

# Configuring Serial Tunnel and Block Serial Tunnel

---

Cisco's serial tunnel (STUN) implementation allows Synchronous Data Link Control (SDLC) protocol devices and High-Level Data Link Control (HDLC) devices to connect to one another through a multiprotocol internetwork rather than through a direct serial link. STUN encapsulates SDLC frames in either the Transmission Control Protocol/Internet Protocol (TCP/IP) or the HDLC protocol. STUN provides a straight passthrough of all SDLC traffic (including control frames, such as Receiver Ready) end-to-end between Systems Network Architecture (SNA) devices.

Cisco's SDLC local acknowledgment provides local termination of the SDLC session so that control frames no longer travel the WAN backbone networks. This means end nodes do not time out, and a loss of sessions does not occur. You can configure your network with STUN, or with STUN and SDLC local acknowledgment. To enable SDLC local acknowledgment, the Cisco IOS software must first be enabled for STUN and routers configured to appear on the network as primary or secondary SDLC nodes. TCP/IP encapsulation must be enabled. Cisco's SDLC Transport feature also provides priority queuing for TCP encapsulated frames.

Cisco's block serial tunnel (BSTUN) implementation enables Cisco series 2500, 4000, and 4500, 4700, 7200 routers to support devices that use the Binary Synchronous Communications (Bisync) datalink protocol and asynchronous security protocols that include Adplex, ADT Security Systems, Inc., Diebold, and asynchronous generic traffic. BSTUN implementation is also supported on the 4T network interface module (NIM) on the Cisco series 4000 and 4500. Our support of the bisync protocol enables enterprises to transport Bisync traffic and SNA multiprotocol traffic over the same network.

This chapter describes how to configure STUN and BSTUN. For a complete description of the STUN and BSTUN commands in this chapter, refer to the "STUN and BSTUN Commands" chapter of the *Bridging and IBM Networking Command Reference*. To locate documentation of specific commands, use the command reference index or search online.

## STUN Configuration Task List

To configure and monitor STUN, or STUN local acknowledgment, complete the tasks in the following sections:

- Enable STUN
- Configure SDLC Broadcast
- Specify STUN Protocol Group
- Enable STUN Keepalive
- Enable STUN Remote Keepalive
- Enable STUN Quick-Response

- Enable STUN Interfaces
- Establish the Frame Encapsulation Method
- Configure STUN with Multilink Transmission Groups
- Set Up STUN Traffic Priorities
- Monitor STUN Network Activity

The “STUN Configuration Examples” section follows these configuration tasks.

## Enable STUN

To enable STUN, perform the following task in global configuration mode:

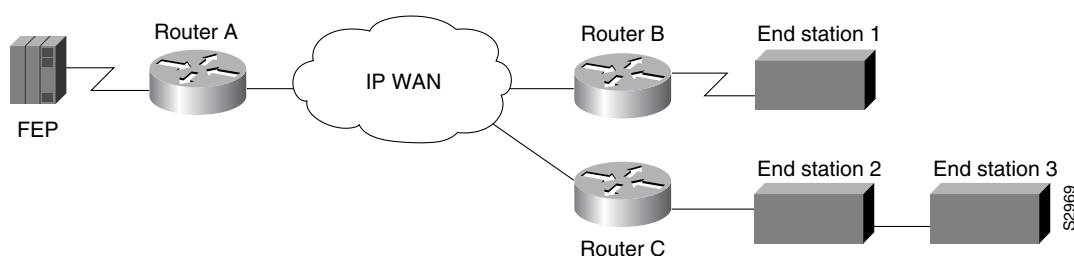
Task	Command
Enable STUN for a particular IP address.	<b>stun peer-name ip-address</b>

When configuring redundant links, ensure that the STUN peer names you choose on each router are the IP addresses of the most stable interfaces on each device, such as a loopback or Ethernet interface. See “STUN Configuration Examples” later in this chapter.

## Configure SDLC Broadcast

The SDLC broadcast feature allows SDLC broadcast address FF to be replicated for each of the STUN peers, so each of the end stations receives the broadcast frame. For example, in Figure 92, the FEP views the end stations 1, 2, and 3 as if they are on an SDLC multidrop link. Any broadcast frame sent from the FEP to Router A is duplicated and sent to each of the downstream routers (B and C).

**Figure 92 SDLC Broadcast across Virtual Multidrop Lines**



To enable SDLC broadcast, perform the following task in interface configuration mode:

Task	Command
Enable SDLC broadcast.	<b>sdlc virtual-multidrop</b>

Only enable SDLC broadcast on the device that is configured to be the secondary station on the SDLC link (Router A in Figure 92).

You must also configure SDLC address FF on Router A for each of the STUN peers. To do so, perform the following task in interface configuration mode:

Task	Command
Configure SDLC address FF on Router A for each STUN peer.	<b>stun route address</b> <i>address-number</i> <b>tcp</b> <i>ip-address</i> [ <b>local-ack</b> ] [ <b>priority</b> ] [ <b>tcp-queue-max</b> ]

## Specify STUN Protocol Group

Place each STUN interface in a group that defines the ISO 3309-compliant framed protocol running on that link. Packets will only travel between STUN interfaces that are in the same protocol group.

There are three predefined STUN protocols:

- Basic
- SDLC
- SDLC transmission group

You can also specify a custom STUN protocol.

If you want to use the STUN Local Acknowledgment feature, you must specify either the SDLC protocol or the SDLC transmission group protocol.

---

**Note** Before you can specify a custom protocol, you must first define the protocol; see the section “Create and Specify a Custom STUN Protocol” later in this chapter for the procedure.

---

### Specify a Basic STUN Group

The basic STUN protocol does not depend on the details of serial protocol addressing and is used when addressing is unimportant. Use this when your goal is to replace one or more sets of point-to-point (not multidrop) serial links by using a protocol other than SDLC. Perform the following task in global configuration mode:

Task	Command
Specify a basic protocol group and assign a group number.	<b>stun protocol-group</b> <i>group-number</i> <b>basic</b>

### Specify an SDLC Group

You can specify SDLC protocol groups to associate interfaces with the SDLC protocol. Use the SDLC STUN protocol to place the routers in the midst of either point-to-point or multipoint (multidrop) SDLC links. To define an SDLC protocol group, perform the following task in global configuration mode:

Task	Command
Specify an SDLC protocol group and assign a group number.	<b>stun protocol-group</b> <i>group-number</i> <b>sdlc</b>

If you specify an SDLC protocol group, you cannot specify the **stun route all** command on any interface of that group.

For an example of how to configure an SDLC protocol group, see the “Configuring Serial Link Address Prioritization Using STUN TCP/IP Encapsulation Example” later in this chapter.

### Specify an SDLC Transmission Group

An SNA transmission group is a set of lines providing parallel links to the same pair of SNA front-end-processor (FEP) devices. This provides redundancy of paths for fault tolerance and load sharing. To define an SDLC transmission group, perform the following task in global configuration mode:

Task	Command
Specify an SDLC protocol group, assign a group number, and create an SNA transmission group.	<b>stun protocol-group</b> <i>group-number</i> <b>sdlc</b> <b>sdlc-tg</b>

All STUN connections in a transmission group must connect to the same IP address and use the SDLC local acknowledgment feature.

### Create and Specify a Custom STUN Protocol

To define a custom protocol and tie STUN groups to the new protocol, perform the following tasks in global configuration mode:

Task	Command
<b>Step 1</b> Create a custom protocol.	<b>stun schema</b> <i>name</i> <b>offset</b> <i>constant-offset</i> <b>length</b> <i>address-length</i> <b>format</b> <i>format-keyword</i>
<b>Step 2</b> Specify the custom protocol group and assign a group number.	<b>stun protocol-group</b> <i>group-number</i> <b>schema</b>

### Enable STUN Keepalive

To define the number of times to attempt a peer connection before declaring the peer connection down, perform the following task in global configuration mode:

Task	Command
Specify the number of times to attempt a peer connection.	<b>stun keepalive-count</b>

### Enable STUN Remote Keepalive

To enable detection of the loss of a peer, perform the following task in global configuration mode:

Task	Command
Enable detection of the loss of a peer.	<b>stun remote-peer-keepalive</b> <i>seconds</i>

## Enable STUN Quick-Response

You can enable STUN quick-response, which improves network performance when used with local acknowledgment. When STUN quick-response is used with local acknowledgment, the router responds to an exchange identification (XID) or a Set Normal Response Mode (SNRM) request with a Disconnect Mode (DM) response when the device is not in the CONNECT state. The request is then passed to the remote router and, if the device responds, the reply is cached. The next time the device is sent an XID or SNRM, the router replies with the cached DM response.

---

**Note** Using STUN quick-response avoids an AS/400 line reset problem by eliminating the Non-Productive Receive Timer (NPR) expiration in the AS/400. With STUN quick-response enabled, the AS/400 receives a response from the polled device, even when the device is down. If the device does not respond to the forwarded request, the router continues to respond with the cached DM response.

---

To enable STUN quick-response, perform the following task in global configuration mode:

Task	Command
Enable STUN quick-response.	<b>stun quick-response</b>

## Enable STUN Interfaces

You must enable STUN on serial interfaces and place these interfaces in the protocol groups you have defined. To enable STUN on an interface and to place the interface in a STUN group, perform the following tasks in interface configuration mode:

Task	Command
<b>Step 1</b> Enable STUN function on a serial interface.	<b>encapsulation stun</b>
<b>Step 2</b> Place the interface in a previously defined STUN group.	<b>stun group <i>group-number</i></b>

When a given serial link is configured for the STUN function, it is no longer a shared multiprotocol link. All traffic that arrives on the link will be transported to the corresponding peer as determined by the current STUN configuration.

## Establish the Frame Encapsulation Method

To allow SDLC frames to travel across a multimedia, multiprotocol network, you must encapsulate them using one of the methods in the following sections:

- Configure HDLC Encapsulation without Local Acknowledgment
- Configure TCP Encapsulation without Local Acknowledgment
- Configure TCP Encapsulation with SDLC Local Acknowledgment and Priority Queuing
- Configure Local Acknowledgment for Direct Frame Relay Connectivity

### Configure HDLC Encapsulation without Local Acknowledgment

You can encapsulate SDLC or HDLC frames using the HDLC protocol. The outgoing serial link can still be used for other kinds of traffic. The frame is not TCP encapsulated. To configure HDLC encapsulation, perform one of the following tasks in interface configuration mode:

Task	Command
Forward all HDLC or SDLC traffic of the identified interface number.	<b>stun route all interface serial <i>number</i></b>
Forward all HDLC or SDLC traffic on a direct STUN link.	<b>stun route all interface serial <i>number</i> direct</b>
Forward HDLC or SDLC traffic of the identified address.	<b>stun route address <i>address-number</i> interface serial <i>number</i></b>
Forward HDLC or SDLC traffic of the identified address across a direct STUN link.	<b>stun route address <i>address-number</i> interface serial <i>number</i> direct</b>

Use the **no** forms of these commands to disable HDLC encapsulation.

---

**Note** You can forward all traffic only when you are using basic STUN protocol groups.

---

### Configure TCP Encapsulation without Local Acknowledgment

If you do not want to use SDLC local acknowledgment and only need to forward all SDLC frames encapsulated in TCP, complete the following tasks in interface configuration mode:

Task	Command
Forward all TCP traffic for this IP address.	<b>stun route all tcp <i>ip-address</i></b>
Specify TCP encapsulation.	<b>stun route address <i>address-number</i> tcp <i>ip-address</i> [local-ack] [priority] [tcp-queue-max]</b>

Use the **no** form of these commands to disable forwarding of all TCP traffic.

This configuration is typically used when two routers can be connected via an IP network as opposed to a point-to-point link.

