US 20020116514A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2002/0116514 A1**

Lee (43) **Pub. Date: Aug. 22, 2002**

(54) **MESSAGE SYSTEM FOR ASYNCHRONOUS TRANSFER MODE**

(76) Inventor: **Kenny Ying Theeng Lee**, Duluth, GA (US)

Correspondence Address:
DISCOVISION ASSOCIATES
INTELLECTUAL PROPERTY DEVELOPMENT
2355 MAIN STREET, SUITE 200
IRVINE, CA 92614 (US)

(21) Appl. No.: **10/063,508**

(22) Filed: **May 1, 2002**

Related U.S. Application Data

(63) Continuation of application No. 09/338,935, filed on Jun. 23, 1999.

(60) Provisional application No. 60/090,441, filed on Jun. 24, 1998.

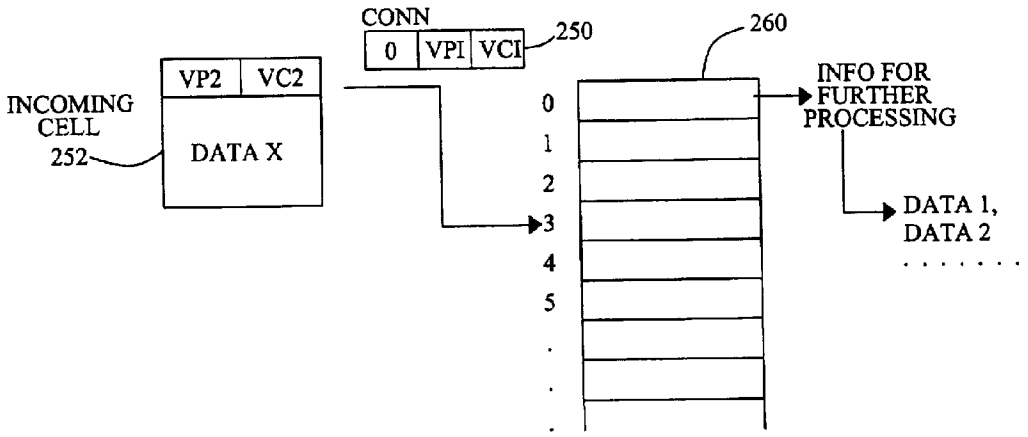
Publication Classification

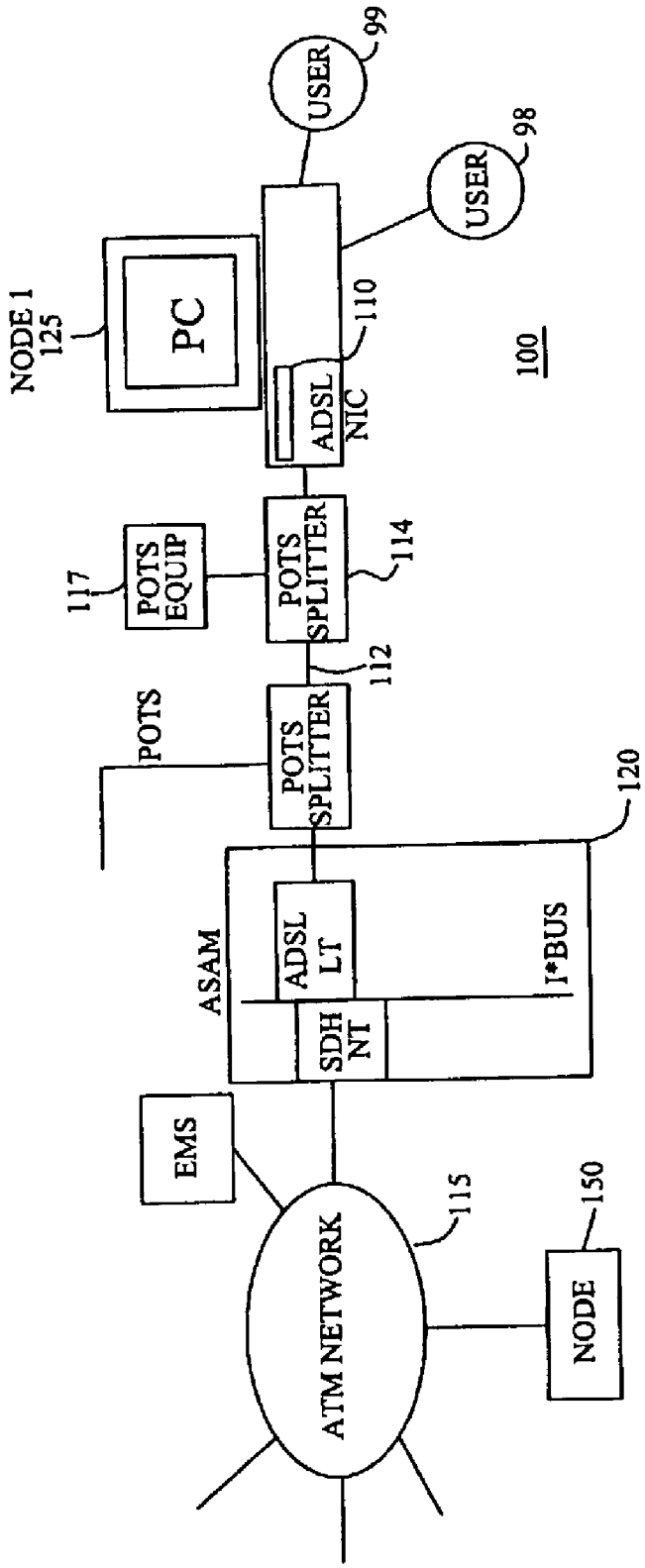
(51) **Int. Cl.**[7] **H04L 12/56**; G06F 15/16; H04L 12/28

(52) **U.S. Cl.** **709/230**; 370/395.1

(57) **ABSTRACT**

An asynchronous transfer mode system operates using virtual addresses VPI and VCI. Cells including these addresses are received. The data associated with these cells is stored in a table. The table contents are advantageously accessed according to a connection number. The proper connection number is more easily found by using two variables. A first variable is associated with a number of active connections. A second variable is associated with the last-used index in the table.





