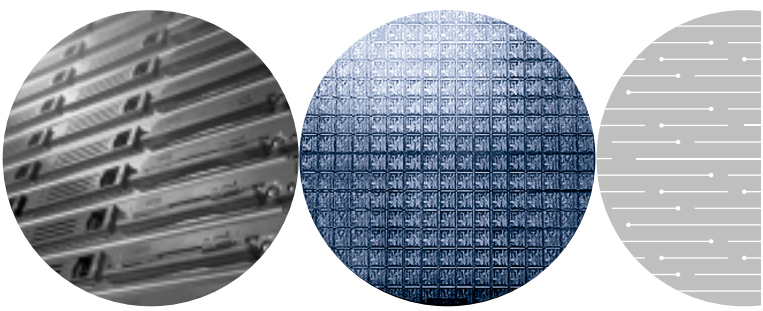
Application Note

Enabling Dual Chassis Fault Tolerance with Intel® NetStructure™ SS7 Boards

Intel in Communications



Contents

Abstract

In order to achieve “five nines” availability and a high degree of fault tolerance in an SS7 environment using Intel® NetStructure™ SS7 components, an SS7 end point spread over two chassis can be built. Splitting the functionality of a signaling point by implementing an SS7 node over more than one chassis isolates the hardware processors on the chassis from each other. This separation lets one processor continue if the other fails, allowing the total system to remain in service.

Intel NetStructure SS7 components are designed for this dual processor approach and provide the architecture for splitting a point code over two active SS7 protocol engines. With this technique, the links in an SS7 link set can be spread between two separate chassis when Intel NetStructure SS7 signaling cards are installed in each. This application note discusses the design and implementation of such a dual-chassis method for achieving high availability and fault tolerance.

Introduction

Because of the high expectation of service reliability by the users of public telephony networks, equipment manufacturers and system integrators demand high levels of fault tolerance and availability, often citing the ‘five nines’ for availability (requiring a system to be operational for 99.999% of the time).

Such systems need to be able to continue to offer service even when partial hardware or software failure has occurred. There are several, well-known methods of achieving this type of reaction to partial failure in the signaling component of communications networks, including:

- Multiple signaling paths (SS7 links and link sets) to each end point
- The distribution of these paths through independent interfaces and cabling
- Distribution of the processing of SS7 terminations at a single signaling point between multiple processing cards in a single chassis
- Physical isolation and duplication of the SS7 interface for a single signaling point (the Intel® NetStructure™ SIU and SS7 boards approach)
- Splitting the functionality of a signaling point, including the application layer, between two chassis (the subject of this paper)

The first three on this list can be achieved in SS7 by implementing multiple links (64 or 56 Kbps channels) between two adjacent inter-communicating points. (By definition, these links will all be in the same link set). The final two options can be achieved by the use of two independent but co-operating SS7 protocol stacks, normally in separate but interconnected chassis to provide a higher degree of resilience to failures.

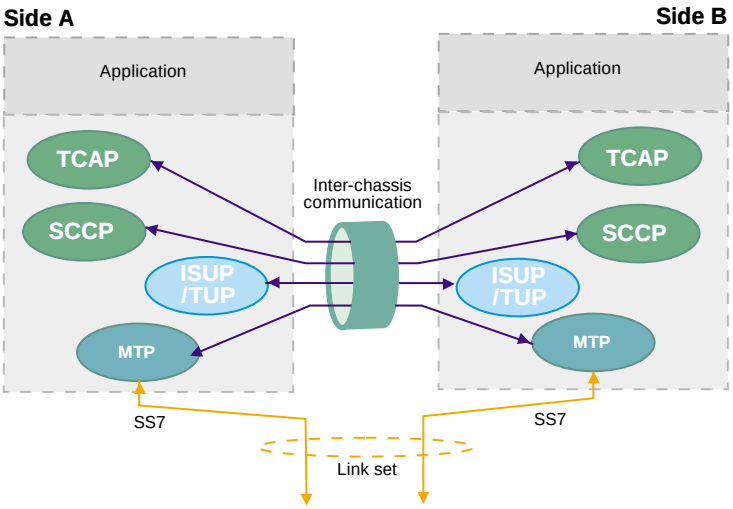


Figure 1 Dual resilient SS7 protocols

Implementing an SS7 Signaling Point on More Than One Chassis

Implementing an SS7 node over more than one chassis isolates one hardware processor from the failure of another, allowing the total system to continue service on complete failure of one of the component processors or chassis. Just as with general fault resilient architectures, there are several methodologies for achieving such a configuration, each with benefits and disadvantages.

The Intel® NetStructure™ SS7 signaling boards take a 2N approach and provides the ability to split a point code over two active SS7 protocol engines. This allows the links in an SS7 link set to be spread between two separate chassis, with SS7 signaling cards installed in each.

