

Guide to BSC and BSTUN

Document ID: 5316

Introduction

Prerequisites

- Requirements
- Components Used
- Conventions

System Overview

BSC/BSTUN Configuration

- Global Commands
- Interface Commands
- TCP Route Configuration
- Serial Route Configuration
- Direct Frame Relay Passthru Configuration
- Direct Frame Relay Local–Ack Configuration
- Passthru Configuration
- Local–Ack Configuration
- Contention Configuration
- Priorities
- Keepalives Configuration

debug Commands

show Commands

- show bstun
- show bsc
- show interface serial <number>

How to Troubleshoot IBM Bisync

- How to Use Passthru FSMs
- How to Use Local–Ack FSM

Common Problems

- Passing 3780 Data to 3270 Config or Vice Versa
- Config Route to a Bad Peer
- Config Bad Group Numbers
- Tandem Hosts
- Difference Between Full and Half Duplex

BSC and BSTUN Examples

- No Device Response Example
- Network Latencies Example

BSC and BSTUN Sample Configurations

- Network Diagram
- Configurations

References

Related Information

Introduction

This document is designed to help you configure and use the Binary Synchronous Communication (BSC) data link protocol and Block Serial Tunneling (BSTUN) on Cisco routers. It also helps you to troubleshoot problems that might occur.

Prerequisites

Requirements

Readers of this document should have knowledge of these topics:

- Binary Synchronous Communications (BSC) concepts.
- General understanding of basic data processing principles.

Components Used

The information in this document is based on Cisco IOS® software with the IBM feature set.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

For more information on document conventions, refer to the Cisco Technical Tips Conventions.

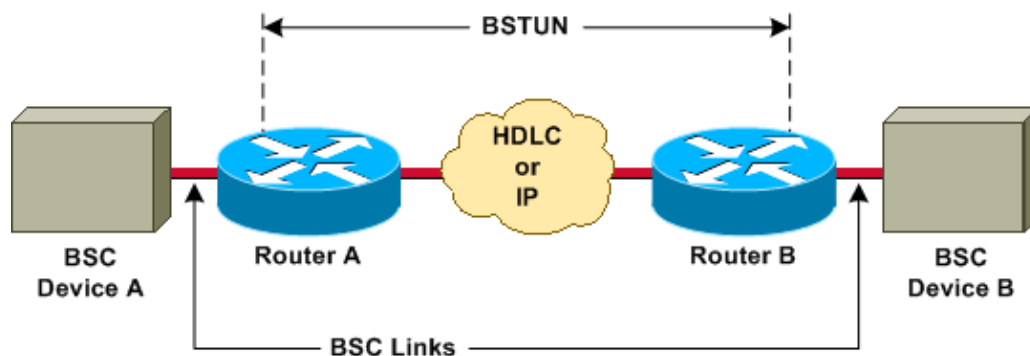
System Overview

Figures 1 and 2 show how an existing BSC link between two devices can be reconfigured to use Cisco routers. This provides the same logical link, without any changes to the existing BSC devices.

Figure 1 Existing BSC Setup



Figure 2 BSC Setup with Cisco Routers



The Cisco routers transport all BSC blocks between the two devices, through the use of Block Serial Tunneling (BSTUN) encapsulation. For each BSC block that is received from the line, an address and control bytes are added to create a BSTUN frame, then BSTUN is used to deliver to the correct destination router.

BSC/BSTUN Configuration

On a clean router, issue these commands, in the order in which they are listed.

Global Commands

[no] bstun peer-name *ip-address*

The *ip-address* defines the address by which this BSTUN peer is known to other BSTUN peers that use the TCP transport.

Note: This command must be configured in Cisco IOS Software Releases earlier than Release 11.3, or it must be configured if TCP/IP addresses are used in route statements.

[no] bstun protocol-group *group-number* {**bsc** | **bsc-local-ack** | **adplex** | **adt-poll** | **adt-poll-select** | **adt-vari-poll** | **diebold** | **async-generic** | **mdi**}

This is a global command to associate group numbers with protocol names. The *group-number* is a decimal integer between 1 and 255. The **bsc** | **bsc-local-ack** | **adplex** are predefined BSTUN protocol keywords. For more information, refer to Defining the Protocol Group in Configuring Serial Tunnel and Block Serial Tunnel.

The selection of the group type is important to determine whether to use passthru or local acknowledgement (local-ack).

Note: This command must always be configured.

Interface Commands

encapsulation bstun

This is an interface command that configures the BSTUN function on a particular serial interface. This command must be configured on an interface before any further BSTUN or BSC commands are configured for this interface.