GENERAL NETWORK ARCHITECTURE

FIG 1

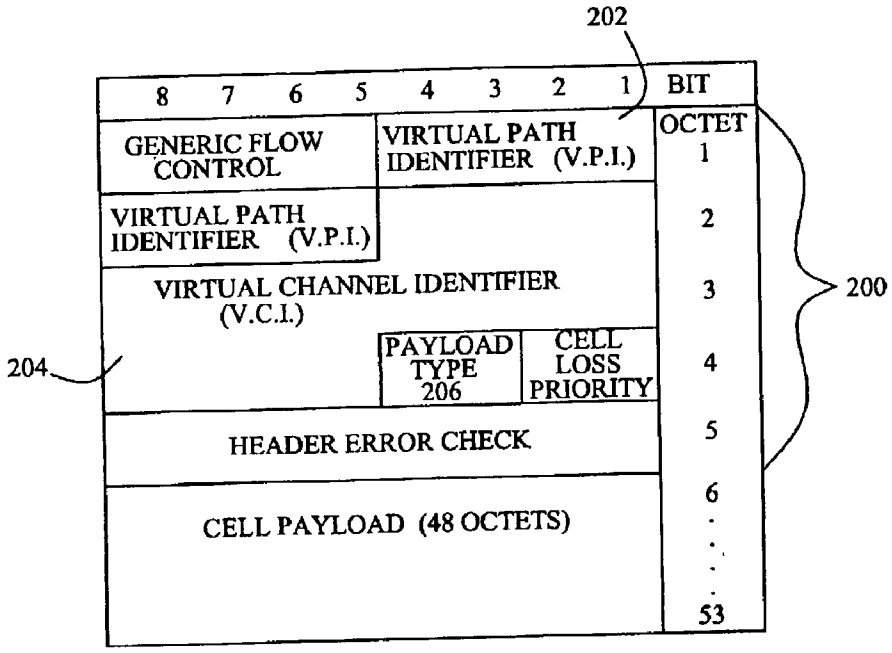

202
8 7 6 5 4 3 2 1
BIT
GENERIC FLOW CONTROL
VIRTUAL PATH IDENTIFIER (V.P.I.)
OCTET
1
VIRTUAL PATH IDENTIFIER (V.P.I.)
2
VIRTUAL CHANNEL IDENTIFIER (V.C.I.)
3
200
204
PAYLOAD TYPE 206
CELL LOSS PRIORITY
4
HEADER ERROR CHECK
5
CELL PAYLOAD (48 OCTETS)
6
53

