combining the differences between these modes within network 10. Accordingly, the action of DIA differs depending on whether DIA is operating within an SNA portion of network 10 or whether it is operating within the non-SNA portion of the network. More specifically, within the SNA portion of network 10, DIA and its supporting programs may be considered "applications" facilities. In this context, DIA resides at the transaction services level of SNA, also known as the Specific Application level of Open Systems Interconnections (OSI, also discussed in chapter 12 of Data and Computer Communications by W. Stallings above noted). However, in either case, it is a level 7 facility.

Within non-SNA portions of network 10, DIA and its supporting programs provide routing transport, sessions, and some transaction facilities. Thus DIA provides a comprehensive network architecture providing OSI level 3, 4, 5 and 7 services.

In accordance with the invention, DIA facilitates "utility session" within network 10. Utility sessions allow partner applications to communicate by means of the single session established between two logical units of the SNA type. In order to reduce the number of resources which must be defined to the network support programs, many user messages may be passed to many different application destinations through logical unit to logical unit (LU-LU) "pipes".

Applications exist on either side of the LU-LU pipe which act to concentrate outbound messages in route to applications resident on the other side of the LU-LU pipe; distribute inbound messages to local applications; and maintain and manage application task boundaries. Users may enter into a conversation with a set of transactions, refined to tasks, which are hereafter noted as "user sessions", and the boundaries of these user sessions (tasks) are indicated by begin session/end session flags.

Another application function supported by DIA is the routing of messages between nodes of network 10. Particularly, a switching application will route messages to the appropriate LU-LU session for transmission to another mode by examining and resolving the DIA destination IDs hereafter described.

In accordance with the invention messages conforming to DIA are composed of two functional parts: message headers and message text. Message Headers are transparent to most applications, but are the primary vehicle for passing information for session layer to session layer or transport layer to transport layer communications. Further, Message Text which is processed by end users, and is transparent to session and transport mechanisms.

In order to reduce program complexity and facilitate maintenance and enhancements, DIA has been structured in a layered fashion. In this regard, the DIA-defined data which flows through network 10 consists of a set a headers preface the end-user to end-user message text. Further, as in the case of objects, messages are organized in a family of types based on the specific form of its header. Particularly, there are "FM0" headers which contain routing and control information; FM2 headers which contain transport level information; FM4 headers which contain gateway information; FM8 headers which obtain information for secondary routing; i.e. messages passed through from node to node; FM9 headers which contain network management information; and FM64 headers which contain application-to-application management information, where, for example, applications running at RS 400 need be rendered compatible with applications running on an external computer connected to network 10 through a gateway 210.

In order to provide SNA compatibility, the first two bytes of all DIA FM headers are formatted such that byte 1 defines the length of header in hexadecimal; and byte 2, bit 0, identifies whether concatenation is provided or not; e.g. if bit 1=0 no other headers follow, but if bit 1=1, then the current header is followed by a concatenated header; while bits 1-7 identify the header type in hexadecimal value.

As will be appreciated to those skilled in the art, this layout is the same as that of SNA Function Management Headers. In an SNA LU0 implementation the DIA FM headers may be treated as SNA Function Management Headers (FMHs). Alternatively, the DIA FMs may be treated as pure data within the SNA Request Unit (RU).

With regard to destination routing, the basic premise of DIA is that each message flowing through network 10 carries a DIA header (FM0) that identifies its source and destination ids. Accordingly, switching applications exist which map destination ids to resources and route messages appropriately. In accordance with the invention, in order to send a reply, the recipient application simply swaps the content of the destination and source id fields and return message.

In the context of DIA the totality of ports, devices, and programs which are managed by a particular Switch and defined as destinations, are referred to as "regions". In this regard, each Switch; i.e. server 205 or cache/concentrator 302 shown in FIG. 2, need only be aware of the destination ids of resources within its own region and of the destination ids of switches resident in immediately adjacent nodes. Since server 205 is the central hub within the network 10 for application message routing, messages destined for end-users unknown to a switch are routed toward server 205 for eventual resolution. Destination id naming conventions then enable server 205 to determine the appropriate switch to which the message should be forwarded. Particularly, "destination id" fields "regions" and "unit" are used for this purpose.

Concerning switch responsibility, a switching application has three primary responsibilities It must forward messages to adjacent switches. It must collect messages from, and distribute messages to resources within its own region. And, it must maintain and manage application task boundaries. Users may enter into a conversation with a set of transactions. This set of transactions is referred to as a "task". These tasks are called user sessions. Further, the boundaries of these tasks are indicated by begin session/end session flags.

