Application Note

Enabling Dual Chassis Fault Tolerance with Dialogic® Signaling Boards

Executive Summary

In order to achieve "five nines" availability and a high degree of fault tolerance in an SS7 signaling environment, an SS7 end point can be spread over two chassis using Dialogic® Signaling Distributed Architecture (SigDiA) enabled components. Splitting the functionality of a signaling point by implementing an SS7 node over more than one chassis allows separate signaling processes to be run on each chassis. This separation lets one signaling process continue if the other fails, allowing the overall system to remain in service.

Dialogic® SS7 components are designed for this dual chassis approach and provide the architecture for splitting a point code over two active SS7 protocol engines. Using this technique, links in an SS7 link set can be spread between two separate chassis, with Dialogic® Signaling Boards installed in each chassis. This application note discusses the design and implementation of such a dual chassis method for achieving high availability and fault tolerance using Dialogic® SS7 signaling boards.

Table of Contents

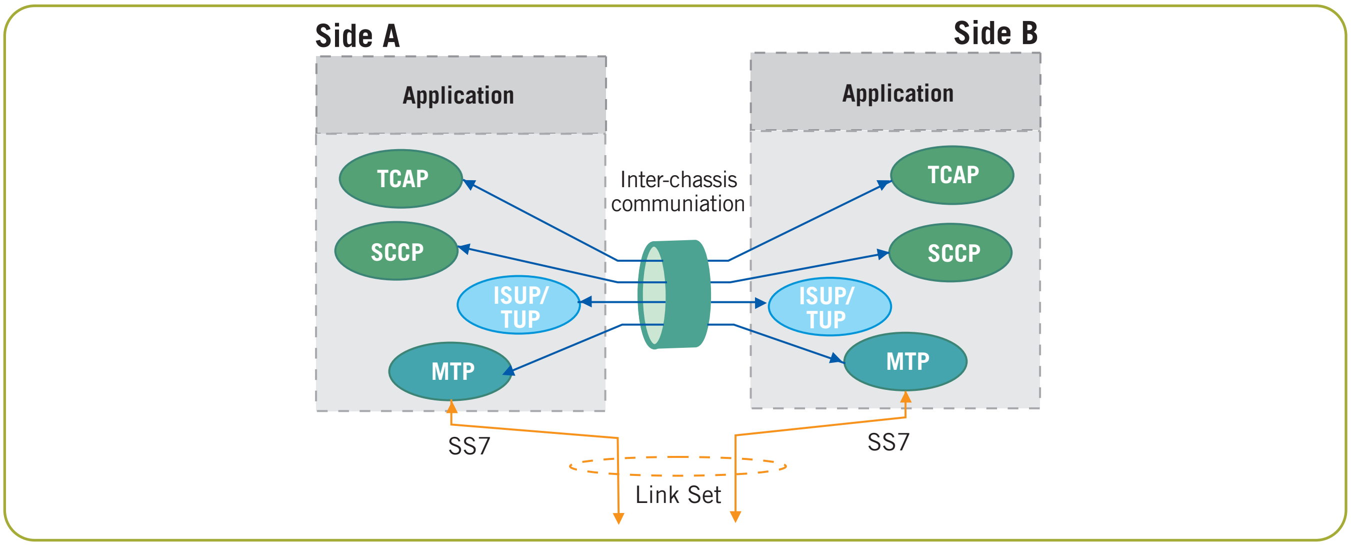

Figure 1. Dual Resilient SS7 Protocols

Introduction

The high expectations of service reliability by users of public telephony networks require equipment manufacturers and system integrators to demand high levels of fault tolerance from telecommunications systems, often citing "five nines" for availability (requiring a system to be operational for 99.999% of the time).

Such systems should continue to offer service even when a partial hardware or software failure has occurred. Several well-known methods of achieving this type of reaction to partial failure in the signaling component of communications networks include:

- Multiple signaling paths (SS7 links and link sets) to end points
- Distribution of signaling paths through independent interfaces and cabling
- Distribution of SS7 terminations at a single signaling point between multiple processing boards in a chassis
- Physical isolation and duplication of an SS7 interface for a single signaling point on independent protocol engines sharing a single point code (the Dialogic® Signaling Server with SIU Option and SS7 boards approach)
- Spreading the functionality of a signaling point, including the application layer between two chassis, which is the subject of this paper

The first three points in the above list can be achieved by implementing SS7 systems with multiple signaling links between two adjacent inter-communicating points. (By definition, these links are in the same link set). The final two options can be achieved by use of two co-operating SS7 protocol stacks, normally in two interconnected chassis, to provide a higher degree of resilience to failures.

Implementing an SS7 Signaling Point on More Than One Chassis

Implementing an SS7 node over more than one chassis isolates one hardware protocol engine from the failure of another, allowing the overall system to continue service following a complete failure of one component or chassis. Several methodologies achieve such a configuration, each with benefits and disadvantages.

Dialogic® SS7 signaling takes a 2N approach and provides an architecture for splitting a point code over two active SS7 protocol engines. This allows links in SS7 link sets to be spread between two separate chassis, with SS7 signaling boards installed in each.

Two communication paths need to be provided between the two system halves, one at the MTP layer, using the MTP2 data link as a transport protocol, the second at layer 4 or the User Part layer, to transport inter-process messages between the two layer 4 protocols on each half. The second is normally achieved by the use of a TCP/IP Ethernet and RSI (Remote Socket Interface) software provided by Dialogic. This combination provides an SS7 interface and a protocol engine split between two physically separate chassis, appearing to the network as a single entity.

Note: The MTP2 communication path between chassis can be achieved using TDM MTP2 or SIGTRAN M2PA

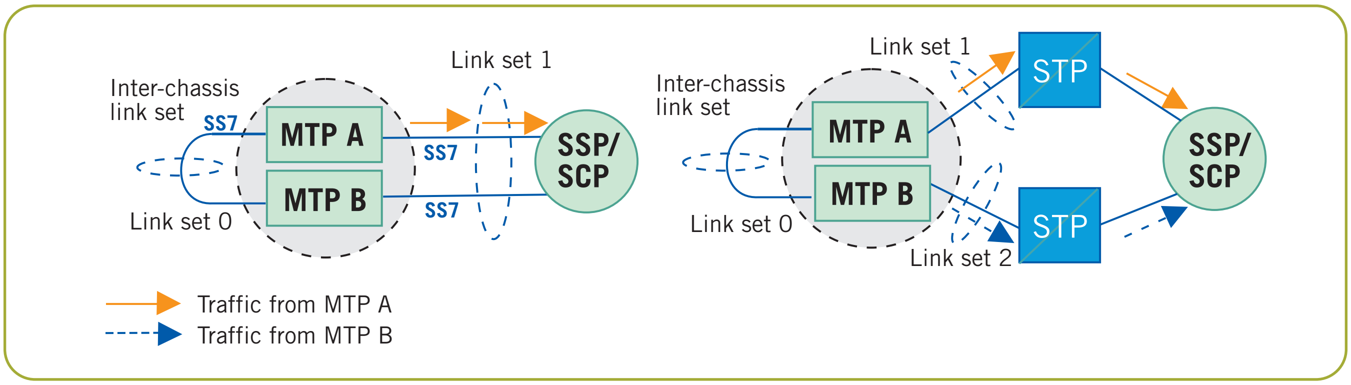

Figure 2. Normal Routing Status

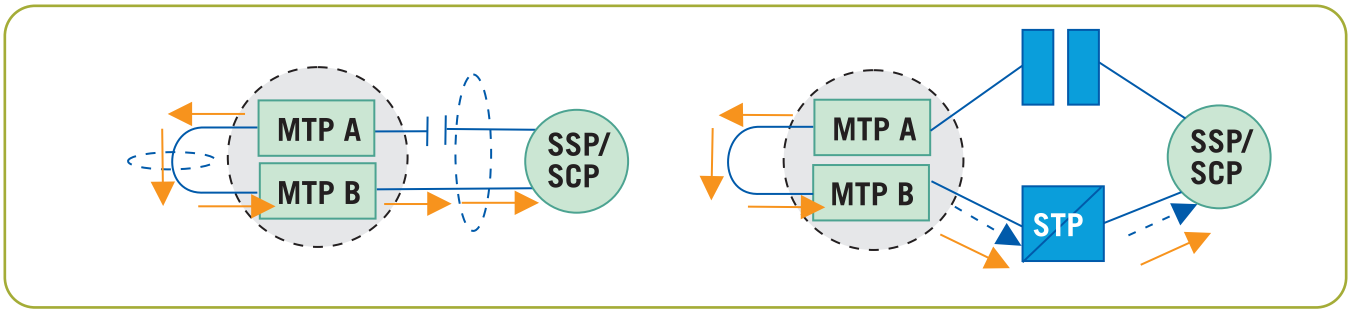

Figure 3. Routing Under Failure of All Links to MTPA

signaling. Use of SIGTRAN M2PA signaling enables an MTP inter-chassis link to be established over IP, without using the resources of SS7 signaling boards in each chassis.

Figure 1 indicates the relationship between the protocols of a dual resilient system.

Also, considerations are required to maintain state information and check pointing at the application layer, and are discussed later in this application note.

Concepts

Dual MTP3 Concepts

In order for two MTP3 processes running on separate hardware to maintain state information and optionally (under certain routing conditions) exchange transmit messages, a special "inter-chassis" link set is required between two systems behaving as a single point code. Figure 2 shows the routing scenarios for such a system, consisting of two halves, A and B, at the MTP3 layer.

Under normal operation, the available links are distributed evenly between MTPA and MTPB. Messages received for transmission by MTPA from the local layer 4 SS7 protocol (User Part) are sent from the links that connect it to the adjacent node. Similarly, messages received by MTPB are sent from its own links to the adjacent node.

When all of the links connecting either MTPA or MTPB fail, messages for transmission are passed over the inter-chassis link set that joins MTPA and MTPB for re-transmission by the other MTP over the available links (see Figure 3).

The link set joining MTPA and MTPB is configured using the same method as any other SS7 link and link set, with an additional data setting indicating that this "inter-chassis" link set should be treated differently from those connecting to adjacent nodes. The "inter-chassis" link set can be populated with TDM MTP2 or SIGTRAN M2PA signaling links.

Messages received by each MTP are delivered to the "local" layer 4 protocol (an assumption is made that local user parts are always available).