FIG 2A

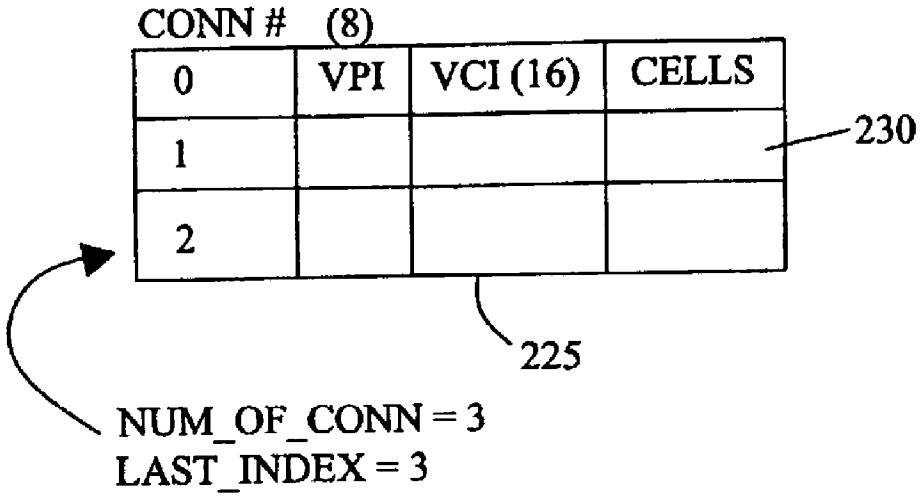


FIG 2B

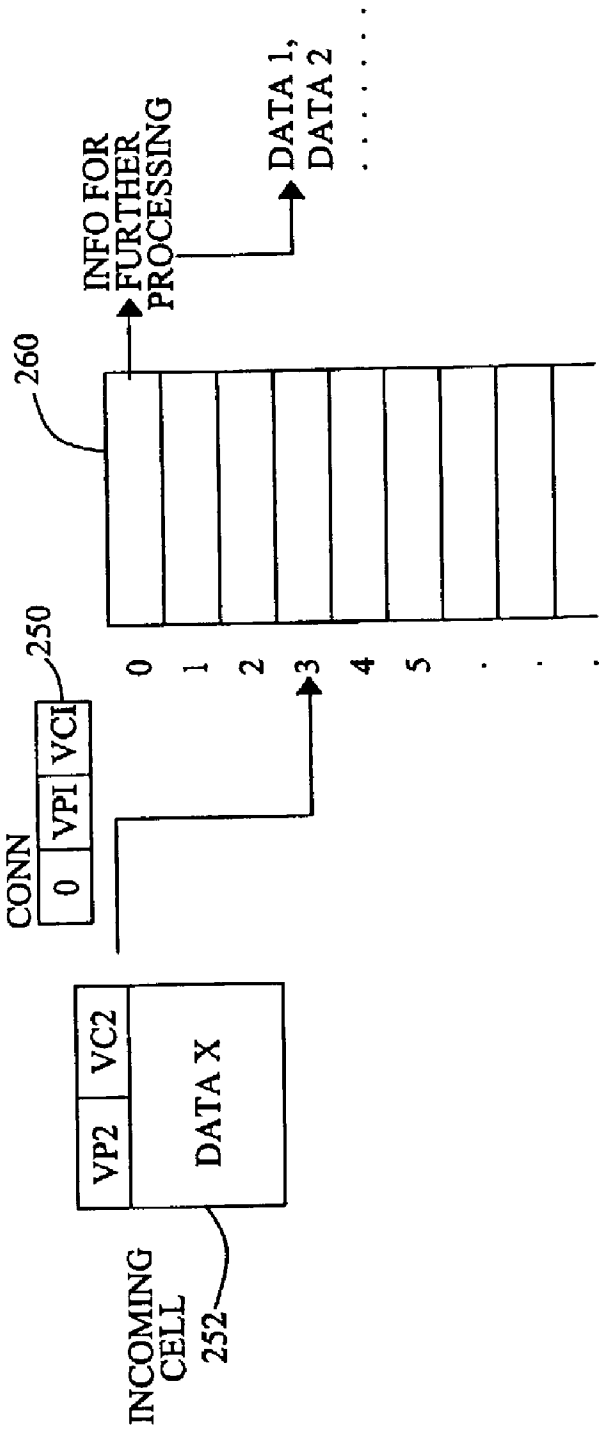

INCOMING CELL
252
VP2
VC2
DATA X
CONN
0
VPI
VCI
250
260
0
1
2
3
4
5
INFO FOR FURTHER PROCESSING
DATA 1, DATA 2

FIG 2C

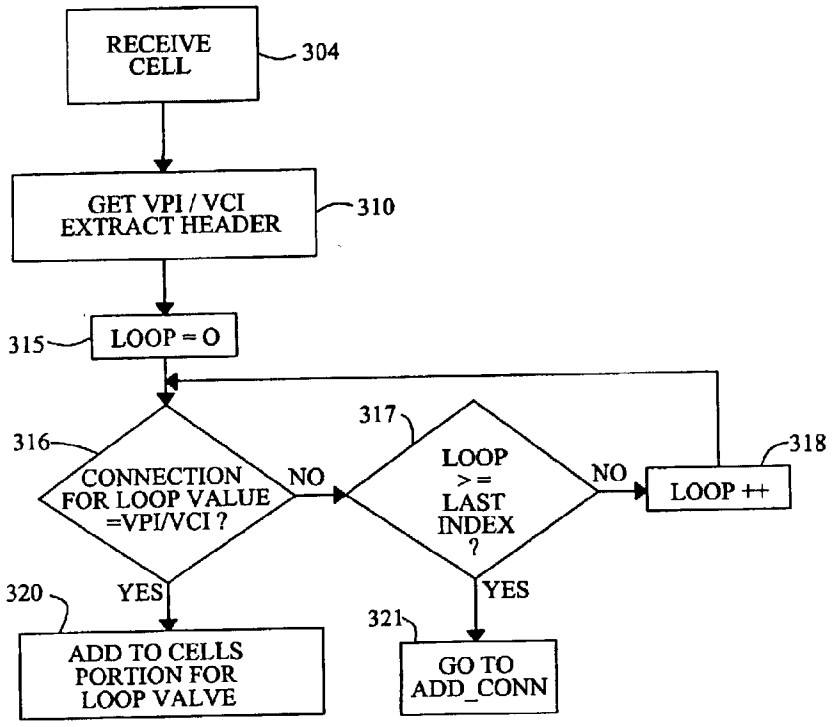

RECEIVE CELL
304
GET VPI / VCI EXTRACT HEADER
310
315
LOOP = O
316
CONNECTION FOR LOOP VALUE =VPI/VCI ?
NO
317
LOOP >= LAST INDEX ?
NO
318
LOOP ++
YES
320
ADD TO CELLS PORTION FOR LOOP VALVE
YES
321
GO TO ADD_CONN