In order to fulfill these functions, a resource definition facility must exist for each switch to map each addressable resource to a destination id. In some cases, particularly on the RS 400, it may be desirable for an application to dynamically define subordinate resources to the switch and to interact with the switch to generate unique destination ids for these subordinate resources. It may also be necessary for the switch to either communicate with, or act within an application subsystem An example of an application subsystem is the Customer Information Control System, (CICS) event where CICS is a commercially available transaction process controller of the IBM Company, well known in the art. CICS, although subordinate to the operating system, is responsible for initiating and managing application "transaction" programs. Routing to specific transactions under the control of an application subsystem may be accomplished by a secondary address. In this case, the subsystem is defined as the primary destination. The transaction is defined as the secondary destination. A switch must only route incoming messages to the subsystem. The subsystem in turn posts to, or initiates the desired transaction.

The use of secondary addressing provides several advantages. Particularly, switch resource tables are not affected by the coming and going of "transaction" applications. Further, since the DR headers are SNA compatible, Type 1 application such as CICS need have no special message routing functions. A switch configured in accordance with the IBM standard VTAM could route incoming messages to CICS. Still further, transactions need not go into "receive loops". It is possible for the subsystem to poll on behalf of many transaction programs. In accordance with DIA secondary addressing is implemented within the application data stream. For instance, CICS transaction ids are, by convention, to be found in the first four bytes of application text.

With regard to the standards for DIA, it will be recalled that DIA messages have a header followed by

the message information. In the preferred embodiment, the DIA headers may be concatenated to one another. Further, the presence of concatenated headers is indicated by the setting of the first bit (bit 0) of the Header Type field.

However, there are two restrictions on the use of concatenated headers. Particularly, concatenated headers are required to be sequenced in ascending order left to right by header type numbers and secondary message text prefaced by concatenated headers (such as FM64 architecture message text) are not permitted to span across message block.

The basic structure of all DIA headers is presented below. As presented, "<>" indicate mandatory elements, "()" indicate optional elements an " . . . " indicate repeat allowed. Further, the "FMX" designations refer to the message header types previously identified and "TTX denotes TRINTEX, the former name of the network developer.

The basic DIA header structure is:

›<Length><Concatenation flag><Type> (FM defined data)!.

For TTX application-to-application messages, the structure is:

›(<FM0> (FM2) (FM8) (<FM64> (64text)) . . . (Appl. Text))!.

For TTX application-to-gateway application messages, the structure is:

›(<FM0> (FM2) (FM4) (FM8) (<FM64>0 (64text)) . . . (Appl. Text))!.

For TTX message to TTX network management, the structure is:

›(<FM0><<FM9> (9text)> . . .)!.

Finally, for internal TTX Switch to Switch messages, the header structure is:

›(<FM0> (Appl. Text))!,

where the FM0 function code is 2x or Cx.

Continuing, the general rules of implementation for DIA messages in the preferred embodiment are as follows. All inter-messages are prefaced by a single FM0. Further, other header types can be optionally concatenated to the FM0. Also, headers should occur in ascending order by header type; i.e. FM0, FM2, FM4, FM8, FM9, FM64. Header and text length values are carried as binary values. Numeric fields contained within DIA headers are carried with the most significant values in the left-most byte(s).

Further, long gateway messages (greater than 1K bytes including headers) are sliced up into blocks. This segmentation is indicated by the presence of the FM2 Header. In the preferred embodiment, the current block number of the FM2 must be correctly set because it acts as a sequence number and provides a means to guarantee message integrity. In this regard, the total number of blocks field must be set correctly when sending the last block of a logical message. Receiving programs can determine end of message by testing block number=total number blocks. If the sender cannot predetermine the total number of blocks in a beginning or middle of message block, the sender must place binary zeros in the total number of blocks field.

Still further, in the preferred embodiment, FM9 architected text may not span message blocks and may not be longer than 255 bytes. Additionally, FM64 architected text may not span message blocks and may not be longer than 512 bytes long. Yet further, only a single instance of FM2 and/or FM4 can be present in a message block. And, messages using FM9 or FM64 headers must be less than 1K bytes, and these messages should not be segmented into blocks.

Continuing with the DIA implementation rules, FM0 and FM2 must be present in each block of a multi-block message when being transported within the network system. Normal application message flow consists of a request/response pair. In normal processing, reception system applications send requests to host applications. Host applications return responses to these requests. The Reception System application initiates this dialogue Sending nodes are responsible for inserting the proper "source id" (SID) and "destination id" (DID) into the FM0. Additionally, the communications manager (CM) of the reception system further described hereafter, acts on behalf of reception system transaction programs. Messages destined to the CM should be considered systems messages (FM0 FUNCTION=Cn). Messages destined to subordinate transactions on reception system 400 should be considered applications message (FM0 Function=0n). Receiving nodes are responsible for swapping SID and DID contents when returning a response. Still further, intermediate nodes (with the exception of CICS switches and Gateways) need only be aware of FM0 and FM2 headers when routing messages to other destinations. CICS switches must be cognizant of all header layouts so that they can find the displacement to the transaction id which is contained within the first four bytes of application text. And server switch 205 provides a facility which allows responses to requests to be deliverable for at least a minimum period after the request was sent, e.g., one minute.