## Configure TCP Encapsulation with SDLC Local Acknowledgment and Priority Queuing

You configure SDLC local acknowledgment using TCP encapsulation. When you configure SDLC local acknowledgment, you also have the option to enable support for priority queuing.

---

**Note** To enable SDLC local acknowledgment, you must specify an SDLC or SDLC transmission group.

---

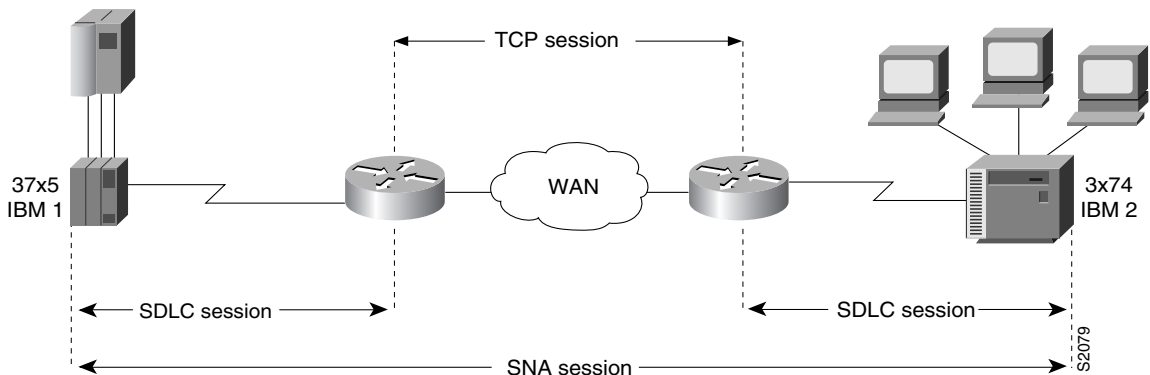
SDLC local acknowledgment provides local termination of the SDLC session so that control frames no longer travel the WAN backbone networks. This means that time-outs are less likely to occur.

Figure 93 illustrates an SDLC session. IBM 1, using a serial link, can communicate with IBM 2 on a different serial link separated by a wide-area backbone network. Frames are transported between Router A and Router B using STUN, but the SDLC session between IBM 1 and IBM 2 is still end-to-end. Every frame generated by IBM 1 traverses the backbone network to IBM 2, which, upon receipt of the frame, acknowledges it.

### Figure 93 SDLC Session without Local Acknowledgment

With SDLC local acknowledgment, the SDLC session between the two end nodes is not end-to-end, but instead terminates at the two local routers, as shown in Figure 94. The SDLC session with IBM 1 ends at Router A, and the SDLC session with IBM 2 ends at Router B. Both Router A and Router B execute the full SDLC protocol as part of SDLC Local Acknowledgment. Router A acknowledges frames received from IBM 1. The node IBM 1 treats the acknowledgments it receives as if they are from IBM 2. Similarly, Router B acknowledges frames received from IBM 2. The node IBM 2 treats the acknowledgments it receives as if they are from IBM 1.

Figure 94 SDLC Session with Local Acknowledgment



To configure TCP encapsulation with SDLC local acknowledgment and priority queuing, perform the tasks in the following sections:

- Assign the Router an SDLC Primary or Secondary Role
- Enable the SDLC Local Acknowledgment Feature
- Establish Priority Queuing Levels

### Assign the Router an SDLC Primary or Secondary Role

To establish local acknowledgment, the router must play the role of an SDLC primary or secondary node. Primary nodes poll secondary nodes in a predetermined order. Secondaries then transmit if they have outgoing data.

For example, in the IBM environment, an FEP is the primary station and cluster controllers are secondary stations. If the router is connected to a cluster controller, the router should appear as an FEP and must therefore be assigned the role of a primary SDLC node. If the router is connected to an FEP, the router should appear as a cluster controller and must therefore be assigned the role of a secondary SDLC node. Devices connected to SDLC primary end-stations must play the role of an SDLC secondary and routers attached to SDLC secondary end stations must play the role of an SDLC primary station.

To assign the router a primary or secondary role, perform one of the following tasks in interface configuration mode:

Task	Command
Assign the STUN-enabled router an SDLC primary role.	<b>stun sdlc-role primary</b>
Assign the STUN-enabled router an SDLC secondary role.	<b>stun sdlc-role secondary</b>

## Enable the SDLC Local Acknowledgment Feature

To enable SDLC local acknowledgment, complete the following task in interface configuration mode:

Task	Command
Establish SDLC local acknowledgment using TCP encapsulation.	<b>stun route address</b> <i>address-number</i> <b>tcp</b> <i>ip-address</i> [ <b>local-ack</b> ] [ <b>priority</b> ] [ <b>tcp-queue-max</b> ]

The **stun route address 1 tcp local-ack priority tcp-queue-max** interface configuration command enables local acknowledgment and TCP encapsulation. Both options are required to use transmission groups. You should specify the SDLC address with the echo bit turned off for transmission group interfaces. The SDLC broadcast address 0xFF is routed automatically for transmission group interfaces. The **priority** keyword creates multiple TCP sessions for this route. The **tcp-queue-max** keyword sets the maximum size of the outbound TCP queue for the SDLC. The default TCP queue size is 100. The value for **hold-queue in** should be greater than the value for **tcp-queue-max**.

You can use the **priority** keyword (to set up the four levels of priorities to be used for TCP encapsulated frames) at the same time you enable local acknowledgment. The **priority** keyword is described in the following section. Use the **no** form of this command to disable SDLC Local Acknowledgment. For an example of how to enable local acknowledgment, see “Configuring Serial Link Address Prioritization Using STUN TCP/IP Encapsulation Example” later in this chapter.

## Establish Priority Queuing Levels

With SDLC local acknowledgment enabled, you can establish priority levels used in priority queuing for serial interfaces. The priority levels are as follows:

- Low
- Medium
- Normal
- High

To set the priority queuing level, perform the following task in interface configuration mode:

Task	Command
Establish the four levels of priorities to be used in priority queuing.	<b>stun route address</b> <i>address-number</i> <b>tcp</b> <i>ip-address</i> [ <b>local-ack</b> ] <b>priority</b> [ <b>tcp-queue-max</b> ]

Use the **no** form of this command to disable priority settings. For an example of how to establish priority queuing levels, see “Configuring Serial Link Address Prioritization Using STUN TCP/IP Encapsulation Example” later in this chapter.

## Configure Local Acknowledgment for Direct Frame Relay Connectivity

To implement STUN with local acknowledgment using direct Frame Relay encapsulation, perform the following task in interface configuration mode:

Task	Command
Configure Frame Relay encapsulation between STUN peers with local acknowledgment.	<b>stun route address</b> <i>sdlc-addr</i> <b>interface</b> <i>frame-relay-port</i> <b>dlci</b> <i>number localsap</i> <b>local-ack</b> <b>cls</b>

## Configure STUN with Multilink Transmission Groups

You can configure multilink SDLC transmission groups across STUN connections between IBM communications controllers such as IBM 37x5s. Multilink transmission groups allow you to collapse multiple WAN leased lines into one leased line.

SDLC multilink transmission groups provide the following features:

- Network Control Program (NCP) SDLC address allowances, including echo and broadcast addressing.
- Remote NCP load sequence. After a SIM/RIM exchange but before a SNRM/UA exchange, NCPs send numbered I-frames. During this period, I-frames are not locally acknowledged, but instead are passed through. After the SNRM/UA exchange, local acknowledgment occurs.
- Rerouting of I-frames sent by the Cisco IOS software to the NCP if a link is lost in a multilink transmission group.
- Flow control rate tuning causes a sending NCP to “feel” WAN congestion and hold frames that would otherwise be held by the Cisco IOS software waiting to be transmitted on the WAN. This allows the NCP to perform its class-of-service algorithm more efficiently based on a greater knowledge of network congestion.

STUN connections that are part of a transmission group must have local acknowledgment enabled. Local acknowledgment keeps SDLC poll traffic off the WAN and reduces store-and-forward delays through the router. It also might minimize the number of NCP timers that expire due to network delay. Also, these STUN connections must go to the same IP address. This is because SNA transmission groups are parallel links between the same pair of IBM communications controllers.

## Design Recommendations

This section provides some recommendations that are useful in configuring SDLC multilink transmission groups.

The bandwidth of the WAN should be larger than or equal to the aggregate bandwidth of all serial lines to avoid excessive flow control and ensure response time does not degrade. If other protocols are also using the WAN, ensure that the WAN bandwidth is significantly greater than the aggregate SNA serial line bandwidth to ensure that the SNA traffic does not monopolize the WAN.

When you use a combination of routed transmission groups and directly connected NCP transmission groups, you need to plan the configuration carefully to ensure that SNA sessions do not stop unexpectedly. Assuming that hardware reliability is not an issue, single-link routed transmission groups are as reliable as direct NCP-to-NCP single-link transmission groups. This is true because neither the NCP nor the Cisco IOS software can reroute I-frames when a transmission group has only one link. Additionally, a multilink transmission group directed between NCPs and a multilink transmission group through a router are equally reliable. Both can perform rerouting.

However, you might run into problems if you have a configuration in which two NCPs are directly connected (via one or more transmission group links) and one link in the transmission group is routed. The NCPs treat this as a multilink transmission group. However, the Cisco IOS software views the transmission group as a single-link transmission group.

A problem can arise in the following situation: Assume that an I-frame is being transmitted from NCP A (connected to router A) to NCP B (connected to router B) and that all SDLC links are currently active. Router A acknowledges the I-frame sent from NCP A and sends it over the WAN. If, before the I-frame reaches Router B, the SDLC link between router B and NCP B goes down, Router B attempts to reroute the I-frame on another link in the transmission group when it receives the I-frame. However, because this is a single-link transmission group, there are no other routes, and Router B drops the I-frame. NCP B never receives this I-frame because Router A acknowledges its

receipt, and NCP A marks it as transmitted and deletes it. NCP B detects a gap in the transmission group sequence numbers and waits to receive the missing I-frame. NCP B waits forever for this I-frame, and does not send or receive any other frames. NCP B is technically not operational and all SNA sessions through NCP B are lost.

Finally, consider a configuration in which one or more lines of an NCP transmission group are connected to a router and one or more lines are directly connected between NCPs. If the network delay associated with one line of an NCP transmission group is different from the delay of another line in the same NCP transmission group, the receiving NCP spends additional time resequencing PIUs.

## Set Up STUN Traffic Priorities

Use the methods described in the following sections to determine the order in which traffic should be handled on the network:

- Assign Queuing Priorities
- Prioritize STUN Traffic over All Other Traffic

### Assign Queuing Priorities

You can assign queuing priorities by one of the following:

- Serial interface address or TCP port
- Logical unit (LU) address

#### Prioritize by Serial Interface Address or TCP Port

You can prioritize traffic on a per-serial-interface address or TCP port basis. You might want to do this so that traffic between one source-destination pair is always sent before traffic between another source-destination pair.

---

**Note** You must first enable local acknowledgment and priority levels as described earlier in this chapter.

---

To prioritize traffic, perform one of the following tasks in global configuration mode:

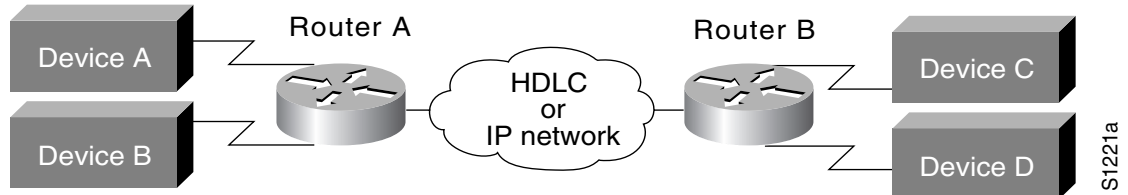
Task	Command
Assign a queuing priority to the address of the STUN serial interface.	<b>priority-list</b> <i>list-number</i> <b>stun queue address</b> <i>group-number</i> <i>address-number</i>
Assign a queuing priority to a TCP port.	<b>priority-list</b> <i>list-number</i> <b>protocol ip queue tcp</b> <i>tcp-port-number</i>

You must also perform the following task in interface configuration mode:

Task	Command
Assign a priority list to a priority group.	<b>priority-group</b> <i>list-number</i>

Figure 95 illustrates serial link address prioritization. Device A communicates with Device C, and Device B communicates with Device D. With the serial link address prioritization, you can choose to give A-C a higher priority over B-D across the serial tunnel.