FIG 3A

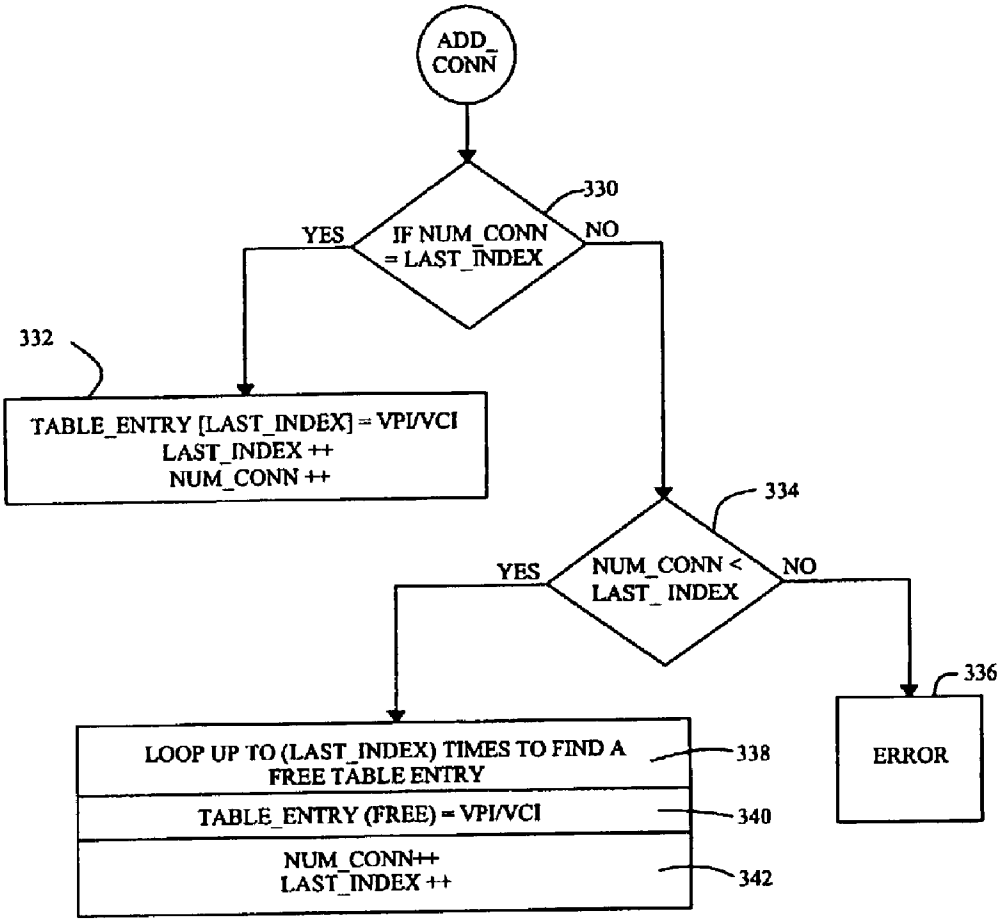

ADD_ CONN
330
IF NUM_CONN = LAST_INDEX
YES
NO
332
TABLE_ENTRY [LAST_INDEX] = VPI/VCI
LAST_INDEX ++
NUM_CONN ++
334
NUM_CONN < LAST_ INDEX
YES
NO
336
ERROR
LOOP UP TO (LAST_INDEX) TIMES TO FIND A FREE TABLE ENTRY
338
TABLE_ENTRY (FREE) = VPI/VCI
340
NUM_CONN++
LAST_INDEX ++
342

FIG 3B

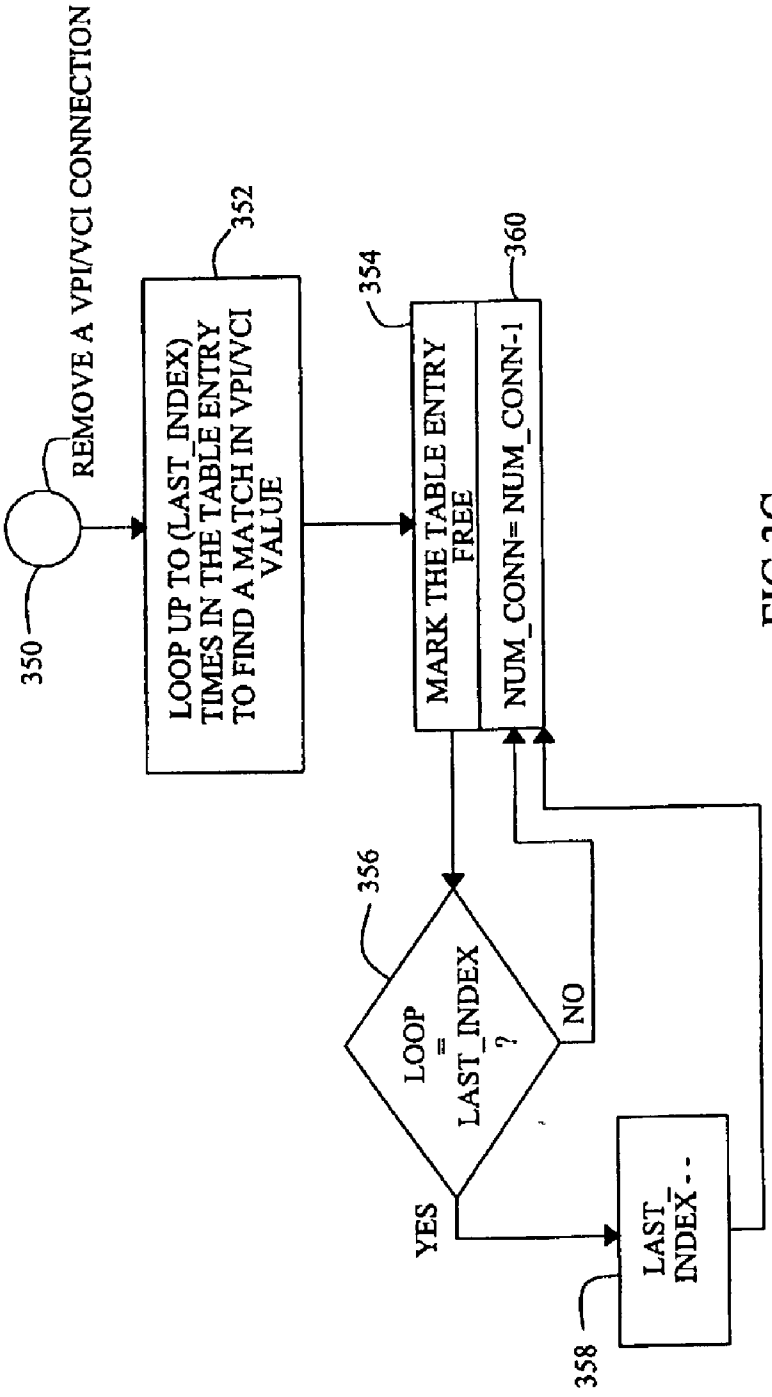

REMOVE A VPI/VCI CONNECTION
350
LOOP UP TO (LAST_INDEX) TIMES IN THE TABLE ENTRY TO FIND A MATCH IN VPI/VCI VALUE
352
MARK THE TABLE ENTRY FREE
354
NUM_CONN= NUM_CONN-1
360
LOOP = LAST_INDEX ?
356
NO
YES
LAST_ INDEX - -
358

FIG 3C

MESSAGE SYSTEM FOR ASYNCHRONOUS TRANSFER MODE

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is a continuation Patent Application that claims priority from co-pending patent application Ser. No. 09/338,935, filed on Jun. 23, 1999, which in turn claims the benefit of U.S. Provisional Application No. 60/090,441, filed Jun. 24, 1998.

BACKGROUND OF INVENTION

[0002] The disclosed system teaches a way of simplifying messages in asynchronous transfer mode. The present system specifically teaches specifying connection ranges among various information to simplify the connection.

[0003] Asynchronous transfer mode or ATM is a telecommunications protocol that allows packet based transfer of information. Cells of information are sent across an information network defined by a number of nodes. The information is sent from node-to-node.

[0004] An ATM transport network (i.e., a communication network, which transmits information using ATM cell packets) is known to include an ATM layer and a physical layer. The ATM layer is based on the virtual path/virtual channel (VP/VC) concept. The VC identifies a unidirectional communication capability through which ATM cells are transported. One or more virtual channels (VCs) can be used in a particular virtual path (VP), which also identifies another level of the communication capability through which the ATM cells are transported.