[no] bstun group *group-number*

This is an interface command that defines the BSTUN group to which this interface belongs. Each BSTUN-enabled interface on a router must be placed in a previously defined BSTUN group. Packets only travel between BSTUN-enabled interfaces that are in the same group. The *group-number* is a decimal integer between 1 and 255.

The group number has already determined whether this interface runs local-ack or passthru.

[no] bsc mode

Here are a list of some of the main options. For a comprehensive list, refer to Configuring Bisync Options on a Serial Interface in Configuring Serial Tunnel and Block Serial Tunnel

No frames are received or sent until the mode is configured for one of these settings:

- **contention** This sets the BSC link that is connected to the serial interface to be for a point-to-point BSC station. Only 3780, and only in passthru mode.

- **contention** *virtual-address* First available in Cisco IOS Software Release 11.3. Used with *dial-contention* to enable multiple remote devices to use the same interface at the host-end router.
- **dial-contention** *timeout* First available in Cisco IOS Software Release 11.3. Used at the host-end router for contention. Enables multiple remote devices to multiplex over the same physical interface.
- **primary** Defines that the router is acting as the primary end of the BSC link and that the attached device or devices are BSC tributary stations.
- **secondary** Defines that the router is acting as the secondary end of the BSC link and that the attached remote device is a BSC control station (such as a front-end processor [FEP] or other host device).

If this command is not configured then the line protocol on the interface will be down and the interface will not operate.

TCP Route Configuration

In this configuration, the transport system is TCP/IP. This can run over any of the physical media over which TCP/IP can run.

[no] bstun route all tcp *ip-address*

[no] bstun route address *address-number* **tcp** *ip-address*

The *ip-address* is the same as the IP address that is specified in the *peer-name* of the partner router.

Serial Route Configuration

In this configuration, the tunnel uses the Cisco proprietary transport. It is much faster than TCP/IP, but goes over a serial interface only.

[no] bstun route all interface serial *interface-number*

[no] bstun route address *address-number* **interface serial** *interface-number*

Direct Frame Relay Passthru Configuration

In this configuration, the tunnel uses a proprietary form of serial encapsulation over Frame Relay, which works as fast as serial routes.

[no] bstun route address *address-number* **interface serial** *interface-number* **dlci** *dlci-number*

Issue this command on the Frame Relay interface:

[no] frame-relay map *dlci-number* **bstun**

Direct Frame Relay Local-Ack Configuration

This configuration uses Logical Link Control, type 2 (LLC2) over Frame Relay encapsulation, to give local-acknowledgment and end-to-end session control. The **lsap** keyword must be included; if not, the encapsulation will go as passthru.

[no] bstun route address *address-number* **interface serial** *interface-number* **dlci** *dlci-number* **lsap** *lsap*

Issue this command on the Frame Relay interface:

[no] frame-relay map *dldci-number llc2*

Note: For more information, refer to *Specifying How Frames Are Forwarded in Configuring Serial Tunnel and Block Serial Tunnel*.

Passthru Configuration

Why Passthru?

Passthru is the basic tunnelling mode. Every frame that is transmitted between the devices gets passed, unchanged, through the BSTUN tunnel. A sequence number and device address are added, to ensure that latencies through the network do not affect the protocol operation. The arrival of late polls or end of transmission (EOT) signals could significantly disrupt an existing session.

When to Use Passthru

Passthru should be used in these circumstances:

- The data that is being transferred does not have an explicit acknowledgment frame sent to verify data integrity.
- The protocol is not pure 3270.
- The user wants end-to-end device connectivity and network latencies are small.

Local-Ack Configuration

Why Local-Ack?

Local-ack removes the overhead of sending all control frames across the tunnel. When the host sends the first poll to a control unit, a special control frame is sent across the tunnel to start the remote polling of that device address. Once the remote device indicates that it is up, a control frame is sent to the host router to tell it to respond to the polls. When the remote device goes down, an indication is sent across the tunnel to tell the host router to no longer respond to polls.

When to Use Local-Ack

Local-ack can be used in these circumstances:

- 3270 bisync is in use.
- Network latency causes bisync session timeouts.
- Excess traffic across the WAN is a problem.

Local-Ack Options

[no] bsc pause *time*

This command specifies the amount of time between the start of one polling cycle and the next. The default value is 30 (that is, 30 tenths or 3 seconds).

It is a good idea to configure this command when there are only one or two controllers on the bisync interface. It effectively slows down the polling and allots more CPU cycles to the attached device.

[no] bsc poll-timeout *time*

This command sets the timeout for a poll or select sequence, in units of one-tenths of a second; the default value is 30 (that is, 30 tenths, or 3 seconds).

The smallest *time* value is determined by the speed of the attached device, and it is of more interest at the host end. If the host that is driving the router reduces its timeout to the smallest value possible, there will be a performance improvement when some devices have failed.

