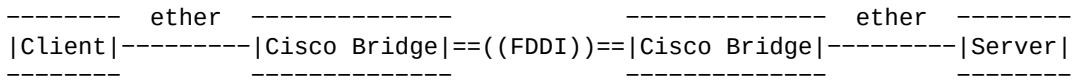
Translational Bridging in Detail

Cisco routers bridge IPX packets from Ethernet to FDDI to Ethernet like this:

Ethernet _SNAP(E) −−> FDDI_SNAP(F) −−> Ethernet_II(E)

Ethernet_II(E) <−− FDDI_SNAP(F) <−− Ethernet_II(E)

Ethernet_802.2(E)<−−>FDDI_802.2(F)<−−>Ethernet_802.2(E)



Using the diagram above and a reference packet of a GNS request from a client, let's take a look at some different IPX encapsulation examples. Each of the four examples below includes a hex dump of the packet in its three forms: sourced from the client, on the intermediate FDDI ring, and on the destination Ethernet.

IPX Ethernet_SNAP

Cisco config command **ipx encapsulation snap**

Ethernet_SNAP(E) −−> FDDI_SNAP(F) −−> Ethernet_II(E)

```
0000  FF FF FF FF FF FF 02 60  8C AD 43 7D 00 2A AA AA
0010  03 00 00 00 81 37 FF FF  00 22 00 11 00 00 00 00  Ethernet
0020  FF FF FF FF FF FF 04 52  00 00 00 00 02 60 8C AD
0030  43 7D 40 06 00 03 00 04  00 0F 53 45

0000- 50 FF FF FF FF FF FF 40  06 31 B5 C2 BE AA AA 03
0010- 00 00 00 81 37 FF FF 00  22 00 11 00 00 00 00 FF  FDDI
0020- FF FF FF FF FF 04 52 00  00 00 00 02 60 8C AD 43
0030- 7D 40 06 00 03 00 04 DF  77 9B C3

0000  FF FF FF FF FF FF 02 60  8C AD 43 7D 81 37 FF FF
0010  00 22 00 11 00 00 00 00  FF FF FF FF FF FF 04 52  Ethernet
0020  00 00 00 00 02 60 8C AD  43 7D 40 06 00 03 00 04
0030  00 00 00 00 20 0B 00 08  00 00 00 01
```

The returning traffic would show:

Ethernet_II(E) <−− FDDI_SNAP(F) <−− Ethernet_II(E)

The original sender which speaks Ethernet_SNAP doesn't understand the response packet in Ethernet_II encapsulation style, so there is no connectivity.

IPX Ethernet_802.3

Cisco config command **ipx encapsulation novell−ether**

Novell netware recognizes only two encapsulation styles for FDDI media: FDDI_SNAP (default) and FDDI_802.2. Since Novell doesn't recognize the encapsulation style of a packet, we have to make up a name for it. Here, we've made up the name FDDI_Raw.

Ethernet_802.3(E) −−> FDDI_Raw(F) −−> Ethernet_802.3(E)

```
0000  FF FF FF FF FF FF 02 60  8C AD 43 7D 00 22 FF FF
0010  00 22 00 11 00 00 00 00  FF FF FF FF FF FF 04 52  Ethernet
0020  00 00 00 00 02 60 8C AD  43 7D 40 06 00 03 00 04
0030  43 7D 40 06 00 03 00 04  0A 73 75 70
```

```
0000- 50 FF FF FF FF FF FF 40  06 31 B5 C2 BE FF FF 00  |
0010- 22 00 11 00 00 00 00 FF  FF FF FF FF FF 04 52 00  | FDDI
0020- 00 00 00 02 60 8C AD 43  7D 40 06 00 03 00 04 4D  |
0030- E1 F9 08                                         _|
                                                       _
0000  FF FF FF FF FF FF 02 60  8C AD 43 7D 00 22 FF FF  |
0010  00 22 00 11 00 00 00 00  FF FF FF FF FF FF 04 52  | Ethernet
0020  00 00 00 00 02 60 8C AD  43 7D 40 06 00 03 00 04  |
0030  00 0F 00 00 00 00 00 00  00 00 00 00              _|
```

If there is a Novell server on the FDDI that understands FDDI_Raw (some third party vendors), then the server can talk to a client on Ethernet using the Ethernet_802.3 encapsulation style. Those drivers supporting FDDI_Raw are from the FDDI card vendors and not from Novell.