Finally, in the preferred embodiment, CICS switches pass all DIA FM headers on to their subordinate applications. The applications are then responsible for responsible for returning the headers with the SID/DID swap) back to the switch for responses. Both fixed length and variable length message headers are supported by the DIA. It must be noted that variable length headers are designed so that only the last field within the header is variable in length.

With regard to mode of conversation under utility sessions, the server switch 205 may engage in multiple sessions with an external CICS. Messages originating from network users may be routed through any of these sessions. Users are not forced to use the same utility session pipe for each message outbound to CICS. Pipes may be selected dynamically based on loading factors. In a switch-driven environment CICS transactions may typically be initiated by means of start commands from the switch. In this arrangement, CICS transactions will pass outbound data back to the switch through a queue.

In accordance with DIA, the potentially dynamic nature of conversation routing dictates the CICS transaction programs not be written in a conversational mode. Rather, the transaction programs are preferably either pseudo-conversational or non-conversational. In this regard it should be noted that conversational transactions send a message and wait for a reply, and non-conversational transactions send a message and expect no reply. In the case of pseudo-conversational transactions, a message is sent, but no reply is expected. However, such messages are coded so as to be able to accept user input in various stages of completion, thus mimicking conversational transactions.

As will be appreciated by those skilled in the art, communications may arise within network 10 that do not require the standards applied to DIA messages. However, non-DIA messages are allowed in the DIA structure. Particularly, non-DIA messages are designated by setting the length portion of the header (i.e., the first byte) to binary zero. Considering header layout, and with input first to FM0 headers, it should be noted that the FM0 header provides routing information to both intermediate and boundary switches. In addition the FM0 contains control fields which allow the sending application (which may be a switch) to communicate information to the switch which "owns" the destination application. When an

originating application wishes to converse with an application resident on the other side of an utility session it must initially pass an FM0 header with a function code representing an "begin session" to its controlling switch. The begin session code requests the assistance of any intervening switches in the establishment of an application session between the requester and the destination application specified in the DID.

When either application session partner wishes to terminate its conversation the session partner must pass an FM0 header to its switch, specifying either a function code representing an "end session", or "end session abnormal", or "request terminate". These function codes request the assistance of any intervening switches in the termination of the application session between the requester and the destination application specified in the DID. In this arrangement an end session function code is unconditional and does not require an acknowledgment. An end session abnormal function code is unconditional and does not require an acknowledgment. And, a request terminate function code is conditional and requires a positive acknowledgement. The positive acknowledgement to a request terminate is an end session. The negative acknowledgement to a request terminate is a function code representing "status Message".

Further, "status/return" function codes "system up", "system down", "echo", "system message" are used by corresponding applications in different regions of network 10 to determine application availability and user session status. Function codes are also used to designate end-to-end user message classes of service. These classes of service refer to a delivery requirement classification and are distinguished from SNA COS. Network class of service allows applications to specify whether or not responses to requests can be delivered after the standard timeout of server 205 has occurred.

In accordance with the invention, the DIA headers are arranged in a predetermined form base on their function. More particularly, FM0 headers, also known as Type "O" headers are required for every message within the network. Header Type 0 provides information necessary for routing and message correlation. Its fields include:

Header Length--Length of header data including length field.

Header Type--Bit 0 is header concatenation flag.

Bits 1-7 indicate current header type.

Function Code--Contains message function.

Data Mode--Indicates attributes of message data.

The "response expected" bit should be turned off if no response is expected, for instance, when sending the response to a request.

Source Id--Identification of end-user sending current message.

Logon Sequence--number which in conjunction with source id

Number provides unique identification of source when source is reception system 400.

Message--used to correlate requests and responses.

Sequence Number

Destination Id--Identification of message destination. All messages are routed by destination id. When responses to messages are sent back to original source, the source id and destination id fields must be swapped.

Text Length--length of all remaining data in the message to the right of this fields. (Includes length of concatenated headers if any are present).