Two communication paths need to be provided between the two halves of the system, one at the MTP layer, using the MTP2 data link as a transport protocol, and the second at the layer 4 or 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 the RSI (Resilient Socket Interface) software provided by Intel. This combination provides an SS7 interface and protocol processing split between two physically separate chassis that appear to the network as a single entity.

The diagram (Figure 1) indicates the relationship between the protocols of a dual resilient system.

There are also considerations required to maintain state information and check pointing at the application layer, which are discussed in later sections of this paper.

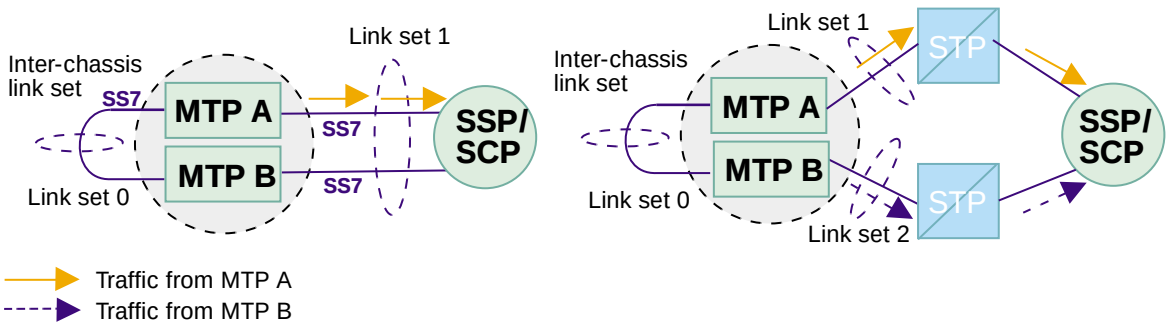


Figure 2 Normal routing status

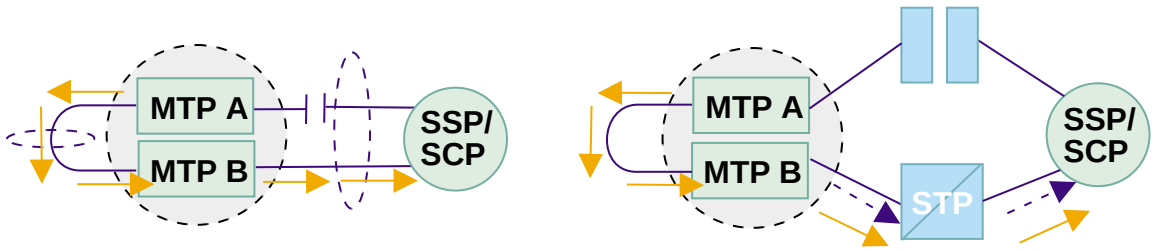


Figure 3 Routing under failure of all links to MTPA

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 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 MTP A and MTP B. Messages received for transmission by MTP A 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 MTP B are sent from its own links to the adjacent node.

When all of the links connecting either MTP A or MTP B fail, messages for transmission are passed over the link set that joins MTP A and MTP B for re-transmission by the other MTP over the available links. See Figure 3.

The link set joining MTP A and MTP B is configured using the same method as any other SS7 link and link set, with an additional data setting indicating that this link set should be treated differently from those connecting to adjacent nodes.

Messages received by each MTP are always delivered to the ‘local’ layer 4 protocol. An assumption is made that the local user part(s) is (are) always available.

State Machines above the MTP

The layer 4 parts of the SS7 protocol maintain state information for each call or transaction being processed. Two approaches are possible: one is to duplicate this state information across the N systems that constitute the signaling point, and the second is 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 will reduce 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.