[no] bsc retries *retry-number*

This command sets the number of retries to attempt before a device is considered dead. The range is 1 to 100; the default is 5 retries.

[no] bsc servlim *value*

This command specifies the servlim (active versus inactive end-station poll ratio) value. The range is 1 to 50; the default is 3.

[no] bsc spec-poll

This command tells the host to handle specific polls as general polls. Use this command when you are working with Tandem Hosts.

For a more information, refer to Configuring Bisync Options on a Serial Interface in Configuring Serial Tunnel and Block Serial Tunnel.

Contention Configuration

Why Contention?

Contention is the 3780 variant of bisync. There are no control unit addresses. The devices are connected point-to-point. Generally, a remote device dials into a central location and assumes that no other devices exist.

When to Use Contention

Use contention *only* when you are using the remote job entry (RJE), 3780, and 2780 protocols. Once you have identified contention, ensure that both ends are configured to use contention.

If you are unsure, then do these steps:

1. Configure **bsc primary**.
2. Turn on **debug bsc packet**.
3. Make the attached device start to poll.

Messages with 1 bytes 2D indicate contention. Any bytes before the 2D are not 3780.

Priorities

When compared with all other traffic that is going over the WAN backbone, bisync traffic is very small and easily swamped by other traffic. A loss of frames in bisync requires a long recovery interval, which is readily apparent to the end devices. To minimize this problem, prioritization of bisync traffic is recommended. You can prioritize traffic either with BSTUN priorities or with custom queuing.

- Priority queuing is a routing feature in which frames in an interface output queue are prioritized based on various characteristics, such as packet size or interface type.

Priority output queuing allows a network administrator to define four priorities of traffic high, normal, medium, and low on a given interface. As traffic comes into the router, it is assigned to one of the four output queues. Packets on the highest priority queue are transmitted first. When that queue empties, traffic on the next highest priority queue is transmitted, and so forth. This mechanism ensures that, during congestion, the highest priority data does not get delayed by lower priority traffic. However, if the traffic sent to a given interface exceeds the bandwidth of that interface, lower priority traffic can experience significant delays.

For example, if you make IP a higher priority than IPX on WAN serial links, the BSC traffic in TCP/IP will take advantage of the fact that IP is being transferred at a higher priority.

- Custom queuing allows a customer to reserve a percentage of bandwidth for specified protocols. Customers can define up to ten output queues for normal data and an additional queue for system messages, such as LAN keepalive messages (routing packets are not assigned to the system queue). Cisco routers service each queue sequentially: they transmit a configurable percentage of traffic on each queue before they move on to the next one. When you use custom queuing, you can guarantee that mission-critical data is always assigned a certain percentage of the bandwidth, while predictable throughput for other traffic is also ensured.

To provide this feature, Cisco routers determine how many bytes should be transmitted from each queue, based on the interface speed and the configured percentage. When the calculated byte count from a given queue has been transmitted, the router completes transmission of the current packet and moves on to the next queue. Eventually, each queue is serviced, in a round-robin fashion.

Refer to Configuring Serial Tunnel and Block Serial Tunnel, and refer to Deciding Which Queuing Policy to Use in Congestion Management Overview.

[no] priority-list *list-number protocol bstun queue [gt | lt packet-size] [address bstun-group bsc-addr]*

Issue the **priority-list protocol bstun global** configuration command to establish BSTUN queuing priorities based on the BSTUN header. Issue the **no** form of the command to revert to normal priorities.

[no] custom-queue-list [*list*]

The *list* is an integer (1 – 16) that represents the number of the custom queue list.

Keepalives Configuration

[no] bstun remote-peer-keepalive *interval*

This command enables BSTUN peer keepalives. This sends a request to the peer whenever the peer has been silent for longer than the *interval* time period. Any frame resets the clock, not just keepalives. The default is 30 seconds.

[no] bstun keepalive-count *number*

When this *number* of keepalives is missed consecutively, the BSTUN connection is brought down. The default is 3.

When to Use Keepalives

Keepalives are useful to protect against tunnel outages when you are running local-ack and TCP/IP. The tunnel brings down an interface only when a signal is received from the remote. If the tunnel is down, no signals are ever received.

In passthru, this is not needed, because end-to-end connectivity is required.

debug Commands

[no] debug bstun event *group*

This command allows you to debug BSTUN connections and status. When enabled, it causes the display of messages that show connection establishment and overall status.

[no] debug bstun packet *group* *buffer-size* *displayed-bytes-size*

This command allows you to debug packets that travel through the BSTUN links.

[no] debug bsc packet *group* *buffer-size* *displayed-byte-size*

This command allows you to debug frames that travel through the BSC feature.