State Machines Above the MTP

The layer 4 parts of the SS7 protocol suite maintain state information for each call or transaction being processed. Two approaches are possible, one being to duplicate this

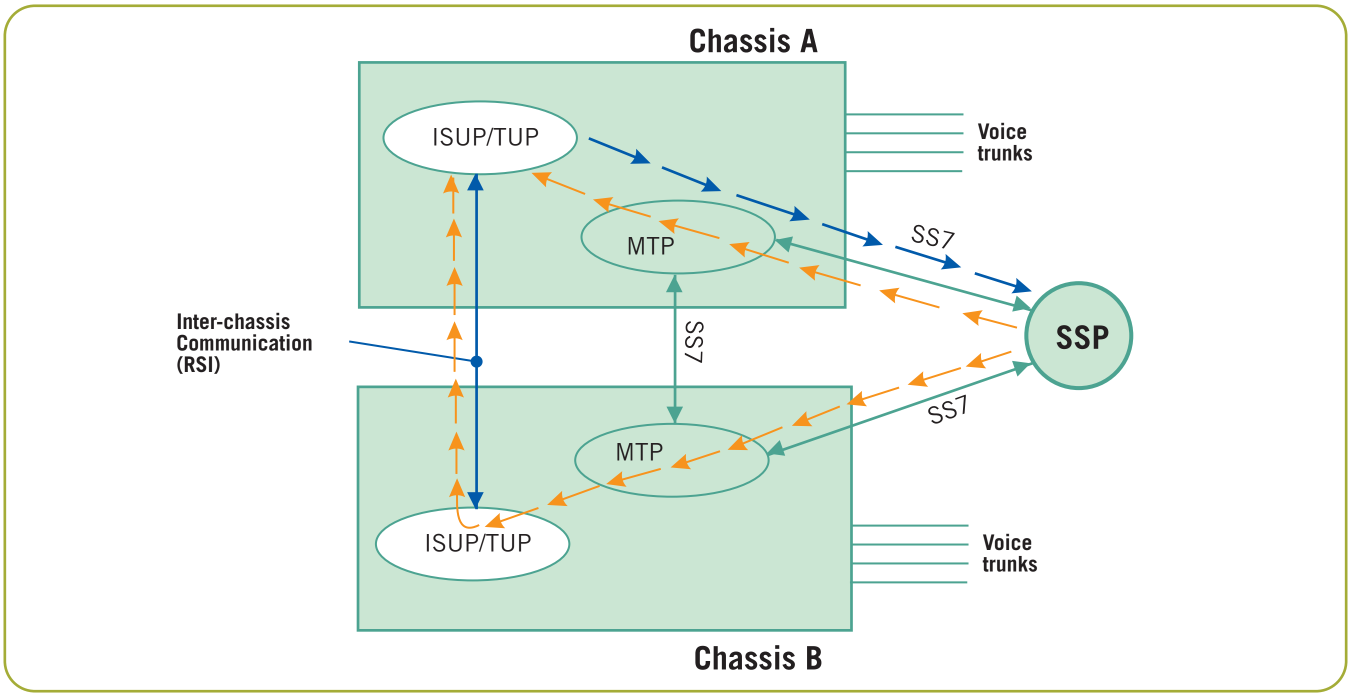

Figure 4. Dual ISUP Operation

state information across the N systems that constitute the signaling point, the second to partition the data, so that half of the state data is stored on each half of the system, such that failure of any one sub-rack reduces the system capacity by 1/N.

The first approach requires a reliable method of state data duplication between the N components of a fault resilient system. This approach has been found to be inefficient in that the N systems spend a large number of CPU cycles attempting to track each other. Since failure could occur at any point during the duplication process, it is difficult to guarantee that all systems contain the same data; therefore, the second, (2N), approach is used here.

Dual Telephony Operation (Based on CIC)

An SS7 system using the ISUP, TUP, or other national telephony layer 4 circuit switched control protocol can achieve fault tolerance by dividing the circuit terminations (each circuit being identified with a unique combination of OPC, DPC, and CIC) between two physical entities (which may be two sets of boards in the same chassis or two separate chassis). The following description discusses the use of two separate chassis.

Note: As an alternate system configuration, Appendix A provides more information on the ISUP fault tolerant approach used with the Dialogic® Signaling Interface Unit.

A dual ISUP/TUP system operates by having half of the circuit protocol state machines active on each chassis. Each ISUP/TUP layer sends transmit traffic through the local MTP3, which attempts to use local links between itself and the adjacent SS7 node for transmission, or fall back to the inter-chassis SS7 link set if all of the former have failed.

The remote SS7 end point is free to load share between any SS7 link that terminates on either chassis; therefore, there is no guarantee that a message received on one chassis will apply to a circuit that terminates on that chassis. In order to handle such a case, the ISUP/TUP layer pre-examines the CIC contained in each received message to determine if the circuit is known within the configuration. If not, the received message is sent (as an MTP3 transfer indication) to a task responsible for resolving messages received for unrecognized circuits. In a dual ISUP/TUP system, this task would be the other ISUP/TUP protocol running on the second chassis. The message is marked as having already been rejected by one of the ISUP/TUP halves, so that if the circuit is not recognized by the second ISUP/TUP protocol, it is treated as a message received for an unrecognized circuit and handled according to the ISUP/TUP protocol. This method protects the system from ISUP messages for unrecognized circuits being continually passed between chassis.

This process is shown in Figure 4.

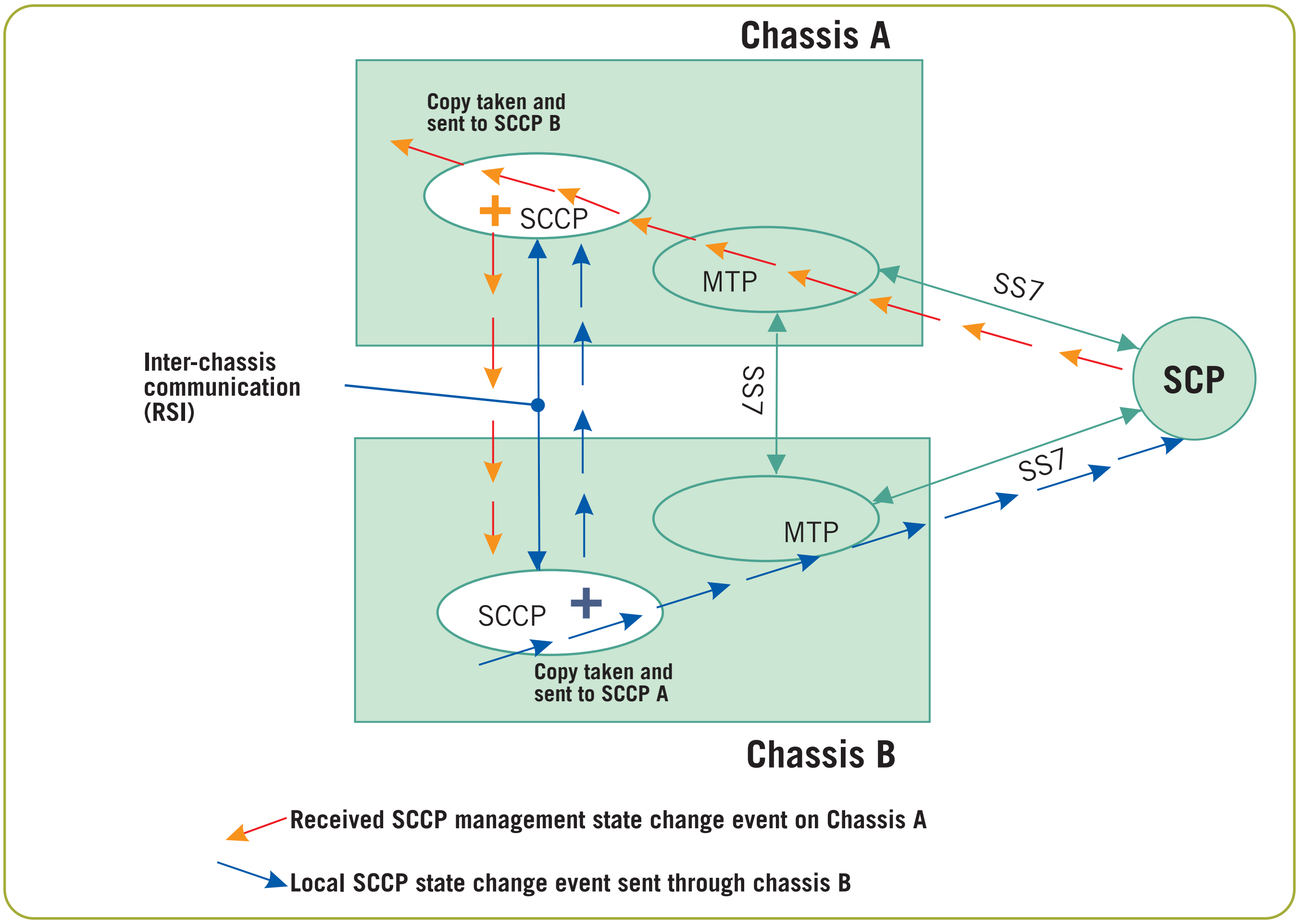
Chassis A
Copy taken and sent to SCCP B
SCCP
MTP
SS7
SS7
SCP
Inter-chassis communication (RSI)
SS7
MTP
SCCP
Copy taken and sent to SCCP A
Chassis B
Received SCCP management state change event on Chassis A
Local SCCP state change event sent through chassis B

Figure 5. Dual SCCP Operation

Dual SCCP Considerations (Maintenance of Sub-System States)

SCCP enhances the routing capabilities of the Message Transfer Part by supporting the concept of addressable sub-systems. The routing availability (the allowed or prohibited state) of all known local and remote sub-systems is maintained within the SCCP layer.

In a system consisting of N SCCP layers (or multiple instances of SCCP), behaving as the same point code, both the user application and the remote end points need to be able to:

- Exchange SCCP data messages through any of the SCCP instances
- Ensure routing availability tables in SCCP instances hold the same information

To do this, the SCCP layer has been enhanced with a broadcast mechanism that transmits any change of local or remote routing status to a broadcast task. The broadcast task is responsible for communicating the changes to the N-1 other SCCP instances. In a dual system, the broadcast task is simply the other SCCP layer, where both layers communicate using the message-based API. In a dual chassis/sub-rack environment, these messages can be conveyed by the RSI and TCP/IP Ethernet combination.