[0005] An ATM cell is the smallest information unit. It includes a header field of 5 bytes or octets, and a payload field of 48 bytes or octets. The header field includes VP and VC identifiers. These identifiers are used for routing the information to an intended destination.

[0006] Communication in known ATM networks is initiated during a connection setup, after which cells belonging to one connection follow a predetermined path defined by the VPI and VCI on a particular link. The connection control information transferred during setup utilizes a unique Signaling VC (SVC) which is included in the VP. The SVC is identified by the virtual path ID (VPI) and virtual channel ID (VCI).

[0007] Cells destined for many different end points are sent over a single physical communications circuit. The header of each cell includes a channel identifier which is used to control the routing of the cell through the ATM system. The channel identifier determines routing of the cell.

[0008] In a typical ATM system there are 256 possible VPIs and 65,536 possible VCls; thus, there are 16,777,216 possible channel identifiers (VPI/VCIs). One of the many challenges in designing an ATM network is how to handle this huge number of corrections.

[0009] Specified traffic control protocols are used to determine the routing of the information. The routing is controlled using conventional addressing techniques.

[0010] Further details of ATM are well known in the art. In addition, different flavors and sub-types of ATM are known, including digital subscriber line (DSL), assymetric digital subscriber line(ADSL), and other flavors of digital subscriber line (XDSL).

[0011] In all of these communication modes, a message is broken into multiple portions or cells. A conventional ATM system breaks the total message to be sent over ATM into 48 byte data portions. A typical data message might be, for example, 1500 bytes in length. Hence, the 1500 byte message is divided up into 31 of the 48 byte cells.

[0012] The ATM message is sent into the ATM environment with 48 byte increments, each addressed by 24 bits of VPI/VCI to instruct where the package is going.

[0013] The recipient node needs to form this data together into the original size. That original recipient receives a mixed message within multiple cells, typically having multiple VPIs and VCIs.

[0014] One way to handle the mixed message is to place each of the messages into a buffer as received, and remove the different cells from the buffer to form one total message.

[0015] This system includes certain limitations. For instance, if the VPI is used as an index to the buffer, the amount of memory for the addressing scheme can increase, and make it difficult to review the contents. Searching the array can take large amounts of processing power and memory. As the number of VPI/VCI connections increases, the array size can grow exponentially.

[0016] The present system teaches a simplified system.

SUMMARY OF INVENTION

[0017] The present specification, in recognition of the above, defines an improved way of handling a message from a system which divides total messages into divided cells. Each of the cells is associated with an address. In a disclosed mode, that address can be a virtual address. A table is stored in memory which includes a list of connection number, address, and data for each of the connection number. Each of the cells is stripped of its address, associated with the connection number, and the data associated with that cell is put into the table. The table stores a plurality of information pieces about a number of simultaneous connections.

[0018] Access to the table is simplified by defining at least one variable associated with all of the simultaneous messages in the table. This variable is a variable that facilitates searching the table. One possible variable is a variable associated with the number of connections. Another variable is associated with the length of the table, e.g., the last-used index. In one disclosed mode, two variables are used, one of which is related to the last index, and the other which is related to the number of connections. If the two variables are equal, then the table is full, and the next entry in the table can be used for a new connection. Otherwise, the system can search to a value no higher than the last-used index, and by so doing, search less than all of the total number of values in the table.

BRIEF DESCRIPTION OF DRAWINGS

[0019] These and other aspects will be described in detail with respect to the accompanying drawings, wherein:

[0020] **FIG. 1** shows a general network architecture of an ATM network;

[0021] FIGS. 2A-2C show a VPI/VCI header according to an embodiment of the present invention; and

[0022] FIGS. 3A-3C show flowcharts of software operations performed by one or more embodiments of the present invention.

DETAILED DESCRIPTION

[0023] The overall block diagram of the general network architecture is shown in **FIG. 1**. The embodiments described herein can operate as part of an ATM system. An ADSL interface card for communicating with an ADSL network is described. More generally, however, this system can operate within any system that carries out data communication by dividing a total message into separate addressed packets, or more specifically in an asynchronous transfer mode system.

[0024] A PC **125** is, for example, an Internet service provider that provides Internet service to a number of users **98**, **99**, and others that are not shown. PC **125** includes an ADSL network interface card or NIC **110**. NIC **110** connects to the telephone line **112** via a plain old telephone system (POTS) splitter **114**. Other POTS equipment **17** can include conventional telephone equipment.

[0025] A conventional ATM subscriber access multiplexer or ASAM **120** connects from telephone line **112** to ATM network **115**. The ASAM **120** multiplexes a number of communications via the ATM network **115**. In this system, the NIC **110** becomes a node connecting to the ATM network **115** which allows routing to other nodes, such as second node **150**. While only one second node **150** is shown, the ATM network is typically connected to literally thousands of other nodes shown generally in **FIG. 1**. Any of the multiple nodes can send or receive a message. The connection among these nodes are based on their VCI/VPI identifiers.

[0026] Node **1** receives a number of cells that will form ATM messages.

[0027] Each cell is of the general form shown in **FIG. 2A**. A 5 byte (Octet) header **200** includes the virtual path identifier **202**, virtual channel identifier **204**, payload type **206**, and other conventional ATM control data.

[0028] When the cells arrive at the NIC **110**, the 5 bytes of header information are used to determine how to reconstruct the entire message among the multiple messages that are sent at once.

[0029] A typical way to operate is to put the entire information, including the 5 byte header information, into a buffer in memory, e.g., an array. The array could be addressed using the VPI and VCI as addresses to the array.

[0030] The disclosed system uses a special memory table **225** shown in **FIG. 2B**. Two additional variables are also maintained, related to all of the entries in the table **225**. A first variable is related to the number of active messages, and defines the total current number of connections (num_of_conn). A second variable defines the last used index in the table (last_index).

[0031] A connection number is defined for each message. The connection number can be, for example, between 0-31, thereby allowing 32 simultaneous connections. This connection number can be expressed as one byte of information.

[0032] The remainder of the table entry includes the 8 bit VPI and the 16 bit VCI corresponding to the connection number. The cell contents from the multiple cells of the message are filled into the table entry field **230**.