The response traffic would show:

Ethernet_802.3(E) <-- FDDI_Raw(F) <-- Ethernet_802.3(E)

The original sender understands the response packet, so this case works fine. Catalysts would translate Ethernet_802.3 to FDDI_SNAP or FDDI_802.2, depending on the encapsulation used by the server.

IPX Ethernet_II

Cisco config command **ipx encapsulation arpa**

Ethernet_II(E) --> FDDI_SNAP(F) --> Ethernet_II(E)

```
                                                       _
0000  FF FF FF FF FF FF 02 60  8C AD 43 7D 81 37 FF FF  |
0010  00 22 00 11 00 00 00 00  FF FF FF FF FF FF 04 52  | Ethernet
0020  00 00 00 00 02 60 8C AD  43 7D 40 06 00 03 00 04  |
0030  03 00 18 00 21 43 00 01  00 0F 53 45              _|
                                                       _
0000- 50 FF FF FF FF FF FF 40  06 31 B5 C2 BE AA AA 03  |
0010- 00 00 00 81 37 FF FF 00  22 00 11 00 00 00 00 FF  |
0020- FF FF FF FF FF 04 52 00  00 00 00 02 60 8C AD 43  | FDDI
0030- 7D 40 06 00 03 00 04 02  00 18 00 1E 3D 00 01 0A  |
0040- 73 75 70 BB 7F DA CD                             _|
                                                       _
0000  FF FF FF FF FF FF 02 60  8C AD 43 7D 81 37 FF FF  |
0010  00 22 00 11 00 00 00 00  FF FF FF FF FF FF 04 52  | Ethernet
0020  00 00 00 00 02 60 8C AD  43 7D 40 06 00 03 00 04  |
0030  03 00 04 06 00 03 00 04  00 0F 53 45              _|
```

Ethernet_II(E) <-- FDDI_SNAP(F) <-- Ethernet_II(E)

No problem here!

IPX Ethernet_802.2

Cisco config command **ipx encapsulation sap**

Ethernet_802.2(E) --> FDDI_802.2(F) --> Ethernet_802.2(E)

```
                                                       _
0000  FF FF FF FF FF FF 02 60  8C AD 43 7D 00 25 E0 E0  |
0010  03 FF FF 00 22 00 11 00  00 00 00 FF FF FF FF FF  | Ethernet
0020  FF 04 52 00 00 00 00 02  60 8C AD 43 7D 40 06 00  |
0030  03 00 04 00 21 43 00 01  00 0F 53 45              _|
                                                       _
```

```
0000- 50 FF FF FF FF FF FF 40  06 31 B5 C2 BE E0 E0 03  |
0010- FF FF 00 22 00 11 00 00  00 00 FF FF FF FF FF FF  | FDDI
0020- 04 52 00 00 00 00 02 60  8C AD 43 7D 40 06 00 03  |
0030- 00 04 C0 8A 83 22                                _|
                                                       _
0000  FF FF FF FF FF FF 02 60  8C AD 43 7D 00 25 E0 E0  |
0010  03 FF FF 00 22 00 11 00  00 00 00 FF FF FF FF FF  | Ethernet
0020  FF 04 52 00 00 00 00 02  60 8C AD 43 7D 40 06 00  |
0030  03 00 04 C0 00 00 00 00  00 00 00 00             _|
```

This one also works fine.

Related Information

- **LAN Product Support**
- **LAN Switching Technology Support**
- **Technical Support & Documentation − Cisco Systems**

Updated: Oct 04, 2005 Document ID: 10673