[no] debug bsc packet

This command allows you to debug frames that travel through the BSC feature. It traces all interfaces that are configured with a BSTUN group number.

[no] debug bsc event *group*

This command allows you to debug events that occur in the BSC feature. If the *group* number is omitted, then it traces all interfaces that are configured with a BSTUN group number.

show Commands

show bstun

This command displays the current status of BSTUN.

```
This peer: 10.10.20.108
 *Serial5 -- interface for ATM: R1710V421 (group 3 [bsc])
route transport address      state      rx_pkts  tx_pkts  drops
C2    TCP      10.10.10.107 open      655630   651332   0
 *Serial6 -- interface for SEC: MST012 (group 2 [bsc])
route transport address      state      rx_pkts  tx_pkts  drops
C2    TCP      10.10.10.107 open      649385   644001   0
```

Check for these problems:

- State closed.
- Drops.
- Low packet count.

Note: Low packet count does not always indicate problems. When you are running local-ack, the count consists only of data frames, which is significantly smaller than the actual number of frames that are sent from the host.

show bsc

This command displays the current status of BSC.

In Passthru

```
BSC pass-through on Serial5:
Output queue depth: 0.
HDX enforcement state: IDLE.
Frame sequencing state: SEC.
Tx-Active: Idle. Rx-Active: False.
Tx Counts: 670239 frames(total). 670239 frames(data). 9288816 bytes.
Rx Counts: 651332 frames(total). 651332 frames(data). 651332 bytes.
```

Check for these problems:

- If `HDX enforcement state` gets stuck in a state other than `IDLE`, then there may be a problem with the attached device or with this router. This usually indicates that the device is not responding. Turn on **bsc event debug**. If you see a lot of `no response from remote` messages, first check that the device is activated, then check duplex. If there are no messages and no eventual recovery, then a transmit completion event has been lost, and a bug has been found that may potentially be catastrophic.
- The `Frame sequence state` tells you which finite state machine (FSM) to check.
- If `Rx-Active` is stuck at `True`, this indicates that something bad has happened with the hardware. Issue a **shut** and then a **no shut** to reset the interface. If this does not work, reload the router.

In Local-Ack

```
BSC local-ack on Serial0:
Secondary state is CU_Idle.
Control units on this interface:

    Poll address: 40. Select address: 60 *CURRENT-CU*
    Current active device address is: 40.
    State is Active.
    Tx Counts: 87228 frames(total). 11 frames(data). 87353 bytes.
    Rx Counts: 87271 frames(total). 5 frames(data). 436312 bytes.

Total Tx Counts: 87228 frames(total). 11 frames(data). 87353 bytes.
Total Rx Counts: 174516 frames(total). 5 frames(data). 523557 bytes.
```

If the state gets stuck in `TCU_Down`, this indicates that something is forcing that interface to stay down. Check clocking and BSC mode and ensure that nothing is down administratively. Occasionally, a **shut** command followed by a **no shut** command starts the interface again.

In General

- An output queue depth greater than 1 indicates a backlog on the interface. Check that half duplex is configured properly.
- Out of `SYN-hunt` mode means either that the interface is down or that the receiver has been disabled. That which applies to `Rx-Active` also applies here.

show interface serial <number>

This command is useful to see the counters that are associated with that serial interface.

```
Received 0 broadcasts, 0 runts, 0 giants
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
```

Note: Any errors mean problems.

Check for these problems:

- `aborts` indicate a bad transmission.
- `ignored` frames are frames that violate the bisync protocol.
- `giants` indicate either that the MTU is too small or a bisync sequence is bad.
- `overrun` indicates CPU resource shortages.
- `CRC` indicates corruption over the line (noisy or other).

If you are using DTE cable and the line seems to be going down frequently, or transmits fail but receives work, then you might need to issue the **ignore-dcd** command. This can be verified with a protocol analyzer. When the DCE transmits, the Data Carried Detect (DCD) is raised. When it finishes, DCD is lowered so the router not able to reply.

- `Hardware is CD2430` indicates the Cirrus chip set.
- `Hardware is HD64570` indicates the Hitachi chip set.

Hitachi uses character interrupts and software-built framing. It does not handle DCD well. Cirrus uses frame interrupts. Frames are built in ucode. It has options to play with DCD. It is important, when you are debugging, that you know the interface type, because there are some differences between them.