**Figure 95 Serial Link Address Prioritization**



To disable priorities, use the **no** forms of these commands.

For an example of how to prioritize traffic according to serial link address, see “Configuring Serial Link Address Prioritization Using STUN TCP/IP Encapsulation Example” later in this chapter.

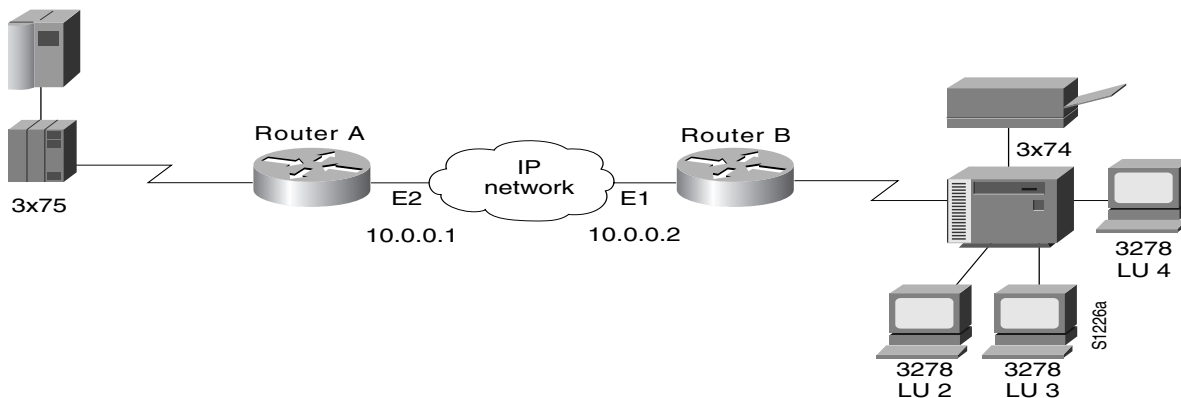
### Prioritize by Logical Unit Address

SNA local logical unit (LU) address prioritization is specific to IBM SNA connectivity and is used to prioritize SNA traffic on either STUN or remote source-route bridging (RSRB). To set the queuing priority by LU address, perform the following task in interface configuration mode:

Task	Command
Assign a queuing priority based on the logical unit address.	<b>locaddr-priority-list</b> <i>list-number address-number queue-keyword</i>

In Figure 96, LU address prioritization can be set so that particular LUs receive data in preference to others or so that LUs have priority over the printer, for example.

**Figure 96 SNA LU Address Prioritization**



To disable this priority, use the **no** form of this command.

For an example of how to prioritize traffic according to logical unit address, see “Configuring LOCADDR Priority Groups for STUN Example” later in this chapter.

## Prioritize STUN Traffic over All Other Traffic

You can prioritize STUN traffic to be routed first before all other traffic on the network. To give STUN traffic this priority, perform the following task in global configuration mode:

Task	Command
Prioritize STUN traffic in your network over that of other protocols.	<b>priority-list</b> <i>list-number</i> <b>stun queue address</b> <i>group-number address-number</i>

To disable this priority, use the **no** form of this command.

For an example of how to prioritize STUN traffic over all other traffic, see “Configuring Serial Link Address Prioritization Using STUN TCP/IP Encapsulation Example” later in this chapter.

## Monitor STUN Network Activity

You can list statistics regarding STUN interfaces, protocol groups, number of packets sent and received, local acknowledgment states, and more. To get activity information, perform the following task in EXEC mode:

Task	Command
List the status display fields for STUN interfaces.	<b>show stun</b>

## STUN Configuration Examples

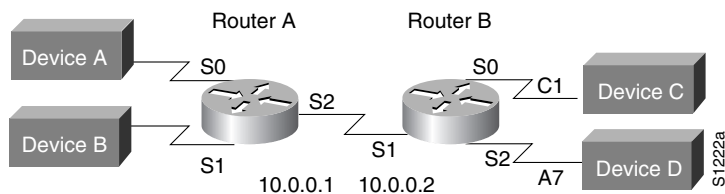
The following sections provide STUN configuration examples:

- Configuring STUN Priorities Using HDLC Encapsulation Example
- Configuring SDLC Broadcast Example
- Configuring Serial Link Address Prioritization Using STUN TCP/IP Encapsulation Example
- Configuring STUN Multipoint Implementation Using a Line-Sharing Device Example
- Configuring STUN Local Acknowledgment for SDLC Example
- Configuring STUN Local Acknowledgment for Frame Relay Example
- Configuring LOCADDR Priority Groups—Simple Example
- Configuring LOCADDR Priority Groups for STUN Example

### Configuring STUN Priorities Using HDLC Encapsulation Example

Assume that the link between Router A and Router B in Figure 97 is a serial tunnel that uses the simple serial transport mechanism. Device A communicates with Device C (SDLC address C1) with a high priority. Device B communicates with Device D (SDLC address A7) with a normal priority.

Figure 97 STUN Simple Serial Transport



The following configurations set the priority of STUN hosts A, B, C, and D.

### Configuration for Router A

```

stun peer-name 1.0.0.1
stun protocol-group 1 sdlc
stun protocol-group 2 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 interface serial 2
!
interface serial 1
no ip address
encapsulation stun
stun group 2
stun route address A7 interface serial 2
!
interface serial 2
ip address 1.0.0.1 255.0.0.0
priority-group 1
!
priority-list 1 stun high address 1 C1
priority-list 1 stun low address 2 A7
    
```

### Configuration for Router B

```

stun peer-name 1.0.0.2
stun protocol-group 1 sdlc
stun protocol-group 2 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 interface serial 1
!
interface serial 1
ip address 1.0.0.2 255.0.0.0
priority-group 1
!
interface serial 2
no ip address
encapsulation stun
stun group 2
stun route address A7 interface serial 1
!
priority-list 1 stun high address 1 C1
priority-list 1 stun low address 2 A7
    
```

## Configuring SDLC Broadcast Example

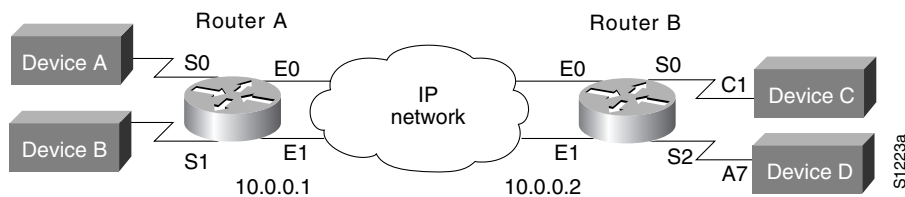
In the following example, an FEP views end stations 1, 2, and 3 as if they were on an SDLC multidrop link. Any broadcast frame sent from the FEP to Router A is duplicated and sent to each of the downstream routers (B and C):

```
stun peer-name xxx.xxx.xxx.xxx
stun protocol-group 1 sdlc
interface serial 1
encapsulation stun
stun group 1
stun sdlc-role secondary
sdlc virtual-multidrop
sdlc address 1
sdlc address 2
sdlc address 3
stun route address 1 tcp yyy.yyy.yyy.yyy local-ack
stun route address 2 tcp zzz.zzz.zzz.zzz local-ack
stun route address 3 tcp zzz.zzz.zzz.zzz local-ack
stun route address FF tcp yyy.yyy.yyy.yyy
stun route address FF tcp zzz.zzz.zzz.zzz
```

## Configuring Serial Link Address Prioritization Using STUN TCP/IP Encapsulation Example

Assume that the link between Router A and Router B is a serial tunnel that uses the TCP/IP encapsulation as shown in Figure 98. Device A communicates with Device C (SDLC address C1) with a high priority. Device B communicates with Device D (SDLC address A7) with a normal priority. The configuration file for each router follows the figure.

**Figure 98 STUN TCP/IP Encapsulation**



### Configuration for Router A

```
stun peer-name 1.0.0.1
stun protocol-group 1 sdlc
stun protocol-group 2 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 tcp 1.0.0.2 local-ack priority
priority-group 1
!
interface serial 1
no ip address
encapsulation stun
stun group 2
stun route address A7 tcp 1.0.0.2 local-ack priority
priority-group 2
!
```

```
interface ethernet 0
  ip address 1.0.0.1 255.0.0.0
!
interface ethernet 1
  ip address 1.0.0.3 255.0.0.0
!
priority-list 1 protocol ip high tcp 1994
priority-list 1 protocol ip medium tcp 1990
priority-list 1 protocol ip normal tcp 1991
priority-list 1 protocol ip low tcp 1992
priority-list 1 stun high address 1 C1
!
priority-list 2 protocol ip high tcp 1994
priority-list 2 protocol ip medium tcp 1990
priority-list 2 protocol ip normal tcp 1991
priority-list 2 protocol ip low tcp 1992
priority-list 2 stun normal address 2 A7
!
hostname routerA
router igrp
network 1.0.0.0
```

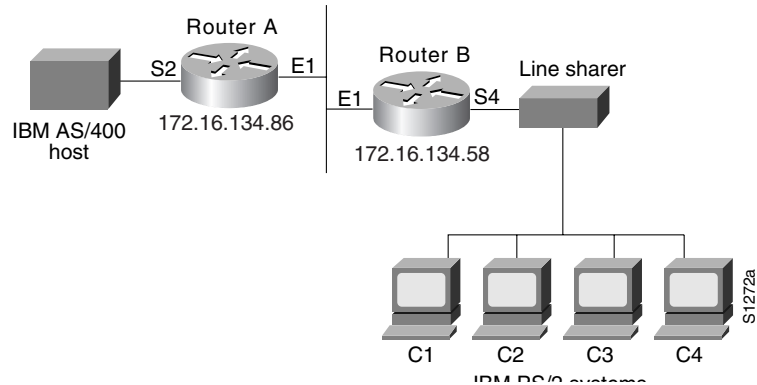
### Configuration for Router B

```
stun peer-name 1.0.0.2
stun protocol-group 1 sdlc
stun protocol-group 2 sdlc
!
interface serial 0
  no ip address
  encapsulation stun
  stun group 1
  stun route address C1 tcp 1.0.0.1 local-ack priority
  priority-group 1
!
interface serial 2
  no ip address
  encapsulation stun
  stun group 2
  stun route address A7 tcp 1.0.0.1 local-ack priority
  priority-group 2
!
interface ethernet 0
  ip address 1.0.0.2 255.0.0.0
!
interface ethernet 1
  ip address 1.0.0.4 255.0.0.0
!
priority-list 1 protocol ip high tcp 1994
priority-list 1 protocol ip medium tcp 1990
priority-list 1 protocol ip normal tcp 1991
priority-list 1 protocol ip low tcp 1992
priority-list 1 stun high address 1 C1
!
priority-list 2 protocol ip high tcp 1994
priority-list 2 protocol ip medium tcp 1990
priority-list 2 protocol ip normal tcp 1991
priority-list 2 protocol ip low tcp 1992
priority-list 2 stun normal address 2 A7
!
hostname routerB
router igrp 109
network 1.0.0.0
```

## Configuring STUN Multipoint Implementation Using a Line-Sharing Device Example

In Figure 99, four separate PS/2 computers are connected to a line-sharing device off of Router B. Each PS/2 computer has four sessions open on an AS/400 device attached to Router A. Router B functions as the primary station, while Router A functions as the secondary station. Both routers locally acknowledge packets from the IBM PS/2 systems.