The local SCCP instance always delivers receive indications to the SCCP user task that exists on the same chassis (see Figure 5).

Dual TCAP Operation

A 2N configuration of the TCAP layer operates in a similar manner to the 2N ISUP/TUP protocols. The transactions managed by the system are divided equally between the two TCAP layers, each maintaining the state and components pending transmission for half of the transactions. Each transaction belongs permanently to one TCAP only.

For transactions initiated by the local TCAP user, this will be the TCAP that received the first transmit data request

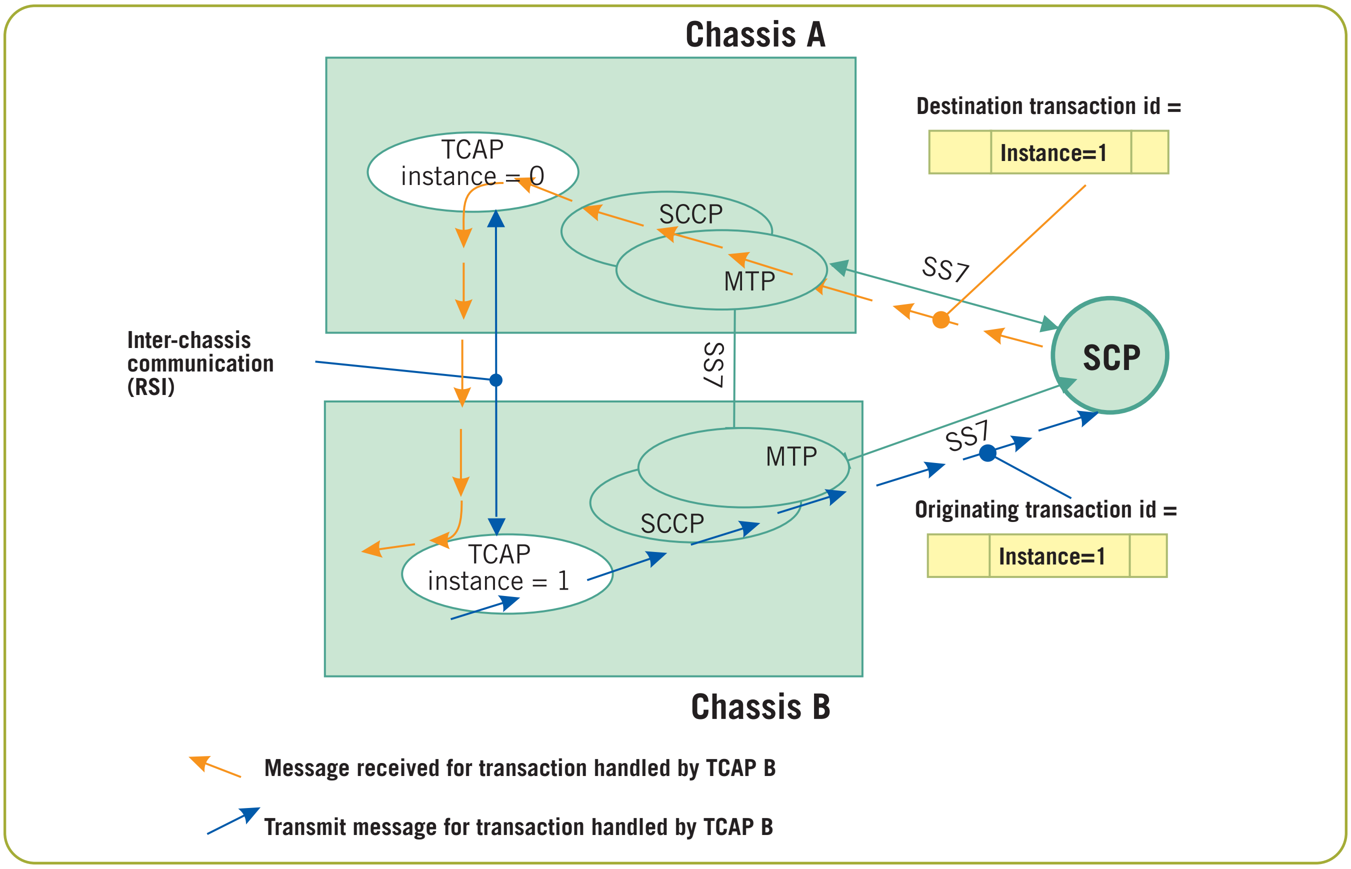
Chassis A
TCAP instance = 0
SCCP
MTP
Destination transaction id =
Instance=1
SS7
SCP
Inter-chassis communication (RSI)
SS7
MTP
SCCP
TCAP instance = 1
SS7
Originating transaction id =
Instance=1
Chassis B
Message received for transaction handled by TCAP B
Transmit message for transaction handled by TCAP B

Figure 6. Dual TCAP Operation

from the TCAP user. For transactions initiated by the remote TCAP entity, the transaction belongs to the TCAP module that received the first message of the transaction (BEGIN or QUERY) from the SCCP layer.

Load sharing for remotely initiated transactions is therefore defined by how the SS7 network distributes the first TCAP message (BEGIN or QUERY) between the SS7 links that terminate on the two system halves.

Dual TCAP Transaction Handling

To allow quick identification of the TCAP module that owns the transaction state machine for each received message, each TCAP module in a dual chassis system is assigned a unique logical identity or instance value (a number [ID]). This is encoded in the originating transaction ID for every transmit message and reflected in the destination transaction ID of each response from remote TCAP entities.

The receive processing of TCAP for a message other than a BEGIN or QUERY begins by recovering the instance bits of the transaction ID to rapidly determine if the message is being processed by the correct TCAP (the one that has an active state machine for this transaction). If not, message recovery is aborted and the message passed to the module configured to handle messages for that instance, as shown in Figure 6. Configuration information for dual chassis TCAP systems is described in the section called *Configuration of TCAP*.

Each TCAP passes receive information to TCAP user tasks on the same chassis. Transmit dialogue and component primitives sent by the local application program issued to the TCAP layer are always issued to the correct TCAP because there is a distinct application layer or user for each TCAP. Receive messages can be load shared over any of the MTP and SCCP layers on either system in a dual environment; therefore, it is possible for one of the TCAP layers to receive a TCAP message that applies to an active transaction handled by the other TCAP.

TCAP Users (GSM-MAP, IS41, CAMEL, INAP)

TCAP users, such as GSM-MAP, IS41, CAMEL, and INAP, are closely coupled to the TCAP layer that they use. In a multi TCAP system, the TCAP users load share transactions between themselves, half each, in a dual system. Resolution of receive messages to the correct TCAP state

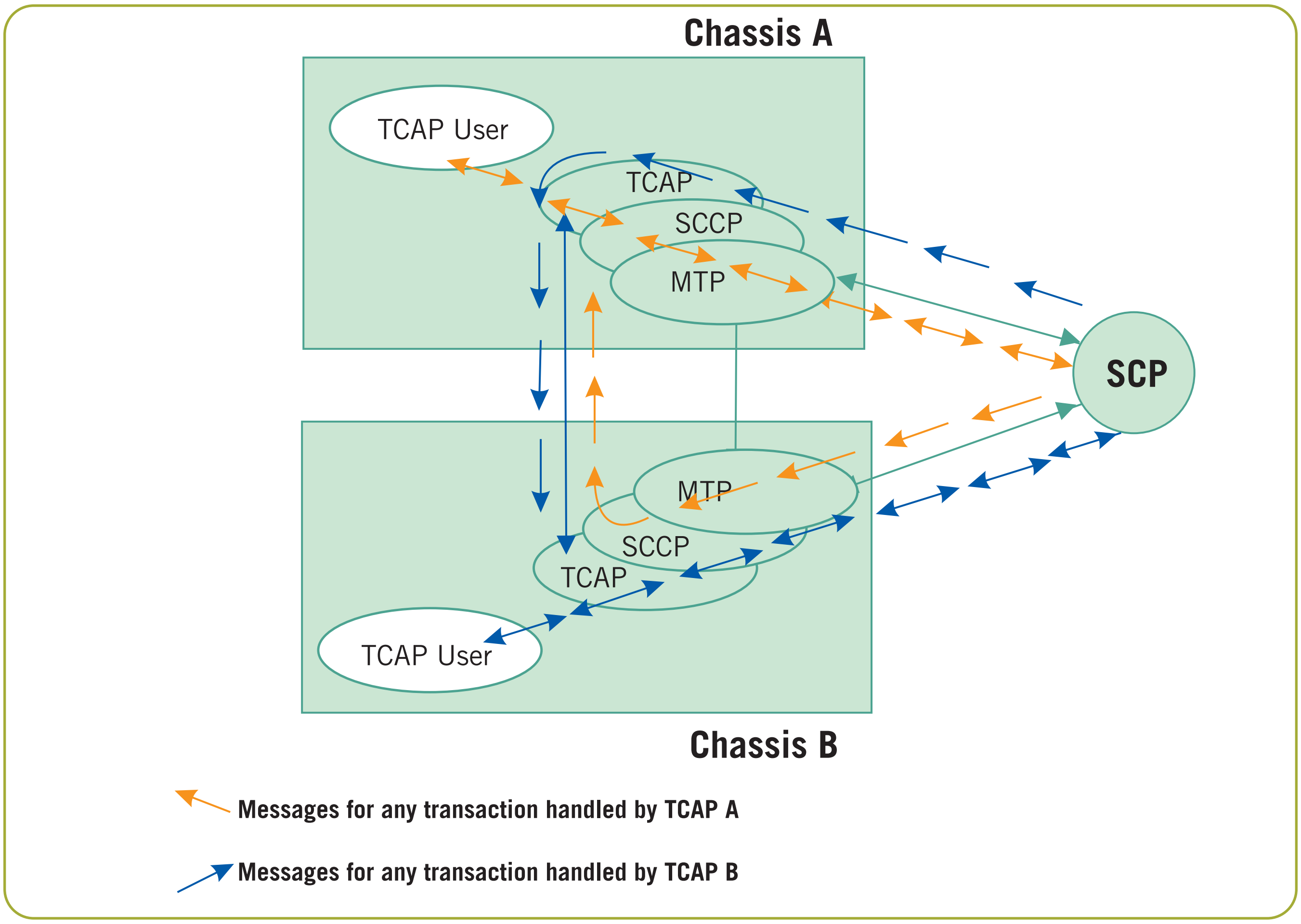
Chassis A
TCAP User
TCAP
SCCP
MTP
SCP
MTP
SCCP
TCAP
TCAP User
Chassis B
Messages for any transaction handled by TCAP A
Messages for any transaction handled by TCAP B

Figure 7. TCAP-User Processing

machine occurs at the TCAP level; therefore, no additional processing is required at the TCAP user layer (see Figure 7).