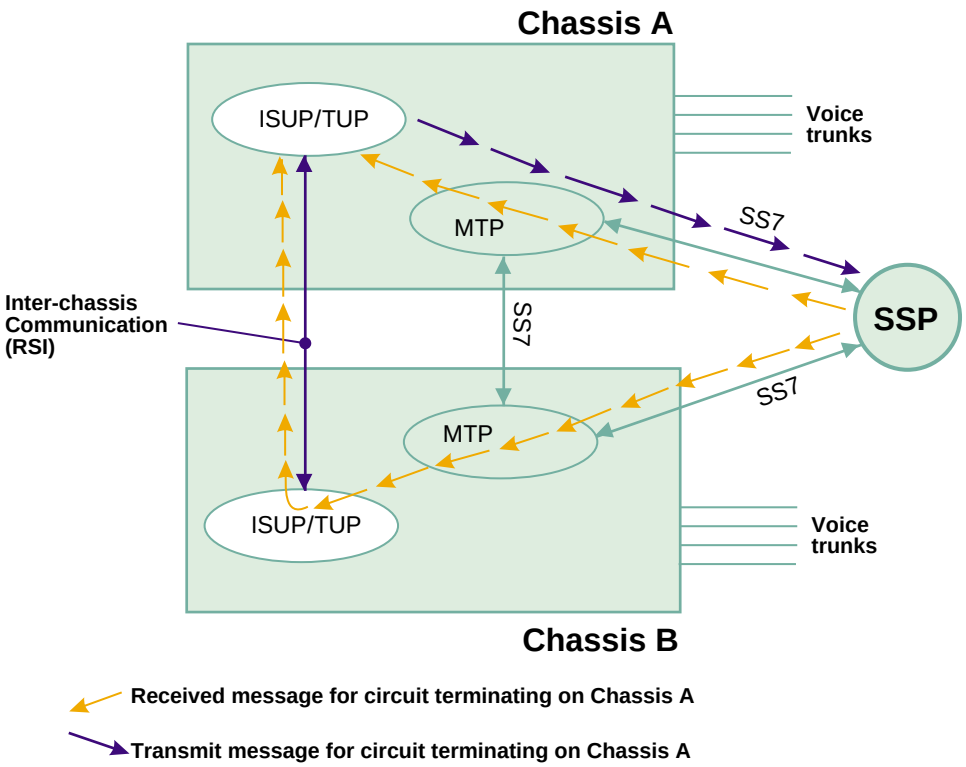


Figure 4 Dual ISUP operation

Since failure could occur at any point during the duplication process, it is difficult to guarantee that all systems contain the same data, hence the second (2N) approach is taken 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 cards in the same chassis or two separate chassis). The following description discusses two chassis, although the same approach can be used for two sets of cards that process SS7 and media.

A dual ISUP/TUP system operates by having half of the circuit protocol state machines active on each chassis. Each ISUP/TUP layer sends all transmit traffic through the local MTP3, which will attempt to use local links between itself and the adjacent SS7 node for transmission, or fall back to the inter-chassis SS7 link 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; hence, 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, therefore, protects from messages being continually passed from one system to another if not recognised by both.

This process is shown in Figure 4.

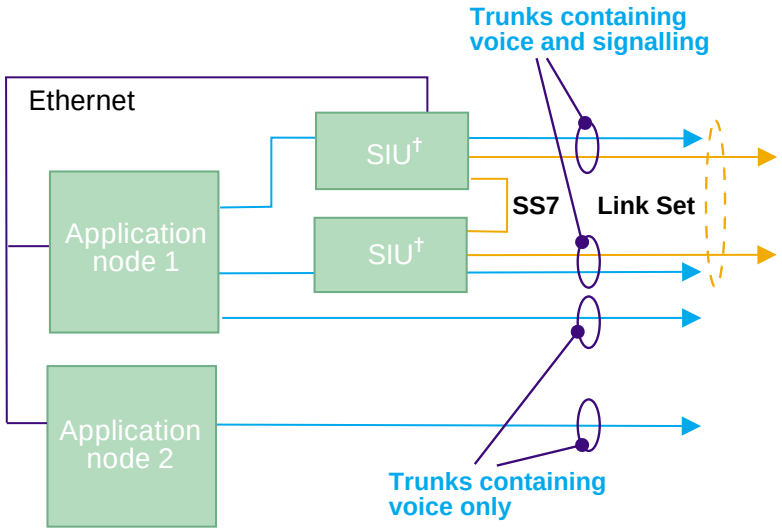


Figure 5 SIU Approach

The message itself is transported from one system to another through an application dependent mechanism, although the RSI and TCP/IP method is possibly the simplest.

The SIU solves the problem 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 5. By separating the SS7 interface, the SIU also allows systems to be built using up to 32 application nodes that process the voice circuits (media).

Dual SCCP Considerations (Maintenance of Sub-System States)

SCCP enhances the routing capabilities of the Message Transfer Part (MTP) 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 and expect the routing availability tables in each SCCP to be the same. In order to achieve 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 with both layers communicating 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.