[0033] The num_of_conn variable represents the number of current active connections within the table. **FIG. 2B** shows three connections, and therefore num_of_conn=3.

[0034] The last_index variable represents the last free index in the table entries, here again 3.

[0035] An alternative table form is shown in **FIG. 2C**. In this system, there are two tables. A first table part **250** translates between the VPI/VCI of an incoming cell **252**, and its connection number. A second table part **260** arranges each of the data**1**, data**2**, data**3** of the cells into a table arranged by connection numbers.

[0036] This table and variables are kept up-to-date with each new connection and each dropped connection as described herein. The ADSL NIC includes an internal controller that operates according to the flowcharts described herein to process the cells. The detailed operation is shown with respect to the flowcharts of FIGS. **3A-3C**.

[0037] A cell arrives at step **304**, having the general form shown in **FIG. 2A**.

[0038] At step **310**, the cell header **200** is extracted, which provides the addressing information from its VPI/VCI. At this time, error checking can be carried out in conventional ways, and the VPI and VCI values are removed. The VPI and VCI values are used as addressing information and translated into connection numbers.

[0039] At step **315**, a loop is formed from 0 up to the value of last_index value. Step **316** compares each VPI/VCI in the table against the current VPI/VCI from the received cell at **316**. If there is a match, flow passes to step **320** where the current cell is added to the cells **230** for the current connection number. If not, the loop is compared against last_index at **317**. If the loop value is greater than or equal to last-index, the current VPI/VCI is not in the table. The add_connection routine is called at **321**. Otherwise, the loop value is incremented at **318**, and the next value is tested.

[0040] Therefore, each VPI/VCI is handled as a connection number of 0-N, where N is the maximum allowed number of simultaneous messages. A typical value for N might be 32.

[0041] Importantly, the total allowed number of simultaneous messages does not increase the length of the search. Instead, the maximum search ends at the last_index value, which represents the last-index that is used. The search length is increased only by the number of existing active connections instead of the numbers of allowed connections.

[0042] The add_connection writes new VPI/VCI values to the table index. This is carried out according to the routine shown in **FIG. 3B**.

[0043] Step **330** first determines if the number of connections variable (num_conn) is equal to the last-index variable (last_index). If so, then the table is currently full. The system then uses the next consecutive entry after last index as shown in step **332**. The table entry corresponding to the last_index is set to the current VPI/VCI address at **332**, and both the last_index and num_conn are incremented.

[0044] As described herein, when a connection is terminated, a value will be removed from the table, leaving a space in the table. The space is noted by setting the VPI/VCI value to all 0's. In that case, num_conn will not be equal to last_index at **330**.

[0045] The flowchart passes to step **334**, which carries out an error checking routine to first determine if num_conn is less than last_index. If not, an error is established at step **336**.

[0046] If the num_conn is less than last_index, however, step **338** illustrates a loop from 0 up to the last_index value, to find a free table entry. That free table entry is then set to the current VPI/VCI at **340**. Num_conn and last_index are both incremented at step **342**.

[0047] A VPI/VCI connection can also be removed as illustrated in **FIG. 3C**. Termination of an ATM message is known in the art. When a message is complete, the message being terminated passes its VPI/VCI address to the routine of **FIG. 3C**. This is received at step **350**. Step **352** loops up to the value of last_index to find a match to the current VPI/VCI value. This is similar to steps **315**, **316**, **317** in **FIG. 3A**.

[0048] At step **354**, the determined table entry is marked as being free by setting the VPI/VCI to all zeros. The number of connections is also decremented. However, last_index is not decremented unless the last entry in the table is being removed. Step **356** shows determining if the current loop value =last_index. If so, last_index is decremented at **358**. In either case, num_conn is decremented at **360**.

[0049] Hence, this addressing becomes relatively simplified. The data from the cells is stored in an improved way. Moreover, the inherent way in which the information is stored automatically sorts the information into a more logical order.

[0050] The use of the two variables, including one that indicates the number of connections, and another that indicates the last information that is free, enables searching fewer than the total number of connections each time a cell is received. When the entire used part of the table is full, no searching needs to be done at all to add a new VPI/VCI. When the table is not full, the search continues only until the first empty point is reached. Even though a connection may be removed anywhere in the table, the search need not always search every entry. In fact, this search technique will never search the entire array, since if the array were full, num_conn would equal last_index.

[0051] Another routine, not shown, could periodically crunch the table **225** to remove blanks therein. This could be done on a timed basis, or when the activity gets below a certain level.

[0052] The previous discussion has referred to flowcharts, and it should be understood that these operations could be carried out by executing code in processors, in dedicated hardware that is formed using hardware definition language to effect these flowcharts, in firmware, or in any other form.

[0053] Although only a few embodiments have been described in detail above, other embodiments are contemplated by the inventor and are intended to be encompassed within the following claims. In addition, other modifications are contemplated and are also intended to be covered.

1. A method of transferring a plurality of messages, comprising: dividing said message into cells of a specified length; obtaining an address on each of said cells which identifies its message; receiving said cells at a receiving node; receiving other cells, from other messages at said receiving node; and at said receiving node, defining at least one variable related to said addresses of at least a plurality of said messages.

2. A method as in claim 1, further comprising forming a message table for said plurality of messages.

3. A method of claim 2, wherein said variable is a number of a last-index in said message table.

4. A method as in claim 2, wherein said variable is a number of active connections in said message table.

5. A method as in claim 2, wherein there are two of said variables, and said variables include a number of active connections in said message table and a last-index which is used by said table.

6. A method as in claim 2, wherein said variable is related to entries in the message table.

7. A method as in claim 2, wherein said variable is related to a number of active messages.

8. A method as in claim 2, further comprising, at the receiving node; receiving a cell, and comparing and address of the cell with said message table by searching said message table only between a lowest possible index and a value based on said variable.

9. A method as in claim 8, further comprising defining a second variable related to said message table.

10. A method as in claim 9, wherein said second variable is related to a fill state of said message table.

11. A method as in claim 10, further comprising comparing said second variable to said number of active messages variable to determine blank entries in said message table.