Configuration

Dialogic® SS7 Development Packages for Linux, Solaris, and Windows® operating systems support the configuration of dual chassis SS7 systems, and provide the necessary commands and parameters to configure the SS7 layers for dual operation. In a dual chassis system, each chassis is treated separately for configuration purposes and has a separate configuration file (config.txt).

Note: The configuration of SCCP, TCAP, and TCAP-User layers is achieved using discrete messages for dual chassis fault tolerant systems, either by embedding this functionality in the user's own program or by using the s7_play utility.

Configuration at MTP

In a dual MTP system, the links in a link set should be divided evenly between the two systems. The logical reference parameters, such as link_id and link set_id, should be numbered from 0 for both halves (each half of the system uses the same range of logical identifiers [tags/handles] for links and link sets).

"Inter-Chassis" Link Set Configuration

An additional "inter-chassis" link set is required to connect the two MTP3 platforms, enabling a single point code to be shared between the two. This is defined in the same way as a standard SS7 link set using the **MTP_LINKSET** configuration command noting the following:

- The same value for <local_pc> and <adjacent_pc> is used (both being set to the value of the point code being shared between the two platforms)
- This link set requires bit 15 of the <flags> parameter to be set to a 1 to indicate that the link set joins two MTP3 protocols with the same point code

Example to be used on both chassis:

```
MTP_LINKSET  <linkset_id>  <adjacent_spc>  <num_links>  <flags>  <local_spc> <ssf>
MTP_LINKSET       0             100             1        0x8000      100      0x8
```

Signaling links for the inter-chassis link set can be either TDM MTP links or SIGTRAN M2PA links.

Note: If SIGTRAN M2PA links are used for the inter-chassis link set, the MTP3 layer should be run as a host protocol.

Configuration of SIGTRAN M2PA Links for "Inter-Chassis" Link Set

SIGTRAN M2PA links can be added to an inter-chassis link set using the "SNSLI" (SIGTRAN Signaling Link Initiate) command.

Example SIGTRAN M2PA link initiate:

Chassis A

```
CNSYS:IPADDR=193.195.185.7,PER=0;
SNSLI:SNLINK=1,IPADDR=193.195.185.8,SNEND=C,SNTYPE=M2PA,M2PA=1,PPORT=3565;
```

Chassis B

```
CNSYS:IPADDR=193.195.185.8,PER=0;
SNSLI:SNLINK=1,IPADDR=193.195.185.7,SNEND=S,SNTYPE=M2PA,M2PA=1,PPORT=3565;
```

(Ensure IP addresses used between chassis refer to each other.)

Example MTP_LINK command to be used on both chassis:

```
* MTP_LINK  <link_id>  <linkset_id> <link_ref>  <slc>  <board_id>  <blink> <stream>  <timeslot>  <flags>
MTP_LINK        0           0           0         0        0          1       0          0     0x80000006
```

Note: If bit 31 of the flags field is set as in the above example, this is an M2PA link and the <blink> parameter is used to identify the SNLINK to be used.

Further information and examples of SIGTRAN M2PA configurations are given in the *Dialogic® Programmer's Manual for SIGTRAN Host Software* (see the *For More Information* section).

Configuration of MTP2 Links for Inter-Chassis Link Set

MTP2 signaling links can be added to an inter-chassis link set using the MTP_LINK command.

The following is an example MTP_LINK command to be used on both chassis:

Note: Example given below is for the Dialogic® SS7HD board. The SS7HD board requires a compound parameter for "blink" of the form "sp_id-sp_channel".

```
*
* MTP_LINK  <link_id>  <linkset_id> <link_ref>  <slc>  <board_id>  <blink> <stream>  <timeslot>  <flags>
MTP_LINK        0           0           0         0        0          0       0        0   16     0x0006
```

Configuration of MTP Routes

The routing for each destination (including the adjacent nodes) should specify two link sets. The primary **<primary_ls>** being the identifier of the link set that connects the local MTP to the adjacent node. The **<secondary_ls>** parameter should be set to identify the link set that joins the two MTP3 layers sharing the local point code. The <options> parameter should be set to 0x0001 to indicate that the secondary link set has been specified and should only be used to route transmit traffic if the route to the destination becomes unavailable through the primary link set.

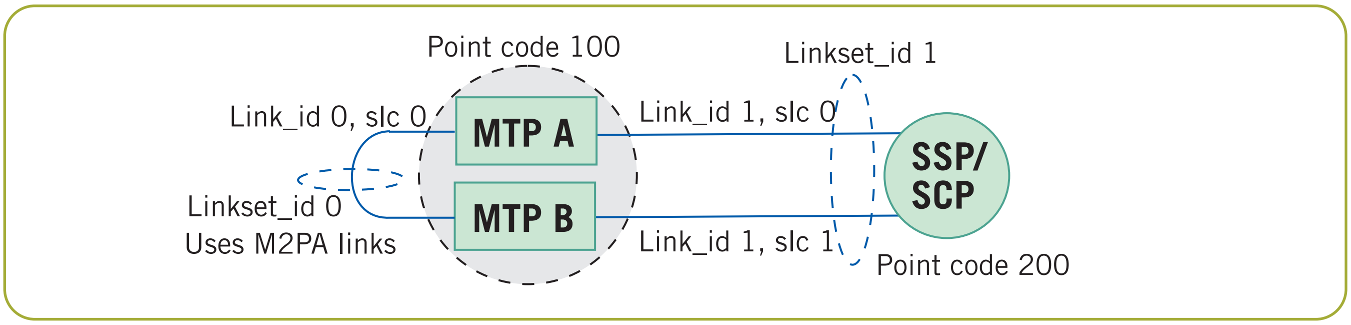

Figure 8. Sample MTP Configuration Direct Connection to an Endpoint

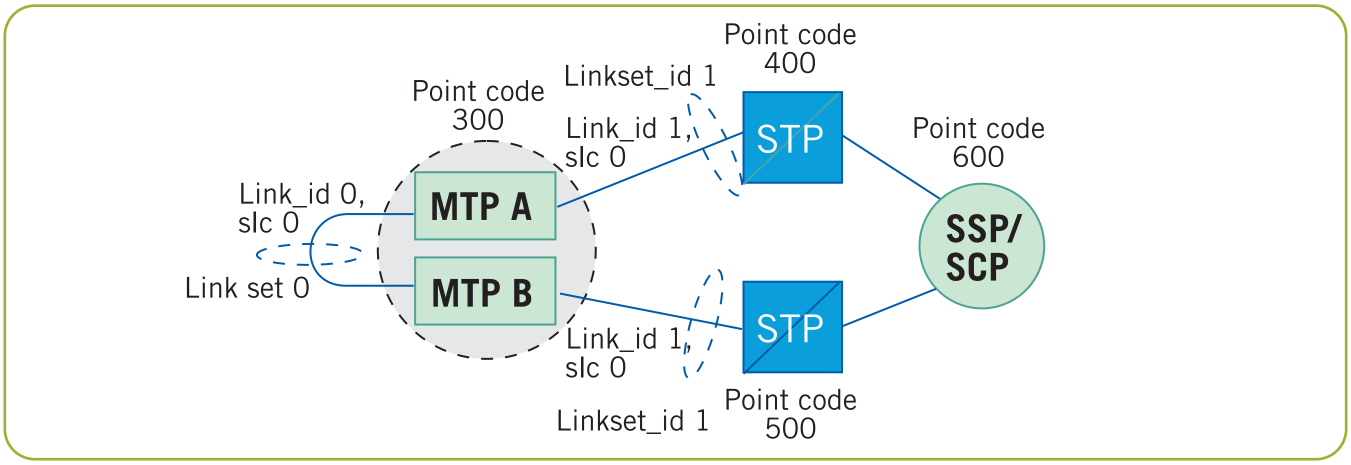

Figure 9. Sample MTP Configuration Connection to STPs

Example Dual Chassis MTP Configurations

Two examples are shown above (see Figures 8 and 9).

The following example gives configuration commands required to build the Dual Chassis MTP system shown in Figure 8, and includes configuration for an inter-chassis link set with 1 SIGTRAN M2PA signaling link.