**Figure 99 STUN Communication Involving a Line-Sharing Device**



The configuration file for the routers shown in Figure 99 follows.

### Configuration for Router A

```

! enter the address of the stun peer
stun peer-name 150.136.134.86
! specify that group 4 uses the SDLC protocol
stun protocol-group 4 sdhc
stun remote-peer-keepalive

interface ethernet 1
! enter the IP address for the Ethernet interface
ip address 150.136.134.86 255.255.255.0
!
! description of IBM AS/400 link
interface serial 2
! description of IBM AS/400 link; disable the IP address on a serial interface
no ip address
! enable STUN encapsulation on this interface
encapsulation stun
! apply previously defined stun group 4 to serial interface 2
stun group 4
! establish this router as a secondary station
stun sdhc-role secondary
! wait up to 63000 msec for a poll from the primary before timing out
sdhc poll-wait-timeout 63000
! list addresses of secondary stations (PS/2 systems) attached to link
sdhc address C1
sdhc address C2
sdhc address C3
sdhc address C4
! use tcp encapsulation to send frames to SDLC stations C1, C2, C3, or
! C4 and locally terminate sessions with these stations
stun route address C1 tcp 150.136.134.58 local-ack
stun route address C2 tcp 150.136.134.58 local-ack
stun route address C3 tcp 150.136.134.58 local-ack
stun route address C4 tcp 150.136.134.58 local-ack

```

### Configuration for Router B

```
! enter the address of the stun peer
stun peer-name 150.136.134.58
! this router is part of SDLC group 4
stun protocol-group 4 sdlc
stun remote-peer-keepalive
!
interface ethernet 1
! enter the IP address for the Ethernet interface
ip address 150.136.134.58 255.255.255.0
!
! description of PS/2 link
interface serial 4
! disable the IP address on a serial interface
no ip address
! enable STUN encapsulation on this interface
encapsulation stun
! apply previously defined stun group 4 to serial interface 2
stun group 4
! establish this router as a primary station
stun sdlc-role primary
sdlc line-speed 9600
! wait 2000 milliseconds for a reply to a frame before resending it
sdlc t1 2000
! resend a frame up to four times if not acknowledged
sdlc n2 4
! list addresses of secondary stations (PS/2 systems) attached to link
sdlc address C1
sdlc address C2
sdlc address C3
sdlc address C4
! use tcp encapsulation to send frames to SDLC stations C1, C2, C3, or
! C4 and locally terminate sessions with these stations
stun route address C3 tcp 150.136.134.86 local-ack
stun route address C1 tcp 150.136.134.86 local-ack
stun route address C4 tcp 150.136.134.86 local-ack
stun route address C2 tcp 150.136.134.86 local-ack
! set the clockrate on this interface to 9600 bits per second
clockrate 9600
```

### Configuring STUN Local Acknowledgment for SDLC Example

The following example shows a sample configuration for a pair of routers performing SDLC local acknowledgment.

### Configuration for Router A

```
stun peer-name 150.136.64.92
stun protocol-group 1 sdlc
stun remote-peer-keepalive
!
interface Serial 0
no ip address
encapsulation stun
stun group 1
stun sdlc-role secondary
sdlc address C1
stun route address C1 tcp 150.136.64.93 local-ack
clockrate 19200
```

### Configuration for Router B

```

stun peer-name 150.136.64.93
stun protocol-group 1 sdlc
stun remote-peer-keepalive
!
interface Serial 0
no ip address
encapsulation stun
stun group 1
stun sdlc-role primary
sdlc line-speed 19200
sdlc address C1
stun route address C1 tcp 150.136.64.92 local-ack
clockrate 19200

```

### Configuring STUN Local Acknowledgment for Frame Relay Example

The following example describes an interface configuration for Frame Relay STUN with local acknowledgment:

```

stun peer-name 10.1.21.1 cls 4
stun protocol-group 120 sdlc
!
interface Serial1
no ip address
encapsulation frame-relay
frame-relay lmi-type ansi
frame-relay map llc2 22
!
interface Serial4
no ip address
encapsulation stun
clockrate 9600
stun group 120
stun sdlc-role secondary
sdlc address C1
sdlc address C2
stun route address C1 interface Serial1 dlci 22 04 local-ack
stun route address C2 interface Serial1 dlci 22 08 local-ack
!

```

### Configuring LOCADDR Priority Groups—Simple Example

The following example shows how to establish queuing priorities on a STUN interface based on an LU address:

```

! sample stun peer-name global command
stun peer-name 131.108.254.6
! sample protocol-group command for reference
stun protocol-group 1 sdlc
!
interface serial 0
! disable the ip address for interface serial 0
no ip address
! enable the interface for STUN
encapsulation stun
! sample stun group command
stun group 2
! sample stun route command
stun route address 10 tcp 131.108.254.8 local-ack priority

```

```
!  
! assign priority group 1 to the input side of interface serial 0  
locaddr-priority 1  
priority-group 1  
!  
interface Ethernet 0  
! give locaddr-priority-list 1 a high priority for LU 02  
locaddr-priority-list 1 02 high  
! give locaddr-priority-list 1 a low priority for LU 05  
locaddr-priority-list 1 05 low
```

## Configuring LOCADDR Priority Groups for STUN Example

The following configuration example shows how to assign a priority group to an input interface:

### Configuration for Router A

```
stun peer-name 1.0.0.1  
stun protocol-group 1 sdlc  
!  
interface serial 0  
no ip address  
encapsulation stun  
stun group 1  
stun route address C1 tcp 1.0.0.2 local-ack priority  
clockrate 19200  
locaddr-priority 1  
priority-group 1  
!  
interface Ethernet 0  
ip address 1.0.0.1 255.255.255.0  
!  
locaddr-priority-list 1 02 high  
locaddr-priority-list 1 03 high  
locaddr-priority-list 1 04 medium  
locaddr-priority-list 1 05 low  
!  
priority-list 1 protocol ip high tcp 1994  
priority-list 1 protocol ip medium tcp 1990  
priority-list 1 protocol ip normal tcp 1991  
priority-list 1 protocol ip low tcp 1992
```

### Configuration for Router B

```
stun peer-name 1.0.0.2  
stun protocol-group 1 sdlc  
!  
interface serial 0  
no ip address  
encapsulation stun  
stun group 1  
stun route address C1 tcp 1.0.0.1 local-ack priority  
clockrate 19200  
locaddr-priority 1  
priority-group 1  
!  
!
```

```
interface Ethernet 0
 ip address 1.0.0.2 255.255.255.0
 !
 locaddr-priority-list 1 02 high
 locaddr-priority-list 1 03 high
 locaddr-priority-list 1 04 medium
 locaddr-priority-list 1 05 low
 !
 priority-list 1 protocol ip high tcp 1994
 priority-list 1 protocol ip medium tcp 1990
 priority-list 1 protocol ip normal tcp 1991
 priority-list 1 protocol ip low tcp 1992
```

## Block Serial Tunneling (BSTUN)

Cisco's implementation of BSTUN provides the following features:

- Encapsulates Bisync, Adplex, ADT Security Systems, Inc., Diebold, and asynchronous generic traffic for transfer over router links. The tunneling of asynchronous security protocols (ASP) feature enables your Cisco 2500, 3600, 4000, 4500, or 7200 series router to support devices that use the following asynchronous security protocols:
  - adplex
  - adt-poll-select
  - adt-vari-poll
  - diebold
  - async-generic
  - mdi
- Provides a tunnel mechanism for BSTUN over Frame Relay, without using TCP/IP encapsulation.
- Supports legacy Bisync devices and host applications without modification.
- Uses standard synchronous serial interfaces on Cisco 2500 series and the 4T network interface module (NIM) on the Cisco 4000 series and Cisco 4500 series.
- Supports point-to-point, multidrop, and virtual multidrop configurations.

---

**Note** The async-generic item, listed above, is not a protocol name. It is a command keyword used to indicate generic support of other asynchronous security protocols that are not explicitly supported.

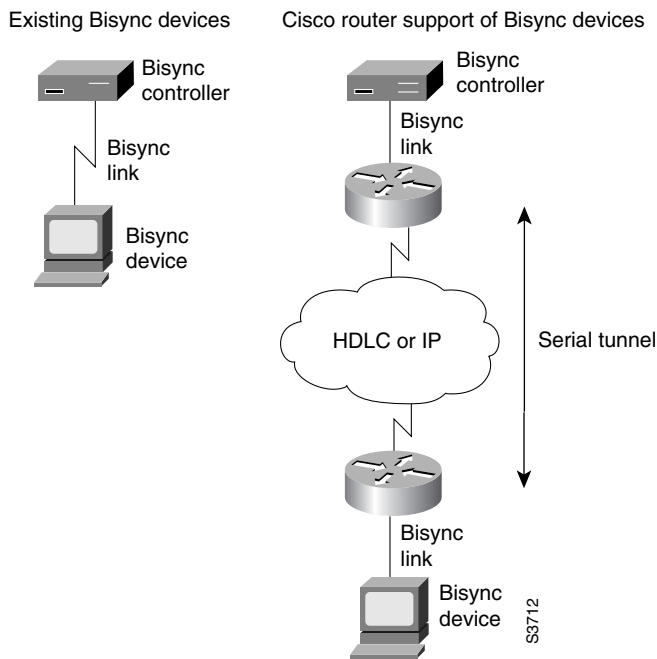
---

## Bisync Network Overview

The Bisync feature enables your Cisco series 2500, 3600, 4000, 4500, 4700, and 7200 series router to support devices that use the Bisync datalink protocol. This protocol enables enterprises to transport Bisync traffic over the same network that supports their SNA and multiprotocol traffic, eliminating the need for separate Bisync facilities.

At the access router, traffic from the attached Bisync device is encapsulated in IP. The Bisync traffic can then be routed across arbitrary media to the host site where another router supporting Bisync will remove the IP encapsulation headers and present the Bisync traffic to the Bisync host or controller over a serial connection. HDLC can be used as an alternative encapsulation method for point-to-point links. Figure 100 shows how you can reconfigure an existing Bisync link between two devices and provide the same logical link without any changes to the existing Bisync devices.

**Figure 100 Routers Consolidate Bisync Traffic by Encapsulation in IP or HDLC**



The routers transport all Bisync blocks between the two devices in pass-through mode using BSTUN as encapsulation. BSTUN uses the same encapsulation architecture as STUN, but is implemented on an independent tunnel.

## Point-to-Point and Multidrop Support

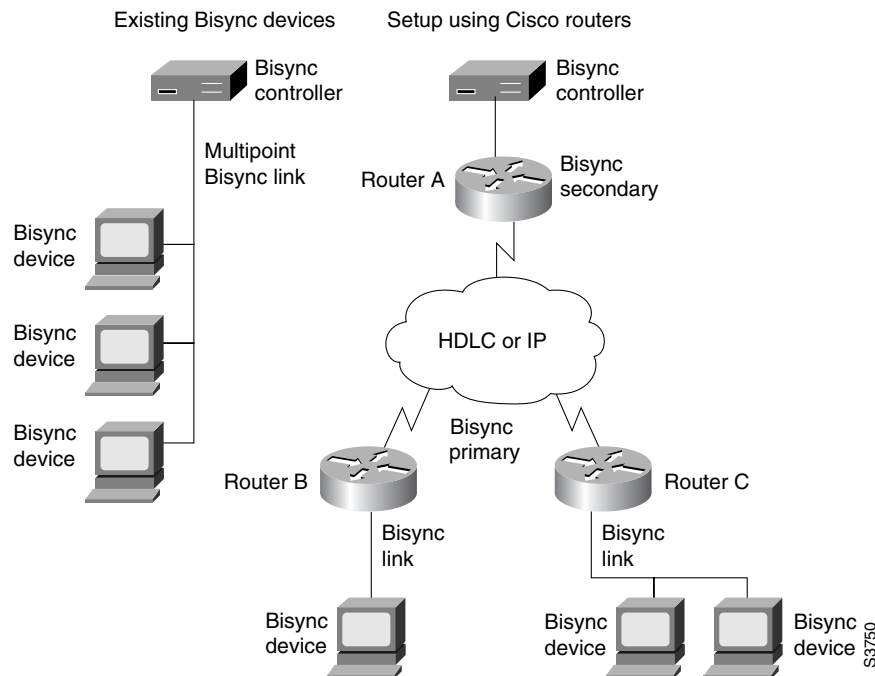
The Bisync feature supports point-to-point, multidrop, and virtual multidrop Bisync configurations.