12. A method as in claim 2, further comprising determining if there are blank spaces in the message table by using said variable, searching said table for said blank entries if so, and using said variable to determine a location for a new message to be added, if not.

13. A method as in claim 1, wherein said communication is via Asynchronous Transfer Mode.

14. A method as in claim 13, wherein said addresses of said cells include a virtual channel identifier (VCI) and virtual path identifier (VPI).

15. A method as in claim 1, wherein said communication is via a digital subscriber line (DSL).

16. A method as in claim 5, wherein said message table further comprises a cell storage area and a connection number entry area, further comprising: associating a connection number with each new address associated with said messages; storing said connection number in said connection number entry area; extracting said cells from said messages and storing said cells in said cell storage area associated with said connection number in said message table.

17. A method as in claim 16, further comprising: receiving a next message having a next address; determining if said next address is associated with a connection number in said message table; extracting a cell from said next message and storing said cell in said cell storage area associated with said connection number, if said next address is associated with a connection number in said message table; and adding a new

connection number to said message table and storing said cell in a cell storage area associated with said new connection number in said message table, if said next address is not associated with a connection number in said message table.

18. A network interface card device for communicating with a plurality of simultaneous communications, comprising: a register, storing a table including a plurality of simultaneous data transmissions, at least a plurality of said data transmissions formed by a plurality of separated cells of information, where each of the cells have a shorter total length than the data transmission, and each data transmission and each cell of the data transmission are represented by an address; said table storing contents of said cells and an address, such that each cell contents is added to an entry in said table which represents the address associated with said each cell, and said memory also storing a first variable related to contents of at least a plurality of said simultaneous data transmissions in said table; and a controller, operating based on a stored instruction set, to receive a cell, to use said variable to search said table in a way that enables searching less than all of said table, to add said cell to a desired entry in said table if an address of said cell is found in said table and to add a new address to said table if said address of said cell is not found in said table.

19. A device as in claim 18, further comprising a second variable related to said plurality of simultaneous data transmissions, wherein said controller also uses said second variable to search said table.

20. A device as in claim 18, wherein said controller also detects an end of message, and removes an entry corresponding to the ended message from said table.

21. A device as in claim 20, wherein there is a second variable related to a number of connections, and wherein said controller is operative to determine whether a blank space exists in said table by comparing said first variable to said second variable.

22. A device as in claim 21, wherein said controller compares said first variable to said second variable, adds a new entry at the end of the table if said first variable equals said second variable, and searches from a lowest value to a value of said first variable if said first and second variables are not equal.

23. A device as in claim 18, wherein said first variable represents a last index in said table that is used, and said second variable represents a number of active connections in said table.

24. A method of operating a synchronous transfer mode (ATM) system, comprising: receiving a plurality of simultaneous messages, each of said simultaneous messages being formed of a plurality of separated cells with addresses which cells collectively have data forming the simultaneous messages; maintaining a table in memory which stores said data associated with said addresses; detecting a new message, which does not have a previous entry in said table; and finding a new location in said table by searching fewer than all locations in said table.

25. A method as in claim 24 further comprising storing a variable associated with said plurality of messages.

26. A method as in claim 24 further comprising storing two variables associated with said plurality of simultaneous messages, one of said variables relating to a number of active messages, another of said variables related to a size of the table.

27. A method as in claim 26 wherein said finding comprises comparing said variables to one another.

28. A method as in claim 26 wherein if said variables are equal, establishing a new value at the end of the table.

29. A method as in claim 27 wherein if said variables are not equal, searching less than all of said table to find a blank space in the table.

30. A method as in claim 24 further comprising assigning a connection number to said cells based on said addresses, and storing said connection number are along with other portions with the same connection number.

31. A method as in claim 24, further comprising removing an inactive message from the table.

32. A method as in claim 31 wherein said inactive message is removed by setting its address to all zeros.

33. A method as in claim 32 further comprising changing values of said variables after removing said inactive message.

34. A method of operating in an asynchronous transfer mode, comprising: receiving a plurality of simultaneous messages, each of said simultaneous messages received as a plurality of separated cells which are addressed with an address; maintaining a table of information from said cells in memory; maintaining a variable associated with said table in memory; and using said variable to search said table in way that enables searching less than all of said table.

35. A method as in claim 34 wherein said variable is related to a number of active connections.

36. A method as in claim 35 wherein said variable is related to a last entry in the table.

37. A method as in claim 34 wherein there are two variables, one related to a number of active connections, another related to an ending point of the table.

38. A method as in claim 37 further comprising adding a new connection by comparing said variables, taking a first action if said variables are equal and a second action if said variables are unequal.

39. A method as in claim 38 wherein said first action comprises adding a new value at the end of the table, and incrementing a first variable.

40. A method as in claim 38 wherein said second action comprises searching between a minimum value and a value of said second variable to look for a blank space in said table, said searching comprising searching less than all values in said table.

41. A method as in claim 40 wherein said second action comprises searching between a minimum value and a value of said second variable to look for a blank space in said table, said searching comprising searching less than all values in said table.

42. A method as in claim 34, wherein said table further comprises a cell storage area and a connection number entry area, further comprising: associating a connection number with each new address associated with said simultaneous messages; storing said connection number in said connection number entry area; extracting said cells from said simultaneous messages and storing said cells in said cell storage area associated with said connection number in said table.

43. A method as in claim 42, further comprising: receiving a next message having a next address; determining if said next address is associated with a connection number in said table; extracting a cell from said next message and storing said cell in said cell storage area associated with said

connection number, if said next address is associated with a connection number in said table; and adding a new connection number to said table and storing said cell in a cell storage area associated with said new connection number in said table, if said next address is not associated with a connection number in said table.

44. A method of operating in asynchronous transfer mode, comprising: maintaining a table in memory having a plurality of simultaneously-transmitted messages, each message transmitted as a plurality of separated cells, said table associating cells that are related to one another; receiving a new cell; receiving an address of said new cell from the new cell; removing the address from said new cell to leave information without said address; forming a first variable associated with said table which represents a last value in said table which is active; forming a loop from a minimum value to said first variable; at each point in said loop, comparing said removed address to a current loop value; and adding said information from said cell to the current loop position if the removed address matches the current loop position.