For MTP A

```
*
CNSYS:IPADDR=193.195.185.7,PER=0;
SNSLI:SNLINK=1,IPADDR=193.195.185.8,SNEND=C,SNTYPE=M2PA,M2PA=1,PPORT=3565;
*
* MTP Parameters:
* MTP_CONFIG <reserved> <reserved> <options>
MTP_CONFIG  0 0  0x0000
*
* Define linksets:
* MTP_LINKSET <linkset_id> <adjacent_spc> <num_links> <flags> <local_spc> <ssf>
MTP_LINKSET 0 100 1 0x8000 100 0x8
MTP_LINKSET 1 200 1 0x0000 100 0x8
*
* Define signaling links:
* MTP_LINK <link_id> <linkset_id> <link_ref> <slc> <board_id> <blink> <stream>
<timeslot> <flags>
```

```
* (Note: For Septel ISA (PCCS6) boards the first LIU port is stream=16
* whilst for other boards the first LIU port is stream=0)
*
* MTP_LINK command for SIGTRAN M2PA interlink
MTP_LINK 0 0 0 0 0 1 0 0 0x80000006
*
* MTP_LINK command for SS7HD board
MTP_LINK 1 1 0 0 0 0-0 0 16 0x0006
*
* Define a route for each remote signaling point:
* MTP_ROUTE <dpc> <linkset_id> <user_part_mask>
MTP_ROUTE 100 0 0x0020
MTP_ROUTE 200 1 0x0020 0x0001 0
*
*
```

For MTP B

```
*
CNSYS:IPADDR=193.195.185.8,PER=0;
SNSLI:SNLINK=1,IPADDR=193.195.185.7,SNEND=S,SNTYPE=M2PA,M2PA=1,PPORT=3565;
* MTP Parameters:
* MTP_CONFIG <reserved> <reserved> <options>
MTP_CONFIG  0 0  0x0000
*
* Define linksets:
* MTP_LINKSET <linkset_id> <adjacent_spc> <num_links> <flags> <local_spc> <ssf>
MTP_LINKSET 0 100 1 0x8000 100 0x8
MTP_LINKSET 1 200 1 0x0000 100 0x8
*
* Define signaling links:
* MTP_LINK <link_id> <linkset_id> <link_ref> <slc> <board_id> <blink> <stream>
<timeslot> <flags>
* (Note: For Septel ISA (PCCS6) boards the first LIU port is stream=16
* whilst for other boards the first LIU port is stream=0)
*
* MTP_LINK command for SIGTRAN M2PA interlink
MTP_LINK 0 0 0 0 0 1 0 0 0x80000006
*
* MTP_LINK command for SS7HD board
MTP_LINK 1 1 0 1 0 0-0 0 16 0x0006
*
* Define a route for each remote signaling point:
* MTP_ROUTE <dpc> <linkset_id> <user_part_mask>
MTP_ROUTE 100 0 0x0020
MTP_ROUTE 200 1 0x0020 0x0001 0
*
*
```

Note that this does not include any commands to configure or define signaling boards. These should be defined as for a standard non-dual system. The MTP_ROUTE `<user_part_mask>` parameter was set for ISUP, (user part Service Indicator = 5) in the preceding examples.

The following example gives configuration commands required to build the Dual Chassis MTP system shown in Figure 9, and includes configuration for an inter-chassis link set with 1 MTP2 signaling link. (MTP_LINK commands are shown for SS7HD boards.)

For MTP A

```
MTP_CONFIG 0 0 0x0000
MTP_LINKSET 0 300 1 0x8000 300 0x8
MTP_LINKSET 1 400 1 0x0000 300 0x8
MTP_LINK 0 0 0 0 0 0-1 0 16 0x0006
MTP_LINK 1 1 0 0 0 0-0 0 16 0x0006
MTP_ROUTE 300 0 0x0020
MTP_ROUTE 400 1 0x0020 0x0001 0
MTP_ROUTE 600 1 0x0020 0x0001 0
```

For MTP B

```
MTP_CONFIG 0 0 0x0000
MTP_LINKSET 0 300 1 0x8000 300 0x8
MTP_LINKSET 1 500 1 0x0000 300 0x8
MTP_LINK 0 0 0 0 0 0-1 0 16 0x0006
MTP_LINK 1 1 0 1 0 0-0 0 16 0x0006
MTP_ROUTE 300 0 0x0020
MTP_ROUTE 500 1 0x0020 0x0001 0
MTP_ROUTE 600 1 0x0020 0x0001 0
```

Considerations for Dual Chassis MTP Systems

If MTP2 signaling links are used, sufficient links should be provided in the inter-chassis link set (between two MTP3 layers) to give sufficient signaling bandwidth to allow transmit traffic from one half to be routed through the other half. For M2PA links, there is no set signaling bandwidth limitation (as there is for 64 kb/s TDM "low speed" signaling links); therefore, one (or two M2PA links for link resilience) would be able to handle the traffic load between chassis.

Also note that in order for one of the MTP3 halves to begin transmitting traffic that would have been handled by its partner, each system half should not be loaded by more than 50% (both processing and signaling channel bandwidth) under normal operating conditions. SS7 systems are normally dimensioned to run at 20% to 40% of available bandwidth.

Inter-chassis links must be activated in the same manner as other links connecting the platform to remote SS7 nodes. This can be achieved using the mtpsl utility, or by issuing an MTP link or link set activation command message from the application program.

Configuration of ISUP

For ISUP, the module_id of the other ISUP protocol (on the other system) is specified by setting the optional <partner_id> parameter of the ISUP_CONFIG command. ISUP_CONFIG options bit ISPF_DUAL should set to indicate that ISUP should hand off any message received from MTP3 for an unrecognized circuit identity to this software task.

Note: Similarly, for TUP, the TUP_CONFIG <partner_id> should be set to the module_id assigned to the TUP module in the second chassis and the TUPF_DUAL options bit set.

The usual task identifier assigned to ISUP is 0x23, and the task identifier to send ISUP messages for unrecognized circuits is 0x73 for Side A and 0x63 for side B. The user should arrange for a LOCAL task to convey messages sent to these tasks to the ISUP protocol layer running on the other platform. This can be achieved by use of the RSI task and the REDIRECTION command as described in the section *Inter-Chassis Communication.*

Note: The normal task identifier assigned to TUP is 0x4a, and the task identifier to send TUP messages for unrecognized circuits is 0x93 for Side A and 0x83 for side B.

Note: When the ISUP protocol is running on the Dialogic® SPCI2S SS7 Interface Board, Dialogic® SPCI4 SS7 Interface Board, or Dialogic® SS7HDP SS7 Interface Board, the "flags" parameter of the SEPTELPCI_BOARD or SS7_BOARD command for side B must be set to include bit 9 (0x0200).

Example of Dual Chassis ISUP Module Configuration

The following are example ISUP configuration commands:

Side A

```
 * Configure ISUP module:
 ISUP_CONFIG <res1> <res2> <user_id> <options> <num_grps> <num_ccts>
[<partner_id>]
 ISUP_CONFIG  0  0  0x4d  0x0416  8  256 0x73
 *
```

Side B

```
 * Configure ISUP module:
 ISUP_CONFIG <res1> <res2> <user_id> <options> <num_grps> <num_ccts>
[<partner_id>]
 ISUP_CONFIG  0  0  0x4d  0x0416  8  256 0x63
 *
```

Dual Chassis Configuration of ISUP Circuit Groups

ISUP circuit groups can either be numbered starting at 0 for each system, or gaps may be left on each half where a circuit group exists on the other half. If the first method is used, the total system capacity is twice that of a single ISUP/TUP protocol layer; if the second method is used, the total capacity is that of a single ISUP/TUP protocol layer. For both cases, each half controls separate CIC ranges, as shown in Figures 10 and 11.

The system that does not duplicate the circuit groups between the two halves of the system (the "single capacity" system shown in Figure 11) has the ability to be able to activate (configure) the unused circuit groups on one of the halves if the other fails. This process is achieved using the xxx_MSG_CNF_GRP message to configure a circuit group. This provides the SS7 signaling resources and ISUP state machines for the circuits that have failed on the other unit.

However, if the circuit terminations themselves remain physically connected to the failed unit, then the only process the signaling is able to achieve for these circuits is hardware blocking. Section *Considerations for a Resilient Application* discusses issuing "hardware blocking" messages for circuit groups affected by chassis failure.

Configuration of SCCP

For dual chassis fault tolerant SCCP systems, the configuration of each SCCP instance is achieved using discrete messages, either by embedding this functionality in the user's own program or by using the s7_play utility.

At the SCCP layer, each SCCP protocol should be configured using the SCCP "Configuration Request" and "Configure Sub-System Resource" messages (SCP_MSG_CONFIG 0x7740, SCP_MSG_CNF_SSR 0x7741) with identical local sub-system, remote signaling point, and remote sub-system data.

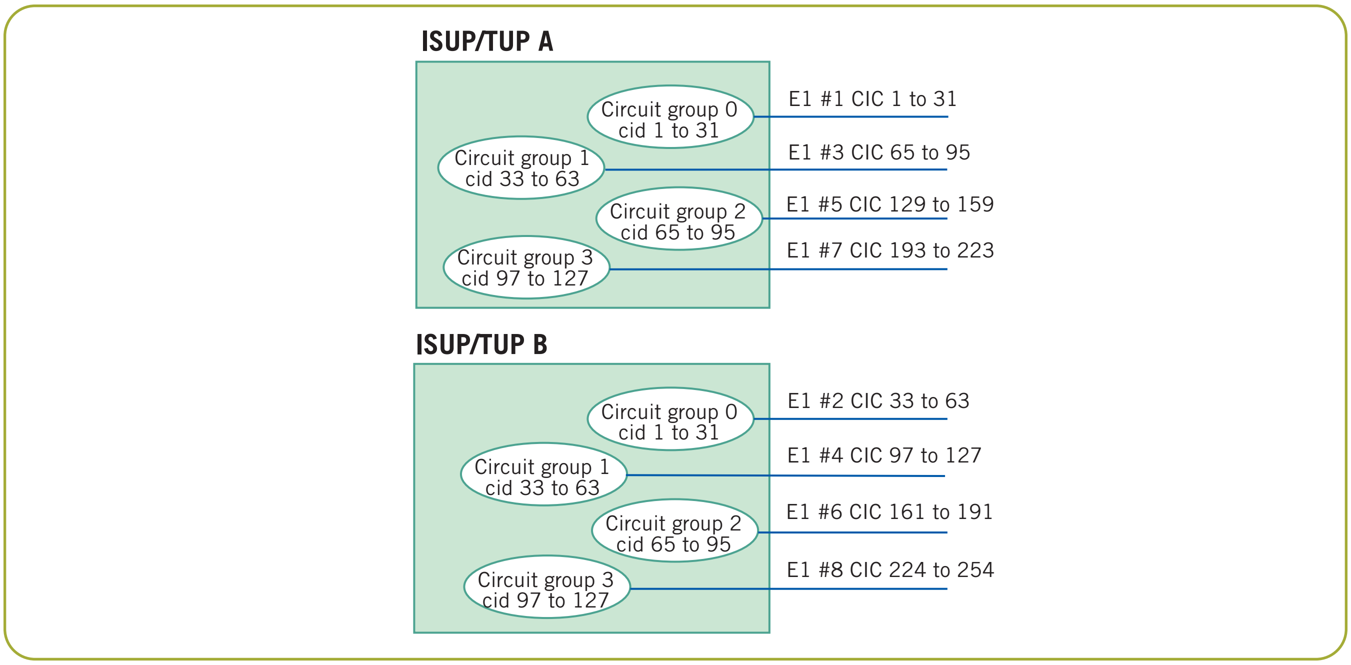
ISUP/TUP A
Circuit group 0 cid 1 to 31
Circuit group 1 cid 33 to 63
Circuit group 2 cid 65 to 95
Circuit group 3 cid 97 to 127
E1 #1 CIC 1 to 31
E1 #3 CIC 65 to 95
E1 #5 CIC 129 to 159
E1 #7 CIC 193 to 223
ISUP/TUP B
Circuit group 0 cid 1 to 31
Circuit group 1 cid 33 to 63
Circuit group 2 cid 65 to 95
Circuit group 3 cid 97 to 127
E1 #2 CIC 33 to 63
E1 #4 CIC 97 to 127
E1 #6 CIC 161 to 191
E1 #8 CIC 224 to 254