### Point-to-Point Operation

In point-to-point operation, the Bisync blocks between the two point-to-point devices are received and forwarded transparently by the Cisco IOS software. The contention to acquire the line for transmission is handled by the devices themselves.

Cisco's Bisync multipoint operation is provided as a logical multipoint configuration. Figure 101 shows how a multipoint Bisync link is reconfigured using Cisco routers. Router A is configured as Bisync secondary. It monitors the address field of the polling or selection block and uses this address information to put into the BSTUN frame for BSTUN to deliver to the correct destination router. To simulate the Bisync multidrop, an EOT block is sent by the Bisync primary router before a poll or selection block. This ensures that Bisync tributary stations are in control mode before being polled or selected.

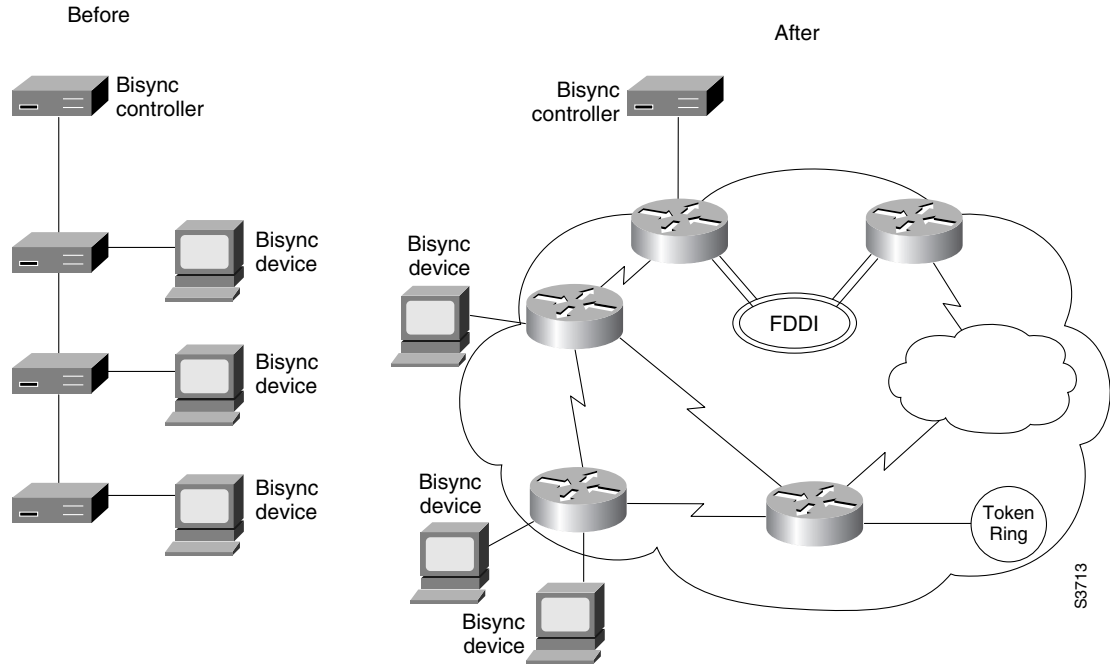
**Figure 101 Multipoint Bisync Link Reconfigured Using Routers**



### Multidrop Configuration

Multidrop configurations are common in Bisync networks where up to 8 or 10 Bisync devices are frequently connected to a Bisync controller port over a single low-speed link. Our allows Bisync devices from different physical locations in the network to appear as a single multidrop line to the Bisync host or controller. Figure 102 illustrates a multidrop Bisync configuration before and after implementing routers.

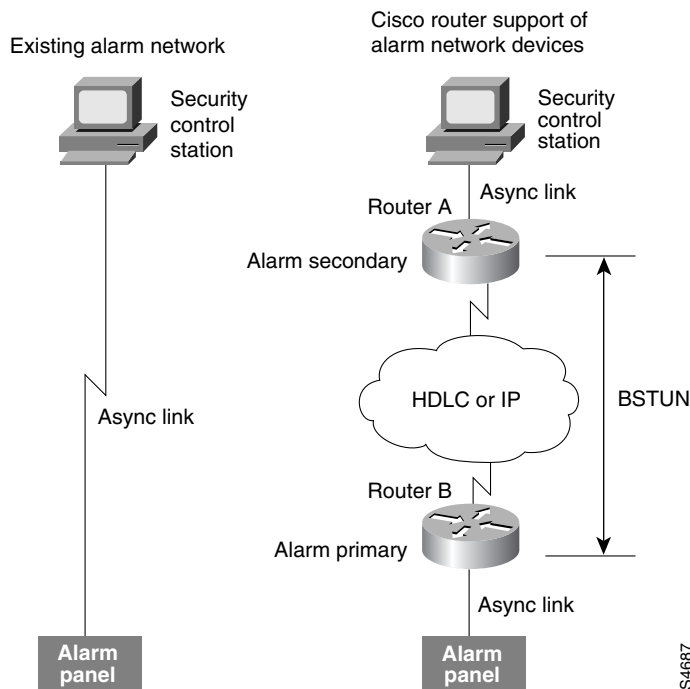
**Figure 102 Integrating Bisync Devices over a Multiprotocol Network**



## Asynchronous Network Overview

These protocols enable enterprises to transport polled asynchronous traffic over the same network that supports their SNA and multiprotocol traffic, eliminating the need for separate facilities. Figure 103 shows how you can reconfigure an existing asynchronous link between two security devices and provide the same logical link without any changes to the existing devices.

**Figure 103 Routers Consolidate Polled Asynchronous Traffic Using Encapsulation in IP or HDLC**



Router A is configured as the secondary end of the BSTUN asynchronous link and is attached to the security control station; Router B is configured as the primary end of the BSTUN asynchronous link and has one or more alarm panels attached to it.

At the downstream router, traffic from the attached alarm panels is encapsulated in IP. The asynchronous (alarm) traffic can be routed across arbitrary media to the host site where the upstream router supporting these protocols removes the IP encapsulation headers and presents the original traffic to the security control station over a serial connection. High-Level Data Link Control (HDLC) can be used as an alternative encapsulation method for point-to-point links.

The routers transport all asynchronous (alarm) blocks between the two devices in passthrough mode using BSTUN for encapsulation. BSTUN uses the same encapsulation architecture as serial tunnel (STUN), but is implemented on an independent tunnel. As each asynchronous frame is received from the line, a BSTUN header is added to create a BSTUN frame, and then BSTUN is used to deliver the frame to the correct destination router.

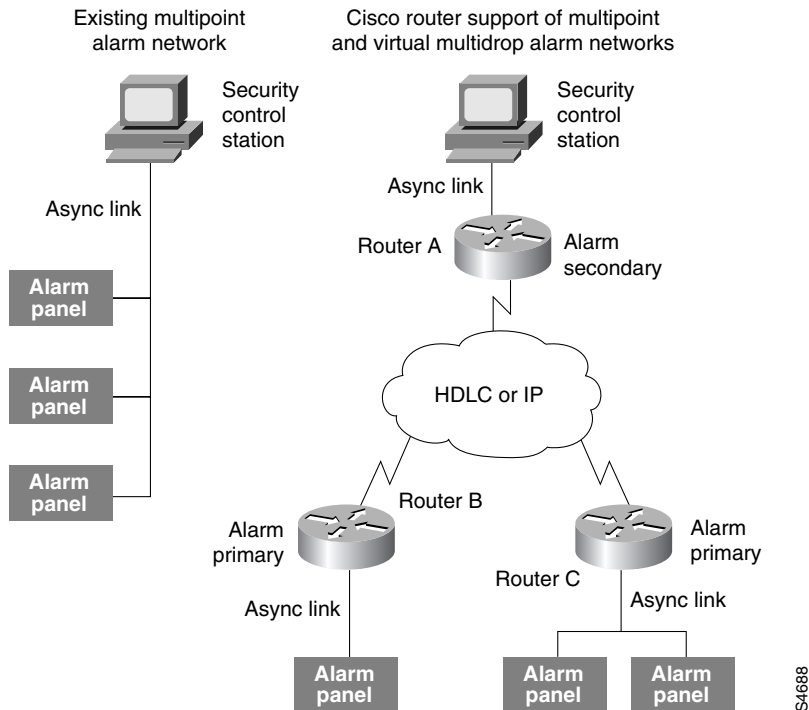
The Cisco routers do not perform any local acknowledgment or cyclic redundancy check (CRC) calculations on the asynchronous alarm blocks. The two end devices are responsible for error recovery in the asynchronous alarm protocol.

## Virtual Multidrop Support for Multipoint Security Network Configurations

Multipoint configurations are common in security networks, where a number of alarm panels are frequently connected to a security control station over a single low-speed link. Our virtual multidrop support allows alarm panels from different physical locations in the network to appear as a single multidrop line to the security control station. Both Adplex and ADT are virtual multidropped protocols.

Multidrop operation is provided as a logical multipoint configuration. Figure 104 shows how a multipoint security network is reconfigured using Cisco routers. Router A is configured as an alarm secondary node, routers B and C are configured as alarm primary nodes. Router A monitors the address field of the polling or selection block and puts this address information in the BSTUN frame so BSTUN can deliver the frame to the correct downstream node.

**Figure 104 Multipoint Asynchronous Security Protocol Link Reconfigured Using Routers**



## Frame Sequencing

Both Bisync and asynchronous alarm protocols are half-duplex protocols; transmission can occur in either direction, but only in one direction at a time. Each block of transmission is acknowledged explicitly by the remote end. To avoid the problem associated with simultaneous transmission, there is an implicit role of primary and secondary station.

### Frame Sequencing in Bisync Networks

In a multidrop setup in Bisync networks, the Bisync control station is primary and the tributary stations are secondary. In a point-to-point configuration, the primary role is assumed by the Bisync device that has successfully acquired the line for transmission through the ENQ bidding sequence. The primary role stays with this station until it sends EOT.

To protect against occasional network latency, which causes the primary station to time out and resend the block before the Bisync block sent by the secondary is received, the control byte of the encapsulating frame is used as a sequence number. This sequence number is controlled and monitored by the primary Bisync router. This allows the primary Bisync router to detect and discard “late” Bisync blocks sent by the secondary router and ensure integrity of the Bisync link.

---

**Note** Frame sequencing is implemented in passthrough mode only.

---

### Frame Sequencing in Asynchronous Networks

Network delays in asynchronous networks make it possible for a frame to arrive “late,” meaning that the poll-cycling mechanism at the security control station has already moved on to poll the next alarm panel in sequence when it receives the poll response from the previous alarm panel.

To protect against this situation, routers configured for adplex or for adt-poll-select protocols use a sequence number built into the encapsulating frame to detect and discard late frames. The “upstream” router (connected to the security control station) inserts a frame sequence number into the protocol header, which is shipped through the BSTUN tunnel and bounced back by the “downstream” router (connected to the alarm panel). The upstream router maintains a frame-sequence count for the line, and checks the incoming frame-sequence number from the downstream router. If the two frame-sequence numbers do not agree, the frame is considered late (out of sequence) and is discarded.