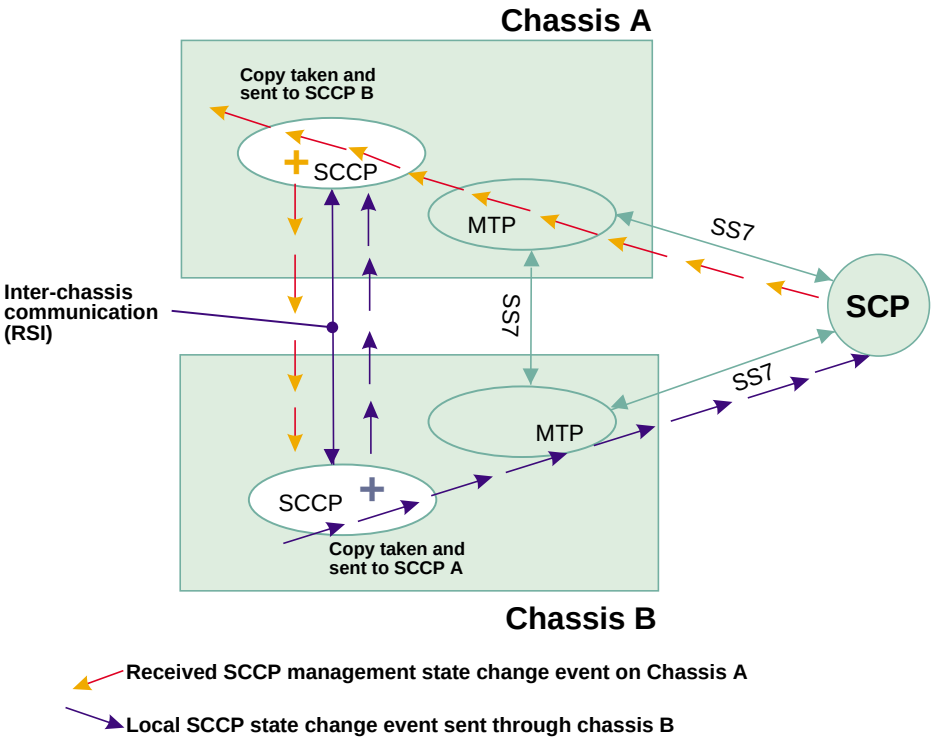


Figure 6 Dual SCCP operation

The SCCP always delivers receive indications to the SCCP user task that exists on the same chassis. See Figure 6.

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

To allow quick identification of the TCAP that owns the transaction state machine for each receive message, each of the TCAP protocols in an mN system is assigned a unique logical identity or instance value (a number). 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 all the TCAP layers for any message other than a BEGIN or QUERY begins by recovery of 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, the message recovery is aborted and the message passed to the module configured to handle messages for that instance as shown in Figure 7.

The module identifier of the other TCAP protocol is configured using the **TCP_MSG_S_TCI** message, which takes two parameters, the TCAP **instance** and **module_id.** (This message is detailed in Appendix A.) For side A, the local TCAP

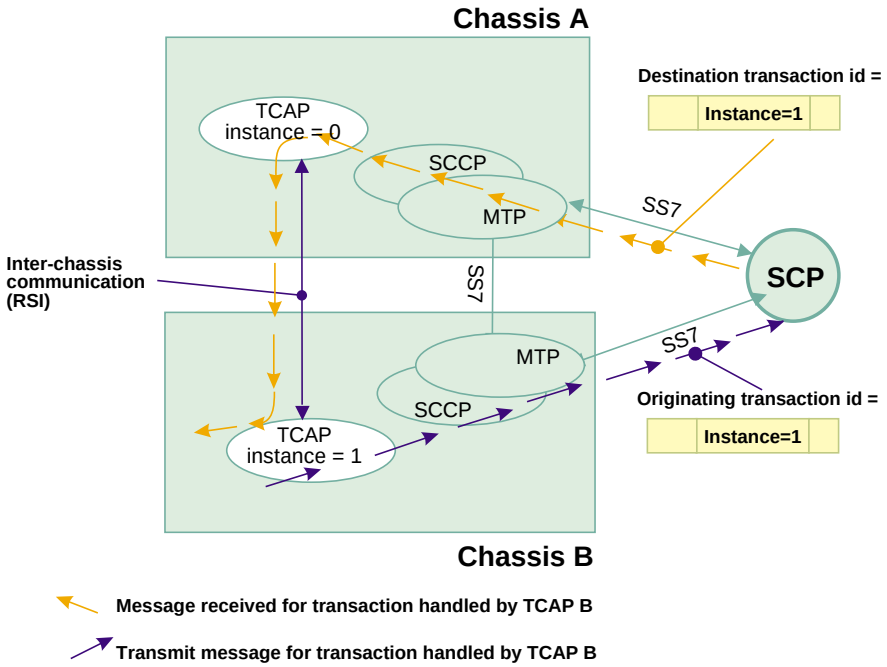


Figure 7 Dual TCAP operation

instance is 0, and the remote is 1; hence, 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 the TCP_MSG_S_TCI_ID message should be used to set the module_id of instance 0 (the remote TCAP as viewed from Side B) to 0x24.

Each TCAP passes all 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 by virtue of there being 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; hence, 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-MAP, INAP)

TCAP users, such as GSM-MAP, IS41, and INAP, are closely coupled to the TCAP layer that they use. In a multi-TCAP system, the TCAP users load share the transactions between themselves, half to each in a dual system. Resolution of receive messages to the correct TCAP state machine occurs at the TCAP level; hence, no additional processing is required at the TCAP user layer. See Figure 8.