The line protocol must be up. If the line protocol is not up, then check that BSC mode is configured.

```
Serial5 is up, line protocol is up
  Hardware is CD2430 in sync mode
  MTU 265 bytes, BW 4 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation BSTUN, loopback not set
  Half-duplex enabled.
  cts-delay 0 millisec
  dcd-txstart-delay 100 millisec
  dcd-drop-delay 100 millisec
  transmit-delay 0 millisec
  Errors - 0 half duplex violation
  Last input 10:27:12, output 1:07:12, output hang never
  Last clearing of "show interface" counters 4d11
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
  3223346 packets input, 3223356 bytes, 0 no buffer
  Received 0 broadcasts, 0 runts, 0 giants
  0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
  3242346 packets output, 45259079 bytes, 0 underruns
  0 output errors, 0 collisions, 8 interface resets, 0 restarts
  0 output buffer failures, 0 output buffers swapped out
  4 carrier transitions
  DCD=up DSR=up DTR=up RTS=down CTS=down
```

How to Troubleshoot IBM Bisync

How to Use Passthru FSMs

Ensure that you are running passthru. You need to find the correct finite state machine (FSM) to follow.

Look at the event debug messages. There are two FSMs to go through. The HDX-FSM is a half-duplex enforcement FSM. It is driven regardless of whether the line is configured full duplex or half duplex. It tries to ensure that the transmit queue of a router does not get backlogged with old data. The FS-FSM ensures that late frames through the network do not destroy established sessions.

To determine where to look, go straight to contention FSM, if contention is configured. Otherwise, look at the state into which it goes after the IDLE state. If you see SEC, look at the secondary frame sequence FSM. If you see PRI, look at the primary frame sequence FSM.

```
BSC: Serial6: HDX-FSM event: RXV old_state: PND_RCV. new_state: IDLE.
BSC: Serial6: FS-FSM event: SDI EOT old_state: SEC. new_state: IDLE.
BSC: Serial6: NDI: Data (8 bytes): C24100C2C27F7F2D
BSC: Serial6: FS-FSM event: NDI BID old_state: IDLE. new_state: SEC.
BSC: Serial6: New Address(C2) New NS(01)
BSC: Serial6: HDX-FSM event: TX old_state: IDLE. new_state: PND_COMP.
BSC: Serial6: HDX-FSM event: CmpOTH old_state: PND_COMP. new_state: PND_RCV.
BSC: Serial6: SDI: Data (1 bytes): 37
BSC: Serial6: HDX-FSM event: RXV old_state: PND_RCV. new_state: IDLE.
```

When you look at the table, you see inputs on the left side and you see states on the top. Each entry in a column is of the form *{next state,action}* The action is done first, then the transition happens.

How to Use Local-Ack FSM

Ensure that you are running local-ack. A **show bsc** command tells you whether the interface is poller or pollee. From this, use the appropriate LACK FSM.

Common Problems

Passing 3780 Data to 3270 Config or Vice Versa



Caution: Do not do this. This does not work reliably.

Config Route to a Bad Peer

You have configured everything and nothing happens. You turn on **debug bsc packet** on the remote router and see nothing. You then turn on **debug bstun packet** and still see nothing. At this stage, turn on **debug bstun event**; you probably still see nothing. Go back to the host end router and turn on **debug bstun event**. You should now see several messages that indicate a bad connection.

Config Bad Group Numbers

This is observed when either end of the tunnel is configured with a different group number. Data either spills out of the wrong interface or gets discarded at the BSTUN level.

Local-ack and passthru group numbers do not mix. Ensure that the protocol-group definitions are consistent

across the entire network. Devices that run contention (3780) also need to be on different group numbers from a 3270.

Tandem Hosts

```
21:55:18: BSC: Serial4: SDI-rx: Data (5 bytes): C7C740402D
21:55:19: BSC: Serial5: SDI-tx: Data (1 bytes): 37
21:55:19: BSC: Serial5: SDI-tx: Data (5 bytes): C2C240402D
21:55:21: BSC: Serial4: SDI-rx: Data (1 bytes): 37
21:55:21: BSC: Serial4: SDI-rx: Data (5 bytes): C7C740402D
21:55:22: BSC: Serial5: SDI-tx: Data (1 bytes): 37
21:55:22: BSC: Serial5: SDI-tx: Data (5 bytes): 404040402D
21:55:24: BSC: Serial4: SDI-rx: Data (1 bytes): 37
```

Tandems do not obey strict 3270 conventions. They do all of their polling with specific polls, which causes a problem for the default LACK FSM. To get tandems to work properly, configure **bsc spec-poll** on the BSC secondary interface.

Difference Between Full and Half Duplex

It is easy to confuse full duplex and half duplex.

- Full duplex can transmit data simultaneously between a sending station and a receiving station.
- Half duplex can transmit data in only one direction at a time, between a sending station and a receiving station.

See the section on the **show bsc** command for more details.

If you have a protocol analyzer or a breakout box available, connect your analyzer in the system without routers.