Because the adt-vari-poll option allows the transmission of unsolicited messages from the alarm panel, frame sequencing is not supported for this protocol.

---

**Note** Polled asynchronous (alarm) protocols are implemented only in passthrough mode. There is no support for local acknowledgment.

---

## BSTUN Configuration Task List

The Bisync feature is configured similar to SDLC STUN, but is configured as a protocol within a BSTUN feature. To configure and monitor Bisync with BSTUN, complete the tasks in the following sections:

- Enable BSTUN
- Define the Protocol Group
- Enable STUN Keepalive
- Enable STUN Remote Keepalive
- Enable Frame Relay Encapsulation
- Define Mapping Between BSTUN and DLCI
- Configure BSTUN on the Serial Interface
- Place a Serial Interface in a BSTUN Group
- Specify How Frames are Forwarded
- Set Up BSTUN Traffic Priorities
- Configure Protocol Group Options on a Serial Interface
- Configure Direct Serial Encapsulation for Passthrough Peers
- Configure Local Acknowledgment Peers
- Monitor the Status of BSTUN

The “BSTUN Configuration Examples” section follows these tasks.

### Enable BSTUN

To enable BSTUN in IP networks, perform the following task in global configuration mode:

Task	Command
Enable BSTUN.	<b>bstun peer-name</b> <i>ip-address</i>
Enable BSTUN for Frame Relay transport.	<b>bstun lissap</b> <i>sap-value</i>

The IP address in the **bstun peer-name** command defines the address by which this BSTUN peer is known to other BSTUN peers that are using the TCP transport. If this command is unconfigured or the **no** form of this command is specified, all BSTUN routing commands with IP addresses are deleted. BSTUN routing commands without IP addresses are not affected by this command.

The **bstun lissap** command specifies a SAP on which to detect incoming calls.

## Define the Protocol Group

Define a BSTUN group and specify the protocol it uses. To define the protocol group, perform the following task in global configuration mode:

Task	Command
Define the protocol group.	<b>bstun protocol-group</b> <i>group-number</i> { <b>bsc</b>   <b>bsc-local-ack</b>   <b>adplex</b>   <b>adt-poll</b>   <b>adt-poll-select</b>   <b>adt-vari-poll</b>   <b>diebold</b>   <b>async-generic</b>   <b>mdi</b> }

The **bsc-local-ack** protocol option only works for 3270 Bisync uses.

The block serial protocols include bsc, bsc-local-ack, adplex, adt-poll-select, adt-vari-poll, diebold, async-generic, and mdi.

Traditionally, the adt-poll-select protocol is used over land-based links, while the adt-vari-poll protocol is used over satellite (VSAT) links. The adt-vari-poll protocol typically uses a much slower polling rate when alarm consoles poll alarm panels because adt-vari-poll allows alarm panels to send unsolicited messages to the alarm console. In an adt-vari-poll configuration, alarm panels do not have to wait for the console to poll them before responding with an alarm, they automatically send the alarm.

Interfaces configured to run the adplex protocol have their baud rate set to 4800 bps, use even parity, 8 data bits, 1 start bit, and 1 stop bit.

Interfaces configured to run the adt-poll-select and adt-vari-poll protocols have their baud rate set to 600 bps, use even parity, 8 data bits, 1 start bit, and 1.5 stop bits. If different line configurations are required, use the **rxspeed**, **txspeed**, **databits**, **stopbits**, and **parity** line configuration commands to change the line attributes.

Interfaces configured to run the diebold protocol have their baud rate set to 300 bps, use even parity, 8 data bits, 1 start bit, and 2 stop bits. If different line configurations are required, use the **rxspeed**, **txspeed**, **databits**, and **parity** line configuration commands to change the line attributes.

Interfaces configured to run the async-generic protocol have their baud rate set to 9600 bps, use no parity, 8 data bits, 1 start bit, and 1 stop bit. If different line configurations are required, use the **rxspeed**, **txspeed**, **databits**, **stopbits**, and **parity** line configuration commands to change the line attributes.

Interfaces configured to run the mdi protocol have their baud rate set to 600 bps, use even parity, 8 data bits, 1 start bit, and 1.5 stop bits. If different line configurations are required, use the **rxspeed**, **txspeed**, **databits**, **stopbits**, and **parity** line configuration commands to change the line attributes. The mdi protocol allows alarm panels to be sent to the MDI alarm console.

## Enable BSTUN Keepalive

To define the number of times to attempt a peer connection before declaring the peer connection down, perform the following task in global configuration mode:

Task	Command
Specify the number of times to attempt a peer connection.	<b>bstun keepalive-count</b>

## Enable BSTUN Remote Keepalive

To enable detection of the loss of a peer, perform the following task in global configuration mode:

Task	Command
Enable detection of the loss of a peer.	<b>bstun remote-peer-keepalive</b> <i>seconds</i>

## Enable Frame Relay Encapsulation

To enable Frame Relay encapsulation, perform the following tasks, beginning in global configuration mode:

Task	Command
Specify a serial port.	<b>interface</b> <i>serial number</i>
Enable Frame Relay encapsulation on the serial port.	<b>encapsulation frame-relay</b>

## Define Mapping Between BSTUN and DLCI

To configure the mapping between BSTUN and the DLCI, perform one of the following tasks in interface configuration mode:

Task	Command
Define the mapping between BSTUN and the DLCI when using BSC passthru.	<b>frame-relay map bstun</b> <i>dlci</i>
Define the mapping between BSTUN and the DLCI when using BSC local acknowledgment.	<b>frame-relay map llc2</b> <i>dlci</i>

---

**Note** Direct encapsulation over Frame Relay is supported only for an encapsulation type of cisco, configured using the **encapsulation frame-relay cisco** command.

---

## Configure BSTUN on the Serial Interface

Configure BSTUN on the serial interface before issuing any further BSTUN or protocol configuration commands for the interface. To configure the BSTUN function on a specified interface, perform the following command in interface configuration mode:

Task	Command
Specify a serial port.	<b>interface</b> <i>serial number</i>
Configure BSTUN on an interface.	<b>encapsulation bstun</b> <sup>1</sup>

1. This command must be configured on an interface before any other BSTUN commands are configured for this interface.

## Place a Serial Interface in a BSTUN Group

Each BSTUN-enabled interface on a router must be placed in a previously defined BSTUN group. Packets will only travel between BSTUN-enabled interfaces that are in the same group. To assign a serial interface to a BSTUN group, perform the following task in interface configuration mode:

Task	Command
Assign a serial interface to a BSTUN group.	<b>bstun group</b> <i>group-number</i>

## Specify How Frames are Forwarded

To specify how frames are forwarded when received on a BSTUN interface, perform one of the following tasks in interface configuration mode:

Task	Command
Propagate the serial frame that contains a specific address. HDLC encapsulation is used to propagate the serial frames.	<b>bstun route address</b> <i>address-number</i> <b>interface serial</b> <i>number</i>
Propagate all BSTUN traffic received on the input interface, regardless of the address contained in the serial frame. HDLC encapsulation is used to propagate the serial frames.	<b>bstun route all interface serial</b> <i>number</i>
Propagate the serial frame that contains a specific address. TCP encapsulation is used to propagate frames that match the entry.	<b>bstun route address</b> <i>address-number</i> <b>tcp</b> <i>ip-address</i>
Propagate all BSTUN traffic received on the input interface, regardless of the address contained in the serial frame. TCP encapsulation is used to propagate frames that match the entry.	<b>bstun route all tcp</b> <i>ip-address</i> <sup>1</sup>
Propagate the serial frame that contains a specific address. Specify the control unit address for the Bisync end station. Frame Relay encapsulation is used to propagate the serial frames.	<b>bstun route address</b> <i>cu-address</i> <b>interface serial</b> <i>serial-int</i> <b>dcli</b> <i>dcli</i>
Propagate all frames regardless of the control unit address for the Bisync end station. Frame Relay encapsulation is used to propagate the serial frames in bisync passthrough mode.	<b>bstun route all interface serial</b> <i>serial-int</i> <b>dcli</b> <i>dcli</i>
Propagate the serial frame that contains a specific address. Specify the control unit address for the bisync end station. Frame Relay encapsulation is used to propagate the serial frames for bisync local acknowledgment mode	<b>bstun route address</b> <i>cu-address</i> <b>interface serial</b> <i>serial-int</i> <b>dcli</b> <i>dcli</i> <b>rsap</b> <b>priority</b> <i>priority</i>
Propagate all BSTUN traffic received on the input interface, regardless of the address contained in the serial frame. Frame Relay encapsulation is used to propagate the serial frames.	<b>bstun route all interface serial</b> <i>serial-int</i> <b>dcli</b> <i>dcli</i> <b>rsap</b> <b>priority</b> <i>priority</i>

1. This command functions in either passthru or local acknowledgment mode.

**Note** Every BSTUN route statement must have a corresponding route statement on the BSTUN peer. For example, a **bstun route address** *address1* **tcp** *peer2ip* statement on PEER1 must have a corresponding **bstun route address** *address1* **tcp** *peer1ip* statement on PEER2. Similarly, a **bstun route address** statement cannot map to a **bstun route all** statement, and vice versa.

For Bisync local acknowledgment, we recommend that you use the **bstun route all tcp** command. This command reduces the amount of duplicate configuration detail that would otherwise be needed to specify devices at each end of the tunnel.

## Set Up BSTUN Traffic Priorities

You can assign BSTUN traffic priorities based on either the BSTUN header or the TCP port. To prioritize traffic, perform one of the following tasks in global configuration mode:

Task	Command
Establish BSTUN queuing priorities based on the BSTUN header.	<b>priority-list</b> <i>list-number</i> <b>protocol bstun</b> <i>queue</i> [ <b>gt</b> <i>packet-size</i> ] [ <b>lt</b> <i>packet-size</i> ] <b>address</b> <i>bstun-group bsc-addr</i>
Assign a queuing priority to TCP port.	<b>priority-list</b> <i>list-number</i> <b>protocol ip</b> <i>queue tcp</i> <i>tcp-port-number</i>

You can customize BSTUN queuing priorities based on either the BSTUN header or TCP port. To customize priorities, perform one of the following tasks in global configuration mode:

Task	Command
Customize BSTUN queuing priorities based on the BSTUN header.	<b>queue-list</b> <i>list-number</i> <b>protocol bstun</b> <i>queue</i> [ <b>gt</b> <i>packet-size</i> ] [ <b>lt</b> <i>packet-size</i> ] <b>address</b> <i>bstun-group bsc-addr</i>
Customize BSTUN queuing priorities based on the TCP port.	<b>queue-list</b> <i>list-number</i> <b>protocol ip</b> <i>queue tcp</i> <i>tcp-port-number</i>

---

**Note** Because the asynchronous security protocols share the same tunnels with Bisync when configured on the same routers, any traffic priorities configured for the tunnel apply to both Bisync and the various asynchronous security protocols.