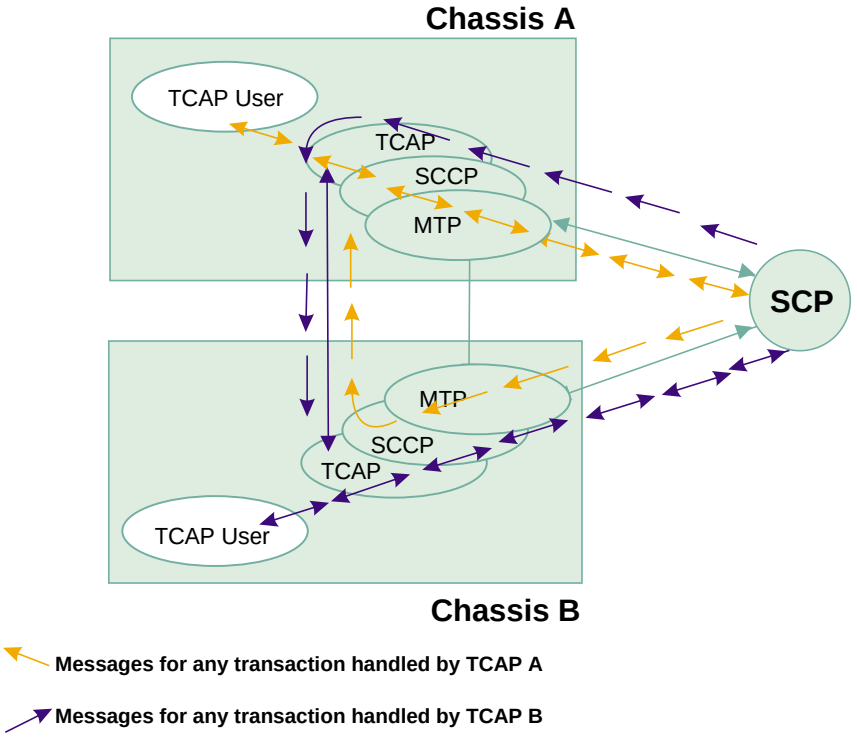


Figure 8 TCAP-User processing

Configuration

Intel development packages for Linux V2.00 and Windows V2.00 support the configuration of dual-chassis SS7 systems, and provide the necessary commands and parameters to configure the MTP, ISUP, TUP, and NUP layers for dual operation. The configuration of the SCCP, TCAP, and TCAP-User layers is achieved in exactly the same way that a single (non-fault resilient) system is configured — using discrete messages, either by embedding this functionality in the user's own program or by using the s7_play utility.

In a dual-chassis system, each half is treated separately for configuration purposes; there are separate configuration files for each.

Configuration at MTP3

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 linkset_id should be numbered from 0 for both halves. Each half of the system will use the same range of logical identifiers (tags/handles) for links and link sets.

An additional 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 and using the same value for **<local_pc>** and **<adjacent_pc>** (both are 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 this link set joins two MTP3 protocols with the same point code. Links are added to the link set using the **MTP_LINK** command.

The routing for each destination (including the adjacent nodes) should specify two link sets. The primary **<primary_ls>** is 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

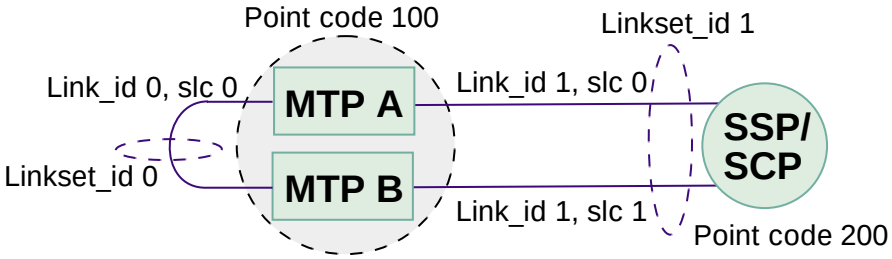


Figure 9 Sample MTP configuration direct connection to an endpoint

unavailable through the primary link set. All other parameters have the same meaning as in single MTP configurations, as documented in the appropriate
user manual.

Two examples are shown. See Figures 9 and 10.

For MTP A

```
* 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 Septel cP / PCI boards the first LIU port is stream=0)
MTP_LINK 0 0 0 0 0 1 0 0 0x4006
MTP_LINK 1 1 0 0 0 0 16 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:

```
* 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 Septel cP / PCI boards the first LIU port is stream=0)
MTP_LINK 0 0 0 0 0 1 0 0 0x6006
MTP_LINK 1 1 0 1 0 0 16 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 cards. This should be defined as if for a standard non-dual system. The MTP_ROUTE **<up_enable>** parameter was set for ISUP, user part SI = 5 for the example above.