- If RTS or CTS changes signal, then you have half duplex; else it is full duplex.
- If DCD seems to change a lot, and the line goes up and down or stays down, you might have switching DCD.

Note: The primary router may be full duplex while the remote router is half duplex, and vice versa. These are separate physical lines, and the control signals from the interfaces are not transported across the tunnel.

BSC and BSTUN Examples

No Device Response Example

This is an example of two interfaces on a secondary router: one local-ack and the other passthru. Neither is receiving a response from the remote. As soon as you see polls come into the secondary router, you need to determine what is happening at the remote end.

```
21:55:18: BSC: Serial4: SDI-rx: Data (5 bytes): C7C77F7F2D
21:55:19: BSC: Serial5: SDI-tx: Data (1 bytes): 37
21:55:19: BSC: Serial5: SDI-tx: Data (5 bytes): C2C27F7F2D
21:55:21: BSC: Serial4: SDI-rx: Data (1 bytes): 37
21:55:21: BSC: Serial4: SDI-rx: Data (5 bytes): C7C77F7F2D
21:55:22: BSC: Serial5: SDI-tx: Data (1 bytes): 37
21:55:22: BSC: Serial5: SDI-tx: Data (5 bytes): 40407F7F2D
21:55:24: BSC: Serial4: SDI-rx: Data (1 bytes): 37
21:55:24: BSC: Serial4: SDI-rx: Data (5 bytes): C7C77F7F2D
21:55:25: BSC: Serial5: SDI-tx: Data (1 bytes): 37
```

```
21:55:25: BSC: Serial5: SDI-tx: Data (5 bytes): C2C27F7F2D
21:55:27: BSC: Serial4: SDI-rx: Data (1 bytes): 37
21:55:27: BSC: Serial4: SDI-rx: Data (5 bytes): C7C77F7F2D
21:55:28: BSC: Serial5: SDI-tx: Data (1 bytes): 37
21:55:28: BSC: Serial5: SDI-tx: Data (5 bytes): C2C27F7F2D
21:55:30: BSC: Serial4: SDI-rx: Data (1 bytes): 37
21:55:30: BSC: Serial4: SDI-rx: Data (5 bytes): C7C77F7F2D
```

When you look at the remote end in the passthru case, you can see frames coming through the tunnel, but the attached device is still quiet.

```
BSC: Serial6: NDI: Data (8 bytes): C24100C2C27F7F2D
BSC: Serial6: NDI: Data (4 bytes): C2C00037
BSC: Serial6: NDI: Data (8 bytes): C24100C2C27F7F2D
BSC: Serial6: NDI: Data (4 bytes): C2C00037
BSC: Serial6: NDI: Data (8 bytes): C24100C2C27F7F2D
BSC: Serial6: NDI: Data (4 bytes): C2C00037
BSC: Serial6: NDI: Data (8 bytes): C24100C2C27F7F2D
BSC: Serial6: NDI: Data (4 bytes): C2C00037
BSC: Serial6: NDI: Data (8 bytes): C24100C2C27F7F2D
BSC: Serial6: NDI: Data (4 bytes): C2C00037
```

Next, determine whether the attached device is dead or whether the router has a bad transmitter: turn on event debugging.

```
BSC: Serial6: NDI: Data (8 bytes): C24100C2C27F7F2D
BSC: Serial6: FS-FSM event: NDI BID old_state: IDLE. new_state: SEC.
BSC: Serial6: New Address(C2) New NS(01)
BSC: Serial6: HDX-FSM event: TX old_state: IDLE. new_state: PND_COMP.
BSC: Serial6: HDX-FSM event: CmpOTH old_state: PND_COMP. new_state: PND_RCV.
BSC: Serial6: Response not received from remote
BSC: Serial6: HDX-FSM event: T/O old_state: PND_RCV. new_state: IDLE.
BSC: Serial6: NDI: Data (4 bytes): C2C00037
BSC: Serial6: FS-FSM event: NDI EOT old_state: SEC. new_state: IDLE.
BSC: Serial6: HDX-FSM event: TX old_state: IDLE. new_state: PND_COMP.
BSC: Serial6: HDX-FSM event: CmpEOT old_state: PND_COMP. new_state: IDLE.
BSC: Serial6: NDI: Data (8 bytes): C24100C2C27F7F2D
BSC: Serial6: FS-FSM event: NDI BID old_state: IDLE. new_state: SEC.
BSC: Serial6: New Address(C2) New NS(01)
```

From the trace, follow the HDX-FSM. If it is stuck in the PND_COMP state, the transmitter is failing. It is probably the case that no clock is being supplied. As you can see in the previous example output, PND_RCV state is reached, and you see the `Response not received from remote`, which points to either a bad receive or an inactive device.