45. A method as in claim 44 further comprising detecting no matches, and adding a new connection.

46. A method as in claim 44 further comprising a second variable related to a number of active connections, and wherein said second variable is incremented when a new connection is added.

47. A method for supporting a flexible addressing scheme for an ADSL interface using a VPI/VCI value in a header of a cell, said VPI/VCI value being within a range of connection values, said method comprising the steps of: setting a num_of_conn variable equal to zero; setting a last variable equal to zero; determining a first entry of a table for storage of a value; said table having a number of entries, said number of entries being less than the number of connection values in said range, including; setting said first entry equal to said last incrementing said last incrementing said num_of_conn; storing a predetermined one of said connection values in said first entry; receiving said cell; extracting said VPI/VCI value from said cell; comparing said VPI/VCI value to entries in said table in a sequential manner starting with said first entry and ending at said last determining a match when said VPI/VCI value is present in one of said entries; and passing the contents of said one of said entries and said cell to a processor for further processing when said match is determined.

48. The method as in claim 47 including the steps of: determining an additional entry of said table for storage of a connection, including; setting said additional entry equal to said last incrementing said last incrementing said num_of_conn; storing an additional predetermined one of said connection values in said additional entry.

49. The method as in claim 48 further including the steps of: recognizing a delete connection value; comparing said delete connection delete connection value to entries in said table in a sequential manner starting with said first entry and ending at said last determining a match when said delete connection value is present in one of said entries; and storing a null value in said one of said entries, said null value being outside said range of connection values; decrementing said num_of_conn.

50. The method as in claim 49 including the steps of: recognizing a second additional connection value; determining a second additional entry of said table for storage of a connection value, said step of determining a second additional entry including; setting said second additional entry equal to said last incrementing said last and incrementing said num_of_conn; comparing said null value to entries in said table in a sequential manner starting with said first entry and ending at said last determining a match when said null value is present in one of said entries; setting said second additional entry equal to said one of said entries when said match is determined, decrementing said last when said match is determined; and storing said second additional connection value in said second additional entry.

51. The method as in claim 47 wherein said number of table entries is equal to **32**.

52. The method as in claim 47 wherein said number of table entries is equal to **64**.

53. An system for supporting a flexible addressing scheme for an ADSL interface using a VPI/VCI value in a header of a cell, said VPI/VCI value being within a range of connection values, said system comprising; first means for determining a first entry of a table for storage of a value, _said table having a number of table entries, said number of table entries being less than the number of connection values in said range, second means for storing a predetermined one of said connection values in said first entry; third means for receiving said cell; fourth means for extracting said VPI/VCI value from said cell; fifth means for comparing said VPI/VCI value to entries in said table in a sequential manner; sixth means for determining a match when said VPI/VCI value is present in one of said entries, and seventh means for passing the contents of said one of said entries and said cell to a processor for further processing when said match is determined.

54. The system as in claim **53** wherein said first means includes: a comparing means for comparing a null value to entries in said table in a sequential manner, said null value being outside said range of connection values; a determining means for determining a null match when said null value is present in one of said entries; and a setting means for setting said first entry as said one of said entries.

55.The system as in claim **53** including: an additional determining means for determining an additional entry of said table for storage of a connection value, said second means storing an additional predetermined one of said connection values in said additional entry.

56. The system as in claim **54** wherein said additional determining means includes: a comparing means for comparing a null value to entries in said table in a sequential manner, said null value being outside said range of connection values; a determining means for determining a null match when said null value is present in one of said entries; and a setting means for setting said additional, entry as said one of said entries.

57. The system as in claim **56** further including: a recognizing means for recognizing a delete connection value, said comparing means comparing said delete connection delete connection value to entries in said table in a sequential manner, said determining means determining a match when said delete connection value is present in one of said entries, and said second means storing said null value in said one of said entries.

58. The system as in claim **57** wherein said recognizing means recognizing an additional connection value and said additional determining means determining said additional entry of said table.

59. The system as in claims **53** wherein said number of table entries is equal to **32**.

60. The system as in claims **53** wherein said number of table entries is equal to **64**.

61. A method for supporting a flexible addressing scheme for an ADSL interface using a VPI/VCI value in a header of a cell, said VPI/VCI value being within a range of connection values, said system comprising: setting a num_of_conn variable equal to zero; setting a last variable equal to zero; determining a first entry of a table for storage of a value said table having a number of entries, said number of entries being less than the number of connection values in said range, including; setting said first entry equal to said last incrementing said last incrementing said num_of_conn, storing a predetermined one of said connection values in said first entry; receiving said cell; extracting said VPI/VCI value from said cell; comparing said VPI/VCI value to entries in said table in a sequential manner starting with said first entry and ending at said last determining a match when said VPI/VCI value is present in one of said entries; and passing the contents of said one of said entries and said cell to a processor for further processing when said match is determined.

62. The method as in claim **61** including the steps of: determining an additional entry of said table for storage of a connection, including; setting said additional entry equal to said last incrementing said last incrementing said num_of_conn; storing an additional predetermined one of said connection values in said additional entry.

63. The method as in claim **62** further including the steps of: recognizing a delete connection value; comparing said delete connection delete connection value to entries in said table in a sequential manner starting with said first entry and ending at said last determining a match when said delete connection value is present in one of said entries; and storing a null value in said one of said entries, said null value being outside said range of connection values; decrementing said num_of_conn.

64. The method as in claim **63** including the steps of: recognizing a second additional connection value; determining a second additional entry of said table for storage of a connection value, said step of determining a second additional entry including; setting said second additional entry equal to said last incrementing said last incrementing said num_of_conn; comparing said null value to entries in said table in a sequential manner starting with said first entry and ending at said last determining a match when said null value is present in one of said entries; setting said second additional entry equal to said one of said entries when said match is determined; decrementing said last when said match is determined; and storing said second additional connection value in said second additional entry.

* * * * *