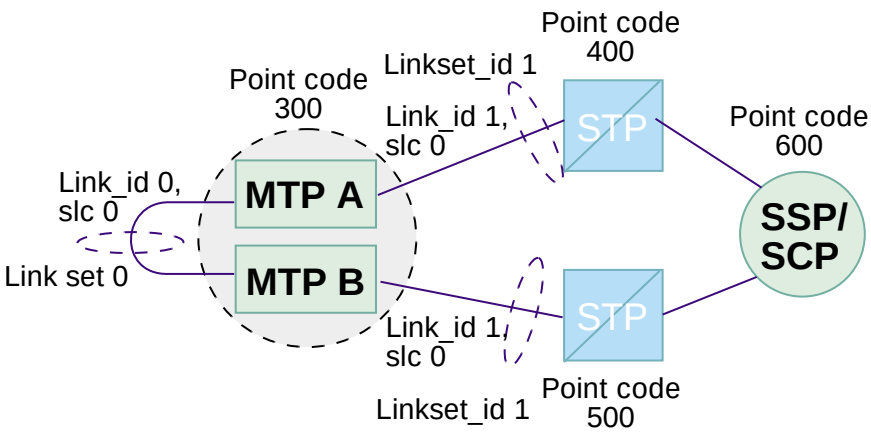


Figure 10 Sample MTP configuration connection to STPs

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 1 0 0 0x4006
MTP_LINK 1 1 0 0 0 0 16 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 1 0 0 0x6006
MTP_LINK 1 1 0 1 0 0 16 16 0x0006
MTP_ROUTE 300 0 0x0020
MTP_ROUTE 500 1 0x0020 0x0001 0
MTP_ROUTE 600 1 0x0020 0x0001 0
```

Sufficient links should be provided in the link set that joins the two MTP3 layers to give sufficient signaling bandwidth to allow all of the transmit traffic from one half to be routed through the second. The designer should 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 loaded at 20 to 40% maximum.

The 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.

3.2 Configuration of ISUP and TUP

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 be set to indicate that ISUP should hand off any message received from MTP3 for an unrecognized circuit identity to this software task.

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 normal task identifier assigned to ISUP is 0x23. For the task of sending ISUP messages for unrecognized circuits, the identifier is 0x73 for Side A and 0x63 for side B. The normal task identifier assigned to TUP is 0x4a. For the task of sending TUP messages for unrecognized circuits, the identifier is 0x93 for Side A and 0x83 for side B. The user should arrange for a LOCAL task to convey messages sent to these tasks to the ISUP or TUP protocol layer running on the other platform. This can be achieved simply by use of the RSI task and the REDIRECTION command as described later.

Note: Running ISUP and TUP on the Intel® NetStructure™ CPM8, SPCI2S, or SPCI4 board.

When the ISUP or TUP protocols are running on the Intel NetStructure SS7 board, the l1_flags parameter of the SEPTEL_CP command for side B must be set to include bit 9 (0x0200).

The following shows example ISUP and TUP configuration commands:

Side A

```
* Configure ISUP module:
* ISUP_CONFIG <reserved> <reserved> <reserved> <options> <num_grps> <num_ccts>
[<partner_id>]
ISUP_CONFIG  0  0  0  0x0435  4  64 0x73
*
* Configure TUP Parameters:
* TUP_CONFIG <reserved> <reserved> <reserved> <options> <num_grps> <num_ccts>
[<partner_id>]
TUP_CONFIG  0  0  0  0x8141  4  64  0x93
*
```

Side B