Figure 10. Circuit Group Distribution – Double Capacity

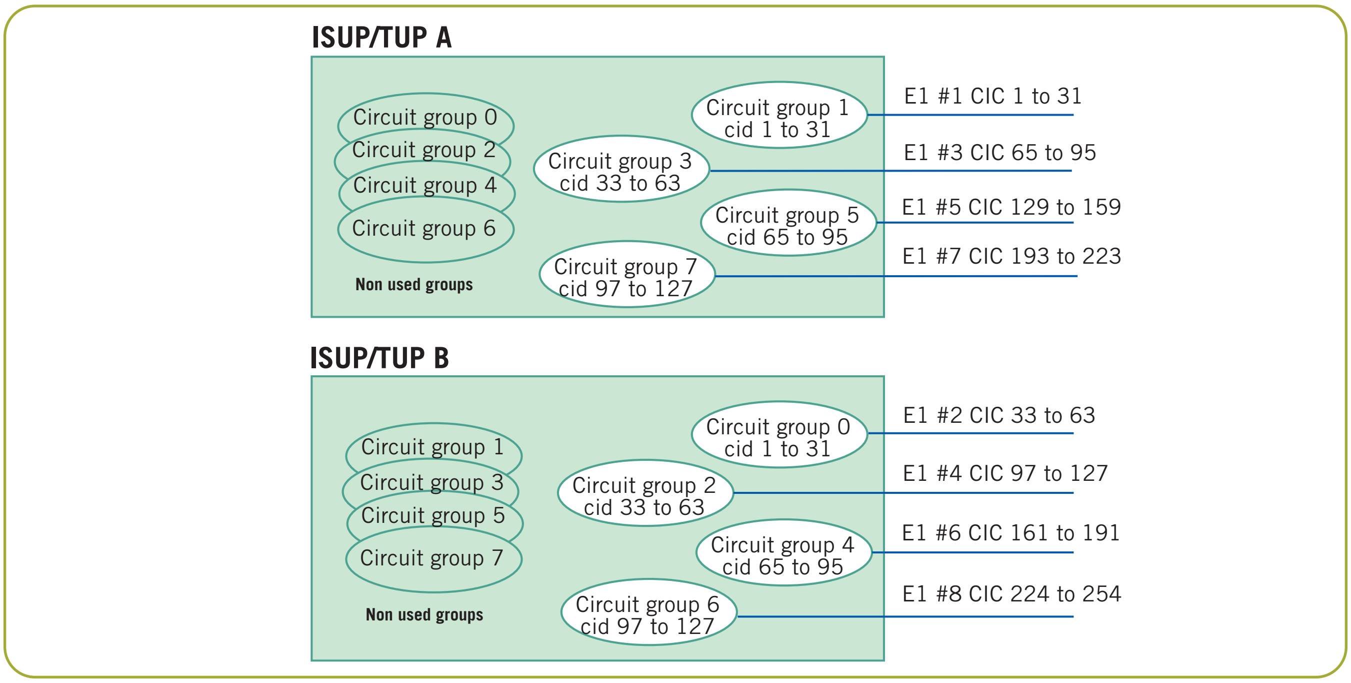

Figure 11. Circuit Group Distribution – Single Capacity

SCCP Configuration Request Notes

The **sccp_instance** parameter should be set uniquely (between 0 and 15) for the SCCP instance running on each chassis.

The SCCP Configuration Request option **SCPF_SMB** should be set to 1 to use the SCCP Management Broadcast (SMB) mechanism to communicate with other SCCP instances at the same signalling point.

The **smb_id** configuration parameter is used to identify the destination for status change updates, and should be set to the identity of a task that can convey this information to the other SCCP module in a dual system.

Generally, SCCP itself takes a task identifier of 0x33 and **smb_id** should be set to 0x53 on Side A and 0x43 on Side B. A REDIRECTION statement set in the system.txt file can then be used to route such messages to the RSI (or similar task), so that they are delivered to the other SCCP layer connected by an Ethernet. Section *Inter-Chassis Communication* gives more information on the use of RSI.

Notes on smb_flags Configuration

The SCCP **smb_flags** configuration parameter should be set to 0x101c. The following table provides information for this configuration:

smb_flags value	smb_flags Configuration Notes
0x0004	Broadcast sub-system **availability** to other SCCP instances at the same signalling point.
0x0008	Broadcast sub-system **status** to other SCCP instances at the same signalling point.
0x0010	Broadcast an indication to other SCCP instances that sub-system test message has been sent to sent to RSS.
0x1000	In the case where an SCCP module is started or restarted, upon reception of an MTP Resume for a remote point code (which matches that of remote sub-systems), a synchronization procedure is triggered. This is optionally enabled using the smb_flags value 0x1000. The procedure causes a message SCP_MSG_SMB_SSR (0x774d) to be sent to the SMB broadcast module ID.

Configuration of TCAP

For dual chassis fault tolerant TCAP systems, the configuration of each TCAP instance is achieved using discrete messages, either by embedding this functionality in the user's own program or by using the s7_play utility.

In a multi-TCAP environment, each TCAP is given a unique instance, this being encoded in the transaction ID at the SCCP boundary to allow quick resolution of the correct destination TCAP for receive messages.

The first TCAP is given an instance value of 0 and the next 1. The values are configured using the TCAP Configuration Request (TCP_MSG_CONFIG 0x7780) **tcap_instance** parameter.

The **tid_ninst** parameter controls how many bits in the transaction ID are used to encode the instance data. In a dual system, 1 bit is sufficient to distinguish between the two TCAPs.

Note: tid_ndref must be large enough to encode the highest dialogue_id value. In a system supporting 32768 simultaneous dialogues, a value of 15 or greater must be used.

The module identifier of the partner TCAP protocol is configured using the **TCP_MSG_S_TCI** message, which takes two parameters, the TCAP **instance** and module_id (this message is described in Appendix B).

The logical dialogue ID ranges can be set to the same for both TCAP halves (so that the two application processes running on the two systems both operate over the same dialogue_id range), or separate ranges can be used.

Inter-Chassis Communication

The Remote Socket Interface (RSI) software takes all messages in its own input queue that have a destination other than that of RSI itself and send those to a peer RSI task on the remote end of a TCP/IP Ethernet. The communication uses TCP/IP sockets, one side acting as a server, the other as a client.

At the receiver, the RSI takes messages received from the Ethernet and delivers them through the local message-passing system to the task identified in the original message destination (header dst field). If the communication between two RSI tasks over an Ethernet fails, messages passed to RSI for transmission over the Ethernet are discarded.

The two RSI tasks running on the system (one on each half) each take the same unique module ID, normally 0xb0. This must be declared in the system.txt file on both systems with a LOCAL definition and started with a **FORK_PROCESS** command. The RSI program takes its module ID as an optional command line parameter, prefixed by '-m', for example:

```
    ./rsi -m0xb0
```

Any message sent to the module ID assigned to RSI is processed by the local RSI task and not passed over the Ethernet.

The RSI links between the master and each slave host are activated using the rsicmd utility. The syntax for the rsicmd utility is as follows:

rsicmd <link_id> <conc_id> <link_type> <rem_addr> <rem_port> [**<rsi_id>**]

<link_id> — This is a logical identifier for this particular communication channel. RSI selects an outgoing channel by matching the message instance value (set by GCT_set_instance, zero by default) to the link_id value. For most dual systems, a single RSI connection between the two halves is sufficient, and this parameter should therefore be set to zero.

<conc_id> — This specifies a module ID that receives a message whenever the RSI link fails. This module should exist within the system, such that when these status messages are issued by RSI, they are received and then released by this module.

<link_type> and **<rem_addr>** should be set according to the following table:

Connection Type	rem_addr	link_type Value
Server	IP address of Side B	0 (client)
Client	0	1 (server)

One side of the system needs to be configured as Client, and the other as Server.

<rem_port> — This specifies the TCP/IP socket port that is used for the connection. Each RSI connection (which has a unique link_id) must take a unique port value, starting from 9000.

<rsi_id> — This is optional and identifies the RSI module for message passing.

Example RSI Link Configuration and Activation

The following shows example RSICMD commands for inter-chassis communication:

rsicmd <link_id> <conc_id> <link_type> <rem_addr> <rem_port> [<rsi_id>]

Chassis A

```
rsicmd     0     0xef     0     193.195.185.8     9000
```

Chassis B

```
rsicmd     0     0xef     1     193.195.185.7     9000
```

Accessing Partner Modules Using RSI

A **REDIRECT** statement is inserted in the system.txt file for destinations that are accessible through the RSI connection to the other half of the system (the module_id value given to the second ISUP, TUP, SCCP, and/or TCAP protocol).

For example, a dual ISUP system consists of two ISUP halves, each with a default module ID of 0x23. The configuration data assigned to ISUP on Side A indicates the other ISUP half is addressed as module_id 0x73; therefore, Side A to redirects messages sent to 0x73 through RSI. On Side B, a REDIRECT statement routes messages received by RSI on Side B for a destination 0x73 to the local ISUP, identified as 0x23. This redirect sequence is illustrated in Figure 12.

Where the protocol modules run on a signaling board, the REDIRECT statement at the receiver (the partner side) should redirect messages through the ssds or ssdh driver (normally taking the module_id 0x20), as shown in Figure 13.