---

## Configure Protocol Group Options on a Serial Interface

Depending on the selected block serial protocol group, you must configure one or more options for that protocol group. The options for each of these protocol groups are explained in the following sections:

- Configure Bisync Options on a Serial Interface
- Configure Asynchronous Security Protocol Options on a Serial Interface

## Configure Bisync Options on a Serial Interface

To configure Bisync options on a serial interface, perform one of the following tasks in interface configuration mode:

Task	Command
Specify the character set used by the Bisync support feature.	<b>bsc char-set</b> { <i>ascii</i>   <i>ebcdic</i> }
Specify an address on a contention interface.	<b>bsc contention</b> <i>address</i>

Task	Command
Specify that the router at the central site will behave as a central router with dynamic allocation of serial interfaces. The timeout value is the length of time an interface can be idle before it is returned to the idle interface pool.	<b>bsc dial-contention</b> <i>time-out</i>
Specify a nonstandard Bisync address.	<b>bsc extended-address</b> <i>poll-address</i> <i>select-address</i>
Specify that the interface can run Bisync in full-duplex mode.	<b>full-duplex</b>
Specify the amount of time between the start of one polling cycle and the next.	<b>bsc pause</b> <i>time</i>
Specify the timeout for a poll or a select sequence.	<b>bsc poll-timeout</b> <i>time</i>
Specify the timeout for a nonreception of poll or a select sequence from the host. If the frame is not received within this time the remote connection will be deactivated.	<b>bsc host-timeout</b> <i>time</i>
Specify that the router is acting as the primary end of the Bisync link.	<b>bsc primary</b>
Specify the number of retries before a device is considered to have failed.	<b>bsc retries</b> <i>retry-count</i>
Specify that the router is acting as the secondary end of the Bisync link.	<b>bsc secondary</b>
Specify specific polls, rather than general polls, used on the host-to-router connection.	<b>bsc spec-poll</b>
Specify the number of cycles of the active poll list that are performed between polls to control units in the inactive poll list.	<b>bsc servlim</b> <i>servlim-count</i>

## Configure Asynchronous Security Protocol Options on a Serial Interface

To configure asynchronous security protocol options on a serial interface, perform one or more of the following tasks in interface configuration mode:

Task	Command
Specify that the router is acting as the primary end of the polled asynchronous link.	<b>asp role primary</b>
Specify that the router is acting as the secondary end of the polled asynchronous link.	<b>asp role secondary</b>
For asynchronous-generic configurations, specify the location of the address byte within the polled asynchronous frame being received.	<b>asp addr-offset</b> <i>address-offset</i>
For asynchronous-generic configurations, specify the timeout period between frames to delineate the end of one frame being received from the start of the next frame.	<b>asp rx-ift</b> <i>interframe-timeout</i>

## Configure Direct Serial Encapsulation for Passthrough Peers

To configure direct serial encapsulation for passthrough peers perform the following task in interface configuration mode:

Task	Command
Configure the Frame Relay interface for passthru.	<b>frame relay map bstun</b>

## Configure Local Acknowledgment Peers

To configure local acknowledgment peers perform the following task in interface configuration mode:

Task	Command
Configure the Frame Relay interface for local acknowledgment.	<b>frame-relay map llc2 dlci</b>

## Monitor the Status of BSTUN

To list statistics for BSTUN interfaces, protocol groups, number of packets sent and received, local acknowledgment states, and other activity information, perform the following task in EXEC mode:

Task	Command
List the status display fields for BSTUN interfaces.	<b>show bstun</b> [ <b>group</b> <i>bstun-group-number</i> ] [ <b>address</b> <i>address-list</i> ]
Display status of the interfaces on which Bisync is configured.	<b>show bsc</b> [ <b>group</b> <i>bstun-group-number</i> ] [ <b>address</b> <i>address-list</i> ]

## BSTUN Configuration Examples

The following sections provide BSTUN configuration examples:

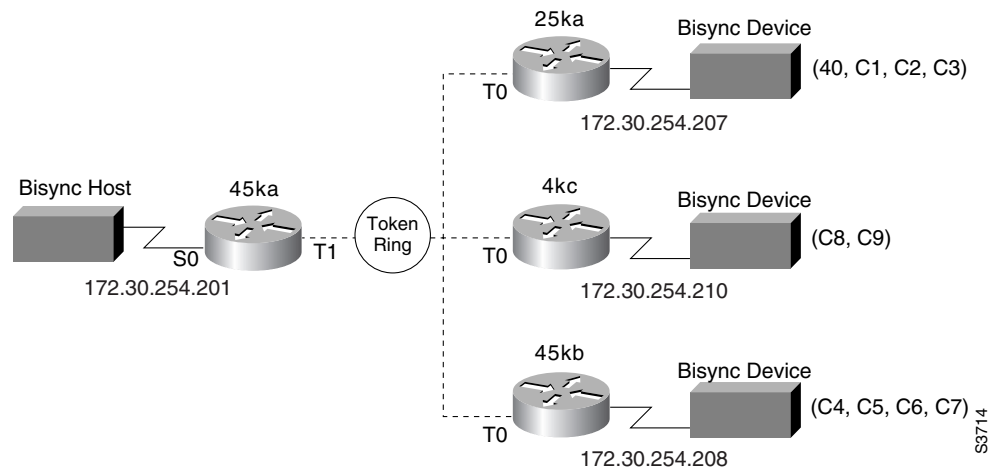
- Simple Bisync Configuration Example
- Bisync Addressing on Contention Interfaces
- Nonstandard Bisync Addressing
- Priority Queuing: Prioritization Based on BSTUN Header Example
- Priority Queuing: Prioritization Based on BSTUN Header and Packet Sizes Example
- Priority Queuing: Prioritization Based on BSTUN Header and Bisync Address Example
- Priority Queuing: Prioritization Based on BSTUN TCP Ports Example
- Priority Queuing: Prioritization Based on BSTUN TCP Ports and Bisync Address Example
- Custom Queuing: Prioritization Based on BSTUN Header Example
- Custom Queuing: Prioritization Based on BSTUN Header and Packet Size Example
- Custom Queuing: Prioritization Based on BSTUN Header and Bisync Address Example
- Custom Queuing: Prioritization Based on BSTUN TCP Ports Example
- Custom Queuing: Prioritization Based on BSTUN TCP Ports and Bisync Address Example

- Asynchronous Configuration Example
- BSTUN-over-Frame Relay Configuration Example for Local Acknowledgment
- BSTUN-over-Frame Relay Configuration Example for Passthrough

## Simple Bisync Configuration Example

Figure 105 shows a simple Bisync configuration example.

**Figure 105 Simple Bisync Configuration**



The configuration files for the routers shown in Figure 105 follow.

### Configuration for Router 45ka

```

version 10.2
!
hostname 45ka
!
no ip domain-lookup
!
bstun peer-name 150.10.254.201
bstun protocol-group 1 bsc
!
interface ethernet 0
 ip address 198.92.0.201 255.255.255.0
 media-type 10BaseT
!
interface ethernet 1
 no ip address
 shutdown
 media-type 10BaseT
!
interface serial 0
 no ip address
 encapsulation bstun
 clockrate 19200
 bstun group 1
 bsc char-set ebcdic
 bsc secondary
 bstun route address C9 tcp 150.10.254.210

```

## BSTUN Configuration Examples

---

```
bstun route address C8 tcp 150.10.254.210
bstun route address C7 tcp 150.10.254.208
bstun route address C6 tcp 150.10.254.208
bstun route address C5 tcp 150.10.254.208
bstun route address C4 tcp 150.10.254.208
bstun route address C3 tcp 150.10.254.207
bstun route address C2 tcp 150.10.254.207
bstun route address C1 tcp 150.10.254.207
bstun route address 40 tcp 150.10.254.207
!
interface serial 1
no ip address
shutdown
!
interface serial 2
no ip address
shutdown
!
interface serial 3
no ip address
shutdown
!
interface tokenring 0
no ip address
shutdown
!
interface tokenring 1
ip address 150.10.254.201 255.255.255.0
ring-speed 16
!
line con 0
line aux 0
line vty 0 4
login
!
end
```

### Configuration for Router 25ka

```
version 10.2
!
hostname 25ka
!
no ip domain-lookup
!
bstun peer-name 150.10.254.207
bstun protocol-group 1 bsc
!
interface serial 0
no ip address
shutdown
!
interface serial 1
no ip address
encapsulation bstun
clockrate 19200
bstun group 1
bsc char-set ebcddic
bsc primary
bstun route address C3 tcp 150.10.254.201
bstun route address C2 tcp 150.10.254.201
bstun route address C1 tcp 150.10.254.201
bstun route address 40 tcp 150.10.254.201
!
```

```
interface tokenring 0
 ip address 150.10.254.207 255.255.255.0
 ring-speed 16
!
interface bri 0
 no ip address
 shutdown
!
line con 0
line aux 0
line vty 0 4
login
!
end
```

### Configuration for Router 4kc

```
version 10.2
!
hostname 4kc
!
no ip domain-lookup
!
bstun peer-name 150.10.254.210
bstun protocol-group 1 bsc
!
interface ethernet 0
 ip address 198.92.0.210 255.255.255.0
 media-type 10BaseT
!
interface serial 0
 no ip address
 encapsulation bstun
 clockrate 19200
 bstun group 1
 bsc char-set ebcidic
 bsc primary
 bstun route address C9 tcp 150.10.254.201
 bstun route address C8 tcp 150.10.254.201
!
interface serial 1
 no ip address
 shutdown
!
interface serial 2
 no ip address
 shutdown
!
interface serial 3
 no ip address
 shutdown
!
interface tokenring 0
 ip address 150.10.254.210 255.255.255.0
 ring-speed 16
!
interface tokenring 1
 no ip address
 shutdown
```

```
!  
line con 0  
line aux 0  
line vty 0 4  
login  
!  
end
```

### Configuration for Router 25kb

```
version 10.2  
!  
hostname 25kb  
!  
no ip domain-lookup  
!  
bstun peer-name 150.10.254.208  
bstun protocol-group 1 bsc  
!  
interface serial 0  
no ip address  
encapsulation bstun  
no keepalive  
clockrate 19200  
bstun group 1  
bsc char-set ebcddic  
bsc primary  
bstun route address C7 tcp 150.10.254.201  
bstun route address C6 tcp 150.10.254.201  
bstun route address C5 tcp 150.10.254.201  
bstun route address C4 tcp 150.10.254.201  
!  
interface serial 1  
no ip address  
shutdown  
!  
interface tokenring 0  
ip address 150.10.254.208 255.255.255.0  
ring-speed 16  
!  
!  
line con 0  
line aux 0  
line vty 0 4  
login  
!  
end
```

## Bisync Addressing on Contention Interfaces

The following two examples show user-configurable addressing on contention interfaces:

### Remote Devices

```
bstun peer-name 1.1.1.20  
bstun protocol-group 1 bsc  
interface serial 0  
bstun group 1  
bsc contention 20  
bstun route address 20 tcp 1.1.1.1
```

## Host Device

```
bstun peer-name 1.1.1.1
bstun protocol-group 1 bsc
interface serial 0
  bstun group 1
  bsc dial-contention 100
  bstun route address 20 tcp 1.1.1.20
  bstun route address 21 tcp 1.1.1.21
```

## Nonstandard Bisync Addressing

This example specifies an extended address on serial interface 0:

```
bstun peer-name 1.1.1.1
bstun protocol-group 1 bsc
!
interface serial 0
  bstun group 1
  bsc extended-address 23 83
  bsc extended-address 87 42
  bsc primary
  bstun route address 23 tcp 1.1.1.20
```

## Priority Queuing: Prioritization Based on BSTUN Header Example

In this example, the output interface examines header info and places packets with the BSTUN header on specified output queue.

```
priority-list 1 protocol bstun normal
interface serial 0
  priority-group 1
interface serial 1
  encapsulation bstun
  bstun group 1
  bstun route all interface serial 0
  ...or...
bstun route address <bsc-addr> interface serial 0
```

## Priority Queuing: Prioritization Based on BSTUN Header and Packet Sizes Example

In this example, the output interface examines header information and packet size and places packets with the BSTUN header that match criteria (gt or lt specified packet size) on specified output queue.

```
priority-list 1 protocol bstun low gt 1500
priority-list 1 protocol bstun hi lt 500
interface serial 0
  priority-group 1
interface serial 1
  encapsulation bstun
  bstun group 1
  bstun route all interface serial 0
  ...or...
bstun route address <bsc-addr> interface serial 0
```

## Priority Queuing: Prioritization Based on BSTUN Header and Bisync Address Example

In this example, the output interface examines header information and Bisync address and places packets with the BSTUN header that match Bisync address on specified output queue.

```
priority-list 1 protocol bstun normal
address <bstun-group> <bsc-addr>
interface serial 0
  priority-group 1
interface serial 1
  encapsulation bstun
  bstun group 1
bstun route address <bsc-addr> interface serial 0
```

## Priority Queuing: Prioritization Based on BSTUN TCP Ports Example

In this example, the output interface examines TCP port number and places packets with the BSTUN port number (1976) on specified output queue.

```
priority-list 1 protocol ip high tcp 1976
interface serial 0
  priority-group 1
interface serial 1
  encapsulation bstun
  bstun group 1
bstun route all tcp <bstun-peer-ip-addr>
```

## Priority Queuing: Prioritization Based on BSTUN TCP Ports and Bisync Address Example

In this example, four TCP/IP sessions (high, medium, normal, and low) are established with BSTUN peers using BSTUN port numbers. The input interface examines the Bisync address and uses the specified output queue definition to determine which BSTUN TCP session to use for sending the packet to the BSTUN peer.