```
* Configure ISUP module:
* ISUP_CONFIG <reserved> <reserved> <reserved> <options> <num_grps> <num_ccts>
[<partner_id>]
ISUP_CONFIG  0  0  0  0x0435  4  64 0x63
*
* Configure TUP Parameters:
* TUP_CONFIG <reserved> <reserved> <reserved> <options> <num_grps> <num_ccts>
[<partner_id>]
TUP_CONFIG  0  0  0  0x8141  4  64  0x83
*
```

The circuit groups can either be numbered starting at 0 for both systems, 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 will be twice that of a single ISUP/TUP protocol layer; if the second, the total capacity will be that of a single ISUP/TUP protocol layer. For both cases, each half will control separate CIC ranges, as shown in Figures 11 and 12.

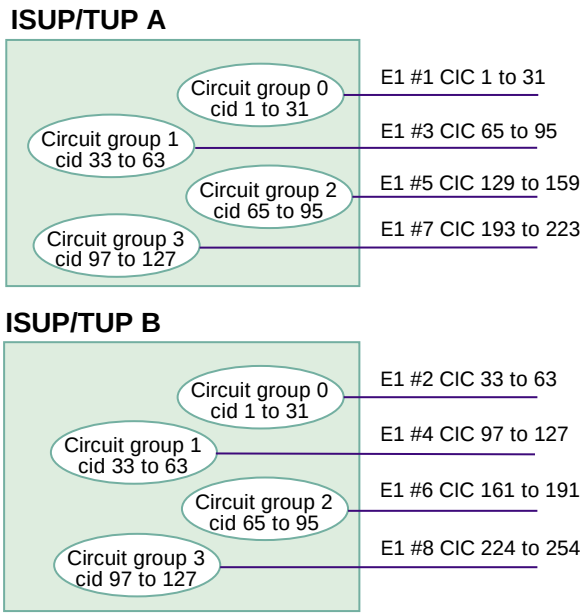


Figure 11 Circuit group distribution – double capacity

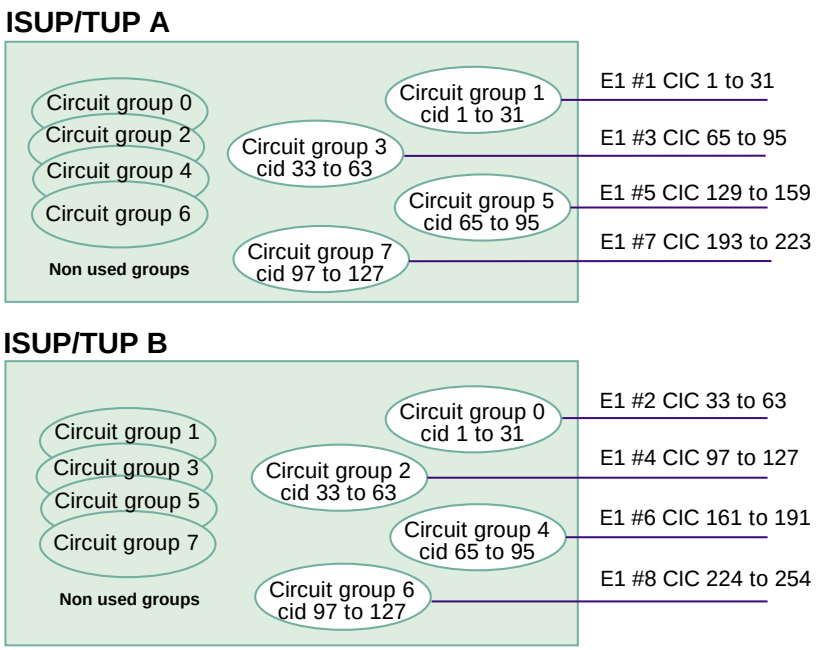


Figure 12 Circuit group distribution – single capacity

The system that does not duplicate the circuit groups between the two halves of the system (the “single capacity” system shown in Figure 12) has the ability 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 will provide the SS7 signaling resources and ISUP or TUP 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, the only process that the signaling will be able to achieve for these circuits is hardware blocking.

Configuration of SCCP

At the SCCP layer, each SCCP protocol should be configured with identical local sub-system, remote signaling point, and remote sub-system data.

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. In most circumstances, 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. In addition, the SCCP **smb_flags** configuration parameter should be set to 0x001c.

Configuration of TCAP

In a multi-TCAP environment, each TCAP is given a unique instance, which is 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 normally given an instance value of 0 and the next 1.

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 that tid_ndref must be large enough to encode the highest dialogue_id value. In a system supporting 2048 simultaneous dialogues a value of 11 or greater must be used).

The logical dialogue_id ranges can be 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 Resilient Socket Interface (RSI) software takes all messages in its input queue that have a destination other than that of the RSI itself and sends 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 and the other as a client.

At the receiver, 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) 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 will be 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 shown below:

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

<link_id> 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, which is zero by default) to the link_id value. For most dual systems, a single RSI connection between the two halves will be sufficient, and this parameter should, therefore, be set to zero.

<conc_id> specifies a module ID that will receive notification (a message) whenever the specified RSI connection fails. This can be set to direct these indications to the application task or a local management event viewer, such as s7_log.

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

Connection type	rem_addr	ink_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, the other as server.

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