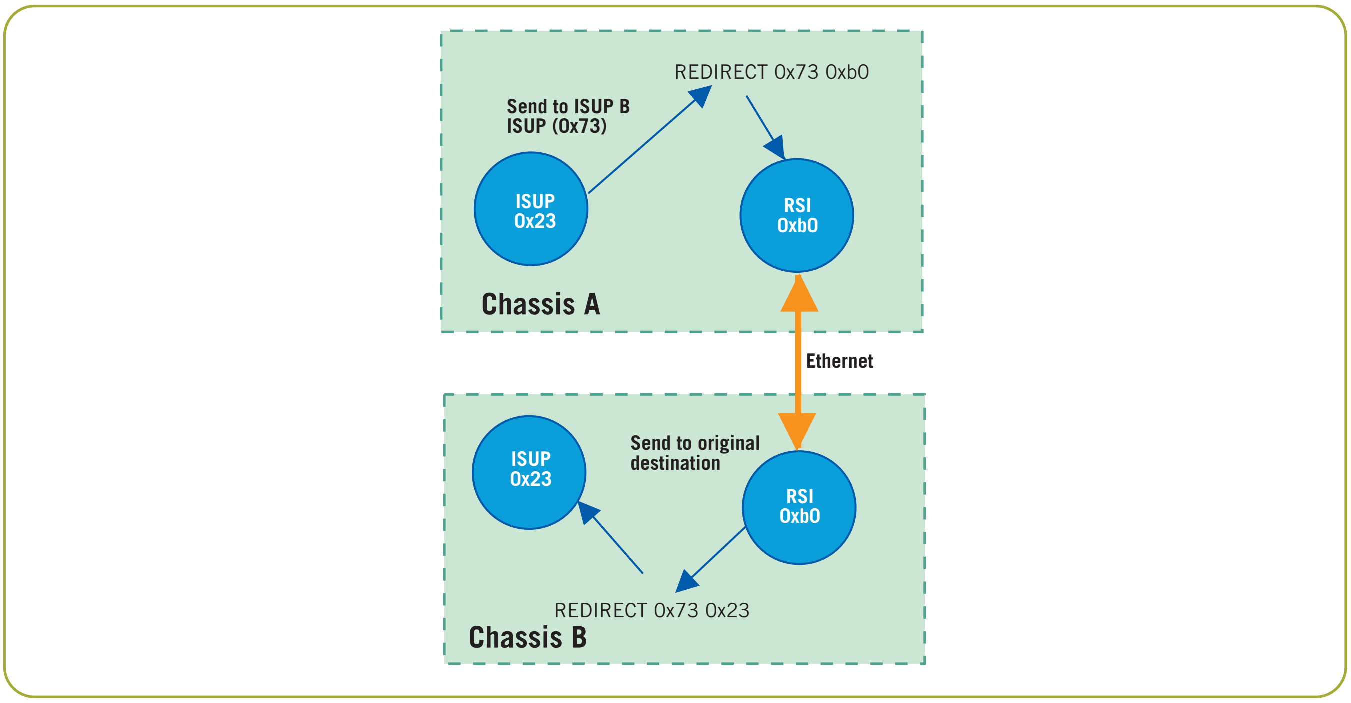

Figure 12. Use of REDIRECT and RSI for ISUP to ISUP Communication

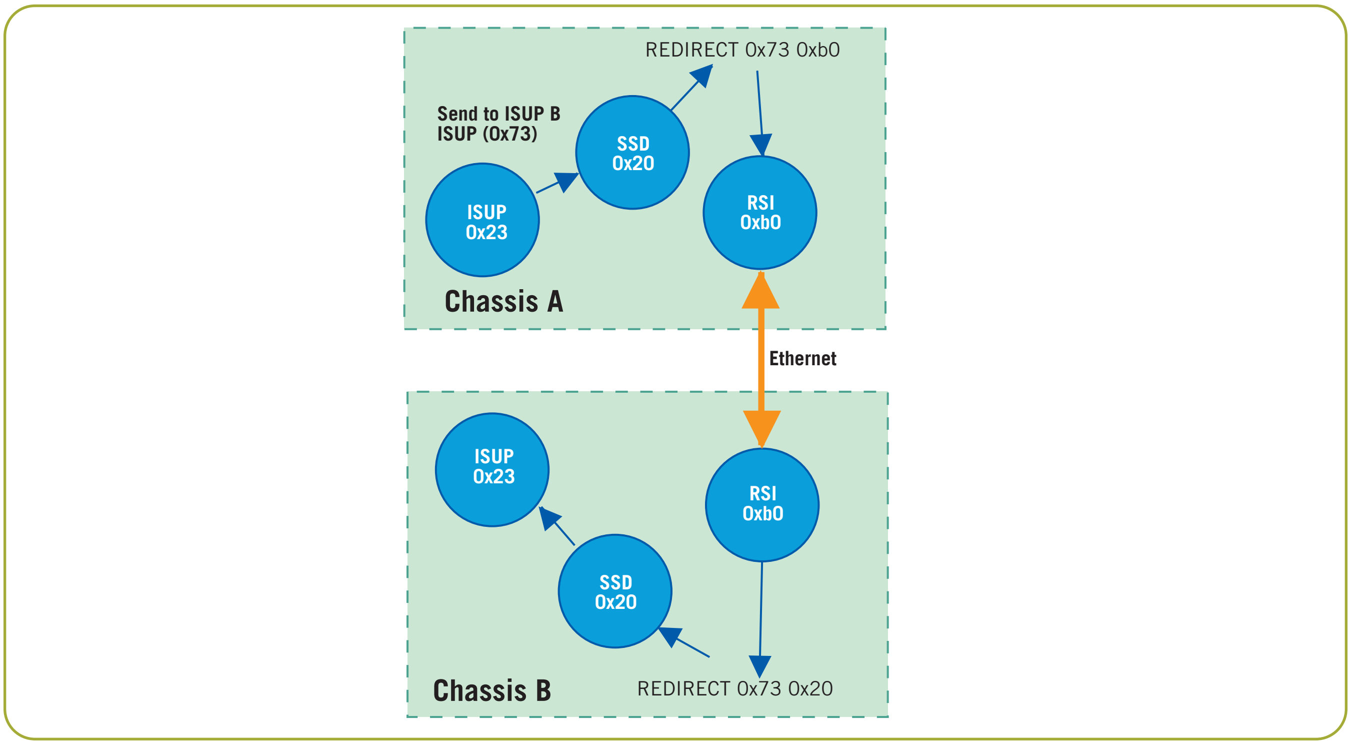

Figure 13. Use of REDIRECT with SSD

The following table gives the module_id values that should be used on both sides of the system:

Module	Side A Setting	Side B Setting
Local ISUP	0x23	0x23
Partner ISUP	0x73	0x63
Local TUP	0x4a	0x4a
Partner TUP	0x93	0x83
Local SCCP	0x33	0x33
Partner SCCP	0x53	0x43
Local TCAP	0x14	0x14
Partner TCAP	0x34	0x24
Local MAP	0x15	0x15
Local IS41	0x25	0x25
Local INAP	0x35	0x35

Considerations for a Resilient Application

In a dual system where the SS7 interface is embedded with the application, failure of the application or the SS7 components results in complete failure of that half of the system.

The applications that communicate with the SS7 protocols operate over the logical circuit ID and dialogue ID ranges pre-configured on each SS7 half. The two system halves can either use the same range of values since these are only private within each half, or the two halves can operate over different ranges.

Resilient Circuit Switching (ISUP) Applications

In a dual circuit switching application (using ISUP or TUP), the physical circuit terminations are distributed between the two chassis that constitute the system; therefore, failure of one of the halves results in the physical failure of one half of the circuits. In SS7 terms, hardware failure is normally handled by issuing hardware blocking (also known as blocking with release). This tears down the active calls on the affected circuits and indicates that these circuits should not be reselected for calls until an unblocking operation has occurred, which would be done once affected circuits had recovered.

However, in the configuration described earlier, the surviving half of the system has no knowledge of the failed circuits (the configuration data for these would be held in the failed half), and would not be able to issue hardware blocking. One method for solving this is to use the circuit group configuration method that leaves holes or gaps where a circuit group exists on the other half, as shown in Figure 11. Following a failure of one half, these “ghost” circuit groups may be configured on the surviving half, enabling hardware blocking to be issued for those circuits physically connected to the failed chassis.

It is also possible to physically isolate the application media/circuit processing and SS7 protocol state information, thereby enabling both applications to communicate with both SS7 interfaces (this is the approach taken by the SIU as described in Appendix A). Physical isolation may be achieved by the use of RSI and Ethernet.

If required, applications may check point each other to detect failures using the message-based API and RSI/Ethernet for communication. A user-specific message would need to be defined for this purpose.

Resilient TCAP Applications

In a dual TCAP system, failure of one half of the system results in loss of the TCAP state information (such as transaction state and any saved components pending transmission). In an Intelligent Networking (IN) environment, the call associated with this transaction should, if possible, be released. In a mobile/wireless environment, any associated pending mobile service (such as short messaging) times out and the operation is reported as failed. (This should then be reattempted on the surviving system half).

In additional feature of the Dialogic® SS7 Protocol Software for TCAP implementation called “TCAP Transaction State Replication” allows dialogues currently owned by a “master” TCAP module to be gracefully aborted by a “backup” TCAP module, either in case of failure or in case of planned maintenance of the system running the master TCAP module.