Network Latencies Example

This is an example of network latencies in a virtual multidrop environment:

```
BSC: Serial10: NDI: Data (5 bytes): C703001061
BSC: Serial10: SDI: Data (1 bytes): 37
BSC: Serial10: SDI: Data (1 bytes): 37
BSC: Serial10: Discard SDI: Data (1 bytes): 37
BSC: Serial10: SDI: Data (5 bytes): 404040402D
BSC: Serial10: NDI: Data (4 bytes): 40C00037
BSC: Serial10: SDI: Data (1 bytes): 37
BSC: Serial10: Discard SDI: Data (1 bytes): 37
```

!--- Output suppressed.

```
BSC: Serial10: SDI: Data (1 bytes): 37
BSC: Serial10: Discard SDI: Data (1 bytes): 37
```

```
BSC: Serial0: SDI: Data (5 bytes): C4C4C4C42D
```

There is a problem here, because C4 has not replied in time:

```
BSC: Serial0: SDI: Data (1 bytes): 37
BSC: Serial0: SDI: Data (1 bytes): 37
BSC: Serial0: Discard SDI: Data (1 bytes): 37
BSC: Serial0: SDI: Data (5 bytes): C5C5C5C52D
BSC: Serial0: NDI: Data (4 bytes): C5C00037
BSC: Serial0: SDI: Data (1 bytes): 37
BSC: Serial0: Discard SDI: Data (1 bytes): 37
BSC: Serial0: SDI: Data (5 bytes): C7C7C7C72D
```

Again, this is lost. Look further, and you see that the problem becomes a little worse:

```
BSC: Serial0: SDI: Data (1 bytes): 37
BSC: Serial0: SDI: Data (1 bytes): 37
BSC: Serial0: Discard SDI: Data (1 bytes): 37
BSC: Serial0: SDI: Data (5 bytes): 404040402D
BSC: Serial0: NDI: Data (4 bytes): 40C00037
BSC: Serial0: SDI: Data (1 bytes): 37
BSC: Serial0: Discard SDI: Data (1 bytes): 37
BSC: Serial0: SDI: Data (5 bytes): C1C1C1C12D
```

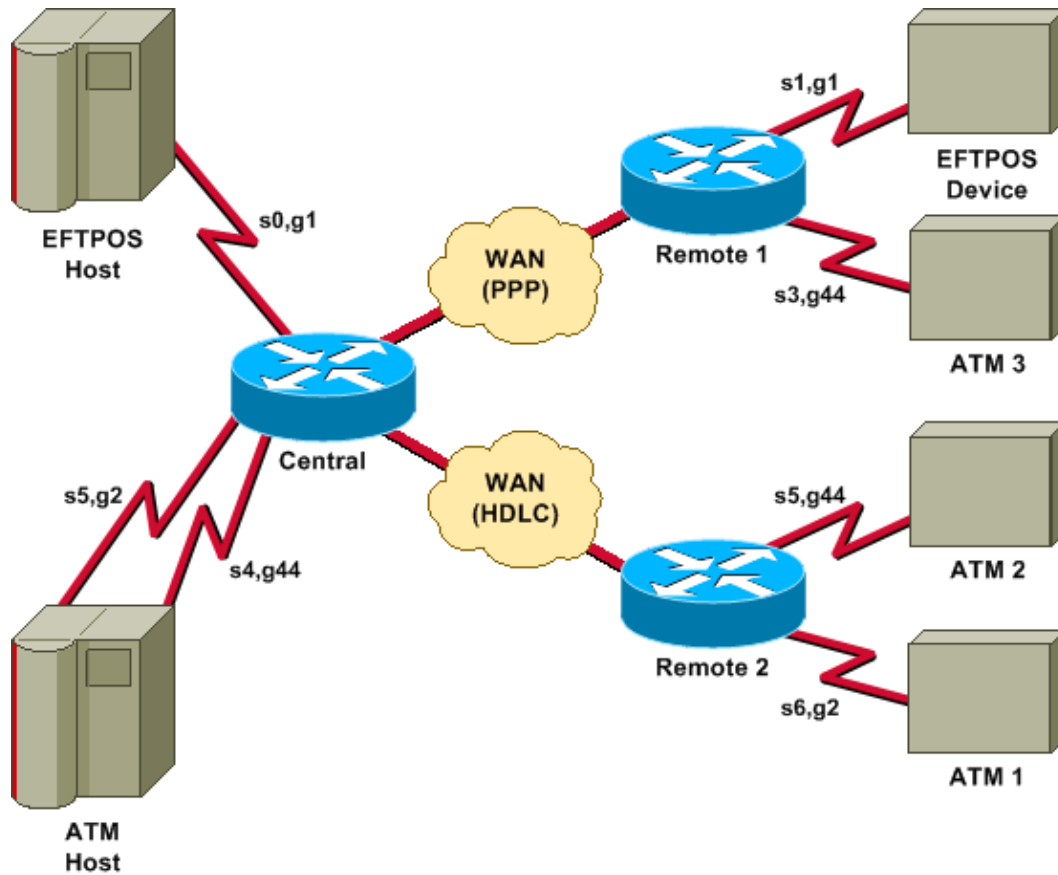
The EOT for C7 has suddenly appeared again. Discard that EOT to recover from this; the next frame is the EOT for C1.