<rsi_id> Is optional and identifies the RSI module for message passing.

A **REDIRECT** statement must be inserted in the system.txt file for all 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 will consist 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; hence, there will be a redirection on Side A to redirect any messages sent to 0x73 through RSI. On Side B, there should be a REDIRECT statement to route all messages received by RSI on Side B for a destination 0x73 to the local ISUP, identified as 0x23. This is illustrated in Figure 13.

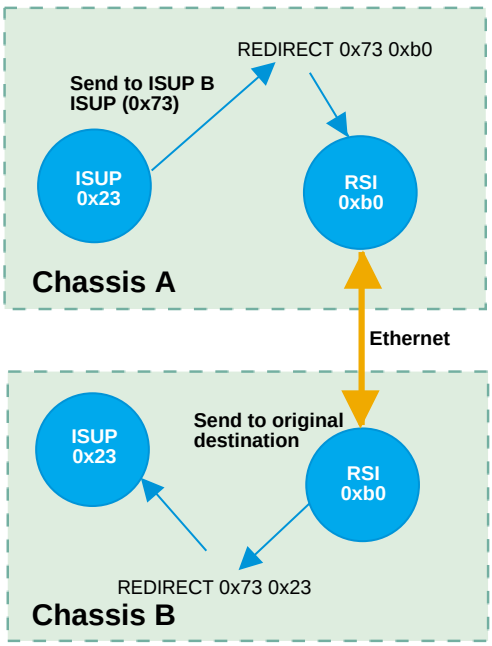


Figure 13 Use of REDIRECT and RSI for ISUP to ISUP communication

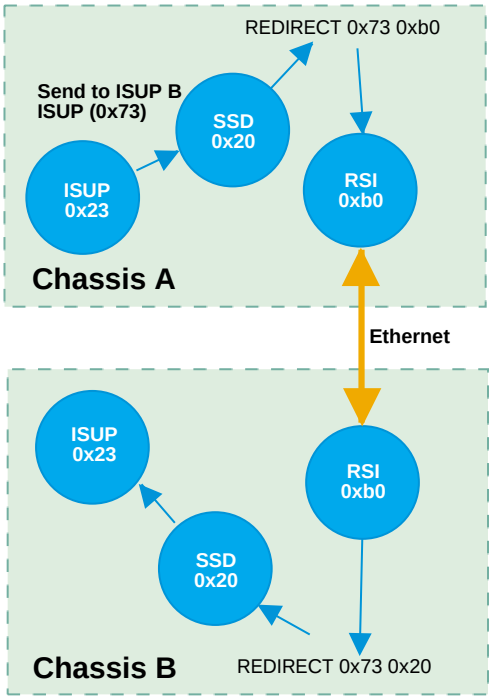


Figure 14 Use of REDIRECT with SSD

Where the protocol modules run on a signaling card, the REDIRECT statement at the receiver (the partner side) should redirect messages through the ssd or ssds driver (normally taking the module_id 0x20), as shown in Figure 14.

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.

In a dual circuit switching application (using ISUP or TUP), the physical circuit terminations are distributed between the two chassis that constitute the system; hence, failure of one of the halves will result 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 all 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 is done once the circuits had recovered.

However, in the configuration described earlier, the surviving half of the system will have no knowledge of the failed circuits (the configuration data for these would be held in the failed half), and would, therefore, 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. Following a failure of one half, these 'ghost' circuit groups may be configured on the surviving half, enabling hardware blocking to be issued for 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). Physical isolation may be achieved by the use of RSI and Ethernet.

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

In a dual TCAP system, failure of one half of the system will result in loss of the TCAP state information (for example, transaction state and any saved components pending transmission). In an Intelligent Networking 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) will time out, and the operation will be reported as failed. The operation should then be reattempted on the surviving system half.

The applications that communicate with the SS7 protocols must 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.

Setting System.txt Values

The following diagram 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. See Figure 15 for an illustration.

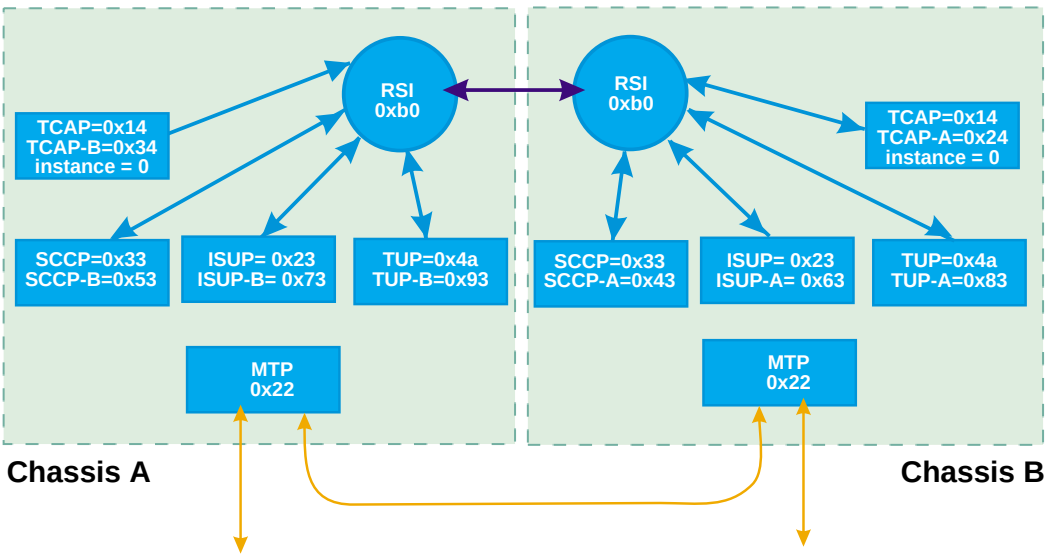


Figure 15 Modules in a dual SS7 protocol system

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), 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 detailed in the appropriate programmers manual.

System.txt for Protocols Running on a Signaling Processor Card

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 detailed in the appropriate programmers manual.

Appendix A — TCAP Set Instance Module ID Message

Description:

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

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.

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

1515 Route Ten
Parsippany, NJ 07054
Phone: 1-973-993-3000
Fax: 1-973-993-3093

Printed in the USA Printed on recycled paper. 03/03 00-8459-002