This feature can be used to ensure TCAP dialogues are

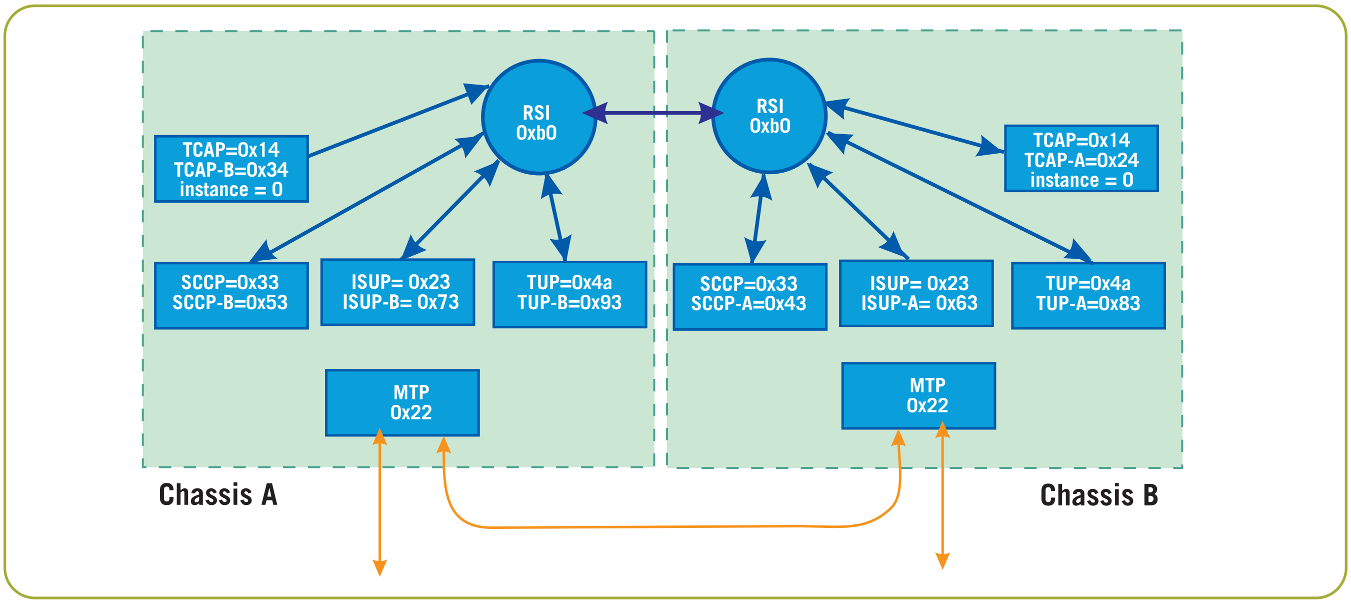

Figure 14. Modules in a Dual SS7 Protocol System

gracefully aborted in the case of a chassis failure, avoiding the issue of "hung" dialogue information being maintained indefinitely by a signaling application. More information is available in an application note entitled, *TCAP Transaction State Replication,* available by requesting it from your local Dialogic Support contact.

Setting system.txt Values

Figure 14 shows the relationship between the modules in a dual ISUP/TUP/SCCP/TCAP system, alongside the appropriate entries in the config.txt files for both halves of the system.

System.txt for Protocols Running on the Host

Side A

```
REDIRECT 0x34 0xb0 * TCAP to chassis B
REDIRECT 0x53 0xb0 * SCCP to chassis B
REDIRECT 0x73 0xb0 * ISUP to chassis B
REDIRECT 0x93 0xb0 * TUP to chassis B
*
REDIRECT 0x24 0x14 * TCAP from chassis B
REDIRECT 0x43 0x33 * SCCP from chassis B
REDIRECT 0x63 0x23 * ISUP from chassis B
REDIRECT 0x83 0x4a * TUP from chassis B
```

Side B

```
REDIRECT 0x24 0xb0 * TCAP to chassis A
REDIRECT 0x43 0xb0 * SCCP to chassis A
REDIRECT 0x63 0xb0 * ISUP to chassis A
REDIRECT 0x83 0xb0 * TUP to chassis A
*
REDIRECT 0x34 0x14 * TCAP from chassis A
REDIRECT 0x53 0x33 * SCCP from chassis A
REDIRECT 0x73 0x23 * ISUP from chassis A
REDIRECT 0x93 0x4a * TUP from chassis A
```

The system.txt file must also contain LOCAL definitions for all local SS7 protocols running on the host, such as ISUP, TUP, SCCP, or TCAP (if any), plus any application tasks, configuration utilities (such as s7_mgt), and debug utilities. There should also be FORK_PROCESS statements to start the appropriate drivers and support tasks, as described in the appropriate programmer's manual.

System.txt for Protocols Running on a Signaling Board

Side A

```
REDIRECT 0x34 0xb0 * TCAP to chassis B
REDIRECT 0x53 0xb0 * SCCP to chassis B
REDIRECT 0x73 0xb0 * ISUP to chassis B
REDIRECT 0x93 0xb0 * TUP to chassis B
*
REDIRECT 0x24 0x20 * TCAP from chassis B through ssd
REDIRECT 0x43 0x20 * SCCP from chassis B through ssd
REDIRECT 0x63 0x20 * ISUP from chassis B through ssd
REDIRECT 0x83 0x20 * TUP from chassis B through ssd
```

Side B

```
REDIRECT 0x24 0xb0 * TCAP to chassis A
REDIRECT 0x43 0xb0 * SCCP to chassis A
REDIRECT 0x63 0xb0 * ISUP to chassis A
REDIRECT 0x83 0xb0 * TUP to chassis A
*
REDIRECT 0x34 0x20 * TCAP from chassis A through ssd
REDIRECT 0x53 0x20 * SCCP from chassis A through ssd
REDIRECT 0x73 0x20 * ISUP from chassis A through ssd
REDIRECT 0x93 0x20 * TUP from chassis A through ssd
```

The system.txt file must also contain LOCAL definitions for all application tasks, configuration utilities (such as s7_mgt), and debug utilities. There should also be FORK_PROCESS statements to start the appropriate drivers and support tasks, as described in the appropriate programmer's manual.

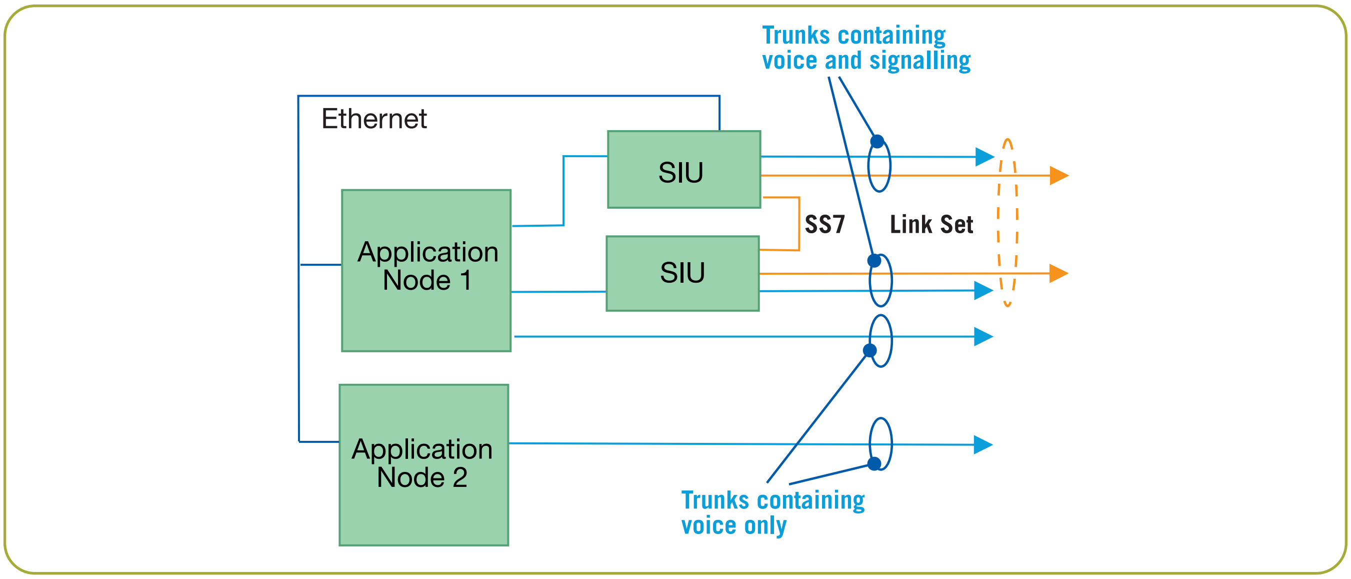
Trunks containing voice and signalling
Ethernet
SIU
SIU
Application Node 1
Application Node 2
SS7
Link Set
Trunks containing voice only

Figure 15. SIU Approach

Appendix A: SIU Fault Tolerance

Description

The SIU resolves the issue of fault resilience by separating the SS7 interface from the processing of the media using an Ethernet, with two SIUs behaving as a single point code for fault tolerance, as shown in Figure 15. By separating the SS7 interface, the SIU allows systems to be built using multiple application nodes to process the media (voice or 3G-324m video) circuits. A Dialogic application note *Building Fault-Tolerant SS7 Systems Using the Dialogic® SS7G2x Signaling Servers with the SIU Option* is available from the Dialogic website (see the *For More Information* section).

Appendix B: TCAP Set Instance Module ID Message

Description

This message configures the module_id for a specific TCAP instance. This module_id is used by TCAP as the destination when sending any message received from the network with the specified instance encoded in the transaction ID.

As described in section *Configuration of TCAP*, for side A, the local TCAP instance is 0, and the remote is 1; therefore, this message should be used to set the module_id of the remote TCAP, instance 1 to module_id 0x34. On Side B, the local TCAP instance is 1 and this message should be used to set the module_id of instance 0 (the remote TCAP as viewed from Side B) to 0x24.

Message Format

MESSAGE HEADER		
FIELD NAME		**MEANING**
Type		TCP_MSG_S_TCI_ID (0x5794)
Id		Instance
src		Sending module_id
dst		TCP_TASK_ID (0x14)
rsp_req		Used to request a confirmation
class		0
status		0
err_info		0
len		1
PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	1	module_id

Parameter Description

Instance

The instance value set in the transaction ID of any received message that should be sent to the specified module_id.

module_id

The module ID to be used as the message destination when TCAP sends messages for the specified instance.

Acronyms

CAMEL	Customized Applications for Mobile network Enhanced Logic
CIC	Circuit Identification Code
IN	Intelligent Networking
INAP	Intelligent Network Application Part
ISUP	ISDN User Part
M2PA	MTP2 Peer To Peer Adaptation Layer
MAP	Mobile Application Part
MTP2	Message Transfer Part Layer 2
RSI	Remote Socket Interface
SSD	System Seven Driver
SIU	Signal Interface Unit
SS7	Signaling System 7
TDM	Time Division Multiplexing
TUP	Telephony User Part

For More Information

Dialogic® Programmer's Manual for SIGTRAN Host Software — http://www.dialogic.com/support/helpweb/signaling/software3.htm

Building Fault-Tolerant SS7 Systems Using the Dialogic® SS7G2x Signaling Servers with the SIU Option — http://www.dialogic.com/goto/?9206

To learn more, visit our site on the World Wide Web at **http://www.dialogic.com**.

Dialogic Corporation
9800 Cavendish Blvd., 5th floor
Montreal, Quebec
CANADA H4M 2V9