In this example, frames from the network are arriving late and out of sequence. This causes a large number of unanswered polls at the host. The solution, in this case, is to configure local-ack.

BSC and BSTUN Sample Configurations

Network Diagram

This diagram is a sample configuration of a site that is running both 3270 and 3780 bisync terminals.



Configurations

That diagram uses these configurations:

- Central
- Remote 1
- Remote 2

Central
<pre> hostname central ! bstun peer-name 10.10.10.107 bstun protocol-group 1 bsc bstun protocol-group 2 bsc bstun protocol-group 44 bsc-local-ack ! interface Serial0 description EFTPOS host no ip address encapsulation bstun no keepalive full-duplex clockrate 19200 bstun group 1 bsc contention 1 bstun route all tcp 10.10.10.108 ! interface Serial2 description WAN-ppp backbone ip address 10.10.10.107 255.255.255.0 encapsulation ppp </pre>

```

clockrate 2000000
!
interface Serial3
description WAN-hdlc
ip address 10.10.20.107 255.255.255.0
bandwidth 2000
no keepalive
clockrate 2000000
!
interface Serial4
description ATM Host
no ip address
encapsulation bstun
no keepalive
full-duplex
bstun group 44
bsc secondary
bstun route all tcp 10.10.20.108
!
interface Serial5
description ATM host
no ip address
encapsulation bstun
no keepalive
bstun group 2
bsc secondary
bstun route address C2 tcp 10.10.20.108
!
end

```

Remote 1

```

hostname remotel
!
bstun peer-name 10.10.10.108
bstun protocol-group 1 bsc
bstun protocol-group 44 bsc-local-ack
!
interface Serial0
description EFTPOS 1
no ip address
encapsulation bstun
no keepalive
full-duplex
clockrate 19200
bstun group 1
bsc char-set ebcdic
bsc contention
bstun route all tcp 10.10.10.107
!
interface Serial1
description ATM 3
no ip address
encapsulation bstun
no keepalive
bstun group 44
bsc char-set ebcdic
bsc primary
bstun route address 40 tcp 10.10.10.107
!
interface Serial3
description WAN -ppp
ip address 10.10.10.108 255.255.255.0
encapsulation ppp
!

```

```
end
```

Remote 2

```
hostname remote2
!
!
bstun peer-name 10.10.20.108
bstun protocol-group 2 bsc
bstun protocol-group 44 bsc-local-ack
bstun protocol-group 10 bsc-local-ack
!
interface Serial0
  description WAN-hdlc
  ip address 10.10.20.108 255.255.255.0
  bandwidth 2000
  no keepalive
!
interface Serial5
  description ATM 1
  mtu 265
  encapsulation bstun
  clockrate 19200
  bstun group 44
  bsc char-set ebcdic
  bsc primary
  bstun route address C2 tcp 10.10.10.107
!
interface Serial6
  description interface for ATM 2
  mtu 265
  encapsulation bstun
  clockrate 19200
  bstun group 2
  bsc char-set ebcdic
  bsc primary
  bstun route address C2 tcp 10.10.10.107
!
ip route 10.10.10.0 255.255.255.0 10.10.20.107
!
end
```

References

General Information – Binary Synchronous Communication, IBM Systems Reference Library, GA27–3004–2.

IBM 3274: Chapter 4: Remote Operations BSC.

IBM 3275: Chapter 9.

BSTUN Commands on the Cisco Documentation CD-ROM (available online in Serial Tunnel and Block Serial Tunnel Commands).

Related Information

- [Configuring and Troubleshooting Serial Tunneling \(STUN\)](#)
- [Technology Support](#)
- [Product Support](#)

• **Technical Support – Cisco Systems**

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

© 2008 – 2009 Cisco Systems, Inc. All rights reserved. [Terms & Conditions](#) | [Privacy Statement](#) | [Cookie Policy](#) | [Trademarks of Cisco Systems, Inc.](#)

Updated: Sep 09, 2005

Document ID: 5316