The output interface examines the TCP port number and places packets with the BSTUN port numbers on the specified output queue.

```
priority-list 1 protocol ip high tcp 1976
priority-list 1 protocol ip medium tcp 1977
priority-list 1 protocol ip normal tcp 1978
priority-list 1 protocol ip low tcp 1979
!
priority-list 1 protocol bstun <outputQ>
address <bstun-group> <bsc-addr>
!
interface serial 0
  priority-group 1
!
interface serial 1
  encapsulation bstun
  bstun group 1
bstun route address <bsc-addr> tcp <bstun-peer-ip-addr> priority
priority-group 1
```

## Custom Queuing: Prioritization Based on BSTUN Header Example

In this example, the output interface examines header info and places packets with the BSTUN header on specified output queue.

```
queue-list 1 protocol bstun normal
!
interface serial 0
  custom-queue-list 1
!
interface serial 1
  encapsulation bstun
  bstun group 1
  bstun route all interface serial 0
```

## Custom Queuing: Prioritization Based on BSTUN Header and Packet Size Example

In this example, the output interface examines header information and packet size and places packets with the BSTUN header that match criteria (gt or lt specified packet size) on specified output queue.

```
queue-list 1 protocol bstun low gt 1500
queue-list 1 protocol bstun high lt 500
!
interface serial 0
  custom-queue-list 1
!
interface serial 1
  encapsulation bstun
  bstun group 1
  bstun route all interface serial 0
```

## Custom Queuing: Prioritization Based on BSTUN Header and Bisync Address Example

In this example the output interface examines header info and Bisync address and places packets with the BSTUN header that match Bisync address on specified output queue.

```
queue-list 1 protocol bstun normal
address <bstun-group> <bsc-addr>
!
interface serial 0
  custom-queue-list 1
!
interface serial 1
  encapsulation bstun
  bstun group 1
  bstun route address <bsc-addr> interface serial 0
```

## Custom Queuing: Prioritization Based on BSTUN TCP Ports Example

In this example, the output interface examines the TCP port number and places packets with the BSTUN port number (1976) on specified output queue.

```
queue-list 1 protocol ip high tcp 1976
!
interface serial 0
  custom-queue-list 1
!
interface serial 1
  encapsulation bstun
  bstun group 1
  bstun route all tcp <bstun-peer-ip-addr>
```

## Custom Queuing: Prioritization Based on BSTUN TCP Ports and Bisync Address Example

In this example, four TCP/IP sessions (high, medium, normal, and low) are established with BSTUN peers using BSTUN port numbers. The input interface examines the Bisync address and uses the specified output queue definition to determine which BSTUN TCP session to use.

The output interface examines the TCP port number and places packets with the BSTUN port numbers on the specified output queue.

For Bisync addressing, output queues map as shown in Table 5:

**Table 5 Bisync Addressing Output Queues**

Output Queue	Session Mapped	BSTUN Port
1	Medium	1977
2	Normal	1978
3	Low	1979
4–10	High	1976

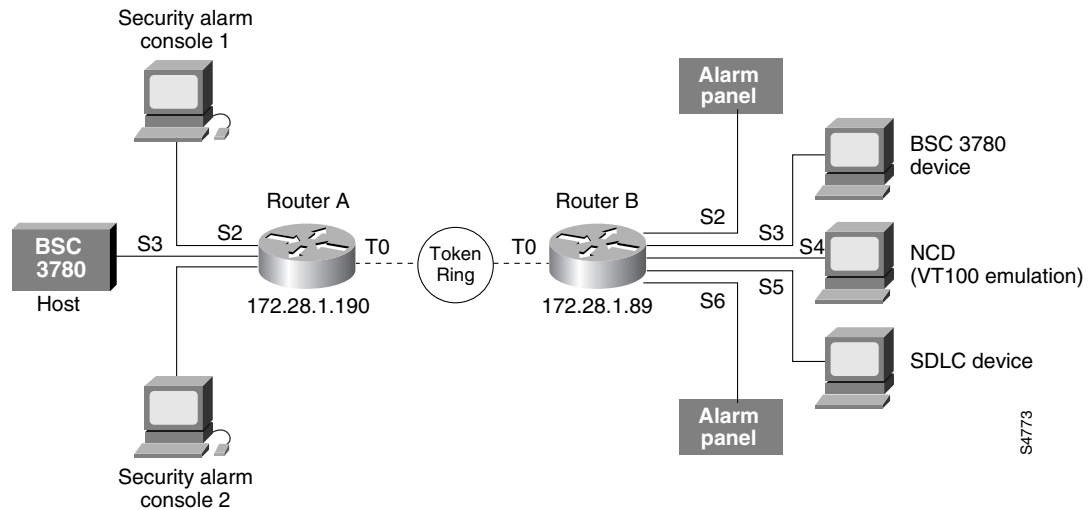
```

queue-list 1 protocol ip high tcp 1976
queue-list 1 protocol ip medium tcp 1977
queue-list 1 protocol ip normal tcp 1978
queue-list 1 protocol ip low tcp 1979
!
priority-list 1 protocol bstun normal
address <bstun-group> <bsc-addr>
!
interface serial 0
 custom-queue-list 1
!
interface serial 1
 encapsulation bstun
 bstun group 1
 bstun route address <bsc-addr> tcp <bstun-peer-ip-addr> priority
 custom-queue-list 1
    
```

## Asynchronous Configuration Example

In this example Router A and Router B are configured for both Adplex and Bisync across the same block serial tunnel, as shown in Figure 106.

**Figure 106 Combined Adplex and Bisync Configuration Example**



### Configuration for Router A

```

version 11.0
!
hostname router-a
!
bstun peer-name 172.28.1.190
bstun protocol-group 1 bsc
bstun protocol-group 2 adplex
bstun protocol-group 3 adplex
!
interface serial 0
 no ip address
!
interface serial 1
 no ip address
!
interface serial 2
 physical-layer async
 description Connection to 1st Security Alarm Console.
 no ip address
 encapsulation bstun
 no keepalive
 bstun group 2
 bstun route address 2 tcp 172.28.1.189
 bstun route address 3 tcp 172.28.1.189
 adplex secondary
!
interface serial 3
 description Connection to BSC 3780 host.
 no ip address
 encapsulation bstun
 no keepalive
 clockrate 9600

```

## BSTUN Configuration Examples

---

```
bstun group 1
bstun route all tcp 172.28.1.189
bsc char-set ebcddic
bsc contention
!
interface serial 4
physical-layer async
description Connection to 2nd Security Alarm Console.
no ip address
encapsulation bstun
no keepalive
bstun group 3
bstun route address 2 tcp 172.28.1.189
bstun route address 3 tcp 172.28.1.189
adplex secondary
!
interface serial 5
no ip address
!
interface serial 6
no ip address
!
interface serial 7
no ip address
!
interface serial 8
no ip address
!
interface serial 9
no ip address
!
interface tokenring 0
ip address 172.28.1.190 255.255.255.192
ring-speed 16
!
interface BRI0
ip address
shutdown
!
ip host ss10 172.28.0.40
ip host s2000 172.31.0.2
ip route 0.0.0.0 0.0.0.0 172.28.1.129
!
snmp-server community public RO
!
line con 0
exec-timeout 0 0
line 2
no activation-character
transport input-all
parity even
stopbits 1
rxspeed 4800
txspeed 4800
line 4
transport input all
parity even
stopbits 1
rxspeed 4800
txspeed 4800
line aux 0
transport input all
line vty 0 4
password mango
login
```

```
!  
end
```

## Configuration for Router B

```
version 11.0  
!  
hostname router-b  
!  
bstun peer-name 172.28.1.189  
bstun protocol-group 1 bsc  
bstun protocol-group 2 adplex  
bstun protocol-group 3 adplex  
source-bridge ring-group 100  
!  
interface serial 0  
no ip address  
!  
interface serial 1  
no ip address  
!  
interface serial 2  
physical-layer async  
description Connection to Security Alarm Panel.  
no ip address  
encapsulation bstun  
no keepalive  
bstun group 2  
bstun route all tcp 172.28.1.190  
adplex primary  
!  
interface serial 3  
description Connection to BSC 3780 device.  
no ip address  
encapsulation bstun  
no keepalive  
clockrate 9600  
bstun group 1  
bstun route all tcp 172.28.1.190  
bsc char-set ebcdic  
bsc contention  
!  
interface serial 4  
physical-layer async  
description Connection to async port on NCD (VT100 terminal emulation).  
no ip address  
!  
interface serial 5  
no ip address  
encapsulation sdlc-primary  
no keepalive  
nrzi-encoding  
clockrate 9600  
sdlc traddr 4000.0000.4100 222 2 100  
sdlc address C1  
sdlc xid C1 05D40003  
sdlc partner 4000.0000.0307 C1  
!  
interface serial 6  
description Connection to alarm panel.  
physical-layer async  
no ip address  
encapsulation bstun  
no keepalive
```

```
bstun group 3
bstun route all tcp 172.28.1.190
adplex primary
!interface serial 7
no ip address
!
interface serial 8
no ip address
!
interface serial 9
no ip address
!
interface tokenring 0
ip address 172.28.1.189 255.255.255.192
ring-speed 16
source-bridge 4 1 100
!
interface BRI0
ip address
shutdown
!
ip host ss10 172.28.0.40
ip host s2000 172.31.0.2
ip route 0.0.0.0 0.0.0.0 172.28.1.129
!
snmp-server community public RO
!
line con 0
exec-timeout 0 0
line 2
no activation-character
transport input-all
parity even
stopbits 1
rxspeed 4800
txspeed 4800
line 4
transport input all
stopbits 1
line 6
transport input all
parity even
stopbits 1
rxspeed 4800
txspeed 4800
line 7
transport input all
line aux 0
transport input all
line vty 0 4
password mango
login
!
end
```

## BSTUN-over-Frame Relay Configuration Example for Local Acknowledgment

```
bstun protocol-group 1 bsc-local-ack

interface Serial1
encapsulation frame-relay ietf
clockrate 125000
frame-relay map llc2 16
```

```
interface Serial4
  no ip address
  encapsulation bstun
  bstun group 1
  bsc secondary
  bstun route address C3 interface Serial1 dlc1 16 C
  bstun route address C2 interface Serial1 dlc1 16 8
  bstun route address C1 interface Serial1 dlc1 16 4
```

## BSTUN-over-Frame Relay Configuration Example for Passthrough

```
bstun protocol-group 1 bsc

interface Serial1
  encapsulation frame-relay
  clockrate 125000
  frame-relay map bstun 16

interface Serial4
  no ip address
  encapsulation bstun
  bstun group 1
  bsc secondary
  bstun route address C3 interface Serial1 dlc1 16
  bstun route address C2 interface Serial1 dlc1 16
  bstun route address C1 interface Serial1 dlc1 16
```