The layout for the Type O header is as follows:

```
Header Type 0 layout:

Byte 0  Header Length (hexadecimal)
Byte 1  Header Type
bit 0     no other headers present; or concatenated header
          present
bits 1-7  current header type
Byte 2  Function Code; i.e.
        Application message (Class of Service)
        Status/Return Code message
        Begin Session
        End Session (normal)
        End Session (error)
        Clear Request (request terminate)
        System Up
        System Down
        Echo
        System Message
        Prepare to bring System Down
Byte 3  Data Mode (bit flags)
bits 0-7  Compaction;
          Encryption;
          Response Expected; Response;
          Unsolicited Message;
          Logging required;
          Timeout Message Required;
          Reserved;
Bytes 4-7
        Source ID
bits 0-7  Region ID (hexadecimal)
bits 8-19 xxxx xxxx xxxx
          Unit: Source application id if in
          Application mode
          xxxx xxxx xxxx
          Unit: Source Concentrator unit if
          in Reception System mode
bits 20-23
          xxxx Id Mode e.g.,
          Reception mode
          Reception mode
          Server 205 Application mode
          Server 205 Application mode
          Cache 302 Application mode
          Reserved
bits 24-31
          xxxx xxxx Sub-unit ID (hexadecimal)
Byte 8  Logon Sequence Number (hexadecimal)
```

```
Byte 9  Message Sequence Number (hexadecimal)
Bytes 10-13
        Destination ID
bits 0-7  Region ID (hexadecimal)
bits 8-19 xxxx xxxx xxxx
             Unit: Destination application ID
             if in Application mode
          xxxx xxxx xxxx
             Unit: Destination Concentrator
             if in Reception System mode
bits 20-23
          xxxx Id Mode; e.g.,
             Reception mode
             Reception mode
             Server 205 Application mode
             Server 205 Application mode
             Cache 302 Application mode
             Reserved
bits 24-31
          xxxx xxxx
          Sub-unit ID (hexadecimal)
Bytes 14-15
        Text Length.
```

With regard to FM2 or Type 2 messages, when an application is transmitting a large message, the sending application or its controlling switch can slice up the message into a number of smaller messages. The FM2 message header is used to indicate how these smaller messages can be reassembled into a single logical message by the receiving application or its controlling switch.

In preferred form, the maximum logical message size is 64K. The maximum message block size is 1K including all headers. Block sequence numbers in the FM2 range from 1 to a maximum of 255. And a single block message will be sequenced as block 1 of 1 in the FM2.

When network objects are large (greater than 1K bytes) they are sliced up into smaller blocks. Each object block is prefaced by an "object block header". Object block headers are found in the application text portion of a message. Object block headers provide sequencing information to cache/concentrator 302. The presence of an object block header does not obviate the requirement for an FM2 DIA header, except in the case of messages from the cache/concentrator down to RS 400. Both an object block header and a FM2 may be present in a message. Sequence numbering within object block headers ranges from 0 to 255. A singleblock Object will be sequenced as block 0 of 0.

Messages larger than 1K are subdivided into 1K blocks when being transmitted between the server switch 205, cache/concentrators 302, and reception systems 400.

Header Type 2 (FM2) message header contain information about this dividing of large messages and is useful when re-constructing large messages. The fields for an FM2 message header are as follows:

Header Length--length of header data including length field.

Header Type--Bit 0 is header concatenation flag.

Bits 1-7 indicate current header type.

Number of--total number of blocks used to transmit the

Blocks logical message. If the total number of blocks cannot be determined at the time the first or middle blocks of a message are being sent, this field may be set to zero. The last block of a message must contain the correct total number of blocks.

Block Number--number of the current message block being transmitted.

The layout for a Type 2 header is as follows:

```
Byte 0      Header Length (hexadecimal)
Byte 1      Header Type
        bit 0 no other headers present; or
              concatenated header present
        bits 1-7
              current header type
Byte 2      Number of Blocks (hexadecimal)
Byte 3      Current Block Number (hexadecimal).
```

With regard to FM4 type headers, also referred to as Type "4", these headers have been designed for communications between network gateway interface applications and external computer systems. For Type 4 Headers, the fields are as follows:

Header Length--length of header data including length field.

Header Type--Bit 0 is header concatenation flag.

Bits 1-7 indicate current header type.

Network User--a seven byte field containing the internal ID of the network user on whose behalf a conversation is being held with the external computer system.

External Data--Reserved Mode

Correlation Id--a field reserved for use by the external computer system. The contents of this field will initially be set to zero when a conversation is initiated across a gateway. The external system may then set the contents of this field to any value desired. Subsequent messages originating from TTX within he bounds of a virtual subscriber to external host session will echo the contents of the Correlation Id field back to the external system.

The layout for a Type 4 header is as follows:

```
Byte 0     Header Length (hexadecimal)
Byte 1     Header Type
bit 0        no other headers present; or
             concatenated header present
```